Sum-Checking SmallWood: Smaller Hash-Based Zero-Knowledge Arguments

Thibauld Feneuil, Killian Provin, and Matthieu Rivain

CryptoExperts, Paris, France
{thibauld.feneuil,killian.provin,matthieu.rivain}@cryptoexperts.com

Abstract. Hash-based zero-knowledge (ZK) arguments are particularly attractive for their conservative post-quantum security, relying only on cryptographic hash functions. While VOLE-in-the-Head proofs, based on GGM trees, are the most compact for very small witnesses, Merkle-tree-based approaches take over for larger ones. In this regime, the recent SmallWood scheme achieves the smallest proof sizes for witness sizes ranging from 2^6 to 2^{16}.

In this work, we present SmallWood$^+$, an enhanced variant of SmallWood that leverages sum-check-based protocols to further reduce proof sizes. Following the paradigm of Longfellow ZK, where a hash-based ZK argument is used to wrap the verification of a sum-check-based protocol, we formalize the notion of *Sum-Check Verification Constraint System* (SVCS), which captures a broad class of such protocols, including GKR for layered circuits and the sum-check protocol for R1CS. We then design generic compilers transforming SVCS instances into ZK arguments from linear commitment schemes and/or multilinear polynomial commitment schemes, which we efficiently instantiate from SmallWood.

Along the way, we develop tools of independent interest: hiding multilinear extensions, which make a multilinear evaluation of the committed witness safe to reveal in the small-field regime, a zero-knowledge version of the ring-switching protocol, and an improved circuit-flattening technique for GKR.

Applied to (layered) arithmetic circuits, hash functions, and lattice-based statements, our constructions establish new state-of-the-art proof sizes for hash-based ZK arguments – for instance, 18 KB and 63 KB to prove 1 and 100 Keccak permutations, against 29 KB and 127 KB for the best previous schemes.

Table of Contents

1 Introduction . . . 3
2 Preliminaries . . . 7
2.1 Polynomials and Multilinear Extensions . . . 7
2.2 The Sum-Check and GKR Protocols . . . 8
2.3 Commit and Prove Systems . . . 11
2.4 Degree-Enforcing Commitment Scheme . . . 17
3 Zero-Knowledge Proof Systems for Sum-Check-Based Protocols . . . 17
3.1 Sum-Check Verification Constraint System . . . 19
3.2 Simplifying SVCS Instances . . . 21
4 Compact Zero-Knowledge Arguments by Sum-Checking SmallWood . . . 23
4.1 Zero-Knowledge Arguments from Sum-Check-Based IP and Commitment Schemes . . . 24
4.2 SmallWood-Style Commitment Schemes . . . 27
4.3 Concrete Zero-Knowledge Argument System: SmallWood$^+$ . . . 33
5 Toolbox for Zero-Knowledge Sum-Check-Based Protocols . . . 35
5.1 Hiding Multilinear Extension . . . 35
5.2 Zero-Knowledge Ring-Switching Protocol . . . 40
5.3 Circuit Flattening for GKR . . . 46
6 Applications . . . 47
6.1 Arithmetic Circuits . . . 48
6.2 Hashing . . . 52
6.3 Lattices . . . 56
A Use of Generative AI . . . 63
B Optimized Sum-Check Protocol . . . 64
C Protocols in the CBIP Paradigm . . . 65
C.1 From Layered Circuits to SVCS: GKR in the CBIP Paradigm . . . 65
C.2 R1CS to SVCS Reduction in the CBIP Paradigm . . . 67
C.3 S-Box Circuit to SVCS Reduction in the CBIP Paradigm . . . 68
D Security Proofs . . . 69
D.1 Round-by-Round Knowledge Soundness of the DECS . . . 69
D.2 Sum-Check-Based Interactive Proofs and Soundness Preservation Under CBIP . . . 70
D.3 Security of Compiler 1 . . . 71
D.4 Security of Compiler 2 . . . 74
D.5 Security of SmallWood-LCS and SmallWood-MPCS . . . 76

1 Introduction

Zero-knowledge (ZK) proofs enable a prover to convince a verifier that a statement is true without revealing anything beyond its validity. Once a purely theoretical object, they have become a fundamental building block of modern cryptography, with applications ranging from signatures and anonymous credentials to verifiable computation and blockchain systems. Among the many families of ZK proof systems, *hash-based* constructions – or, more generally, *symmetric-crypto-based* constructions – hold a special place: they require no trusted setup, rely on cryptographic hash functions (and/or pseudorandom functions) as their only assumption, and thereby offer conservative, plausibly post-quantum security. This work focuses on minimizing the proof size of hash-based zero-knowledge arguments.

The state of the art in this area is best described in terms of witness size. For very small witnesses – a few dozen field elements, as typically arising in signature schemes – the MPC-in-the-Head paradigm [IKOS07] and its modern incarnations, VOLE-in-the-Head (VOLEitH) [BBD+23, BBB+24] and Threshold-Computation-in-the-Head (TCitH) [FR25], yield the most compact proofs. At the other end of the spectrum, for very large statements, FRI-based systems such as STARK [BBHR18b] – further improved by recent advances on the underlying proximity tests such as STIR [ACFY24a] and WHIR [ACFY24b] – achieve polylogarithmic proof sizes, but with substantial constants that make them competitive only when the witness counts hundreds of thousands of elements or more. In between lies a wide regime – witnesses from a few dozen to a few tens of thousands of elements, covering many statements of practical interest such as hash preimages and evaluations of symmetric primitives – where Merkle-tree-based approaches in the lineage of Ligero [AHIV17, AHIV23] remain the most compact. The recent SmallWood scheme [FR26], built on Ligero-style and TCitH techniques, currently achieves the smallest proof sizes in this regime, for witness sizes ranging from 2^6 to 2^{16}. This intermediate regime is the focus of the present work.

A complementary line of work leverages *sum-check-based interactive proofs*, such as the GKR protocol, whose communication grows only logarithmically in the width of the computation and which only require committing to a small part of it (e.g., the inputs and outputs of a layered circuit rather than its full trace). These protocols are not zero-knowledge on their own, but they can be compiled into zero-knowledge arguments by letting the prover commit to its secret values and prove the verifier's checks in zero knowledge, a blueprint that conceptually dates back to the seminal work of Ben-Or, Goldreich, Goldwasser, Håstad, Kilian, Micali and Rogaway [BGG+90]. This approach underlies several GKR-based argument systems, such as Hyrax [WTs+18], Libra [XZZ+19] and Virgo [ZXZS20], respectively instantiated with discrete-logarithm, pairing-based and FRI-based polynomial commitments. The recent Longfellow ZK system [Fas26, FS26] revisits this blueprint with a Ligero-style commit-and-prove system, which is particularly effective in the intermediate witness-size regime discussed above: the prover commits to the witness, runs the sum-check-based protocol with masked messages, and finally proves the verifier's checks on the committed values. The resulting arguments have proven compelling in practice: Longfellow ZK notably enables proving statements about legacy ECDSA-signed identity credentials in a few hundred milliseconds on a mobile device, with a proof size of a few hundred kilobytes [Fas26].

In this work, we build upon this approach and generalize it into a framework applicable to any sum-check-based interactive proof and any commitment scheme supporting linear openings and/or multilinear polynomial evaluations. We then optimize the resulting framework by instantiating it with SmallWood-style commitment schemes, which outperform Ligero precisely in the targeted witness-size regime. Along the way, we introduce different techniques to increase the efficiency and flexibility of the framework, and we address technical challenges raised by this instantiation to preserve zero knowledge.

Our Contributions. We revisit and generalize the approach of Longfellow ZK [Fas26, FS26], which wraps the verification of a sum-check-based protocol into a hash-based zero-knowledge argument, and we instantiate the resulting framework with SmallWood-style commitment schemes [FR26] to obtain hash-based zero-knowledge arguments with state-of-the-art proof sizes. Throughout the paper, we refer to this methodology – committing to the witness and a random *trace mask*, running the interactive protocol with masked prover messages, and finally proving the verifier's residual checks on the committed values – as the *commit, blindly interact, and prove* (CBIP) paradigm. In more detail, our contributions are the following:

- *The SVCS abstraction.* We introduce the *sum-check verification constraint system* (SVCS), a constraint system capturing the residual checks left to the verifier when a sum-check-based protocol is run with masked prover messages: multilinear evaluation constraints on the witness and low-degree constraints on the trace mask. Sum-check-based protocols – in particular the GKR protocol – naturally reduce to (extended) SVCS instances, and simple techniques (merging of evaluation claims, linearization of trace constraints) further reduce any SVCS instance to constraints provable with commitment schemes supporting linear openings and multilinear polynomial evaluations.
- *Generic compilers.* We describe two compilers turning any sum-check-based interactive proof into a zero-knowledge argument in the CBIP paradigm. The first compiler relies on a single *linear commitment scheme* (LCS); the second one splits the work between an LCS for the trace mask and a *multilinear polynomial commitment scheme* (MPCS) for the witness, which yields better efficiency when the witness is heavier. We prove both compilers round-by-round knowledge sound and zero-knowledge, making them compatible with the Fiat–Shamir transform.
- *SmallWood-style commitment schemes.* We instantiate the two compilers with new commitment schemes in the style of SmallWood [FR26], both built on the *degree-enforcing commitment scheme* (DECS) [FR25, FR26], a Merkle-tree-based small-domain polynomial commitment scheme in the lineage of Ligero [AHIV17, AHIV23]. Specifically, we construct a linear commitment scheme SmallWood-LCS and a multilinear polynomial commitment scheme SmallWood-MPCS, and we show how to merge their two DECS instances into a single Merkle tree, thus mutualizing the commitment and opening costs when both schemes are used jointly (as in our second compiler).
- *Toolbox for zero-knowledge sum-check-based protocols.* To build our concrete zero-knowledge arguments, we develop several tools that we believe to be of independent interest:
  - First, some protocols require revealing a multilinear evaluation of the witness in the clear, which may compromise zero knowledge. This is, for example, the case in our second compiler. To address this issue, we introduce the notion of a *hiding multilinear extension* (hiding MLE), which revisits the random-padding principle of [FKR26] in the small-field regime, where a handful of masks from the base field $\mathbb{F}$ must hide an evaluation lying in the extension field $\mathbb{K}$. We show how to achieve zero knowledge in this setting by carefully choosing the placement of the masks in the MLE, and we further propose a rejection-based approach ensuring it constructively while sampling the challenge point.
  - Our second tool is a zero-knowledge variant of the ring-switching protocol introduced in [DP26]. This protocol provides a way to commit to elements of a tiny field (*e.g.* the binary field) while avoiding the communication overhead that would otherwise arise when the underlying commitment scheme only supports commitments to elements of larger fields. We achieve zero knowledge by combining our hiding MLE technique with an execution of the underlying sum-check protocol in the CBIP paradigm.
  - Finally, we improve the circuit-flattening technique for GKR introduced in [Lon26], which trades circuit depth against witness size by committing to the intermediate states of the computation. On the one hand, the flexibility of the extended SVCS enforces for free that the output of a block and the input of the next one are read from the same coordinates of the committed witness, dispensing with the additional consistency gates required in [Lon26]; on the other hand, the block structure of the flattened circuit can be exploited within the GKR sum-checks to further reduce communication.
- *Applications and concrete proof sizes.* We apply the resulting argument systems to several classes of statements: unstructured arithmetic circuits (via R1CS), layered arithmetic circuits (via GKR), standard hash functions (Keccak and Poseidon), and lattice-based statements. For arithmetic circuits of size below 2^{15}, our constructions outperform the previous state of the art, namely SmallWood [FR26] for unstructured circuits and Longfellow ZK [FR26, Fas26] for layered circuits. When proving knowledge of a preimage of Keccak (respectively, 10 preimages), our schemes achieve proof sizes of approximately 18 KB (resp. 31 KB), compared with 60 KB (resp. 82 KB) for the previous state of the art, corresponding to an improvement by a factor of roughly three. We also obtain smaller proofs for proving knowledge of solutions to LWE instances in relevant parameter regimes. For instance, for instances with parameters corresponding to Kyber512 and Dilithium2, we achieve proof sizes of 12.1 KB and 13.2 KB, respectively, compared with 14.1 KB and 17.5 KB for the previous hash-based approach.

– *Round-by-round knowledge soundness for commit-and-prove systems in the ROM.* As a side contribution of independent interest, we adapt round-by-round knowledge soundness [CCH+19, CMS19] to interactive commit-and-prove systems in the random oracle model, where the statement is only known once the commitment has been sent and the prover's hash queries must be accounted for. The resulting notion composes which is what allows our compilers to inherit the security of generic commitment schemes, and we show that the DECS of [FR26] satisfies it.

Technical Overview. In the CBIP paradigm, the prover first commits to the witness $w \in \mathbb{F}^\omega$ together with a random trace mask $t \in \mathbb{K}^\tau$ using a commit-and-prove system. The prover and the verifier then run a (sum-check-based) interactive protocol, with the twist that each prover message is masked by a fresh coordinate of t: instead of m_i, the prover sends $m_i + t_i$. The verifier thus learns nothing from the transcript, but can no longer perform its checks in clear; instead, each check becomes a constraint on the committed pair (w, t).

The SVCS abstraction. The SVCS abstraction captures exactly the constraints that arise this way. Two types of constraints occur: *multilinear evaluation constraints* $\widetilde{w}(r) = \psi(t)$, stating that the multilinear extension of the witness at a verifier-chosen point equals an affine combination of the masked messages, and *trace constraints* $f(t) = 0$ of low degree, accounting for the few nonlinear checks of the verifier (typically, multiplications between prover messages). The point of this design is efficiency: the sum-check machinery handles all the nonlinear work of the statement, while the commitment scheme is only ever asked to open *linear* functions (or *multilinear* polynomial evaluations) of the committed values. In the case of GKR, an additional benefit is that only the circuit's inputs and outputs need to be committed, not the full computation trace.

We propose two generic simplifications making SVCS instances friendly to the hash-based commitment schemes considered in this work. First, we consider an *extended* version of SVCS in which the evaluation constraints may bear, instead of on $\widetilde{w}$ itself, on the multilinear extensions of vectors $p_i = A_i \cdot w + b_i$ obtained from the witness through public affine maps. This extension brings valuable flexibility: in GKR for instance, the input and output of the circuit may be partly public, partly secret, redundant, or laid out in an arbitrary order, and the matrices A_i relate the committed witness to the vectors actually processed by the protocol at no re-encoding cost. This extension also allows us to avoid sum-check executions whose sole purpose is to perform a linear computation on the witness. For example, it reduces the protocol of [Set20] for proving R1CS satisfiability from two sum-check executions to a single one. We then show how to *merge* all these evaluation claims into a single one: writing each claimed evaluation as

$$\widetilde{p_i}(r_i) = \sum_{y \in \{0,1\}^\ell} \left(\widetilde{A_i}(r_i, y) \cdot \widetilde{w}(y) + \widetilde{b_i}(r_i, y)\right) ,$$

a random linear combination of the claims forms a new degree-2 sum-check statement, whose execution in the CBIP paradigm collapses all the constraints into a single one of the form $\widetilde{w}(r) = \psi'(t)$. Second, the degree-$d$ trace constraints are *linearized*: using an interpolation technique inspired from linear PCPs [BBC+19], m multiplication constraints on t are traded for m extra mask coordinates and two linear constraints. After these steps, every constraint is either a linear constraint on (w, t) or a single multilinear evaluation of $\widetilde{w}$.

Compilers and commitment schemes. Our first compiler then discharges everything with a single linear commitment scheme (LCS), since a multilinear evaluation $\widetilde{w}(r)$ is itself a linear function of w. Our second compiler splits the work: a multilinear polynomial commitment scheme (MPCS) proves the evaluation claim on $\widetilde{w}$, while an LCS handles the trace mask. Since an MPCS is a weaker primitive than an LCS, it admits a more compact instantiation, and the split pays off whenever $\omega \gg \tau$. We instantiate both commitment schemes from the degree-enforcing commitment scheme (DECS) underlying SmallWood. The committed vector w is arranged as a matrix $W \in \mathbb{F}^{n \times s}$ for a packing parameter s, whose rows are interpolated – together with a few random values ensuring zero-knowledge – into a vector polynomial $\boldsymbol{P}$ satisfying $\boldsymbol{P}(\Omega) = W$ over a public domain Ω; the DECS commits to the evaluations of $\boldsymbol{P}$ over a small domain through a Merkle tree, and enforces the degree bound by a random-linear-combination test in the spirit of Ligero. In SmallWood-LCS, a batch of linear constraints $\langle a_i, w \rangle = b_i$ is proven Ligero-style, by revealing a single masked random combination Q of the polynomials $\langle \boldsymbol{A}_i, \boldsymbol{P} \rangle$ (where $\boldsymbol{A}_i$ encodes a_i)

and checking that $\sum_{x\in\Omega} Q(x) = \sum_i \gamma_i \cdot b_i$, together with a few Merkle openings. SmallWood-MPCS additionally exploits the tensor structure of multilinear evaluations, which rewrite as $\widetilde{w}(r) = v_{\mathrm{L}}^{\top} \cdot W \cdot v_{\mathrm{R}}$ for public vectors $v_{\mathrm{L}}, v_{\mathrm{R}}$ derived from r, so that proving $\widetilde{w}(r) = v$ amounts to proving the correctness of a single row combination $\langle v_{\mathrm{L}}, \boldsymbol{P}\rangle$. In the second compiler, the DECS instances of the LCS and the MPCS share a single Merkle tree, so that the commitment and opening overheads are paid only once. The resulting proof size decomposes as the masked trace plus the commitment openings, which grow sublinearly in ω; concrete formulas and parameter optimizations are given in Section 4.2.

Toolbox. Beyond the framework itself, our concrete schemes rely on a few additional tools that we believe to be of independent interest. First, some protocols require revealing in the clear a multilinear evaluation $v = \widetilde{w}(r)$ at some point r, which may leak information about the witness w. To address this issue, we revisit the idea of random padding of MLE proposed in [FKR26], which we formalize through the notion of *hiding multilinear extension* (hiding MLE) and extend to the small-field regime. The idea is to pad w with k random masks $m \in \mathbb{F}^k$ before taking its multilinear extension, so that the revealed value becomes $v = \widetilde{w}(r) + \sum_{x\in M} \chi_x(r)\, m_x$, where $M \subseteq \{0,1\}^\ell$ denotes the set of mask positions and the χ_x's are the multilinear Lagrange basis polynomials. Since the masks lie in $\mathbb{F}$ while r and v live in $\mathbb{K}$, the masking term is uniformly distributed over $\mathbb{K}$ if and only if the coefficients $\{\chi_x(r) : x \in M\}$ span $\mathbb{K}$ as an $\mathbb{F}$-vector space. We exhibit a mask placement $M^\star$ for which this spanning condition fails only with negligible probability while using barely more than $[\mathbb{K} : \mathbb{F}]$ masks. The placement $M^\star$, however, only applies in a restricted parameter range and uses slightly more masks than the information-theoretic minimum. Our second observation removes both limitations: whether a point r is hiding depends only on public data, namely the placement M and the point itself, so that the prover can *test* the condition and the verifier can re-check it, which yields a rejection-based sampling of the evaluation point r. In the non-interactive setting obtained through the Fiat–Shamir transformation, the coordinates of r are derived one by one from the prover's messages and, whenever appending a coordinate breaks the (partial) spanning condition, the prover increments a counter and re-derives that coordinate alone, the counters being included in the proof. The evaluation point is then hiding *by construction*, so that zero knowledge holds perfectly and with the smallest possible mask, namely $k = [\mathbb{K} : \mathbb{F}]$ field elements. Resampling coordinate-wise rather than restarting the execution is what makes this affordable, since for some of our applications the rejection probability is far from negligible. Finally, this mechanism gives no advantage to a malicious prover: the rejection rule discards a predetermined set of challenge values, independently of the witness and of the cheating strategy, so that the number of hash evaluations needed to reach a challenge breaking soundness is not reduced.

Our second tool is a zero-knowledge variant of the ring-switching technique of [DP26]. We first generalize ring switching to a three-field setting, in which the MLE evaluation points and the commitment need not be defined over the same field. A direct application of the CBIP paradigm to the entire ring-switching protocol is problematic because the protocol requires decomposing intermediate values over a subfield, an operation that is not conveniently represented within the CBIP paradigm. Instead, we use hiding MLEs to allow the prover to reveal the required partial evaluations in the clear without leaking information about the witness. The required subfield decompositions can then be performed directly on these public values. The subsequent sum-check protocol, which proves a relation involving the resulting public values, can finally be executed in the CBIP paradigm.

Finally, since the proof size of GKR grows linearly with the circuit depth, deep circuits benefit from being *flattened* before entering our framework. The circuit is decomposed into K blocks as $C = C_K \circ \cdots \circ C_1$ and the intermediate states $z_i = C_i(z_{i-1})$ are committed as part of the witness, thereby trading circuit depth for witness size. Our treatment improves upon the recent proposal of [Lon26] in two ways. First, the flexibility of extended SVCS allows us to directly enforce that the output of C_i and the input of C_{i+1} are read from the same coordinates of the committed witness, eliminating the additional consistency gates required in [Lon26]. Second, the block structure of the flattened circuit can be exploited within the GKR sum-check protocols to further reduce communication.

Related Works. Endowing sum-check-based protocols with zero knowledge has been addressed along two main directions. The first direction makes the sum-check protocol itself zero-knowledge, by having the prover mask the summed polynomial with a committed low-degree random polynomial; this algebraic approach was introduced by Chiesa, Forbes and Spooner [CFS17] and is used, e.g., in Libra [XZZ+19] and

Virgo [ZXZS20]. The second direction, which we follow after Longfellow ZK [Fas26], leaves the sum-check protocol unchanged but masks each prover message with a committed random value, and defers the verifier's checks to the commit-and-prove system; this is an instance of the general methodology of Ben-Or et al. [BGG+90]. The latter direction keeps the interactive protocol untouched: any sum-check-based protocol can be plugged in as a black box, which is what our SVCS abstraction captures and exploits.

More generally, two standard methods exist for adding zero knowledge to a (non-ZK) hash-based succinct proof system. One can compose it with an outer zero-knowledge proof that the base verifier would have accepted – as done, e.g., in Orion [XZS22] and in the final wrapping phase of modern zkVMs – at the cost of proving cryptographic hash computations (e.g., Merkle path verifications) inside the outer system. Alternatively, one can make each component of the proof system natively zero-knowledge [CFS17, BCF+17, BCR+19, XZZ+19, RW24, CFW26], which avoids this overhead but requires intrusive, protocol-specific modifications. The recent work VEIL [DHRR26] strikes a balance between the two and, like us, builds upon and generalizes the approach of Longfellow ZK, but with complementary goals: VEIL targets a minimal *prover overhead* for making existing large-scale IOP-based proof systems (e.g., zkVMs with traces of 2^{25} elements and beyond) zero-knowledge in a non-intrusive way, with proof sizes reaching hundreds of kilobytes, whereas we target minimal *proof size* in the small-to-medium witness regime, which leads us to co-design the whole argument system rather than wrap an existing one.

Finally, our toolbox builds upon several previous works. The idea of padding the witness with random values before taking its multilinear extension is due to [FKR26], in a setting where all the objects live in a cryptographically large field: rejection is then negligibly rare, so that a leaking evaluation point can simply abort the protocol. Our contribution is to make the technique work in the small-field regime, where the placement of the masks becomes critical and where the rejection probability may be far from negligible, which we address by a coordinate-wise rejection sampling of the challenge point. Our second tool is a zero-knowledge variant of the ring-switching protocol of [DP26], further generalized to three fields, and our flattening technique improves the proposal of [Lon26] in the two ways described above.

2 Preliminaries

2.1 Polynomials and Multilinear Extensions

Throughout the paper, $\mathbb{F}$ denotes a finite field and $\mathbb{K}$ denotes an extension field of $\mathbb{F}$. We write $\mathbb{F}[X]$ for the ring of univariate polynomials over $\mathbb{F}$, and $\mathbb{F}[X_1, \ldots, X_n]$ (or $\mathbb{F}[X]$ with $X = (X_1, \ldots, X_n)$ when n is clear from context) for the ring of n-variate polynomials over $\mathbb{F}$. For a polynomial $p \in \mathbb{F}[X_1, \ldots, X_n]$, we denote by $\deg_i(p)$ the partial degree of p in the variable X_i and by $\deg(p)$ its total degree, defined as the maximum, over all monomials of p, of the sum of the exponents of the variables in the monomial. We write $\mathbb{F}[X]^{\leq d}$ for the set of all polynomials over $\mathbb{F}$ of (total) degree at most d.

A polynomial p is said to be *multilinear* if $\deg_i(p) \leq 1$ for all $i \in \{1, \ldots, n\}$. Any polynomial in $\mathbb{F}[X_1, \ldots, X_n]$ can be evaluated over an extension field $\mathbb{K}$ of $\mathbb{F}$, and we use the same notation p for the polynomial and for the corresponding function $\mathbb{K}^n \to \mathbb{K}$, $x \mapsto p(x)$, when no confusion arises. We recall that the Schwartz-Zippel lemma states that any non-zero polynomial $p \in \mathbb{K}[X_1, \ldots, X_n]$ of total degree d vanishes on a uniformly random point $r \in S^n$, with $S \subseteq \mathbb{K}$, with probability at most $d/|S|$.

Finally, for a bit string $x \in \{0,1\}^\ell$, we denote $\langle x \rangle$ the integer with binary representation x.

Multilinear Extensions. Let $f : \{0,1\}^\ell \to \mathbb{F}$ be a function defined over the Boolean hypercube. The *multilinear extension* (MLE) of f is the unique multilinear polynomial $\tilde{f} \in \mathbb{F}[X_1, \ldots, X_\ell]$ such that $\tilde{f}(x) = f(x)$ for all $x \in \{0,1\}^\ell$. By identifying a vector $u = (u_0, \ldots, u_{2^\ell - 1}) \in \mathbb{F}^{2^\ell}$ with the function $f_u : \{0,1\}^\ell \to \mathbb{F}$ defined by $f_u(x) = u_{\langle x \rangle}$, the MLE of u refers to the MLE of f_u and is denoted $\tilde{u}$. The MLE admits the following closed-form expression in the Lagrange basis:

$$\tilde{f}(X_1, \ldots, X_\ell) = \sum_{x \in \{0,1\}^\ell} f(x) \cdot \widetilde{\mathrm{eq}}(x, X) \ ,$$

where $\widetilde{\mathrm{eq}}$ is the multilinear extension of the equality function $\mathrm{eq} : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}$, given by

$$\widetilde{\mathrm{eq}}(X, Y) = \prod_{i=1}^{\ell} \big(X_i \cdot Y_i + (1 - X_i)(1 - Y_i) \big) \ .$$

Although $\tilde{f}$ has coefficients in $\mathbb{F}$, it can be evaluated over any extension field $\mathbb{K}$ of $\mathbb{F}$, in which case we view it as a function $\tilde{f} : \mathbb{K}^n \to \mathbb{K}$. Multivariate functions $f : \{0,1\}^{n_1} \times \cdots \times \{0,1\}^{n_k} \to \mathbb{F}$ admit a unique *multilinear extension* that is multilinear in each block of variables; we use the same notation $\tilde{f}$ for this extension. In particular, an $\omega \times \omega$ matrix A over $\mathbb{F}$ (with $\omega = 2^\ell$) is identified with the function $A : \{0,1\}^\ell \times \{0,1\}^\ell \to \mathbb{F}$ and admits a bivariate multilinear extension $\tilde{A} : \mathbb{F}^\ell \times \mathbb{F}^\ell \to \mathbb{F}$.

Virtual Polynomials. Throughout the paper, we shall consider polynomials whose definition involves the evaluation of one or several other polynomials. We refer to such polynomials as *virtual polynomials* in the spirit of [CBBZ23, Tha25]. Formally, given a family of polynomials $p_1, \ldots, p_k$ with $p_i \in \mathbb{F}[Y_{i,1}, \ldots, Y_{i,\ell_i}]$, a *virtual polynomial* with respect to $p_1, \ldots, p_k$ is a polynomial $g \in \mathbb{F}[X_1, \ldots, X_\ell]$ that can be expressed as

$$g(X) = F\big(X,\, p_1(L_1(X)),\, \ldots,\, p_k(L_k(X))\big)\ ,$$

where $F \in \mathbb{F}[X_1, \ldots, X_\ell, Z_1, \ldots, Z_k]$ is a public polynomial and each $L_i : \mathbb{F}^\ell \to \mathbb{F}^{\ell_i}$ is a public affine map. The *total degree* of g as a virtual polynomial with respect to $p_1, \ldots, p_k$ is the total degree of F when each Z_i is assigned degree $\deg(p_i)$. In our applications, the underlying polynomials p_i will typically be the multilinear extensions of committed vectors (*e.g.* the witness $\tilde{w}$ or some matrix-related polynomials $\tilde{A}, \tilde{B}, \tilde{C}$), and a virtual polynomial gives a compact way to describe a polynomial whose evaluations are not directly available but can be derived from evaluations of the p_i's.

2.2 The Sum-Check and GKR Protocols

The sum-check protocol is a fundamental interactive proof protocol introduced by Lund, Fortnow, Karloff and Nisan in [LFKN92]. This protocol allows a prover to convince a verifier that the sum of a multivariate polynomial $g \in \mathbb{F}[X_1, \ldots, X_\ell]$, over the Boolean hypercube $\{0,1\}^\ell$ equals a claimed value. Namely, the protocol proves a statement of the form:

$$\sum_{(x_1,\ldots,x_\ell)\in\{0,1\}^\ell} g(x_1, \ldots, x_\ell) = c$$

for some constant c. The sum-check protocol reduces the above statement to a claim of the form:

$$g(r_1, \ldots, r_\ell) = c'$$

where $r_1, \ldots, r_\ell \in \mathbb{K}$ are random field elements chosen by the verifier, for $\mathbb{K}$ an extension field of $\mathbb{F}$. Namely, proving the sum over the Boolean hypercube is reduced to proving a single evaluation of the polynomial at a random point of $\mathbb{K}^\ell$.

This protocol can be used in many contexts where the access to the polynomial g might differ. This only impacts the final step, namely checking the correctness of the evaluation of g at a random point. A description of the sum-check protocol is given in Figure 1.

In the standard version of the sum-check protocol described in Figure 1, the prover sends the full polynomial $g_i(X_i)$ at each iteration, which requires $\deg(g_i)+1 = \deg_i(g)+1$ field elements, where $\deg_i(g)$ is the degree of g in variable X_i, making a total of $\ell + \sum_{i=1}^{\ell} \deg_i(g)$ field elements. This communication can be reduced by ℓ field elements by sending only the evaluations of g_i at $1, \ldots, d$ where $d = \deg(g_i)$ and deducing the constant term by $g_i(0) = g_{i-1}(r_{i-1}) - g_i(1)$. In this variant, all the intermediate checks performed by the verifier are satisfied by construction and the only check performed by the verifier is the final check $c' = g(r_1, r_2, \ldots, r_\ell)$. Along this paper, we shall only consider this variant of the sum-check protocol. The obtained (prover to verifier) communication of the sum-check protocol is then given by $\sum_{i=1}^{\ell} \deg_i(g)$ field elements. For $\deg(g) = d$ the total degree of g, the communication is at most of $\ell \cdot d$ field elements.

The GKR Protocol. The GKR protocol is an interactive proof for verifying the correctness of computations represented as layered arithmetic circuits based on the sum-check protocol. It was introduced by Goldwasser, Kalai and Rothblum in [GKR08] and improved in several subsequent works [CMT12, Tha13, WJB+17, WTs+18]. We use the generalization of layered arithmetic circuits from [Fas26] where each gate is a general quadratic function and further generalize it to the case of functions of prescribed

The following steps are repeated for $i = 1, \ldots, \ell$:

- $\mathcal{P}$ sends a polynomial $g_i(X_i)$ to $\mathcal{V}$, claiming that

$$g_i(X_i) = \sum_{x_{i+1},\ldots,x_\ell} g(r_1, \ldots, r_{i-1}, X_i, x_{i+1}, \ldots, x_\ell) \ .$$

- $\mathcal{V}$ checks that

$$g_i(0) + g_i(1) = \begin{cases} c & \text{if } i = 1 \ , \\ g_{i-1}(r_{i-1}) & \text{if } i > 1 \ . \end{cases}$$

- $\mathcal{V}$ samples a field element $r_i \in \mathbb{K}$ and sends it to $\mathcal{P}$.

End of the repetition.

- $\mathcal{V}$ computes $c' = g_\ell(r_\ell)$.
- $\mathcal{P}$ and $\mathcal{V}$ run a protocol to check that

$$c' = g(r_1, r_2, \ldots, r_\ell) \ .$$

Fig. 1: The sum-check protocol.

degree d. For the sake of simplicity, we first assume that these functions are homogeneous, i.e., all the monomials of the function have the same degree (we discuss the non-homogeneous case in Remark 1). In this setting, the circuit is represented as a sequence of layers of gates, with the output of each layer serving as the input to the next layer. The state at step i is a vector of 2^{ℓ_i} field elements, indexed by $k \in \{0,1\}^{\ell_i}$, and defined as:

$$V_i(k) = \sum_{j_1,\ldots,j_d \in \{0,1\}^{\ell_{i-1}}} f_i(k, j_1, \ldots, j_d) \ , \tag{1}$$

with

$$f_i(k, j_1, \ldots, j_d) = F_i(k, j_1, \ldots, j_d) \cdot \prod_{u=1}^{d} V_{i-1}(j_u) \ , \tag{2}$$

where $F_i \in \mathbb{F}[K, J_1, \ldots, J_d]$, for $K = (K_1, \ldots, K_{\ell_i})$ and $J_u = (J_{u,1}, \ldots, J_{u,\ell_{i-1}})$, are multilinear polynomials that encode the circuit's degree-d gates at layer i. Namely, $V_i(k)$ is a general homogeneous degree-d polynomial function of the state $\{V_{i-1}(j)\}_{j \in \{0,1\}^{\ell_{i-1}}}$ at step $i-1$.

Let n be the number of layers in the circuit so that $(V_n(k))_{k \in \{0,1\}^{\ell_n}}$ is the output of the circuit. The GKR protocol proves a statement of the form $C(x) = y$ where C is the function computed by the circuit, x is the input and y is the claimed output. In this claim, x and y have been somehow committed to the verifier (e.g. sent in clear and through some binding commitment scheme). The GKR protocol reduces this claim to the following claims about the input and output of the circuit:

$$\begin{cases} \tilde{y}(s_n) = c_n \ , \\ \tilde{x}(r_{1,m}) = v_{0,m} \text{ for all } m \in \{1, \ldots, d\} \ , \end{cases}$$

where $\tilde{y}$ and $\tilde{x}$ are the multilinear extensions of y and x respectively, $s_n \in \mathbb{K}^{\ell_n}$ and $r_{1,m} \in \mathbb{K}^{\ell_0}$ for $m = 1, \ldots, d$ are random points chosen by the verifier, $c_n \in \mathbb{K}$ and $v_{0,m} \in \mathbb{K}$ are field elements derived from the prover messages. In case the output y is fully known to the verifier, the claim $c_n = \tilde{y}(s_n)$ can be directly checked by the verifier and the prover does not need to send c_n. In case the output y is only partially known or even completely private, the claim $c_n = \tilde{y}(s_n)$ needs to be checked by the verifier from a previous commitment to y.

The GKR protocol is described in Figure 2. Each iteration involves a sum-check protocol, with communication $2\,d \cdot \ell_{i-1}$ (virtual polynomial of degree 2 in $d \cdot \ell_{i-1}$ variables)[1] and sending the claimed

[1] This communication assumes the optimized version of the sum-check protocol (see Appendix B) and does not include the communication of the final check $c' = g(r_1, \ldots, r_n)$ in the sum-check protocol.

– $\mathcal{V}$ samples a random point $s_n \in \mathbb{K}^{\ell_n}$ and sends it to $\mathcal{P}$.
– $\mathcal{P}$ computes $c_n = \tilde{y}(s_n)$ and sends it to $\mathcal{V}$, claiming $c_n = \tilde{y}(s_n)$ and $c_n = V_n(s_n)$. The former claim is either directly checked by the verifier (if y is fully known) or needs to be checked by the verifier from a previous commitment to y (if y is partially or fully private). The latter claim is checked by the rest of the GKR protocol.

  At this stage $\mathcal{V}$ needs to check the claim $V_n(s_n) = c_n$. This is done by iteratively checking the correctness of each layer of the circuit, starting from the output layer n down to the input layer 0.

  The following steps are repeated for $i = n$ down to 1.

– $\mathcal{P}$ and $\mathcal{V}$ defines the virtual polynomial g_i as follows:

$$g_i(j_1, \ldots, j_d) := \begin{cases} f_n(s_n, j_1, \ldots, j_d) & \text{if } i = n, \\ \alpha_{i,1} \cdot f_i(r_{i+1,1}, j_1, \ldots, j_d) + \cdots + \alpha_{i,d} \cdot f_i(r_{i+1,d}, j_1, \ldots, j_d) & \text{otherwise.} \end{cases}$$

  where f_i is defined in Equation (2). In this iteration, the protocol aims to check the following claim:

$$\sum_{j_1, \ldots, j_d \in \{0,1\}^{\ell_{i-1}}} g_i(j_1, \ldots, j_d) = c_i \ .$$

– $\mathcal{P}$ and $\mathcal{V}$ runs the sum-check protocol to check this claim. At the end of this sum-check protocol, the above claim is reduced to the claim $g_i(r_{i,1}, \ldots, r_{i,d}) = c'_i$ for $r_{i,1}, \ldots, r_{i,d} \in \mathbb{K}^{\ell_{i-1}}$ the random points chosen by the verifier and c'_i the final field element computed by the verifier in the sum-check protocol.

– $\mathcal{P}$ sends field elements $v_{i-1,1}, \ldots, v_{i-1,d}$ claiming that $v_{i-1,m} = V_{i-1}(r_{i,m})$ for $m = 1, \ldots, d$.
– $\mathcal{V}$ checks that

$$c'_i = \begin{cases} F_i(s_n, r_{i,1}, \ldots, r_{i,d}) \cdot \prod_{u=1}^{d} v_{i-1,u} & \text{if } i = n, \\ \sum_{m=1}^{d} \alpha_{i,m} \cdot F_i(r_{i+1,m}, r_{i,1}, \ldots, r_{i,d}) \cdot \prod_{u=1}^{d} v_{i-1,u} & \text{otherwise.} \end{cases}$$

  At this stage, the claim is reduced to checking that $V_{i-1}(r_{i,m}) = v_{i-1,m}$ for all $m = 1, \ldots, d$.
– If $i > 1$ (*this is not the last iteration*), $\mathcal{V}$ samples random field elements $\alpha_{i-1,1}, \ldots, \alpha_{i-1,d} \in \mathbb{K}$ and sends them to $\mathcal{P}$. $\mathcal{P}$ and $\mathcal{V}$ compute:

$$c_{i-1} := \alpha_{i-1,1} \cdot v_{i-1,1} + \cdots + \alpha_{i-1,d} \cdot v_{i-1,d} \ .$$

  The claim is reduced to checking $\alpha_{i-1,1} \cdot V_{i-1}(r_{i,1}) + \cdots + \alpha_{i-1,d} \cdot V_{i-1}(r_{i,d}) = c_{i-1}$. The latter implies $V_{i-1}(r_{i,m}) = v_{i-1,m}$ for all $m = 1, \ldots, d$ with probability at least $1 - 1/|\mathbb{K}|$ over the randomness of $\alpha_{i-1,1}, \ldots, \alpha_{i-1,d}$. This claim is checked by the next iteration.

  End of the repetition. At this stage, the claim is reduced to checking that $V_0(r_{1,m}) = v_{0,m}$ for $m = 1, \ldots, d$ for some random points $r_{1,1}, \ldots, r_{1,d} \in \mathbb{K}^{\ell_0}$ and some field elements $v_{0,1}, \ldots, v_{0,d} \in \mathbb{K}$.

– $\mathcal{P}$ and $\mathcal{V}$ runs a protocol to check that $v_{0,m} = \tilde{x}(r_{1,m})$ for $m = 1, \ldots, d$ where $\tilde{x}$ is the multilinear extension of the input x committed to the verifier.

Fig. 2: The GKR protocol with general degree-d gates and random linear combinations.

evaluations $v_{i,j}$'s (d field elements). The total communication is hence:

$$\underbrace{2\,d \cdot \sum_{i=1}^{n} \ell_{i-1}}_{\text{sum-check}} + \underbrace{n \cdot d}_{v_{i,j}\text{'s}} .$$

If all the layers of the circuit have the same width 2^ℓ, the communication is $n \cdot d \cdot (2\,\ell + 1)$.

Remark 1 (GKR Generalizations). The GKR protocol can be generalized to the case of non-homogeneous degree-d gates, by defining the multivariate polynomial V_i at layer i as:

$$V_i(k) = \sum_{j_1,\ldots,j_d \in \{0,1\}^{\ell_{i-1}}} f_i(k, j_1, \ldots, j_d) \tag{3}$$

with

$$f_i(k, j_1, \ldots, j_d) = \sum_{m=0}^{d} F_i^{(m)}(k, j_1, \ldots, j_d) \cdot \prod_{u=1}^{m} V_{i-1}(j_u) \ , \tag{4}$$

where $F_i^{(m)} \in \mathbb{F}[K, J_1, \ldots, J_d]$ for every $m \in \{0, \ldots, d\}$. We note that different polynomials $F_i^{(m)}$, for $m < d$, can lead to equivalent functions f_i by definition and we can assign $F_i^{(m)}(k, j_1, \ldots, j_d) = 0$ for all $(j_{m+1}, \ldots, j_d) \neq 0$ to ensure a unique representation. While making the expression slightly more complex, this generalization does not change the communication complexity of the GKR protocol. The GKR protocol can also be generalized to the case of general gates with different degrees d_i at each layer i. In this case, the communication complexity of the GKR protocol is given by $2\sum_{i=1}^{n} d_i \ell_{i-1} + \sum_{i=0}^{n-1} d_i$.

2.3 Commit and Prove Systems

A *commit-and-prove* (CP) system is a primitive that allows a prover to commit to a witness w and to subsequently prove that w satisfies some public statement. In this paper, we consider transparent, interactive CP systems, in the random oracle model (ROM).

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 interactive protocol, namely interactive protocols between a stateful (PPT) prover $\mathcal{P}$ and a stateful (PPT) verifier $\mathcal{V}$. Along this paper, $\mathcal{V}$ is always a *public-coin verifier* meaning that all the messages sent by $\mathcal{V}$ are fresh uniform random values (over some space depending on the context). We denote

$$\Pi : \langle \mathcal{P}(in_{\mathcal{P}}), \mathcal{V}(in_{\mathcal{V}}) \rangle \to \mathcal{P}(out_{\mathcal{P}}), \mathcal{V}(out_{\mathcal{V}})$$

for running a protocol Π with $in_{\mathcal{P}}$ and $in_{\mathcal{V}}$ as input of $\mathcal{P}$ and $\mathcal{V}$ respectively, getting $out_{\mathcal{P}}$ and $out_{\mathcal{V}}$ as output of $\mathcal{P}$ and $\mathcal{V}$ respectively. In addition, we consider protocols in the ROM, where the prover and verifier have access to a random oracle $\mathcal{H}$. We shall denote $\mathcal{P}^{\mathcal{H}}$ and $\mathcal{V}^{\mathcal{H}}$ when we want to stress that the parties have access to $\mathcal{H}$.

Definition 1 (Commit-and-Prove System). *Let $\mathcal{W}$ be a witness space, $\mathcal{X}$ be a statement space, and $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{X}$ be a relation. A* commit-and-prove *(CP) system in the ROM for $\mathcal{R}$ is a pair of interactive protocols $(\mathsf{Commit}^{\mathcal{H}}, \mathsf{Prove}^{\mathcal{H}})$ defined as follows:*

- $\mathsf{Commit}^{\mathcal{H}} : \langle \mathcal{P}(w), \mathcal{V}() \rangle \to \mathcal{P}(\mathsf{key}), \mathcal{V}(\mathsf{com})$ *is a protocol where $\mathcal{P}$ takes as input a witness $w \in \mathcal{W}$ and $\mathcal{V}$ takes no input. The output of $\mathcal{P}$ is a commitment key* key *and the output of $\mathcal{V}$ is a commitment* com *to w.*
- $\mathsf{Prove}^{\mathcal{H}} : \langle \mathcal{P}(\mathsf{key}, x), \mathcal{V}(\mathsf{com}, x) \rangle \to \mathcal{V}(out_{\mathcal{V}})$ *is a protocol where $\mathcal{P}$ takes as input a commitment key* key *and statement $x \in \mathcal{X}$ and $\mathcal{V}$ takes as input a commitment* com *and statement $x \in \mathcal{X}$. The output of $\mathcal{V}$ is $out_{\mathcal{V}} \in \{\textsc{Accept}, \textsc{Reject}\}$.*

The CP system is complete*, meaning that for every $(w, x) \in \mathcal{R}$,*

$$\Pr\left[out_{\mathcal{V}} = \textsc{Accept} \;\middle|\; \begin{array}{l} \mathsf{Commit}^{\mathcal{H}} : \langle \mathcal{P}(w), \mathcal{V}() \rangle \to \mathcal{P}(\mathsf{key}), \mathcal{V}(\mathsf{com}) \\ \mathsf{Prove}^{\mathcal{H}} : \langle \mathcal{P}(\mathsf{key}, x), \mathcal{V}(\mathsf{com}, x) \rangle \to \mathcal{V}(out_{\mathcal{V}}) \end{array} \right] = 1 \ ,$$

where the probability is taken over the random oracle $\mathcal{H}$ and the randomness of $\mathcal{P}$ and $\mathcal{V}$.

Let us now formally define several flavors of commitment schemes that we shall use in this work as special cases of commit-and-prove systems for specific relations.

Definition 2 (Polynomial Commitment Scheme). *Let $\mathbb{F}$ a finite field and $\mathbb{K}$ a finite extension of $\mathbb{F}$. Let n and d be positive integers. A* polynomial commitment scheme *(PCS) with parameters $(n, d, \mathbb{F}, \mathbb{K})$ is a CP system for the relation*

$$\mathcal{R} = \{(p, (r, v)) : p(r) = v\} \subseteq \left(\mathbb{F}[X]^{\leq d}\right)^n \times (\mathbb{K} \times \mathbb{K}^n) \ .$$

A small-domain PCS *with parameters $(n, d, \mathbb{E}, \mathbb{F})$ is a PCS for which the evaluation points r in the above relation are restricted to a small subset of $\mathbb{E} \subseteq \mathbb{F}$.*

Definition 3 (Multilinear Polynomial Commitment Scheme). *Let $\mathbb{F}$ a finite field and $\mathbb{K}$ a finite extension of $\mathbb{F}$. Let ω be a positive integer and $\ell = \lceil \log_2(\omega) \rceil$. A* multilinear polynomial commitment scheme *(MPCS) is a CP system for the relation*

$$\mathcal{R} = \{(w, (r, v)) : \tilde{w}(r) = v\} \subseteq \mathbb{F}^\omega \times (\mathbb{K}^\ell \times \mathbb{K})$$

where $\tilde{w}$ is the multilinear extension of w.

Definition 4 (Linear-Map Vector Commitment Scheme). *Let $\mathbb{F}$ a finite field and $\mathbb{K}$ a finite extension of $\mathbb{F}$. Let n, m be positive integers. A* linear-map vector commitment scheme *(LVCS) with parameters $(n, m, \mathbb{F}, \mathbb{K})$ is a CP system for the relation*

$$\mathcal{R} = \{(W, (c, v)) : c^T \cdot W = v\} \subseteq \mathbb{F}^{n \times m} \times (\mathbb{K}^n \times \mathbb{K}^m) \ .$$

Definition 5 (Linear Commitment Scheme). *Let $\mathbb{F}$ a finite field and $\mathbb{K}$ a finite extension of $\mathbb{F}$. Let ω be a positive integer. A* linear commitment scheme *(LCS) with parameters $(\omega, \mathbb{F}, \mathbb{K})$ is a CP system for the relation*

$$\mathcal{R} = \{(w, (a, b)) : \langle a, w \rangle = b\} \subseteq \mathbb{F}^\omega \times (\mathbb{K}^\omega \times \mathbb{K}) \ .$$

Definition 6 (Functional Commitment Scheme). *Let $\mathbb{F}$ a finite field and $\mathbb{K}$ a finite extension of $\mathbb{F}$. Let ω and d be positive integers. A* functional commitment scheme *(FCS) with parameters $(\omega, d, \mathbb{F}, \mathbb{K})$ is a CP system for the relation*

$$\mathcal{R} = \{(w, (f, v)) : f(w) = v\} \subseteq \mathbb{F}^\omega \times (\mathcal{F}_d \times \mathbb{K})$$

where $\mathcal{F}_d$ is the set of all functions $f : \mathbb{K}^\omega \to \mathbb{K}$ that can be represented as a polynomial from $\mathbb{K}[X_1, \ldots, X_\omega]^{\leq d}$.

We have the following relations between the above defined commitment schemes:

$$\left.\begin{array}{r} \mathrm{PCS}^{(n=1)} \subseteq \mathrm{MPCS} \\ \mathrm{PCS}^{(n\geq 1)} \\ \mathrm{LVCS} \end{array}\right\} \subseteq \mathrm{LCS} = \mathrm{FCS}^{(d=1)} \subseteq \mathrm{FCS}^{(d\geq 1)} \ .$$

where XCS $\subseteq$ YCS means that XCS can be naturally derived from YCS. Indeed, an LCS can be obtained from an FCS of degree $d \leq 1$ since constraints supported by an LCS are a subset of those supported by an FCS. Then, evaluation constraints arising in a PCS, MPCS or LVCS can all be expressed as linear constraints supported by an LCS. Finally, a PCS (for $n = 1$) with degree d can be emulated by an MPCS with $\ell = \lceil \log_2(d + 1) \rceil$ variables (i.e. $\omega = d + 1$) by encoding the coefficients of the committed polynomial as the evaluations of a multilinear polynomial at the Boolean hypercube $\{0, 1\}^\ell$. Then, for every $p \in \mathbb{F}[X]^{\leq d}$ there exists $w \in \mathbb{F}^\omega$ such that $p(r) = \tilde{w}(1, r, r^2, \ldots, r^{2^\ell})$ for every $r \in \mathbb{K}$.

For the sake of simplicity, we define the above CP systems for single-query relations (where a single evaluation can be proved). They can all be easily extended to multi-query relations (where multiple evaluations can be proved).

In the following, we define the security properties required for our purposes: *honest-verifier zero-knowledge* and *round-by-round knowledge soundness*. The former ensures that an honest execution of the CP system reveals nothing about the witness w beyond the truth of the statement x. The latter is a strengthening of the standard knowledge soundness which controls the soundness error round-by-round and admits an explicit witness extractor.

Honest-Verifier Zero-Knowledge. The zero-knowledge property is formalized in the programmable random oracle model: a simulator generates a transcript and a list of query-answer pairs that define the random oracle on those points, and indistinguishability is required against a (possibly unbounded) distinguisher with oracle access to $\mathcal{H}$.

Definition 7 (Honest-Verifier Zero-Knowledge). *A CP system $\Pi = (\mathsf{Commit}^{\mathcal{H}}, \mathsf{Prove}^{\mathcal{H}})$ for a relation $\mathcal{R}$ is* (q, ε)-honest-verifier zero-knowledge *(HVZK) if there exists a probabilistic polynomial-time simulator* Sim *such that, for every $(w, x) \in \mathcal{R}$, and every adversary $\mathcal{A}^{\mathcal{H}}$ making at most q queries to $\mathcal{H}$, the advantage of $\mathcal{A}$ in distinguishing the following two distributions is at most ε:*

- ***Real:*** $\{(\mathsf{com}, \pi_{\mathsf{com}}, \pi_{\mathsf{prov}}, \mathcal{H})\}$*, sampled by drawing $\mathcal{H}$ uniformly at random and running*

$$\mathsf{Commit}^{\mathcal{H}} : \langle \mathcal{P}(w), \mathcal{V}() \rangle \to \mathcal{P}(\mathsf{key}), \mathcal{V}(\mathsf{com})$$
$$\mathsf{Prove}^{\mathcal{H}} : \langle \mathcal{P}(\mathsf{key}, x), \mathcal{V}(\mathsf{com}, x) \rangle \to \mathcal{V}(\textsc{Accept})$$

  where π_{com} and π_{prov} are the transcripts of the two protocols;
- ***Simulated:*** $\{(\mathsf{com}, \pi_{\mathsf{com}}, \pi_{\mathsf{prov}}, \mathcal{H})\}$ *generated by* Sim *on input x.* Sim *programs $\mathcal{H}$ on points that lie in the simulated transcript* $(\mathsf{com}, \pi_{\mathsf{com}}, \pi_{\mathsf{prov}})$*; on all other points, $\mathcal{H}$ is sampled uniformly at random.*

Round-by-Round Knowledge Soundness. Round-by-round (RBR) soundness, introduced by Canetti, Chen, Holmgren, Lombardi, Rothblum, Rothblum and Wichs [CCH+19], is a strengthening of standard soundness in which a soundness error is associated to each verifier message rather than to the protocol as a whole – which is what makes it well suited to the Fiat–Shamir transform of multi-round protocols. The protocol admits a *state function* that classifies every partial transcript as either "*alive*" or "*doomed*": for a false statement the initial state is doomed, the prover alone cannot escape a doomed state, a complete doomed transcript is rejected, and the probability of moving from a doomed to an alive state across a single verifier message is bounded by a small per-round error. We follow the *knowledge* variant of this notion, introduced by Chiesa, Manohar and Spooner [CMS19] in the interactive oracle proof (IOP) setting, in which a polynomial-time extractor must recover a valid witness from any doomed transcript that the prover has a noticeable chance of bringing back to life; the per-round error then bounds the probability, over the next verifier message, that the state becomes alive *while* extraction fails.

We adapt this notion to commit-and-prove systems in the ROM. Adapting the IOP setting to the ROM setting roughly consists in replacing oracle access to the proof string by the prover's queries to $\mathcal{H}$. Accordingly, the state function and the extractor take as input the transcript *extended* with these queries and their answers, which is exactly what the prover knows about $\mathcal{H}$ at a given point of the execution. This has a subtlety with no counterpart in the IOP model: the prover's hash queries can, by themselves, change the state of a transcript – typically when an answer of the oracle happens to collide with a hash value sent earlier – so that, besides the per-round error $\varepsilon_{\mathsf{rbr}}$, the notion comes with a second error $\varepsilon_{\mathcal{H}}$ bounding the probability of such bad events over a whole execution of a query-bounded prover.

Adapting the notion to commit-and-prove systems is less immediate: the statement x is not known at the time the commitment is produced, but only when the prove protocol starts, so that the initial state of the execution cannot be anchored to the truth of x as for interactive proofs or IOPs. We therefore include the statement in the transcript, between the two protocols, and move that anchoring to the point where it becomes known. During the commit phase, the state does not tell whether the prover can still convince the verifier – there is no statement yet – but whether it is on its way to being bound to a single committed value, which calls for three states rather than two, as described below. Then the arrival of the statement turns the state doomed unless the committed value – as recovered by an extractor depending on the commit phase only – is a valid witness for x. From that point on, one is back to the standard setting. We discuss these choices further below. As usual, we stress that the state function is merely required to exist and need not be efficiently computable; only the extractor is required to be efficient.

Formally, a (partial) transcript of a CP system $(\mathsf{Commit}^{\mathcal{H}}, \mathsf{Prove}^{\mathcal{H}})$ is either a (possibly partial) transcript π_{com} of the commit protocol or a string of the form

$$\pi \;=\; \pi_{\mathsf{com}} \,\|\, x \,\|\, \pi_{\mathsf{prov}} \;,$$

where π_{com} is a complete commit transcript, $x \in \mathcal{X}$ is the statement to be proven and π_{prov} is a (possibly empty or partial) transcript of the prove protocol. An *extended transcript* θ is a transcript in which

the hash queries made by the prover (*i.e.* its queries to $\mathcal{H}$), together with their answers, are inserted in chronological order at the points where they were made; its *message part* is the transcript obtained by removing them. In the rest of this section, θ always denotes an extended transcript, while π denotes a plain one. We write $\theta\|m$ (resp. $\theta\|r$, $\theta\|(h,a)$) for the extended transcript obtained by appending a prover message m (resp. a verifier message r, a hash query h with its answer a).

Extended transcripts. The state function and the extractor take as input an extended transcript θ, which records, besides the messages exchanged, the hash queries made by the prover up to the point where θ stops: this is exactly what the prover knows about $\mathcal{H}$ at that point, together with the order in which it learned it relative to the messages it sent. The prover may make hash queries at any time, in particular between its last message and the verifier's decision. In the soundness experiment below, the statement x is sent by the prover after the commit phase, and it is treated as a prover message; we also assume, without loss of generality, that the hash queries involved in the checks of the verifier are made by the prover. An extended transcript θ is a *commit transcript* if its message part is a (possibly partial or complete) transcript of the commit protocol, and a *prove transcript* if its message part is of the form $\pi_{\mathsf{com}}\|x\|\pi_{\mathsf{prov}}$. A prove transcript thus reads

$$\theta \;=\; \theta_{\mathsf{com}} \,\|\, x \,\|\, \theta_{\mathsf{prov}} \;,$$

where θ_{com}, the longest prefix of θ which is a commit transcript, is a complete commit transcript ending just before x (it thus includes the hash queries made before x is sent), and where θ_{prov}, the possibly empty or incomplete remainder, records the messages of the prove protocol and the hash queries made since. The extractor is run on a complete commit transcript, possibly followed by hash queries, at whatever point of the execution.

The random oracle may, by chance, answer a hash query of the prover in a way that is inconsistent with what the prover has done so far – typically, an answer colliding with an earlier one, or with a hash value that the prover sent before querying any preimage of it. We let the state function give up on such transcripts by mapping them to a special state $\perp$, and we bound their probability separately. An extended transcript is *bad* if its state is $\perp$ and *good* otherwise; the conditions below only concern good transcripts, and unless stated otherwise they quantify over all of them, whether or not they can arise in an execution.

States. During the commit phase, a partial transcript is in one of three states: *legit* when the prover has, so far, behaved in a way that will leave it bound to a single committed value; *trashed* when it has produced a commitment that the verifier will never accept an opening of; and *cheating* when the prover has gained the opportunity to later open the commitment inconsistently. The states *alive* and *doomed* are only used once the statement is known, and the state $\perp$ is used in both phases, for the transcripts on which the state function gives up.

Definition 8 (State Function). *Let* $(\mathsf{Commit}^{\mathcal{H}}, \mathsf{Prove}^{\mathcal{H}})$ *be a CP system for a relation* $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{X}$. *A* state function *for this CP system is a function* State *mapping any extended transcript* θ *to a state* $\mathsf{State}(\theta)$*, which belongs to* $\{\mathsf{legit}, \mathsf{trashed}, \mathsf{cheating}, \perp\}$ *when* θ *is a (possibly partial or empty) commit transcript, and to* $\{\mathsf{alive}, \mathsf{doomed}, \perp\}$ *when* θ *is a prove transcript, such that:*

1. Initial state. *The empty transcript is legit:* $\mathsf{State}(\emptyset) = \mathsf{legit}$.
2. Prover messages cannot cheat. *For every good transcript* $\theta\|m$ *made of a partial commit transcript* θ *and a prover message* m*, if* $\mathsf{State}(\theta) \neq \mathsf{cheating}$ *then* $\mathsf{State}(\theta\|m) \neq \mathsf{cheating}$.
3. Statement arrival. *For every good transcript* $\theta\|x$ *made of a complete commit transcript* θ *and a statement* $x \in \mathcal{X}$*, it holds that* $\mathsf{State}(\theta\|x) = \mathsf{doomed}$ *whenever* $\mathsf{State}(\theta) = \mathsf{trashed}$*, and* $\mathsf{State}(\theta\|x) = \mathsf{alive}$ *whenever* $\mathsf{State}(\theta) = \mathsf{cheating}$. *(A legit commit transcript may become either alive or doomed, depending on* x*.)*
4. Prover messages preserve doom. *For every good transcript* $\theta\|m$ *made of a prove transcript* θ *and a prover message* m*, if* $\mathsf{State}(\theta) = \mathsf{doomed}$ *then* $\mathsf{State}(\theta\|m) = \mathsf{doomed}$.
5. Doomed full transcripts are rejected. *For every good transcript* θ *whose message part is a full transcript and which contains all the hash queries made by* $\mathcal{V}$ *when verifying it, so that the output of* $\mathcal{V}$ *is determined by* θ*, this output is* Reject *whenever* $\mathsf{State}(\theta) = \mathsf{doomed}$.

6. Bad transcripts and hash queries. *If* $\mathsf{State}(\theta) = \bot$ *then* $\mathsf{State}(\theta') = \bot$ *for every extension* θ' *of* θ*; and for every transcript* θ *and hash query* (h, a)*,* $\mathsf{State}(\theta\|(h,a)) \in \{\mathsf{State}(\theta), \bot\}$*: a hash query may make a transcript bad, but otherwise does not change its state.*

Definition 9 (Round-by-Round Knowledge Soundness). *A CP system* $(\mathsf{Commit}^{\mathcal{H}}, \mathsf{Prove}^{\mathcal{H}})$ *for a relation* $\mathcal{R}$ *has* $(q, \varepsilon_{\mathcal{H}}, \varepsilon_{\mathsf{rbr}})$-round-by-round knowledge soundness *if there exist a state function* State *(as per Definition 8) and a probabilistic polynomial-time* commitment extractor $\mathsf{Ext}_{\mathsf{com}}$*, which on input a complete commit transcript* θ *(possibly followed by hash queries) returns an element of* $\mathcal{W} \cup \{\bot\}$*, with* $\bot \notin \mathcal{W}$ *being a witness for no statement, and which is* stable*:* $\mathsf{Ext}_{\mathsf{com}}(\theta\|(h,a)) = \mathsf{Ext}_{\mathsf{com}}(\theta)$ *for every good* $\theta\|(h,a)$ *with* (h,a) *a hash query (the extractor being run with the same coins on both inputs); such that the four following conditions hold.*

1. Commit phase. *For every good partial commit transcript* θ *which is to be followed by a verifier message* r *and such that* $\mathsf{State}(\theta) \neq \mathsf{cheating}$*,*

$$\Pr_r \Big[\mathsf{State}(\theta\|r) = \mathsf{cheating} \Big] \leq \varepsilon_{\mathit{rbr}} ;$$

2. Statement arrival. *For every good transcript* $\theta\|x$ *made of a complete commit transcript* θ *with* $\mathsf{State}(\theta) = \mathsf{legit}$ *and a statement* $x \in \mathcal{X}$*,*

$$\mathsf{State}(\theta\|x) = \mathsf{alive} \iff \big(\mathsf{Ext}_{\mathsf{com}}(\theta),\, x\big) \in \mathcal{R} .$$

3. Prove phase. *For every good prove transcript* $\theta = \theta_{\mathsf{com}}\|x\|\theta_{\mathsf{prov}}$ *which is to be followed by a verifier message* r *and such that* $\mathsf{State}(\theta) = \mathsf{doomed}$*,*

$$\Pr_r \Big[\mathsf{State}(\theta\|r) = \mathsf{alive} \ \wedge \ \Big(\big(\mathsf{Ext}_{\mathsf{com}}(\theta_{\mathsf{com}}),\, x\big) \notin \mathcal{R} \ \vee \ \mathsf{State}(\theta_{\mathsf{com}}) \neq \mathsf{legit} \Big) \Big] \leq \varepsilon_{\mathit{rbr}} ;$$

i.e. the prover escapes doom while no valid witness has been extracted from the commit phase, the extractor's output on a trashed or cheating commit transcript being irrelevant;

4. Bad transcripts. *For every (computationally unbounded) malicious prover* $\tilde{\mathcal{P}}$ *making at most* q *queries to* $\mathcal{H}$*, consider an execution of* $\mathsf{Commit}^{\mathcal{H}}$ *followed by* $\mathsf{Prove}^{\mathcal{H}}$ *in which* $\tilde{\mathcal{P}}$ *chooses the statement* x *and also chooses the verifier's messages, and makes its hash queries at any time, and let* θ *be the final extended transcript of this execution (which includes the hash queries of the verifier). Then*

$$\Pr_{\mathcal{H}} \big[\mathsf{State}(\theta) = \bot \big] \leq \varepsilon_{\mathcal{H}} ,$$

where the probability is over $\mathcal{H}$ *and the coins of* $\tilde{\mathcal{P}}$*. By condition 6 of Definition 8, this bounds the probability that any transcript reached along the execution is bad.*

Figure 3 illustrates the above definitions by summarizing the allowed state transitions with their associated conditions and probabilities. Let us further comment on these definitions.

During the commit phase, no statement is defined, so that the state cannot be anchored to the truth of a statement; it is anchored instead to the binding of the commitment. A legit transcript is one along which the prover is on its way to being bound to a single committed value – the one the extractor recovers. A trashed transcript is one along which the prover has produced a commitment that it will never be able to open, *e.g.* by sending a hash value of which it has queried no preimage: hash-based schemes cannot prevent this behavior, but it is harmless, since such a transcript is doomed from the arrival of the statement on, whatever the statement, and hence always rejected (unless a later hash query happens to return the sent hash value, which makes the transcript bad). A cheating transcript is one along which the prover has gained the ability to open its commitment inconsistently. Looking ahead, for the DECS of Figure 4, the prover enters the cheating state when the committed evaluations are not consistent with a vector polynomial of degree at most d and the challenge Γ is nevertheless such that the degree-enforcing test can be passed; bounding the probability of this event over Γ is the heart of the analysis of the DECS.

The initial state is legit and, by condition 2 of Definition 8, a prover message never makes the prover enter the cheating state: this can only result from a verifier message, with probability at most $\varepsilon_{\mathsf{rbr}}$ by the commit-phase condition of Definition 9, or from lucky oracle answers, which make the transcript bad

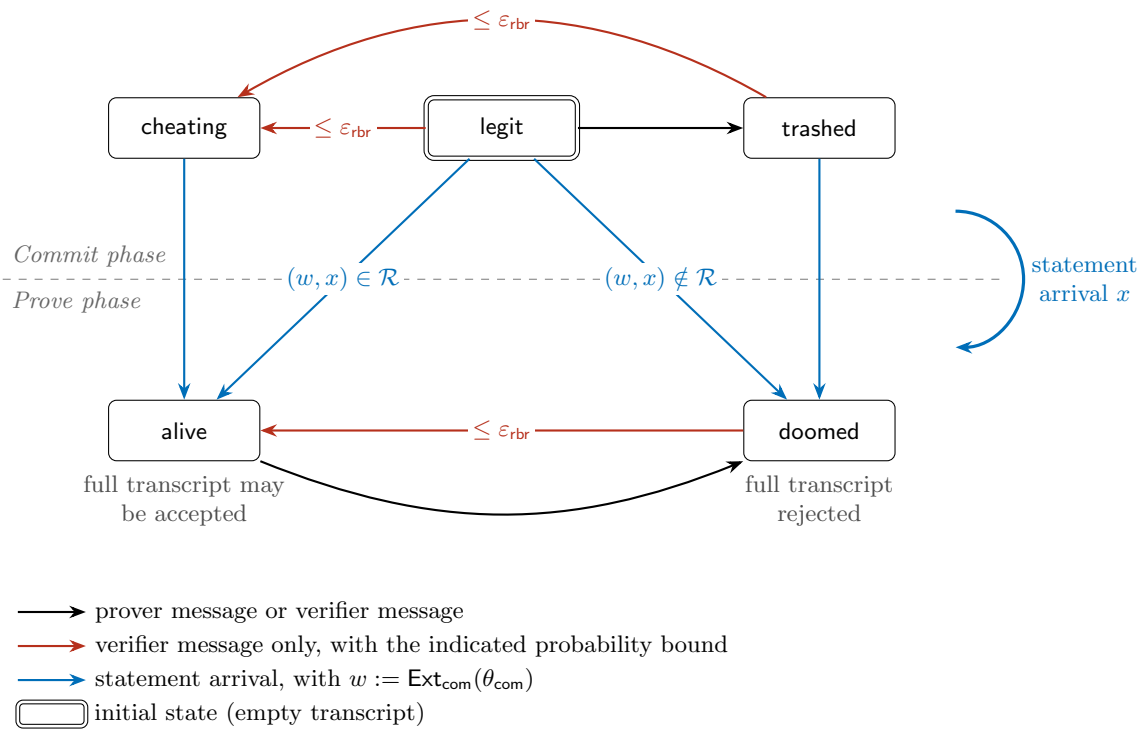


Fig. 3: Transitions allowed by Definitions 8 and 9. A verifier message enters cheating or leaves doomed with probability at most $\varepsilon_{\mathsf{rbr}}$ (in the prove phase, only when the extracted witness is not valid for x); a prover message never does, and neither does the answer to a hash query, unless it makes the transcript bad, which has probability at most $\varepsilon_{\mathcal{H}}$ over a whole execution. The transitions to the state $\perp$ (from any state, on a hash query), the self-loops, as well as the transitions that the definitions allow but that play no role (leaving cheating, and trashed to legit), are omitted; the double border marks the initial state.

and whose probability $\varepsilon_{\mathcal{H}}$ bounds. Once the prover is cheating, it may be able to open its commitment to any value, so that nothing can be asked of the extractor: Definition 9 accordingly requires nothing of it on a cheating commit transcript, and the arrival of the statement turns a cheating transcript alive whatever the statement. The commit-phase error thus plays, for the entrance into the cheating state, the role that the prove-phase error plays for doomed-to-alive transitions: both bound the probability that a verifier message gives the prover a chance to make the verifier accept while no valid witness is extracted.

The role played by the initial state in the standard interactive-proof setting, where a false statement is doomed from the outset [CCH+19], is here played by the arrival of the statement: for a legit commit transcript, aliveness then means exactly that the extracted value is a witness for x, which is what prevents the notion from being vacuous. From that point on, one is back to the standard setting: prover messages preserve doom, and the probability that a verifier message revives a doomed state while extraction fails is bounded by $\varepsilon_{\mathsf{rbr}}$.

The state $\perp$ and the stability conditions deserve a few more words. Conditions 1 to 3 of Definition 9, as well as the conditions of Definition 8, are required to hold for every good transcript, whether or not it can be produced by a query-bounded prover; condition 4 is the only one that refers to an execution. Conditions 2 and 4 of Definition 8 reflect the fact that a prover message carries no randomness, and condition 6 that, as long as the transcript is good, neither does an oracle answer: the only randomness that can change the state for the better (for the cheating prover) is that of the verifier's messages, which conditions 1 and 3 bound. The oracle's answers may of course be lucky – *e.g.* an answer colliding with an earlier one, or with a hash value that the prover sent before querying any preimage of it – and this is what the state $\perp$ is for: a scheme is free to give up on such transcripts, which exempts them from all the conditions, at the price of bounding their probability by $\varepsilon_{\mathcal{H}}$ in condition 4. Letting $\tilde{\mathcal{P}}$ choose the verifier's messages makes condition 4 stronger than its counterpart with uniformly random messages, which is what the Fiat–Shamir transform requires.

Finally, the extractor is given neither the statement nor the transcript of the prove phase: the extracted witness is entirely determined by the commit phase. This property, which hash-based commitment schemes achieve through straight-line extraction from the prover's oracle queries [FR25, FR26], is what makes such CP systems usable when the statement is only determined after the commitment has been sent – as in the compilers of Section 4.1, where the statement proven by the commitment scheme results from the execution of a sum-check-based protocol – since a single extracted witness then serves as the reference value for the whole execution.

2.4 Degree-Enforcing Commitment Scheme

The *degree-enforcing commitment scheme* (DECS) is a zero-knowledge small-domain PCS with parameters $(n, d, \mathbb{F}, \mathbb{E})$. Originally introduced in the Threshold-Computation-in-the-Head (TCitH) framework [FR25] as a variant of Ligero [AHIV17, AHIV23], this scheme commits to a (vector) polynomial $\boldsymbol{P} \in \left(\mathbb{F}[X]^{(\leq d)}\right)^n$ through a Merkle tree built over hash commitments of its evaluations on a public domain $\mathbb{E} \subseteq \mathbb{F}$, while *enforcing* – rather than merely testing – that the committed evaluations are consistent with a polynomial of degree at most d. To this end, the commitment phase is completed by a degree-enforcing test in which the prover reveals a random linear combination of $\boldsymbol{P}$ with random masking polynomials of degree at most d, while the verifier checks that the revealed combination is indeed a polynomial of degree d. In contrast with low-degree (proximity) tests such as those underlying Ligero [AHIV17, AHIV23] or FRI [BBHR18a], which only guarantee proximity to a low-degree polynomial, the DECS provides a strict degree guarantee on the extracted polynomial. The original formalization [FR25] was subsequently refined in the SmallWood paper [FR26], which recasts the DECS as a small-domain PCS achieving extractable knowledge soundness and honest verifier zero-knowledge in the ROM. We restate its security below in terms of the round-by-round knowledge soundness notion of Definition 9.

The DECS is recalled on Figure 4. It relies on an internal parameters η which is the number of linear combinations used in the degree-enforcing test (a.k.a. the dimension of the degree-enforcing polynomial $\boldsymbol{R}$). We further denote $N = |\mathbb{E}|$ the number of Merkle leaves and λ the security parameter. The DECS relies on a random oracle $\mathsf{Hash} : \{0,1\}^* \to \{0,1\}^{2\lambda}$ and a Merkle tree construction $\mathsf{MerkleRoot} : (\{0,1\}^{2\lambda})^N \to \{0,1\}^{2\lambda}$. The hash function is used for (i) committing to the evaluations of the polynomial $\boldsymbol{P}$, (ii) committing to the degree-enforcing polynomial $\boldsymbol{R}$, and (iii) as the compression function in the Merkle tree construction. We assume strict domain separation between these uses, leaving it implicit in what follows, along with the serialization of field elements into hash inputs.

Remark 2. Rather than transmitting $\boldsymbol{R}$ in plain in the commitment, the prover commits to it with a hash and later sends only the evaluations needed to reconstruct it: since the verifier recovers $\boldsymbol{R}(E)$ from $\boldsymbol{P}(E)$ and $\boldsymbol{M}(E)$, and $\deg \boldsymbol{R} \leq d$, it suffices to send the $d + 1 - |E|$ missing evaluations and check them against the hash. This optimization is mentioned in [FR26] but not described explicitly. It leaves the soundness and zero-knowledge guarantees unchanged.

Theorem 1 (DECS Security). *The DECS in Figure 4 is $(q, \varepsilon_{\mathrm{ZK}})$-HVZK and has $(q, \varepsilon_{\mathcal{H}}, \varepsilon_{\mathrm{RBR}})$-round-by-round knowledge soundness where*

$$\varepsilon_{\mathrm{ZK}} = \frac{q}{2^{\lambda}} \ , \quad \varepsilon_{\mathcal{H}} = \frac{3q(q+1)}{2^{2\lambda+1}} \ , \quad \textit{and} \quad \varepsilon_{\mathrm{RBR}} = \binom{N}{d+2} \cdot \frac{1}{|\mathbb{F}|^{\eta}}$$

for any positive integer q.

The proof is given in Appendix D.1; it adapts the extractable knowledge soundness proof of [FR26] to Definition 9.

3 Zero-Knowledge Proof Systems for Sum-Check-Based Protocols

This section describes our general approach to build zero-knowledge proof systems from sum-check-based protocols. As in [Fas26], our approach follows the general methodology of [BGG+90] which is to wrap an interactive proof protocol (in our case, a sum-check-based interactive protocol) into a non-interactive zero-knowledge argument system using zero-knowledge commit-and-prove systems. To obtain a general

DECS.Commit:

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

- $\mathcal{P}$ samples $\boldsymbol{M} = (M_1, \ldots, M_\eta) \leftarrow (\mathbb{F}[X]^{(\leq d)})^\eta$ and $(\rho_1, \ldots, \rho_N) \leftarrow (\{0,1\}^\lambda)^N$.
- $\mathcal{P}$ computes $u_j = \mathsf{Hash}(\boldsymbol{P}(e_j), \boldsymbol{M}(e_j), j, \rho_j)$ for all $e_j \in \mathbb{E}$ and $\mathsf{root} = \mathsf{MerkleRoot}(u_1, \ldots, u_N)$. $\mathcal{P}$ sends root to $\mathcal{V}$.
- $\mathcal{V}$ samples a matrix $\Gamma = (\gamma_{k,i})_{k,i} \leftarrow \mathbb{F}^{\eta \times n}$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ computes the polynomials

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

  and a hash commitment $h_R = \mathsf{Hash}(\boldsymbol{R})$ of the vector polynomial $\boldsymbol{R} := (R_1, \ldots R_\eta)$. $\mathcal{P}$ sends h_R to $\mathcal{V}$.
- At the end of the protocol, $\mathcal{V}$ holds the commitment $\mathsf{com} := (\mathsf{root}, \Gamma, h_R)$ and $\mathcal{P}$ holds the key $\mathsf{key} := (\boldsymbol{P}, \boldsymbol{M})$.

DECS.Prove:

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

- $\mathcal{P}$ sends the evaluations $\boldsymbol{M}(E)$, evaluations $\boldsymbol{R}(S)$ for a set $S \subseteq \mathbb{E} \setminus E$ of size $|S| = d+1-|E|$, 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.
- $\mathcal{V}$ computes $\boldsymbol{R}(e) = \boldsymbol{M}(e) + \Gamma \cdot \boldsymbol{P}(e)$ for all $e \in E$ and recover $\boldsymbol{R}$ from $\boldsymbol{R}(E \cup S)$ using polynomial interpolation (this $\boldsymbol{R}$ is of degree at most d by construction). $\mathcal{V}$ then 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.
  - Check that $h_R = \mathsf{Hash}(\boldsymbol{R})$.

Fig. 4: DECS from [FR25, FR26].

compiler for sum-check-based interactive proofs, we introduce a new constraint system, called the *sum-check verification constraint system* (SVCS), that captures the constraints that arise in the sum-check protocol, on the verifier's side.

The general methodology to build a zero-knowledge proof system for sum-check-based protocols is based on the following *commit, blindly interact, and prove* (CBIP) paradigm:

1. *Commit:* The prover commits to a witness $w \in \mathbb{F}^\omega$ and a random *trace mask* $t \in \mathbb{K}^\tau$ using a zero-knowledge commit-and-prove system and sends the commitment to the verifier.
2. *Blindly interact:* The prover and the verifier run a (sum-check-based) interactive protocol to prove that w satisfies some statement. In this protocol, each message $m_i \in \mathbb{K}$ from the prover is masked by an element t_i of the trace mask, namely the prover sends $m_i + t_i$ instead of m_i. At the end of this interactive protocol, all the checks performed by the verifier are reduced to an SVCS instance with witness (w, t).
3. *Prove:* Based on the commit-and-prove system, the prover shows that the committed witness (w, t) satisfies the SVCS instance defined by the interactive protocol.

In the following, we formally define the sum-check verification constraint system (SVCS) and explain how sum-check-based interactive protocols can be naturally reduced to SVCS instances. We further describe some techniques to simplify SVCS instances, which are useful for the compilers described in the next sections.

3.1 Sum-Check Verification Constraint System

We formally define the sum-check verification constraint system (SVCS) as follows:

Definition 10 (SVCS). *An SVCS instance* $\mathbb{x} = (\mathbb{F}, \mathbb{K}, \omega, \tau, d, \mathcal{E}, \mathcal{F})$ *is defined with respect to a finite field* $\mathbb{F}$*, an extension field* $\mathbb{K}$ *of* $\mathbb{F}$*, integers* $\omega, \tau, d, \ell \in \mathbb{N}$*, with* $\ell = \lceil \log_2 \omega \rceil$*, and two sets* $\mathcal{E} = \{(r_1, \psi_1), \ldots, (r_{n_\mathcal{E}}, \psi_{n_\mathcal{E}})\}$ *and* $\mathcal{F} = \{f_1, \ldots, f_{n_\mathcal{F}}\}$*, with* $r_i \in \mathbb{K}^\ell$ *and* $\psi_i : \mathbb{K}^\tau \to \mathbb{K}$ *a* $\mathbb{K}$*-affine map for every* $i \in [n_\mathcal{E}]$*, and* $f_i : \mathbb{K}^\tau \to \mathbb{K}$ *a polynomial function of total degree* d *for every* $i \in [n_\mathcal{F}]$*. The witness of an SVCS instance is a pair* $(w, t) \in \mathbb{F}^\omega \times \mathbb{K}^\tau$ *satisfying the following constraints:*

- Multilinear evaluation constraints: *For each* $(r, \psi) \in \mathcal{E}$*, we have* $\widetilde{w}(r) = \psi(t)$*, where* $\widetilde{w} : \mathbb{K}^\ell \to \mathbb{K}$ *is the multilinear extension of* w*.*
- Trace constraints: *For each* $f \in \mathcal{F}$*, we have* $f(t) = 0$*.*

We denote $\mathsf{Sat}_{\mathrm{SVCS}}(\mathbb{x}, (w, t)) = 1$ *to mean that the witness* (w, t) *satisfies the above constraints and* $\mathsf{Sat}_{\mathrm{SVCS}}(\mathbb{x}, (w, t)) = 0$ *otherwise.*

The two kinds of constraints of an SVCS instance are exactly those that arise, on the verifier's side, when a sum-check-based protocol is executed with masked prover messages: the multilinear evaluation constraints come from the reduction of a sum-check claim to a single evaluation of the summed polynomial, while the trace constraints account for the few remaining checks of the verifier, typically products of prover messages. This is made precise below, after having extended the definition to capture richer evaluation constraints.

Extended SVCS. We also consider an extended version of SVCS where the multilinear evaluation constraints are of the form $\widetilde{p}(r) = \psi(t)$ for some polynomial p defined as a linear combination of multilinear extensions of subsets of the witness coordinates. This extended version of SVCS is useful to capture more general sum-check-based protocols in the CBIP paradigm, where the claim to be proven is not directly on the multilinear extension of the witness but on some linear combination of such extensions. Specifically, the extended SVCS instance is defined as in Definition 10 except that the multilinear evaluation constraints are of the form $\widetilde{p}_i(r) = \psi(t)$ for a set $\{\widetilde{p}_i\}$ of multilinear extensions of vectors $p_i = A_i \cdot w + b_i$ defined from the witness w and some public matrices A_i and vectors b_i. An extended SVCS instance has the same parameters as a standard SVCS instance plus the set of matrices $\{A_i\}$ and vectors $\{b_i\}$, and its set $\mathcal{E}$ is of the form $\mathcal{E} = \{(r_i, \psi_i)\}$ to encode the constraints $\widetilde{p}_i(r_i) = \psi_i(t)$. We capture multiple evaluation constraints on the same MLE by possibly considering several copies of this polynomial among the $\widetilde{p}_i$'s (i.e. duplicates in the pairs (A_i, b_i)).

Extended SVCS instances typically capture instances arising with the GKR protocol that involves multilinear evaluation claims on the input MLE $\widetilde{x}$ and output MLE $\widetilde{y}$ which might be partly public and/or partly redundant (in case of flattening of the layered circuit – see e.g. Section 6). In this case, the witness is simply made of all the secret coordinates of x and y which are then mapped from w to $\widetilde{x}$ and $\widetilde{y}$ thanks to the A_i's and b_i's. In the next section, we show how to reduce an extended SVCS instance to a standard SVCS instance using the sum-check protocol.

From Interactive Proofs to SVCS Instances. Let us now describe, in full generality, how an interactive proof is turned into a protocol producing an SVCS instance. The transformation is purely syntactic: the protocol itself is left untouched, only the prover's messages are masked and the verifier's checks are re-expressed as constraints on the committed pair. We first formally define sum-check-based interactive proofs.

Definition 11 (Sum-Check-Based Interactive Proof). *Let $\mathcal{R}_{\text{IP}} \subseteq \mathbb{F}^\omega \times \mathbb{X}_{\text{IP}}$ be a relation. A* sum-check-based interactive proof *(SC-IP) for $\mathcal{R}_{\text{IP}}$ is a public-coin interactive protocol between a prover holding w and a verifier, both holding $\mathbb{x}$, in which the prover messages are vectors over $\mathbb{K}$ and at the end of which both parties compute, from $\mathbb{x}$ and the transcript π, a set of* residual checks *on the witness w and the prover messages $m = (m_1, \ldots, m_\nu) \in \mathbb{K}^\nu$, made of*

- evaluation checks $\widetilde{p}_i(r_i) = \varphi_i(m)$, *where $p_i = A_i \cdot w + b_i$ for public A_i, b_i, $r_i \in \mathbb{K}^{\ell_i}$, and $\varphi_i : \mathbb{K}^\nu \to \mathbb{K}$ is a public affine map;*
- constraint checks $f_j(m) = 0$, *where $f_j : \mathbb{K}^\nu \to \mathbb{K}$ is a public polynomial function of total degree at most d.*

We write $\mathsf{Chk}_{\mathbb{x},\pi}(w, m) = 1$ when all these checks hold, and require completeness: an honest execution on $(w, \mathbb{x}) \in \mathcal{R}_{\text{IP}}$ yields $\mathsf{Chk}_{\mathbb{x},\pi}(w, m) = 1$.

Consider an SC-IP as per Definition 11, whose prover messages amount to $m = (m_1, \ldots, m_\nu) \in \mathbb{K}^\nu$ field elements and whose residual checks, computed by both parties from the instance and the transcript, are the evaluation checks $\widetilde{p}_i(r_i) = \varphi_i(m)$ and the constraint checks $f_j(m) = 0$. In the CBIP paradigm, the prover has committed to a trace mask $t \in \mathbb{K}^\tau$ with $\tau \geq \nu$ and sends $\bar{m}_i = m_i + t_i$ instead of m_i, so that the verifier learns nothing from the transcript but can no longer evaluate its checks. It can however *express* them on the committed pair: substituting $m = \bar{m} - t$, the residual checks become

$$\widetilde{p}_i(r_i) = \psi_i(t) \quad \text{with} \quad \psi_i(t) := \varphi_i(\bar{m} - t)\ , \qquad\qquad f'_j(t) = 0 \quad \text{with} \quad f'_j(t) := f_j(\bar{m} - t)\ ,$$

which is precisely an (extended) SVCS instance $\mathbb{x}_{\text{SVCS}} = (\mathbb{F}, \mathbb{K}, \omega, \tau, d, \mathcal{E}, \mathcal{F})$ with $\mathcal{E} = \{(r_i, \psi_i)\}_i$ and $\mathcal{F} = \{f'_j\}_j$: the substitution being affine in t, the maps ψ_i are affine because the φ_i are, and the trace constraints f'_j have the same degree bound as the original checks f_j. The cost of the transformation is thus one trace element per transmitted field element, and nothing else.

Two features of this reduction deserve to be stressed. First, the split between the two kinds of constraints of an SVCS instance mirrors the split between the two kinds of verifier checks: those that are *affine* in the prover messages – typically, the reduction of a sum-check claim to the evaluation of the summed polynomial – give rise to the multilinear evaluation constraints, while the remaining ones, of degree d, give rise to the trace constraints. Second, since the masking is a bijection for a fixed t and leaves the verifier's messages untouched, the transformation preserves the round-by-round soundness of the original protocol, with the same error; this is stated and proven as Lemma 3 in Appendix D, and is what allows our compilers to be instantiated with *any* sound sum-check-based interactive proof, oblivious of the CBIP paradigm.

Examples. Let us illustrate this transformation on two protocols. The first one is a single sum-check call, the elementary building block of all the protocols we consider: it consumes $\ell \cdot d$ trace elements – one per transmitted field element – and produces *no* trace constraint at all, every check of the sum-check verifier being affine in the prover messages. The protocol is depicted in Figure 5. It is a step-by-step translation of the optimized sum-check protocol of Figure 20: the prover masks each of its $\ell \cdot d$ messages with a fresh coordinate of the trace mask, and the verifier, which can no longer evaluate the quantities it used to compute, computes instead the public affine maps of t that represent them – the claimed value c

becoming the map $\hat{c} = \psi$ and the final value c' the map $\hat{c}' = \psi'$. Every step carries over verbatim, since the operations the verifier performs on these quantities, namely the recovery of $g_i(0)$, the interpolation of g_i and its evaluation at r_i, are all $\mathbb{K}$-linear and therefore well defined on affine maps of t. In particular, the execution produces a single multilinear evaluation constraint $g(r) = \psi'(t)$ and no trace constraint at all.

As second example, we consider the GKR protocol, where trace constraints do appear: one per layer, of degree the degree of that layer, arising from the consistency check between the final value of the layer's sum-check and the claimed evaluations of the previous state. The description of the GKR protocol in the CBIP paradigm is provided in Appendix C.

3.2 Simplifying SVCS Instances

This section describes some techniques to simplify SVCS instances, which are useful for the compilers described in the next sections. On the one hand, we show how to reduce an extended SVCS instance to a standard one while merging multiple multilinear evaluation constraints into a single one. On the other hand, we show how to linearize the trace constraints, namely how to reduce them to degree-1 constraints of the form $f(t) = \psi(t)$ for some affine map $\psi : \mathbb{K}^\tau \to \mathbb{K}$. Looking ahead, this allows us to reduce the requirement to the commitment scheme involved in the CBIP paradigm to a multilinear polynomial commitment scheme (MPCS) and/or a linear commitment scheme (LCS) for which more efficient constructions can be obtained (as we describe in Section 4.2).

Merging Multilinear Evaluation Constraints. We describe hereafter a general technique to reduce an extended SVCS instance with multiple evaluation constraints into a standard SVCS instance with a single evaluation constraint. Namely, we show how to merge the evaluation claims $\widetilde{p}_i(r_i) = \psi_i(t)$ for $i = 1, \ldots, n$ into a single claim of the form $\widetilde{w}(r) = \psi'(t)$ for some random $r \in \mathbb{K}^\ell$ and some public affine map ψ'. First note that we can write $\widetilde{p}_i$ as

$$\widetilde{p}_i(X) = \sum_{y \in \{0,1\}^\ell} \left(\widetilde{A}_i(X, y) \cdot \widetilde{w}(y) + \widetilde{b}_i(X, y) \right) ,$$

where $\widetilde{A}_i$ is the MLE of the matrix A_i and $\widetilde{b}_i$ is defined such that $\widetilde{b}_i(X, 0)$ is the MLE of the vector b_i and $\widetilde{b}_i(X, y) = 0$ for all $y \in \{0,1\}^\ell \setminus \{0\}$. The n evaluation claims on the $\widetilde{p}_i$'s are reduced to a single claim on $\widetilde{w}$ by the following reduction in the CBIP paradigm:

- $\mathcal{V}$ chooses random field elements $\gamma_1, \ldots, \gamma_n$ and sends them to $\mathcal{P}$. The rest of the protocol aims to prove the claim $\sum_i \gamma_i \cdot \widetilde{p}_i(r_i) = \psi(t)$ where $\psi = \sum_i \gamma_i \cdot \psi_i$. This claim is equivalent to the initial claims on the $\widetilde{p}_i$'s with probability $1 - 1/|\mathbb{K}|$ over the choice of $\gamma_1, \ldots, \gamma_n$.
- $\mathcal{P}$ and $\mathcal{V}$ run a sum-check protocol on the claim

$$\sum_{y \in \{0,1\}^\ell} g(y) = \psi(t) , \quad \text{where } \; g(y) = \sum_{i=1}^{n} \gamma_i \cdot \left(\widetilde{A}_i(r_i, y) \cdot \widetilde{w}(y) + \widetilde{b}_i(r_i, y) \right) .$$

- At the end of the sum-check protocol, the claim is reduced to a single evaluation claim on $\widetilde{w}$ of the form $\widetilde{w}(r) = \psi'(t)$ for some random $r \in \mathbb{K}^\ell$ and some public affine map $\psi' : \mathbb{K}^\tau \to \mathbb{K}$ defined from ψ and the prover messages in the sum-check protocol.

The communication complexity of this reduction is 2ℓ field elements since it relies on a sum-check on a degree-2 polynomial with ℓ variables. In the CBIP paradigm, this means that the committed trace mask t must include 2ℓ extra elements.

Linearizing Trace Constraints. In most sum-check-based interactive protocols, the verifier checks besides the multilinear evaluations are merely a few multiplication checks between (linear combinations of) the prover messages. In the CBIP paradigm, these checks translate to trace constraints of the form $\psi_1(t) \cdot \psi_2(t) = \psi_3(t)$ for some affine maps $\psi_1, \psi_2, \psi_3 : \mathbb{K}^\tau \to \mathbb{K}$. Assume that we have m such multiplication

Context:

- $\mathcal{P}$ has previously committed to a witness $w \in \mathbb{F}^\omega$ and a trace mask $t \in \mathbb{K}^\tau$.
- $\mathcal{P}$ claims that w and t satisfy
$$\sum_{x \in \{0,1\}^\ell} g(x) = \psi(t) \ ,$$
where $g : \mathbb{K}^\ell \to \mathbb{K}$ is an ℓ-variate virtual polynomial with respect to $\widetilde{w}$ of total degree d and $\psi : \mathbb{K}^\tau \to \mathbb{K}$ is a public affine map.
- The trace mask t includes $\ell \cdot d \leq \tau$ elements labelled as $t_{i,j}$ for $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots, d\}$ in the following description.
- For any value $v \in \mathbb{K}$ computed by the verifier in the standard protocol (Figure 20), the verifier of the CBIP version can no longer compute v in plain, but only a public affine map $\hat{v} : \mathbb{K}^\tau \to \mathbb{K}$ which agrees with v when applied to the trace mask in the honest case (i.e., $\hat{v}(t) = v$). Accordingly, $\hat{g}_i(X_i, t)$ denotes a polynomial of degree at most d in X_i whose coefficients are affine in t, and which satisfies $\hat{g}_i(X_i, t) = g_i(X_i)$ in the honest case – the dependence on t then cancelling out. We further let $\hat{c}(t) := \psi(t)$.
- At the end of the protocol, the claim is reduced to checking $g(r) = \psi'(t)$, for $r = (r_1, \ldots, r_\ell) \in \mathbb{K}^\ell$ the random point chosen by the verifier and $\psi' : \mathbb{K}^\tau \to \mathbb{K}$ a public affine map depending on ψ, on r and on the prover messages.

Protocol:

The following steps are repeated for $i = 1, \ldots, \ell$:

- $\mathcal{P}$ sends the masked evaluations $m_{i,j} = g_i(j) + t_{i,j}$ for $j = 1, \ldots, d$ of the polynomial
$$g_i(X_i) = \sum_{x_{i+1}, \ldots, x_\ell} g(r_1, \ldots, r_{i-1}, X_i, x_{i+1}, \ldots, x_\ell) \ ,$$
and $\mathcal{P}$ and $\mathcal{V}$ let $\hat{g}_i(j, t) = m_{i,j} - t_{i,j}$ for $j = 1, \ldots, d$.
- $\mathcal{V}$ lets
$$\hat{g}_i(0, t) = -\hat{g}_i(1, t) + \begin{cases} \hat{c}(t) & \text{if } i = 1 \ , \\ \hat{g}_{i-1}(r_{i-1}, t) & \text{if } i > 1 \end{cases}$$
and retrieves $\hat{g}_i(X_i, t)$ from $\hat{g}_i(0, t), \ldots, \hat{g}_i(d, t)$ by interpolation in X_i.
- $\mathcal{V}$ samples a field element $r_i \in \mathbb{K}$ and sends it to $\mathcal{P}$.

End of the repetition.

- $\mathcal{V}$ computes $\hat{c}'(t) = \hat{g}_\ell(r_\ell, t)$.
- Let $\psi' := \hat{c}'$. Checking the initial claim is reduced to checking the multilinear evaluation constraint
$$g(r_1, r_2, \ldots, r_\ell) = \psi'(t) \ .$$

Fig. 5: Sum-check protocol in the CBIP paradigm.

checks and denote $\psi_1^{(1)}, \psi_2^{(1)}, \psi_3^{(1)}, \ldots, \psi_1^{(m)}, \psi_2^{(m)}, \psi_3^{(m)}$ the corresponding affine maps. We need to verify that for each $j \in \{1, \ldots, m\}$, we have

$$\psi_1^{(j)}(t) \cdot \psi_2^{(j)}(t) = \psi_3^{(j)}(t)$$

which is an R1CS instance with m constraints and τ variables, that the trace mask t must satisfy. In this instance, the number of constraints (m) is typically much lower than the number of variables (τ), which is in contrast to standard R1CS instances arising e.g. in arithmetic circuit satisfiability (where the number of constraints and variables are comparable). Using a technique inspired from linear PCPs [BBC+19], we can reduce these m multiplication constraints to a single *linear* constraint. The protocol requires $m+1$ extra elements in the trace mask t, which we denote as $t_1, \ldots, t_{m+1}$, and goes as follows:

- $\mathcal{P}$ defines polynomials $p_1, p_2, p_3 \in \mathbb{K}[X]$ by interpolation such that $p_i(j) = \psi_i^{(j)}(t)$ for $i \in [3]$ and $j \in [m]$ and $p_1(m+1) = t_{m+1}$. We have $\deg(p_1) \leq m$ and $\deg(p_2), \deg(p_3) \leq m-1$.
- $\mathcal{P}$ computes the polynomial $q \in \mathbb{K}[X]$ satisfying $q(X) \cdot \prod_{j=1}^{m}(X-j) = p_1(X) \cdot p_2(X) - p_3(X)$ and $\deg(q) \leq m-1$. The existence of q satisfying this identity implies that for each $j \in [m]$, we have $p_1(j) \cdot p_2(j) = p_3(j)$, which is equivalent to the initial multiplication constraints. The rest of the protocol aims to prove the existence of such a polynomial q in a zero-knowledge way.
- $\mathcal{P}$ sends $q'(X) = q(X) + \mu(X)$ to $\mathcal{V}$, where $\mu(X)$ is the polynomial defined by interpolation such that $\mu(j) = t_j$ for $j \in [m]$.
- $\mathcal{V}$ samples a random field element $r \leftarrow \mathbb{K} \setminus [m]$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ sends $v = p_1(r)$ to $\mathcal{V}$.
- At this stage, the polynomial identity $q(X) \cdot \prod_{j=1}^{m}(X-j) = p_1(X) \cdot p_2(X) - p_3(X)$ can be checked on the random point r by checking two linear constraints on the trace mask t, as explained hereafter.

Let $\psi_1, \psi_2, \psi_3 : \mathbb{K}^\tau \to \mathbb{K}$ the affine maps such that $\psi_i(t) = p_i(r)$ for $i \in [3]$ (which can be computed by interpolation from $\psi_i^{(1)}, \ldots, \psi_i^{(m)}$ and r) and $\psi_4 : \mathbb{K}^\tau \to \mathbb{K}$ the affine map such that $\psi_4(t) = q(r)$ (which can be computed by interpolation from q' and r). The two following linear trace constraints are included in the SVCS instance in place of the m multiplication constraints:

$$\psi_1(t) = v \quad \text{and } \psi'(t) = 0, \quad \text{where } \quad \psi'(t) := v \cdot \psi_2(t) - \psi_3(t) - \prod_{j=1}^{m}(r-j) \cdot \psi_4(t) \ .$$

The first constraint ensures that v is correctly defined as $p_1(r)$ and the second constraint ensures that $v \cdot p_2(r) - p_3(r) = q(r) \cdot \prod_{j=1}^{m}(r-j)$, which together implies the existence of a polynomial q of degree at most $m-1$ such that $q(X) \cdot \prod_{j=1}^{m}(X-j) = p_1(X) \cdot p_2(X) - p_3(X)$, with probability $1-(2m-1)/(|\mathbb{K}|-m)$ over the choice of r (by the Schwartz-Zippel lemma). This in turn implies that for each $j \in [m]$, we have $\psi_1^{(j)}(t) \cdot \psi_2^{(j)}(t) = \psi_3^{(j)}(t)$.

This way, by adding $m+1$ extra elements to the trace mask t and running the above protocol in the CBIP paradigm, one can drop the m multiplication constraints on t from the SVCS instance and replace them with two linear constraints. In particular, this allows us to reduce the requirement to the commitment scheme involved in the CBIP paradigm to a multilinear polynomial commitment scheme (MPCS) and/or a linear commitment scheme (LCS) for which more efficient constructions can be obtained (as we describe in Section 4.2).

Remark 3. We note that the above protocol has a communication complexity of $m+1$ field elements in $\mathbb{K}$ (m for the polynomial q' and one for the value v) and requires $m+1$ extra elements in the trace mask t. In our context, we do not focus on optimizing the communication complexity of this protocol since it is typically negligible compared to the communication complexity of the core sum-check-based protocol. However, we stress that those two $O(m)$ overheads can be reduced to $O(\sqrt{m})$ or $O(\log m)$ using techniques from [BMRS21, BDK+21b, DOT21].

4 Compact Zero-Knowledge Arguments by Sum-Checking SmallWood

This section describes our zero-knowledge arguments built by combining sum-check-based interactive proofs and SmallWood-style commitment schemes in the CBIP paradigm. We first describe two compilers to build zero-knowledge arguments from sum-check-based interactive proofs and commitment schemes, then we describe concrete SmallWood-style commitment schemes to instantiate these compilers and analyze the proof sizes of the resulting schemes.

4.1 Zero-Knowledge Arguments from Sum-Check-Based IP and Commitment Schemes

In this section, we describe two compilers to build zero-knowledge arguments from sum-check-based interactive proofs and commitment schemes. Both schemes follow the CBIP paradigm described in Section 3 where the execution of the sum-check-based protocol gives rise to an SVCS instance $\mathbb{x}_{\text{SVCS}}$ as defined in Definition 10. Thanks to the properties of the commitment scheme(s), one can then prove in zero-knowledge that (w, t) satisfies this instance $\mathbb{x}_{\text{SVCS}}$. The difference between the two compilers relies on the type of commitment scheme they use: the first one relies on a linear commitment scheme (LCS) while the second one relies on a combination of a LCS and a multilinear polynomial commitment scheme (MPCS). We describe these two compilers in more details hereafter.

Compiler 1. This first compiler is based on two ingredients:

- a sum-check-based interactive proof (SC-IP) for a relation $\mathcal{R}_{\text{IP}} = \{(w, \mathbb{x})\} \subseteq \mathbb{F}^{\omega} \times \mathbb{X}_{\text{IP}}$, which in the CBIP paradigm gives rise to an SVCS instance $\mathbb{x}_{\text{SVCS}} = (\mathbb{F}, \mathbb{K}, \omega, \tau, d, \mathcal{E}, \mathcal{F})$ where the trace constraints $\mathcal{F}$ can be expressed as m_{mult} multiplication constraints;
- a zero-knowledge linear commitment scheme (ZK-LCS) with parameters $(\omega + \tau', \mathbb{F}, \mathbb{K})$ (see Definition 5), where $\tau' = \tau + m_{\text{mult}} + 1$ is the trace size in the CBIP version of the SC-IP (including the trace constraint linearization).

From these two ingredients, we derive a zero-knowledge argument system for the relation $\mathcal{R}_{\text{IP}}$. The scheme is described in Figure 6. In this protocol, the linearized trace constraints in $\mathcal{F}'$ and the multilinear evaluation constraints in $\mathcal{E}$ are jointly considered as linear constraints on the pre-committed witness (w, t), which are directly proved using the LCS.

Remark 4. While the protocol is described assuming that the CBIP execution of the protocol results in a standard SVCS instance, it works as well if the resulting instance is a *extended* SVCS instance as defined in Section 3.1 where the multilinear evaluation constraints are richer. This comes from the fact that these richer constraints are still linear constraints on the pre-committed witness (w, t). In particular, the technique described in Section 3.2 to simplify and merge multilinear evaluation constraints is not required by the first compiler.

The security of the obtained scheme is stated in the following theorem. The proof is provided in Appendix D.

Theorem 2 (Security of Compiler 1). *The zero-knowledge argument system described in Figure 6 is* $(q, \varepsilon_{\text{ZK}})$*-HVZK and has* $(q, \varepsilon_{\mathcal{H}}, \varepsilon_{\text{RBR}})$*-round-by-round knowledge soundness where*

$$\varepsilon_{\text{ZK}} = \varepsilon_{\text{ZK}}^{\mathsf{lcs}} \ , \quad \varepsilon_{\mathcal{H}} \leq \varepsilon_{\mathcal{H}}^{\mathsf{lcs}} \quad \textit{and} \quad \varepsilon_{\text{RBR}} \leq \max\left(\varepsilon_{\text{RBR}}^{\mathsf{ip}}, \varepsilon_{\text{RBR}}^{\mathsf{lcs}}, \frac{2m}{|\mathbb{K}|}\right)$$

for any positive integer q*, assuming* $m \leq \sqrt{|\mathbb{K}|/2}$*, and where* $\varepsilon_{\text{RBR}}^{\mathsf{ip}}$ *is the RBR-soundness error of the underlying SC-based IP and where* $\varepsilon_{\text{ZK}}^{\mathsf{lcs}}$*,* $\varepsilon_{\mathcal{H}}^{\mathsf{lcs}}$ *and* $\varepsilon_{\text{RBR}}^{\mathsf{lcs}}$ *are the zero-knowledge and round-by-round knowledge soundness errors of the underlying LCS, respectively.*

Compiler 2. This second compiler is similar to the first one but combines a ZK-LCS and a ZK-MPCS instead of only a ZK-LCS. The ZK-MPCS proves the multilinear evaluation constraints in $\mathcal{E}$, while the ZK-LCS proves the linear trace constraints in $\mathcal{F}$. The motivation for this separation is that an MPCS is typically more efficient than an LCS (since an LCS is strictly more general than an MPCS). This separation allows us to apply an MPCS for the witness w and use an LCS only for the trace mask t, which is beneficial when w is significantly heavier than t.

Hiding multilinear extensions. One pitfall of this separation is that the proven multilinear evaluations must be revealed in clear which might break the zero-knowledge property of the scheme. In order to circumvent this issue, we use the notion of *hiding* multilinear extension (MLE) described in Section 5.1.

Context:

- $\mathcal{P}$ and $\mathcal{V}$ hold an instance $\mathbb{x} \in \mathbb{X}_{\text{IP}}$.
- $\mathcal{P}$ holds a witness $w \in \mathbb{F}^{\omega}$ such that $(w, \mathbb{x}) \in \mathcal{R}_{\text{IP}}$.

Protocol:

- $\mathcal{P}$ samples a random trace mask $t \in \mathbb{K}^{\tau'}$, where $\tau' = \tau + m_{\text{mult}} + 1$.
- $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{LCS.Commit}$ on prover witness (w, t). This protocol results in a commitment com for $\mathcal{V}$ and a key key for $\mathcal{P}$.
- $\mathcal{P}$ and $\mathcal{V}$ runs the SC-based IP with instance $\mathbb{x}$, witness w and trace mask t in the CBIP paradigm. This protocol results in a shared SVCS instance $\mathbb{x}_{\text{SVCS}} = (\mathbb{F}, \mathbb{K}, \omega, \tau, d, \mathcal{E}, \mathcal{F})$.
- $\mathcal{P}$ and $\mathcal{V}$ runs the trace constraint linearization protocol described in Section 3.2 which results in a new SVCS instance $\mathbb{x}'_{\text{SVCS}} = (\mathbb{F}, \mathbb{K}, \omega, \tau', 1, \mathcal{E}, \mathcal{F}')$ and $\mathcal{F}'$ is a set of linear constraints on t.
- $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{LCS.Prove}$ on commitment com and prover key key for the linear constraints in $\mathcal{E}$ and $\mathcal{F}$.
- $\mathcal{V}$ returns the output of the $\mathsf{LCS.Prove}$ protocol (Accept or Reject).

Fig. 6: Compiler 1: From SC-based IP and ZK-LCS to ZK-ARK.

The compiler. Based on a hiding MLE, we can describe our second compiler. It relies on three ingredients:

- a sum-check-based interactive proof (SC-IP) for a relation $\mathcal{R}_{\text{IP}} = \{(w, \mathbb{x})\} \subseteq \mathbb{F}^{\omega} \times \mathbb{X}_{\text{IP}}$, which in the CBIP paradigm gives rise to an SVCS instance $\mathbb{x}_{\text{SVCS}} = (\mathbb{F}, \mathbb{K}, \omega + k, \tau, d, \mathcal{E}, \mathcal{F})$ of witness $w^* = (w, m)$, for some random witness mask m, where the trace constraints $\mathcal{F}$ can be expressed as m_{mult} multiplication constraints;[2]
- a zero-knowledge linear commitment scheme (ZK-LCS) with parameters $(\tau', \mathbb{F}, \mathbb{K})$ (see Definition 5), where $\tau' = \tau + m_{\text{mult}} + 1$ is the trace size in the CBIP version of the SC-IP (including the trace constraint linearization);
- a zero-knowledge multilinear polynomial commitment scheme (ZK-MPCS) with parameters $(\omega + k, \mathbb{F}, \mathbb{K})$ for some positive integer k (size of the mask in the hiding MLE).

From these three ingredients, we get a zero-knowledge argument system for the relation $\mathcal{R}_{\text{IP}}$. The scheme is described in Figure 7. In the protocol description, we assume that the set of multilinear evaluation constraints $\mathcal{E}$ is made of a single element (r, ψ). To prove $\widetilde{w}^*(r) = \psi(t)$ in the split setting, the prover first computes $v = \widetilde{w}^*(r)$ and sends v to the verifier, then proves the multilinear evaluation constraint $\widetilde{w}^*(r) = v$ using the MPCS, while adding the linear constraints $\psi(t) - v$ to $\mathcal{F}'$ to be proved using the LCS. Thanks to the properties of the hiding MLE, the revealed evaluation $v = \widetilde{w}^*(r)$ does not leak information about w except with probability at most ε_{HID} (the hiding error).

Remark 5. If the CBIP execution of the SC-IP results in an *extended* SVCS instance as defined in Section 3.1 (or a standard SVCS instance with several evaluation constraints in $\mathcal{E}$), the protocol includes an additional step after the execution of the SC-IP: the prover and verifier runs the multilinear constraint merging protocol described in Section 3.2. This reduces the (richer) multilinear evaluation constraints to a single constraint of the form $\widetilde{w}^*(r) = \psi(t)$. For the sake of simplicity, we assume that (when necessary) this extra step is already included in the SC-IP execution and that the resulting SVCS instance is a standard one with a single multilinear evaluation constraint.

The security of the obtained scheme is stated in the following theorem. The proof is provided in Appendix D.

[2] The SC-IP is executed on the extended witness $w^* = (w, m) \in \mathbb{F}^{\omega+k}$, whose k mask coordinates are *nullified* throughout the execution: every sum over the Boolean hypercube arising in the protocol carries a public factor vanishing on the mask positions (equivalently, the public matrices A_i of the underlying extended SVCS instance have zero columns there), so that the mask coordinates are left unconstrained by the statement and only intervenes in the final multilinear evaluation constraint, which bears on $\widetilde{w}^*$ instead of $\widetilde{w}$.

Context:

- $\mathcal{P}$ and $\mathcal{V}$ hold an instance $\mathbb{x} \in \mathbb{X}_{\text{IP}}$.
- $\mathcal{P}$ holds a witness $w \in \mathbb{F}^{\omega}$ such that $(w, \mathbb{x}) \in \mathcal{R}_{\text{IP}}$.

Protocol:

- $\mathcal{P}$ samples a random trace mask $t \in \mathbb{K}^{\tau'}$ and a random witness mask $m \in \mathbb{F}^{k}$, where $\tau' = \tau + m_{\text{mult}} + 1$.
- $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{LCS.Commit}$ on prover witness t. This protocol results in a commitment com_t for $\mathcal{V}$ and a key key_t for $\mathcal{P}$.
- $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{MPCS.Commit}$ on prover witness $w^* = (w, m)$. This protocol results in a commitment com_w for $\mathcal{V}$ and a key key_w for $\mathcal{P}$.
- $\mathcal{P}$ and $\mathcal{V}$ runs the SC-based IP with instance $\mathbb{x}$, witness $w^* = (w, m)$ and trace mask t in the CBIP paradigm. This protocol results in a shared SVCS instance $\mathbb{x}_{\text{SVCS}} = (\mathbb{F}, \mathbb{K}, \omega + k, \tau, d, \mathcal{E}, \mathcal{F})$.
- $\mathcal{P}$ and $\mathcal{V}$ runs the trace constraint linearization protocol described in Section 3.2 which results in a new SVCS instance $\mathbb{x}'_{\text{SVCS}} = (\mathbb{F}, \mathbb{K}, \omega + k, \tau', 1, \mathcal{E}, \mathcal{F}')$ and $\mathcal{F}'$ is a set of linear constraints on t.
- Let $\mathcal{E} = \{(r, \psi)\}$ where $r \in \mathbb{K}^{\ell}$ and $\psi : \mathbb{K}^{\tau} \to \mathbb{K}$ an affine map so that the multilinear evaluation constraint to be proved is $\widetilde{w}^*(r) = \psi(t)$. $\mathcal{P}$ computes $v = \widetilde{w}^*(r)$ and sends v to $\mathcal{V}$.
- $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{MPCS.Prove}$ on commitment com_w and prover key key_w for the multilinear evaluation constraint $\widetilde{w}^*(r) = v$.
- $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{LCS.Prove}$ on commitment com_t and prover key key_t for the linear constraints in $\mathcal{F}' \cup \{\psi(t) - v\}$.
- If the two above protocols ($\mathsf{MPCS.Prove}$ and $\mathsf{LCS.Prove}$) are successful, $\mathcal{V}$ returns Accept, otherwise $\mathcal{V}$ returns Reject.

Fig. 7: Compiler 2: From SC-based IP, ZK-LCS and ZK-MPCS to ZK-ARK.

Theorem 3 (Security of Compiler 2). *The zero-knowledge argument system described in Figure 7 is* $(q, \varepsilon_{\text{ZK}})$*-HVZK and has* $(q, \varepsilon_{\mathcal{H}}, \varepsilon_{\text{RBR}})$*-round-by-round knowledge soundness where*

$$\varepsilon_{\text{ZK}} = \varepsilon_{\text{ZK}}^{\mathsf{lcs}} + \varepsilon_{\text{ZK}}^{\mathsf{mpcs}} + \varepsilon_{\text{HID}} \ , \quad \varepsilon_{\mathcal{H}} \leq \varepsilon_{\mathcal{H}}^{\mathsf{lcs}} + \varepsilon_{\mathcal{H}}^{\mathsf{mpcs}} \quad \textit{and} \quad \varepsilon_{\text{RBR}} \leq \max\left(\varepsilon_{\text{RBR}}^{\mathsf{ip}}, \varepsilon_{\text{RBR}}^{\mathsf{lcs}}, \varepsilon_{\text{RBR}}^{\mathsf{mpcs}}, \frac{2m}{|\mathbb{K}|}\right)$$

for any positive integer q *and where* $\varepsilon_{\text{RBR}}^{\mathsf{ip}}$ *is the RBR-soundness error of the underlying SC-based IP and where* $\varepsilon_{\text{ZK}}^{\mathsf{lcs}}$, $\varepsilon_{\text{ZK}}^{\mathsf{mpcs}}$, $\varepsilon_{\text{RBR}}^{\mathsf{lcs}}$, *and* $\varepsilon_{\text{RBR}}^{\mathsf{mpcs}}$ *together with* $\varepsilon_{\mathcal{H}}^{\mathsf{lcs}}$ *and* $\varepsilon_{\mathcal{H}}^{\mathsf{mpcs}}$ *are the zero-knowledge and round-by-round knowledge soundness errors of the underlying LCS and MPCS, respectively.*

Remark 6. We present the output of our compilers as zero-knowledge argument systems, but the protocols of Figures 6 and 7 are in fact commit-and-prove systems, for the relation $\mathcal{R}_1 := \{((w,t),\mathbb{x}) : (w,\mathbb{x}) \in \mathcal{R}_{\text{IP}}\}$ in the case of Compiler 1: the prover first commits to the witness and the trace mask, and only then engages in a protocol proving the statement $\mathbb{x}$ – which, in particular, need not be known at commitment time. We adopt this view in the security proofs, as it allows us to apply Definitions 8 and 9 verbatim, with the statement inserted at the end of the commit phase.

4.2 SmallWood-Style Commitment Schemes

In this section we introduce two commitment schemes to efficiently instantiate the compilers described in the previous section. In the style of SmallWood [FR26], both schemes rely on the *degree enforcing commitment scheme* (DECS), which is a small-domain polynomial commitment scheme (see Section 2.4 for details). We show how to use the DECS to build a linear commitment scheme, which we name SmallWood-LCS and a multilinear polynomial commitment scheme, which we name SmallWood-MPCS.

Commitment Protocol. We first describe the commitment protocol which is common to both schemes. These schemes are based on two parameters s and s'. The parameter s is the *packing parameter* which determines how many entries of the witness are packed together in a single polynomial. In case of the MPCS, s must be a power of 2. The parameter s' is a security parameter corresponding to the number of evaluations in the DECS openings. We further denote $n = \lceil \omega/s \rceil$, the dimension of the witness polynomial, and d_Q the degree of the polynomial Q involved in the proof of constraints, which is defined as

$$d_Q := \begin{cases} 2s + s' - 1 & \text{for the LCS} \\ s + s' - 1 & \text{for the MPCS} \end{cases} \tag{5}$$

The witness w is first arranged as a matrix W of s columns and $n = \lceil \omega/s \rceil$ rows (where the last row is possibly padded with 0's over $\mathbb{F}$). The prover samples a random matrix R of n rows and s' columns over $\mathbb{F}$. For two fixed and distinct sets $\Omega \subseteq \mathbb{F} \setminus \mathbb{E}$ and $\Omega' \subseteq \mathbb{F}$ of size s and s' respectively, the prover computes the vector polynomial $\boldsymbol{P}$ by interpolation from W and R, as

$$\boldsymbol{P} = \mathcal{I}_{\Omega,\Omega'}(W, R) \ ,$$

which by definition implies that $\boldsymbol{P}(\Omega) = W$ and $\boldsymbol{P}(\Omega') = R$ (see Section 2.1). The prover further samples a random *masking* polynomial $M \in \mathbb{K}[X]^{\leq d_Q}$. In the case of the LCS, M is sampled such that $\sum_{x\in\Omega} M(x) = 0$ while in the case of the MPCS, M is sampled uniformly at random. The commitment to w simply consists in the DECS commitment to $\boldsymbol{P}$ and M.

Remark 7. Let us stress that the DECS enables committing to polynomials of a fixed degree over a given base field $\mathbb{F}$. Here, this fixed degree is d_P and the base field is $\mathbb{F}$. For the commitment of $M \in \mathbb{K}[X]^{\leq d_Q}$, one commits to η polynomials $M_1, \ldots, M_\eta \in \mathbb{F}[X]^{\leq d_Q}$ such that $M(X) = \sum_{i=1}^{\eta} \beta_i \cdot M_i(X)$ where $\{\beta_i\}_{i=1}^{\eta}$ is a basis of $\mathbb{K}$ over $\mathbb{F}$. The polynomials M_i are further split into polynomials $M_{i,1}, \ldots, M_{i,\theta} \in \mathbb{F}[X]^{\leq d_P}$ such that $M_i(X) = M_\theta(X) \cdot X^{d_Q - d_P} + \sum_{j=1}^{\theta-1} X^{(j-1)\cdot(d_P+1)} \cdot M_{i,j}(X)$ where $\theta = \lceil (d_Q+1)/(d_P+1) \rceil$. To sum up, committing to M requires committing to $n_M := \eta \cdot \lceil (d_Q+1)/(d_P+1) \rceil$ polynomials of degree at most d_P over $\mathbb{F}$.

SmallWood-LCS. As introduced in Section 2.3, a linear commitment scheme (LCS) is a CP system for the relation $\mathcal{R} = \{(w, (a, b)) : \langle a, w\rangle = b\}$. We consider a multi-query version of LCS, where the relation is

$$\mathcal{R} = \{(w, ((a_1, b_1), \ldots, (a_m, b_m))) : \forall i, \langle a_i, w\rangle = b_i\}$$

for some positive integer m, where $a_1, \ldots, a_m \in \mathbb{K}^\omega$ and $b_1, \ldots, b_m \in \mathbb{K}$. We show how to efficiently instantiate such a scheme using ideas from Ligero [AHIV17, AHIV23] and TCitH [FR25]. In a nutshell, the construction described below relies on DECS commitments to the witness w and it uses Ligero's way to prove linear relations on committed values.

For each linear constraint $\langle a_i, w\rangle = b_i$, there exists a vector polynomial $\boldsymbol{A}_i \in (\mathbb{K}[X]^{\leq s})^n$ whose coefficients are the entries of a_i such that:

$$\langle a_i, w\rangle = \sum_{x\in\Omega} Q_i(x) \quad \text{where} \quad Q_i = \langle \boldsymbol{A}_i, \boldsymbol{P}\rangle \ .$$

By definition, we have $\deg(Q_i) \leq d_Q$ for all i, where d_Q is given in Equation (5). To prove that w satisfies the m linear constraints in zero-knowledge, the prover receives a random challenge $(\gamma_1, \ldots, \gamma_m) \in \mathbb{K}^m$ from the verifier and computes the polynomial

$$Q(X) = M(X) + \sum_{i=1}^{m} \gamma_i \cdot Q_i(X) \ \in \mathbb{K}[X]^{\leq d_Q}$$

and sends it to the verifier (implicitly implying that $\deg(Q) \leq d_Q$). The verifier checks that $\sum_{x\in\Omega} Q(x) = \sum_{i=1}^{m} \gamma_i \cdot b_i$ which implies that all the linear constraints are satisfied with probability at least $1 - 1/|\mathbb{K}|$ over the random choice of $(\gamma_1, \ldots, \gamma_m)$. The prover and verifier then run the DECS opening protocol to prove that Q is correctly computed from the committed $\boldsymbol{P}$ and M. By opening s' evaluations, the verifier is guaranteed that Q is correctly computed with probability at least $1 - \binom{d_Q}{s'}/\binom{|\mathbb{E}|}{s'}$ by the Schwartz-Zippel lemma. The overall protocol is described in Figure 8.

Remark 8. In practice, the prover does not need to send the entire polynomial Q to the verifier. Instead, the prover can first send a hash commitment to Q, then, after some evaluations $\boldsymbol{P}(E)$, $M(E)$ have been revealed (with $|E| = s'$), revealing some evaluations $Q(S)$ where $|S| = d_Q + 1 - s'$. This way, the communication cost of sending Q is reduced from $d_Q + 1$ field elements to $d_Q + 1 - s'$.

The security of SmallWood-LCS is stated in the following theorem.

Theorem 4 (Security of SmallWood-LCS). *The linear commitment scheme described in Figure 8 is $(q, \varepsilon_{\text{ZK}})$-HVZK and has $(q, \varepsilon_{\mathcal{H}}, \varepsilon_{\text{RBR}})$-round-by-round knowledge soundness where*

$$\varepsilon_{\text{ZK}} = \varepsilon_{\text{ZK}}^{\mathsf{decs}} \ , \quad \varepsilon_{\mathcal{H}} \leq \varepsilon_{\mathcal{H}}^{\mathsf{decs}} \ , \quad \textit{and} \quad \varepsilon_{\text{RBR}} \leq \max\left(\varepsilon_{\text{RBR}}^{\mathsf{decs}}, \frac{1}{|\mathbb{K}|}, \frac{\binom{d_Q}{s'}}{\binom{|\mathbb{E}|}{s'}}\right)$$

for any positive integer q and where $\varepsilon_{\text{ZK}}^{\mathsf{decs}}$, $\varepsilon_{\mathcal{H}}^{\mathsf{decs}}$ and $\varepsilon_{\text{RBR}}^{\mathsf{decs}}$ are the zero-knowledge and round-by-round knowledge soundness errors of the underlying DECS, respectively.

SmallWood-MPCS As introduced in Section 2.3, a multilinear polynomial commitment scheme (MPCS) is a CP system for the relation

$$\mathcal{R} = \{(w, (r, v)) : \widetilde{w}(r) = v\} \subseteq \mathbb{F}^\omega \times (\mathbb{K}^\ell \times \mathbb{K})$$

where $\widetilde{w}$ is the multilinear extension of w. We show how to efficiently instantiate such a scheme using ideas from Brakedown [GLS+23] and SmallWood [FR26]. In a nutshell, the construction described below is reminiscent of the SmallWood PCS construction based on a LVCS with a simplified way of dealing with zero-knowledge.

Recall that in the context of the MPCS, the packing parameter s must be a power of 2. We denote σ th integer such that $s = 2^\sigma$. The matrix W is now arranged in n rows and $s = 2^\sigma$ columns, where $n = \lceil \omega/s \rceil$. For a point $r = (r_1, \ldots, r_\ell) \in \mathbb{K}^\ell$, the evaluation $\widetilde{w}(r)$ can be computed as

$$\widetilde{w}(r) = \sum_{i=1}^{n}\sum_{j=1}^{s} W(i,j) \cdot \chi_j(r_1, \ldots, r_\sigma) \cdot \chi_i(r_{\sigma+1}, \ldots, r_\ell) = v_{\text{L}}^T \cdot W \cdot v_{\text{R}}$$

LCS.Commit

- At the beginning, $\mathcal{P}$ holds a witness $w \in \mathbb{F}^{\omega}$. Let W be the matrix obtained by arranging w in n rows and s columns, where s is a packing parameter and $n = \lceil \omega/s \rceil$.
- $\mathcal{P}$ samples a random matrix $R \in \mathbb{F}^{n \times s'}$ and a random polynomial $M \in \mathbb{K}[X]^{\leq d_Q}$ such that $\sum_{x \in \Omega} M(x) = 0$. $\mathcal{P}$ computes $\boldsymbol{P} = \mathcal{I}_{\Omega,\Omega'}(W, R)$.
- $\mathcal{P}$ and $\mathcal{V}$ runs the protocol DECS.Commit to commit to $(\boldsymbol{P}, M)$. At the end of the protocol, $\mathcal{V}$ holds a commitment $\mathsf{com}_{\mathsf{decs}}$ and $\mathcal{P}$ holds a key $\mathsf{key}_{\mathsf{decs}}$.
- The protocol ends with $\mathcal{P}$ keeping $\mathsf{key}_{\mathsf{lcs}} = (\mathsf{key}_{\mathsf{decs}}, \boldsymbol{P}, M)$ and $\mathcal{V}$ keeping $\mathsf{com}_{\mathsf{lcs}} = \mathsf{com}_{\mathsf{decs}}$.

LCS.Prove

- At the beginning, $\mathcal{P}$ holds a key $\mathsf{key}_{\mathsf{lcs}} = (\mathsf{key}_{\mathsf{decs}}, \boldsymbol{P}, M)$ and $\mathcal{V}$ holds a commitment $\mathsf{com}_{\mathsf{lcs}} = \mathsf{com}_{\mathsf{decs}}$. They both hold elements $a_1, \ldots, a_m \in \mathbb{K}^{\omega}$ and $b_1, \ldots, b_m \in \mathbb{K}$ defining the linear constraints to be proven. For each i, let $\boldsymbol{A}_i$ be the vector polynomial whose coefficients are the entries of a_i.
- $\mathcal{V}$ samples a random challenge $(\gamma_1, \ldots, \gamma_m) \in \mathbb{K}^m$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ computes the polynomial $Q = M + \sum_{i=1}^{m} \gamma_i \cdot Q_i$ where $Q_i = \langle \boldsymbol{A}_i, \boldsymbol{P} \rangle$ and sends it to the verifier.
- $\mathcal{V}$ checks that $\sum_{x \in \Omega} Q(x) = \sum_{i=1}^{m} \gamma_i \cdot b_i$. If the check fails, $\mathcal{V}$ returns Reject and the protocol ends.
- $\mathcal{V}$ samples a random set $E \subseteq \mathbb{E}$ of size s' and sends it to $\mathcal{P}$.
- $\mathcal{P}$ sends the evaluations $\boldsymbol{P}(E)$ and $M(E)$ to $\mathcal{V}$.
- $\mathcal{V}$ checks that $Q(e) = M(e) + \sum_{i=1}^{m} \gamma_i \cdot Q_i(e)$, for all $e \in E$, where $Q_i(e)$ is computed from the evaluation $\boldsymbol{P}(e)$. If any of the checks fails, $\mathcal{V}$ returns Reject and the protocol ends.
- $\mathcal{P}$ and $\mathcal{V}$ run the protocol DECS.Prove to check the correctness of the claimed evaluations $\boldsymbol{P}(E)$ and $M(E)$. If the protocol fails, $\mathcal{V}$ returns Reject. If the protocol succeeds, $\mathcal{V}$ returns Accept.

Fig. 8: The SmallWood-LCS linear commitment scheme.

where χ_i and χ_j are the i-th and j-th Lagrange basis polynomials (in $\ell - \sigma$ variables and σ variables, respectively) as defined in Section 2.1, and where

$$v_{\mathrm{R}} = \begin{bmatrix} 1 - r_1 \\ r_1 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 - r_\sigma \\ r_\sigma \end{bmatrix} \in \mathbb{K}^s \quad \text{and} \quad v_{\mathrm{L}} = \mathsf{First}_n \left(\begin{bmatrix} 1 - r_{\sigma+1} \\ r_{\sigma+1} \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 - r_\ell \\ r_\ell \end{bmatrix} \right) \in \mathbb{K}^n \tag{6}$$

for First_n the function that takes the first n entries of a vector.

To prove a constraint of the form $\widetilde{w}(r) = v$, the prover first sends

$$\delta = \langle M(\Omega), v_{\mathrm{R}} \rangle$$

to the verifier. The latter replies with a random challenge $\gamma \in \mathbb{K}$. The prover then sends the polynomial

$$Q(X) = M(X) + \gamma \cdot Q'(X) \quad \text{with} \quad Q'(X) = \langle v_{\mathrm{L}}, \boldsymbol{P}(X) \rangle \ \in \mathbb{K}[X]^{\leq d_Q} \ .$$

The verifier checks that $\langle Q(\Omega), v_{\mathrm{R}} \rangle = \delta + \gamma \cdot v$. In case of success, this implies that $\langle Q'(\Omega), v_{\mathrm{R}} \rangle = v$ with probability at least $1 - 1/|\mathbb{K}|$ over the random choice of γ. The prover and verifier then run the DECS opening protocol to check that Q is correctly computed from the committed $\boldsymbol{P}$ and M. By opening s' evaluations, the verifier is guaranteed that Q is correctly computed with probability at least $1 - \binom{d_Q}{s'} / \binom{|\mathbb{E}|}{s'}$ by the Schwartz-Zippel lemma. The overall protocol is described in Figure 9.

MPCS.Commit

- At the beginning, $\mathcal{P}$ holds a witness $w \in \mathbb{F}^\omega$. Let W be the matrix obtained by arranging w in n rows and $s = 2^\sigma$ columns, where s is a packing parameter and $n = \lceil \omega / s \rceil$.
- $\mathcal{P}$ samples a random matrix $R \in \mathbb{F}^{n \times s'}$ and a random polynomial $M \in \mathbb{K}[X]^{\leq d_Q}$. $\mathcal{P}$ computes $\boldsymbol{P} = \mathcal{I}_{\Omega,\Omega'}(W, R)$.
- $\mathcal{P}$ and $\mathcal{V}$ runs the protocol DECS.Commit to commit to $(\boldsymbol{P}, M)$. At the end of the protocol, $\mathcal{V}$ holds a commitment $\mathsf{com}_{\mathsf{decs}}$ and $\mathcal{P}$ holds a key $\mathsf{key}_{\mathsf{decs}}$.
- The protocol ends with $\mathcal{P}$ keeping $\mathsf{key}_{\mathsf{mpcs}} = (\mathsf{key}_{\mathsf{decs}}, \boldsymbol{P}, M)$ and $\mathcal{V}$ keeping $\mathsf{com}_{\mathsf{mpcs}} = \mathsf{com}_{\mathsf{decs}}$.

MPCS.Prove

- At the beginning, $\mathcal{P}$ holds a key $\mathsf{key}_{\mathsf{mpcs}} = (\mathsf{key}_{\mathsf{decs}}, \boldsymbol{P}, M)$ and $\mathcal{V}$ holds a commitment $\mathsf{com}_{\mathsf{mpcs}} = \mathsf{com}_{\mathsf{decs}}$. They both hold elements $r = (r_1, \ldots, r_\ell) \in \mathbb{K}^\ell$ and $v \in \mathbb{K}$ defining the multilinear evaluation to be proven. Let v_{R} and v_{L} be as defined in Equation (6).
- $\mathcal{P}$ computes $\delta = \langle M(\Omega), v_{\mathrm{R}} \rangle$ and sends it to $\mathcal{V}$.
- $\mathcal{V}$ samples a random challenge $\gamma \in \mathbb{K}$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ computes the polynomial $Q(X) = M(X) + \gamma \cdot Q'(X)$, where $Q'(X) = \langle v_{\mathrm{L}}, \boldsymbol{P}(X) \rangle$, and sends it to $\mathcal{V}$.
- $\mathcal{V}$ checks that $\langle Q(\Omega), v_{\mathrm{R}} \rangle = \delta + \gamma \cdot v$. If the check fails, $\mathcal{V}$ returns Reject and the protocol ends.
- $\mathcal{V}$ samples a random set $E \subseteq \mathbb{E}$ of size s' and sends it to $\mathcal{P}$.
- $\mathcal{P}$ sends the evaluations $\boldsymbol{P}(E)$ and $M(E)$ to $\mathcal{V}$.
- $\mathcal{V}$ checks that $Q(e) = M(e) + \gamma \cdot Q'(e)$, for all $e \in E$, where $Q'(e)$ is computed from the evaluation $\boldsymbol{P}(e)$. If any of the checks fails, $\mathcal{V}$ returns Reject and the protocol ends.
- $\mathcal{P}$ and $\mathcal{V}$ run the protocol DECS.Prove to check the correctness of the claimed evaluations $\boldsymbol{P}(E)$ and $M(E)$. If the protocol fails, $\mathcal{V}$ returns Reject. If the protocol succeeds, $\mathcal{V}$ returns Accept.

Fig. 9: The SmallWood-MPCS multilinear polynomial commitment scheme.

The security of SmallWood-MPCS is stated in the following theorem.

Theorem 5 (Security of SmallWood-MPCS). *The multilinear polynomial commitment scheme described in Figure 9 is $(q, \varepsilon_{\mathrm{ZK}})$-HVZK and has $(q, \varepsilon_{\mathcal{H}}, \varepsilon_{\mathrm{RBR}})$-round-by-round knowledge soundness where*

$$\varepsilon_{\mathrm{ZK}} = \varepsilon_{\mathrm{ZK}}^{\mathsf{decs}} \ , \quad \varepsilon_{\mathcal{H}} \leq \varepsilon_{\mathcal{H}}^{\mathsf{decs}} \quad \textit{and} \quad \varepsilon_{\mathrm{RBR}} \leq \max \left(\varepsilon_{\mathrm{RBR}}^{\mathsf{decs}}, \frac{1}{|\mathbb{K}|}, \frac{\binom{d_Q}{s'}}{\binom{|\mathbb{E}|}{s'}} \right)$$

for any positive integer q and where $\varepsilon_{\mathrm{ZK}}^{\mathsf{decs}}$, $\varepsilon_{\mathcal{H}}^{\mathsf{decs}}$ and $\varepsilon_{\mathrm{RBR}}^{\mathsf{decs}}$ are the zero-knowledge and round-by-round knowledge soundness errors of the underlying DECS, respectively.

We stated above that the packing parameter s should be a power of 2. While this restriction simplifies both the description and the implementation, it is not necessarily optimal in terms of opening-proof size. We can relax this requirement and allow arbitrary values of s.

As before, W is arranged into n rows and s columns, but s is no longer required to be a power of 2. Let $\sigma := \lceil \log_2(s) \rceil$. Then, in Equation (6), v_{R} is defined as

$$v_{\mathrm{R}} = \mathsf{First}_s \left(\begin{bmatrix} 1 - r_1 \\ r_1 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 - r_\sigma \\ r_\sigma \end{bmatrix} \right) \in \mathbb{K}^s \ ,$$

where First_s denotes the truncation to the first s coordinates.

One consequence of this relaxation is that the coordinates of w no longer correspond to evaluations of its multilinear extension at the first ω points of the Boolean hypercube $\{0,1\}^\ell$, namely

$$\left\{ (b_0, \ldots, b_{\ell-1}) : \sum_{i=0}^{\ell-1} b_i 2^i < \omega \right\}.$$

Instead, they correspond to evaluations at the points

$$\left\{ (b_0, \ldots, b_{\ell-1}) : \sum_{j=0}^{\sigma-1} b_j 2^j < s \quad \text{and} \quad \sum_{i=0}^{\ell-\sigma-1} b_{\sigma+i} 2^i < n \right\}.$$

In other words, the valid evaluation points now form the Cartesian product of the first s points of a σ-dimensional hypercube and the first n points of an $(\ell - \sigma)$-dimensional hypercube. Moreover, ℓ is no longer necessarily equal to $\lceil \log_2(\omega) \rceil$, but instead

$$\ell = \lceil \log_2(s) \rceil + \lceil \log_2(\omega/s) \rceil \ .$$

Merging DECS Instances. For our second compiler, we use both the SmallWood LCS and MPCS, which means that we need to run two DECS instances in parallel. We can merge these two instances into a single one using a single Merkle tree to commit to the DECS evaluations of both the LCS and MPCS. This way, we only have one DECS commitment and one DECS opening protocol instead of two. In this context, the two instances can share a single Merkle tree, so that the authentication paths are paid only once. We describe the merge for two instances; it extends to any number of them in the obvious way.

Consider two DECS instances $i \in \{1,2\}$ with respective parameters $(n_i, d_i, \mathbb{F}_i, \mathbb{E}_i)$ and degree-enforcing dimension η_i, committing to vector polynomials $\boldsymbol{P}_i$. We only assume that the two evaluation domains have the same size, $|\mathbb{E}_1| = |\mathbb{E}_2| = N$, and index their points compatibly: $e_{i,j}$ denotes the j-th point of $\mathbb{E}_i$, for $j \in [N]$. Following Figure 4, write

$$u_{i,j} = \mathsf{Hash}\big(\boldsymbol{P}_i(e_{i,j}), \boldsymbol{M}_i(e_{i,j}), j, \rho_{i,j}\big)$$

for the leaf that instance i would produce on its own at index j.

The merged DECS protocol is described in Figure 10. The two instances are committed exactly as in DECS.Commit, except that their leaves are combined index-wise into a *single* Merkle tree. Two matrices Γ_1, Γ_2 are sampled and sent to the prover, who computes the two $\boldsymbol{R}_i$ and a single hash commitment h_R. The merged commitment is $\mathsf{com} = (\mathsf{root}, \Gamma_1, \Gamma_2, h_R)$.

The merged opening of public evaluations $\boldsymbol{P}_1(E_1)$ and $\boldsymbol{P}_2(E_2)$, with $E_i \subseteq \mathbb{E}_i$ for $i \in \{1,2\}$, requires $E_1 \subseteq E_2$, so that the merged tree is opened exactly at the leaves indexed by E_2. To open:

- For each $i \in \{1,2\}$, $\mathcal{P}$ sends $\boldsymbol{M}_i(E_i)$, the completion evaluations $\boldsymbol{R}_i(S_i)$ with $|S_i| = d_i + 1 - |E_i|$, and the random tapes $\{\rho_{i,j} : j \in E_i\}$ – as in a standalone opening of instance i.
- $\mathcal{P}$ additionally sends the leaves $\{u_{1,j} : j \in E_2 \setminus E_1\}$ of the first instance, and the authentication paths π_{MT} for the merged leaves $\{u_j : j \in E_2\}$ in the single Merkle tree.

The verifier proceeds as follows:

MergedDECS.Commit:

On input two vector polynomials $\boldsymbol{P}_i \in \left(\mathbb{F}_i[X]^{(\leq d_i)}\right)^{n_i}$ $(i \in \{1,2\})$, for two DECS instances with parameters $(n_i, d_i, \mathbb{F}_i, \mathbb{E}_i)$, degree-enforcing dimension η_i and common domain size $|\mathbb{E}_1| = |\mathbb{E}_2| = N$:

- For each $i \in \{1,2\}$, $\mathcal{P}$ samples $\boldsymbol{M}_i = (M_{i,1}, \ldots, M_{i,\eta_i}) \leftarrow (\mathbb{F}_i[X]^{(\leq d_i)})^{\eta_i}$ and $(\rho_{i,1}, \ldots, \rho_{i,N}) \leftarrow (\{0,1\}^\lambda)^N$.
- For each $i \in \{1,2\}$ and $j \in [N]$, $\mathcal{P}$ computes the per-instance leaves

$$u_{i,j} = \mathsf{Hash}\big(\boldsymbol{P}_i(e_{i,j}), \boldsymbol{M}_i(e_{i,j}), j, \rho_{i,j}\big) \ ,$$

  where $e_{i,j}$ is the j-th point of $\mathbb{E}_i$.
- $\mathcal{P}$ forms the merged leaves $u_j = \mathsf{Hash}(u_{1,j}, u_{2,j})$ for all $j \in [N]$ and $\mathsf{root} = \mathsf{MerkleRoot}(u_1, \ldots, u_N)$. $\mathcal{P}$ sends root to $\mathcal{V}$.
- $\mathcal{V}$ samples $\varGamma_i = (\gamma_{k,l}^{(i)})_{k,l} \leftarrow \mathbb{F}_i^{\eta_i \times n_i}$ for $i \in \{1,2\}$ and sends $(\varGamma_1, \varGamma_2)$ to $\mathcal{P}$.
- For each $i \in \{1,2\}$, $\mathcal{P}$ computes

$$R_{i,k}(X) = M_{i,k}(X) + \sum_{l=1}^{n_i} \gamma_{k,l}^{(i)} \cdot P_{i,l}(X) \quad \text{for all } k \in [\eta_i] \ ,$$

  sets $\boldsymbol{R}_i := (R_{i,1}, \ldots, R_{i,\eta_i})$, and a single hash commitment $h_R = \mathsf{Hash}(\boldsymbol{R}_1, \boldsymbol{R}_2)$. $\mathcal{P}$ sends h_R to $\mathcal{V}$.
- $\mathcal{V}$ holds $\mathsf{com} := (\mathsf{root}, \varGamma_1, \varGamma_2, h_R)$ and $\mathcal{P}$ holds $\mathsf{key} := (\boldsymbol{P}_1, \boldsymbol{M}_1, \boldsymbol{R}_1, \boldsymbol{P}_2, \boldsymbol{M}_2, \boldsymbol{R}_2)$.

MergedDECS.Prove:

Both parties take opening sets $E_1 \subseteq E_2 \subseteq \{1, \ldots, N\}$ (indexing the leaves) with $E_1 \subseteq E_2$, and the evaluations $\boldsymbol{P}_1(E_1)$, $\boldsymbol{P}_2(E_2)$. We write $s_i' = |E_i|$.

- For each $i \in \{1,2\}$, $\mathcal{P}$ sends:
  - the mask evaluations $\boldsymbol{M}_i(E_i)$ and the completion evaluations $\boldsymbol{R}_i(S_i)$ for a set $S_i \subseteq \mathbb{E}_i \setminus E_i$ with $|S_i| = d_i + 1 - s_i'$;
  - the random tapes $\{\rho_{i,j} : j \in E_i\}$.
- $\mathcal{P}$ additionally sends the first-instance leaves $\{u_{1,j} : j \in E_2 \setminus E_1\}$ and the authentication paths π_{MT} for the merged leaves $\{u_j : j \in E_2\}$ w.r.t. root.
- $\mathcal{V}$ performs the following checks:
  - *(Degree enforcement.)* For each $i \in \{1,2\}$: compute $\boldsymbol{R}_i(e_{i,j}) = \boldsymbol{M}_i(e_{i,j}) + \varGamma_i \cdot \boldsymbol{P}_i(e_{i,j})$ for all $j \in E_i$, recover $\boldsymbol{R}_i$ from $\boldsymbol{R}_i(E_i \cup S_i)$ by interpolation (of degree $\leq d_i$ by construction), then check $h_R = \mathsf{Hash}(\boldsymbol{R}_1, \boldsymbol{R}_2)$.
  - *(Merkle consistency.)* Recompute $u_{2,j} = \mathsf{Hash}(\boldsymbol{P}_2(e_{2,j}), \boldsymbol{M}_2(e_{2,j}), j, \rho_{2,j})$ for all $j \in E_2$, and $u_{1,j} = \mathsf{Hash}(\boldsymbol{P}_1(e_{1,j}), \boldsymbol{M}_1(e_{1,j}), j, \rho_{1,j})$ for all $j \in E_1$; take $u_{1,j}$ for $j \in E_2 \setminus E_1$ from the proof. Form $u_j = \mathsf{Hash}(u_{1,j}, u_{2,j})$ for all $j \in E_2$ and verify π_{MT} against root.

Fig. 10: Merged DECS for two instances, sharing a single Merkle tree (see Figure 4 for the single-instance scheme).

– For each i, it recovers $\boldsymbol{R}_i$ exactly as in Figure 4 and checks the single equality $h_R = \mathsf{Hash}(\boldsymbol{R}_1, \boldsymbol{R}_2)$.
– It then rebuilds the merged leaves on E_2: it recomputes $u_{2,j}$ for all $j \in E_2$ and $u_{1,j}$ for $j \in E_1$ from the opened data, takes $u_{1,j}$ for $j \in E_2 \setminus E_1$ directly from the proof, sets $u_j = \mathsf{Hash}(u_{1,j}, u_{2,j})$ for $j \in E_2$, and verifies π_{MT} against root.

The merged leaf u_j hashes the two per-instance leaves rather than their joint preimage, precisely so that the verifier can recompute u_j at the indices $E_2 \setminus E_1$ where only the second instance is opened: there it has no access to $\boldsymbol{P}_1(e_{1,j})$, but the opaque value $u_{1,j}$ supplied in the proof suffices to rebuild u_j. Since each $u_{1,j}$ is itself a salted hash, it hides $\boldsymbol{P}_1(e_{1,j})$, so revealing it leaks nothing about the first instance and zero-knowledge is preserved. Round-by-round soundness is inherited from the two underlying instances: the degree-enforcing check of each instance is performed unchanged, and the merged tree binds both sets of leaves to a single root exactly as a standalone DECS tree binds one (the error $\varepsilon_{\mathcal{H}}$ is essentially unchanged, the lucky events of the merged tree being those of a single tree with one more hash commitment h_R).

Remark 9. While this generalization is not needed for our MPCS, the DECS-merging technique can be extended to settings involving more than two DECS. Suppose that we have k DECS with evaluation sets satisfying $E_1 \subset E_2 \subset \cdots \subset E_k$. For each leaf index j, the corresponding merged leaf is obtained through a hash chain:

$$u_j = \mathsf{Hash}\big(\ldots \mathsf{Hash}(\mathsf{Hash}(u_{1,j}, u_{2,j}), u_{3,j}), \ldots, u_{k,j}\big).$$

With this construction, at most one additional hash digest needs to be revealed for each opened merged leaf.

4.3 Concrete Zero-Knowledge Argument System: SmallWood$^+$

Applying one of our compilers with our SmallWood-style commitment schemes, we obtain a concrete zero-knowledge non-interactive argument system from any sum-check-based interactive protocol. We refer to this family of argument systems as SmallWood$^+$. We further denote the two variants obtained from the two compilers and underlying commitment schemes as:

– SmallWood$^+$ v1 based on Compiler 1 with SmallWood-LCS, and
– SmallWood$^+$ v2 based on Compiler 2 with SmallWood-LCS and SmallWood-MPCS (using a merged DECS instance).

We provide hereafter detailed formulas for the proof sizes of these two schemes in terms of the parameters of the underlying SVCS instance and the parameters of the commitment schemes. In the following, we let the base field $\mathbb{F}$ and the extension field $\mathbb{K}$ implicit and denote $\eta = [\mathbb{K} : \mathbb{F}]$. We consider a sum-check-based interactive proof (SC-IP) which when executed in the CBIP paradigm results in an (extended) SVCS instance $\mathbb{x}_{\text{SVCS}}$ with witness size ω and trace size τ and such that the trace constraints in $\mathcal{F}$ include m_{mult} multiplication constraints and m_{lin} linear constraints. Applying the trace constraint linearization protocol of Section 3.2 results in a new SVCS instance $\mathbb{x}'$ with trace size $\tau' = \tau + m_{\text{mult}} + 1$ and with $m_{\text{lin}} + 2$ linear trace constraints in $\mathcal{F}'$.

For the second argument system relying on Compiler 2, we denote k the size of the mask in the hiding MLE and $\ell = \lceil \log_2(\omega + k) \rceil$. Whenever $\mathbb{x}_{\text{SVCS}}$ is an *extended* SVCS instance, or with more than one evaluation constraint in $\mathcal{E}$, Compiler 2 applies the evaluation merging protocol of Section 3.2 which results in a new SVCS instance $\mathbb{x}''$ with trace size $\tau'' = \tau' + 2\ell$. In case this reduction is not required, we simply have $\tau'' = \tau'$. We first ignore the merged DECS optimization for the sake of simplicity, and address it later.

Proof sizes. The proof sizes (in bits) of SmallWood$^+$ v1 and SmallWood$^+$ v2 are given by:

$$|\pi_1| = \tau' \cdot \log_2 |\mathbb{K}| + f_{\mathsf{lcs}}(\omega + \tau' \cdot \eta)$$
$$|\pi_2| = \tau'' \cdot \log_2 |\mathbb{K}| + f_{\mathsf{lcs}}(\tau'' \cdot \eta) + f_{\mathsf{mpcs}}(\omega + k)$$

where $f_{\mathsf{lcs}}(\omega')$ and $f_{\mathsf{mpcs}}(\omega')$ are the proof sizes of SmallWood LCS and MPCS (Commit and Prove) for a witness of size ω' (over $\mathbb{F}$) and any number of constraints (over $\mathbb{K}$). In both formulas, the first term corresponds to the masked trace in the SC-based IP.

Proof sizes of the LCS and MPCS. The size formula for the LCS and MPCS depends on the following parameters:

- s the packing parameter,
- s' the security parameter,
- $N = |\mathbb{E}|$ the size of the Merkle tree in the DECS.

From these parameters (which can be set to different values for the LCS and MPCS), both f_{lcs} and f_{mpcs} are given by the following formula:

$$f_{\{\mathsf{l},\mathsf{mp}\}\mathsf{cs}}(\omega') = \underbrace{(d_Q + 1 - s') \cdot \log_2 |\mathbb{K}|}_{Q} + \underbrace{s' \cdot (n + n_M) \cdot \log_2 |\mathbb{F}|}_{\boldsymbol{P}(E), M(E)} + f_{\mathsf{decs}}(d_P, s') + |\delta|$$

where $|\delta| = 0$ for the LCS and $|\delta| = \log_2 |\mathbb{K}|$ for the MPCS, where $f_{\mathsf{decs}}(d', s')$ is the proof size of the DECS (Commit and Prove) for polynomials of degree at most d' and s' evaluations revealed (see formula below), and where the parameters n, n_M, d_P and d_Q are defined as follows:

- $n = \lceil \omega'/s \rceil$ is the number of rows of the witness polynomial $\boldsymbol{P}$,
- $n_M = \eta \cdot \lceil (d_Q + 1)/(d_P + 1) \rceil$ is the number of $\mathbb{F}[X]^{\leq d_P}$ polynomials to commit M (see Remark 7),
- $d_P = s + s' - 1$ is the degree of the vector polynomial $\boldsymbol{P}$,
- d_Q is the degree of the polynomial Q, given by Equation 5.

Remark 10. The parameters s and s' can be set independently for the LCS and MPCS, and can be optimized to minimize the overall proof size.

Proof size of the DECS. The proof size of the DECS follows from Figure 4. Hereafter we add the subscript decs to the DECS masking polynomial $\boldsymbol{M}$ and to the DECS parameter η to avoid confusion with the masking polynomial M and the parameter η used in the LCS and MPCS. After the Fiat–Shamir transform the challenge matrix Γ is recomputed by the verifier and is not transmitted; the prover's messages consist of the root root and the hash commitment h_R (from Commit), and of the mask evaluations $\boldsymbol{M}_{\mathsf{decs}}(E)$, the completion evaluations $\boldsymbol{R}(S)$, the leaf random tapes $\{\rho_j : e_j \in E\}$ and the Merkle authentication paths π_{MT} (from Prove). The opened evaluations $\boldsymbol{P}(E)$ are *not* included here, as they are accounted for separately in the LCS/MPCS size formula. Writing $a_{\mathsf{MT}}(N, s')$ for the number of internal nodes on the union of the s' authentication paths, we obtain

$$f_{\mathsf{decs}}(d', s') \;=\; \underbrace{(4 + s')\,\lambda}_{\mathsf{root},\ h_R,\ \rho} \;+\; \underbrace{\eta_{\mathsf{decs}} \cdot (d' + 1)\,\log_2 |\mathbb{F}|}_{\boldsymbol{M}_{\mathsf{decs}}(E),\ \boldsymbol{R}(S)} \;+\; \underbrace{2\lambda \cdot a_{\mathsf{MT}}(N, s')}_{\pi_{\mathsf{MT}}}\,,$$

where η_{decs} is fixed as the smallest integer satisfying

$$\frac{\binom{N}{d'+2}}{|\mathbb{F}|^{\eta_{\mathsf{decs}}}} \leq \frac{1}{2^{\lambda}}\,.$$

Merged DECS optimization. The merged DECS of Figure 10 differs from two independent DECS instances only in that the root root, the hash commitment h_R and the authentication paths π_{MT} are shared. Writing $s'_i = |E_i|$ with $E_1 \subseteq E_2$ (so $s'_1 \leq s'_2$) and $d'_i = \deg \boldsymbol{P}_i$, the size of the merged DECS proof is

$$f^{\mathsf{merge}}_{\mathsf{decs}}(d'_1, s'_1, d'_2, s'_2) \;=\; \underbrace{4\lambda}_{\mathsf{root},\ h_R} \;+\; \sum_{i \in \{1,2\}} \Big(\underbrace{s'_i\,\lambda}_{\rho_i} \;+\; \underbrace{\eta_{\mathsf{decs},i} \cdot (d'_i + 1)\,\log_2 |\mathbb{F}|}_{\boldsymbol{M}_{\mathsf{decs},i}(E_i),\ \boldsymbol{R}_i(S_i)} \Big) \;+\; \underbrace{(s'_2 - s'_1) \cdot 2\lambda}_{\{u_{1,j}\}_{E_2 \setminus E_1}} \;+\; \underbrace{2\lambda \cdot a_{\mathsf{MT}}(N, s'_2)}_{\pi_{\mathsf{MT}}}\,,$$

where $\eta_{\mathsf{decs},i}$ is fixed as the smallest integer satisfying $\binom{N}{d'_i+2}/|\mathbb{F}_i|^{\eta_{\mathsf{decs},i}} \leq 2^{-\lambda}$.

For SmallWood$^+$ v2, the LCS and MPCS share a single merged DECS instance, so the two DECS contributions in $|\pi_2|$ must not be counted twice. Concretely, the term $f_{\mathsf{lcs}}(\tau'' \cdot \eta) + f_{\mathsf{mpcs}}(\omega + k)$ double-counts one root, one h_R and the full second set of authentication paths. Pulling the shared DECS out, the proof size becomes

$$|\pi_2| \;=\; \tau'' \cdot \log_2 |\mathbb{K}| \;+\; \big(f^{\circ}_{\mathsf{lcs}} + f^{\circ}_{\mathsf{mpcs}}\big) \;+\; f^{\mathsf{merge}}_{\mathsf{decs}}\,,$$

where $f^{\circ}_{\{\mathsf{l},\mathsf{mp}\}cs}(\omega') := f_{\{\mathsf{l},\mathsf{mp}\}cs}(\omega') - f_{\mathsf{decs}}(\cdot)$ denotes the LCS/MPCS size stripped of its own DECS term and $f^{\mathsf{merge}}_{\mathsf{decs}}$ is instantiated with instance 1 (LCS) and instance 2 (MPCS) parameters $(d'_1, s'_1) = (d_P^{\mathsf{lcs}}, s'^{\mathsf{lcs}})$, $(d'_2, s'_2) = (d_P^{\mathsf{mpcs}}, s'^{\mathsf{mpcs}})$, polynomial counts $n^{\mathsf{lcs}} + n_M^{\mathsf{lcs}}$ and $n^{\mathsf{mpcs}} + n_M^{\mathsf{mpcs}}$, and a common N. The nesting $E_1 \subseteq E_2$ is enforced by deriving the LCS opening set as a random subset of the MPCS one.

5 Toolbox for Zero-Knowledge Sum-Check-Based Protocols

In this section, we introduce useful tools for zero-knowledge sum-check-based (or GKR-based) protocols. Our first tool is the notion of *hiding multilinear extension*, which allows us to make an MLE evaluation zero-knowledge. The second tool is a zero-knowledge version of the ring-switching protocol from [DP26] which enables to efficiently handle sum-check-based protocols over small fields by reducing the final MLE claim over the small field to a claim over a larger field. Our final tool is a circuit flattening technique that improves the efficiency of GKR-based protocols. All these tools will be useful for the applications presented in Section 6.

5.1 Hiding Multilinear Extension

We revisit the idea of random padding of MLE proposed in [FKR26], which we formalize through the notion of *hiding multilinear extension* (hiding MLE). The hiding MLE of a vector $w \in \mathbb{F}^\omega$ is the MLE of the vector w^* obtained by extending w with a uniformly random coordinates (the *mask*). We generalize this approach to the small-field regime: the witness and its mask live in a base field $\mathbb{F}$, while the evaluation point is drawn from an extension $\mathbb{K}$. We must then size and place the mask in a way to ensure that an evaluation $\widetilde{w}^*(r)$ reveals no information about the original vector w, with overwhelming probability over a random choice of the evaluation point r.

The hiding MLE approach is useful to achieve zero-knowledge for a protocol where the final MLE evaluation is revealed in plain, as in the context of our second compiler (see Section 4.1). We shall further use it later on for our zero-knowledge variant of the ring-switching protocol.

Definition 12 (Hiding MLE). *Let ω and k be two positive integers and let $\ell = \lceil \log_2(\omega + k) \rceil$. Let $\sigma \in S_{2^\ell}$ a permutation. For $w \in \mathbb{F}^\omega$ and $m \in \mathbb{F}^k$, a* k-hiding multilinear extension *of w (with mask m under permutation σ) is defined as the multilinear extension $\widetilde{w}^*(X_1, \ldots, X_\ell)$ of $w^* = \sigma(w, m, 0^\nu) \in \mathbb{F}^{2^\ell}$ where $\nu = 2^\ell - (\omega + k)$. For a point $r \in \mathbb{K}^\ell$, let $\mathcal{D}_r(w)$ be the distribution of $\widetilde{w}^*(r)$ when m is uniformly sampled in $\mathbb{F}^k$. We say that r is a* k-hiding point *if $\mathcal{D}_r(w)$ is the uniform distribution over $\mathbb{K}$ for all $w \in \mathbb{F}^\omega$. The following probability is then called the* k-hiding error*:*

$$\varepsilon_{\text{HID}} := \Pr_{r \leftarrow \mathbb{K}^\ell}[r \text{ is not a } k\text{-hiding point}] \ .$$

We stress that the hiding error is defined with respect to $\mathbb{F}$, $\mathbb{K}$, ω, k, σ and does not depend on a specific witness value w.

Remark 11. The goal of the permutation σ is to enable to place the mask m at any specific set of positions $M \subseteq \{0,1\}^\ell$ in the multilinear extension w^*. One can use the identity for σ as a particular case, inducing $M = \{\omega, \omega + 1, \ldots, \omega + k - 1\}$. However, the placement of the mask has an impact on the hiding error ε_{HID}. In particular, we will show that a specific M allows us to make the hiding error ε_{HID} negligible.

Following the above definition, we ensure the zero-knowledge property of the scheme by committing to $w^* = (w, m)$ for a random mask $m \in \mathbb{F}^k$ and we later prove the multilinear evaluation constraints on the hiding MLE $\widetilde{w}^*$ instead of the standard MLE $\widetilde{w}$, which ensures that an MLE evaluation $\widetilde{w}^*(r)$ for a random point r does not leak information about w unless with probability at most ε_{HID} (i.e. unless r is not a hiding point). We show below how to choose the parameters of the scheme to make ε_{HID} negligible.

On the Hiding Error. Recall that the hiding MLE pads the witness w with a uniform mask $m \leftarrow \mathbb{F}^k$ placed at a set of positions $M \subseteq \{0,1\}^\ell$, $|M| = k$, yielding $w^* \in \mathbb{F}^{2^\ell}$ with $w^*_x = m_x$ for $x \in M$. For an evaluation point $r \in \mathbb{K}^\ell$,

$$\widetilde{w}^*(r) \ = \ \underbrace{\sum_{x \notin M} w^*_x \, \chi_x(r)}_{\text{determined by } w} \ + \ \sum_{x \in M} m_x \, \chi_x(r) \ , \qquad \chi_x(r) = \prod_{j=1}^{\ell} \big(x_j r_j + (1 - x_j)(1 - r_j) \big) \ .$$

Here $\chi_x(r) = \widetilde{\text{eq}}(x, r)$ is the Lagrange basis polynomial associated with the hypercube point x. Since the masks m_x lie in $\mathbb{F}$ while the coefficients $\chi_x(r)$ lie in $\mathbb{K}$, the random part ranges exactly over the $\mathbb{F}$-linear

span of $\{\chi_x(r)\}_{x\in M}$. Hence $\widetilde{w}^*(r)$ is uniformly distributed over $\mathbb{K}$ (and the masking perfectly hides $\widetilde{w}(r)$) if and only if

$$\mathrm{Span}_{\mathbb{F}}\big\{\chi_x(r) \;:\; x \in M\big\} \;=\; \mathbb{K}\;, \tag{7}$$

so that the *hiding error* can be redefined as the probability, over a uniformly random evaluation point, that this spanning condition fails:

$$\varepsilon_{\text{HID}} \;:=\; \Pr_{r\leftarrow\mathbb{K}^\ell}\Big[\,\mathrm{Span}_{\mathbb{F}}\{\chi_x(r) : x \in M\} \neq \mathbb{K}\,\Big]\;.$$

This error depends on ℓ, k, $\eta := [\mathbb{K} : \mathbb{F}]$, and the placement M: the masks only supply $\mathbb{F}$-scalars on the k fixed coefficients $\chi_x(r)$, and it is the geometry of these coefficients that determines whether they span $\mathbb{K}$.

Lemma 1. *For every placement M with $|M| = k < \eta$, we have $\varepsilon_{\text{HID}} = 1$. As a consequence, we must have $k \geq \eta$ to achieve a hiding error less than 1.*

Proof. For any r, the span in (7) is generated by k vectors, hence has $\mathbb{F}$-dimension at most $k < \eta = \dim_{\mathbb{F}} \mathbb{K}$, and therefore cannot equal $\mathbb{K}$. The spanning condition fails for every r.

Star Placement. For a suitable placement, k slightly above η already makes the hiding error negligible. We use a "star placement": a center point and $k-1$ arms, each obtained by flipping a distinct coordinate of the center.

Lemma 2 (Hiding Bound). *For $\eta < k \leq \ell + 1$, the star placement is $M^\star = \{p_0, p_1, \ldots, p_{k-1}\}$ where $p_0 = 2^\ell - 1$ (the all-ones point) and $p_i = p_0 - 2^{i-1}$ for $i \in [k-1]$ (the all-ones point with its i-th bit cleared). Then, using the star placement $M^\star$, the hiding error is bounded as*

$$\varepsilon_{\text{HID}} \;\leq\; \frac{4\ell}{|\mathbb{K}|} \;+\; 2\,|\mathbb{F}|^{-(k-\eta+1)}\;.$$

Proof. Let us start by defining the "bad event", denoted Bad, as the event:

$$\mathsf{Bad} \;:=\; \big\{\, r \in \mathbb{K}^\ell \;:\; (\exists\, j \in [\ell],\; r_j = 0) \;\text{ or }\; (\exists\, i \in [k-1],\; r_i = 1)\,\big\}\;.$$

Then we have

$$\varepsilon_{\text{HID}} \;\leq\; \Pr_{r\leftarrow\mathbb{K}^\ell}\Big[\,\mathrm{Span}_{\mathbb{F}}\{\chi_x(r) : x \in M\} \neq \mathbb{K} \;\Big|\; \neg\mathsf{Bad}\,\Big] \;+\; \underbrace{\Pr[\mathsf{Bad}]}_{\leq 2\ell/|\mathbb{K}|}\;.$$

Excluding bad events. Now, given $\neg\mathsf{Bad}$, we have $r_j \neq 0$ for all $j \in [\ell]$ and $r_i \neq 1$ for all $i \in [k-1]$. We can then define $\theta_i = (1 - r_i)/r_i$, satisfying $\theta_i \neq 0$, for $i \in [k-1]$. Since p_i differs from p_0 only in coordinate i, we have $\chi_{p_i}(r) = \chi_{p_0}(r) \cdot \theta_i$ and since $\chi_{p_0}(r) \neq 0$ we get

$$\mathrm{Span}_{\mathbb{F}}\{\chi_{p_i}(r)\}_i = \chi_{p_0}(r) \cdot \mathrm{Span}_{\mathbb{F}}\{1, \theta_1, \ldots, \theta_{k-1}\}\;.$$

As $\chi_{p_0}(r) \neq 0$, multiplication by it is an $\mathbb{F}$-linear automorphism of $\mathbb{K}$ and thus preserves the property of spanning $\mathbb{K}$. So $\{\chi_{p_i}(r)\}_i$ fails to span $\mathbb{K}$ over $\mathbb{F}$ iff $\{1, \theta_1, \ldots, \theta_{k-1}\}$ fails to span $\mathbb{K}$ over $\mathbb{F}$, which gives:

$$\Pr_{r\leftarrow\mathbb{K}^\ell}\Big[\,\mathrm{Span}_{\mathbb{F}}\{\chi_x(r) : x \in M\} \neq \mathbb{K} \;\Big|\; \neg\mathsf{Bad}\,\Big] \;=\; \Pr_{r\leftarrow\mathbb{K}^\ell}\Big[\,\mathrm{Span}_{\mathbb{F}}\{1, \theta_1, \ldots, \theta_{k-1}\} \neq \mathbb{K} \;\Big|\; \neg\mathsf{Bad}\,\Big]\;.$$

Reducing to the uniform case. The θ_i's are independent (each a function of a distinct coordinate r_i), and given $\neg\mathsf{Bad}$ each is uniform over $\mathbb{K} \setminus \{0, -1\}$. Let $\mathcal{D}_{\text{true}}$ be their joint distribution given $\neg\mathsf{Bad}$, and $\mathcal{D}_{\text{unif}}$ the distribution of $(u_1, \ldots, u_{k-1})$ i.i.d. uniform over $\mathbb{K}$. Each marginal has statistical distance $2/|\mathbb{K}|$ from uniform, and since $\mathcal{D}_{\text{true}}$ is a product distribution, $\mathrm{SD}(\mathcal{D}_{\text{true}}, \mathcal{D}_{\text{unif}}) \leq (k-1) \cdot 2/|\mathbb{K}|$. Therefore

$$\Pr_{\{\theta_i\}\leftarrow\mathcal{D}_{\text{true}}}\big[\mathrm{Span}_{\mathbb{F}}\{1, \theta_1, \ldots, \theta_{k-1}\} \neq \mathbb{K}\big] \;\leq\; \Pr_{\{u_i\}\leftarrow\mathcal{D}_{\text{unif}}}\big[\mathrm{Span}_{\mathbb{F}}\{1, u_1, \ldots, u_{k-1}\} \neq \mathbb{K}\big] \;+\; \frac{2(k-1)}{|\mathbb{K}|}\;.$$

Uniform spanning probability. The set $\{1, u_1, \ldots, u_{k-1}\}$ fails to span $\mathbb{K}$ iff all its elements lie in a common $\mathbb{F}$-hyperplane of $\mathbb{K}$. As 1 must lie in it, we union-bound only over the hyperplanes containing 1, of which there are $\frac{|\mathbb{F}|^{\eta-1}-1}{|\mathbb{F}|-1} \leq 2|\mathbb{F}|^{\eta-2}$. Each such hyperplane has $|\mathbb{F}|^{\eta-1}$ elements, so it traps a uniform u_i with probability $1/|\mathbb{F}|$, and all $k-1$ of them with probability $|\mathbb{F}|^{-(k-1)}$. Thus

$$\Pr_{\{u_i\} \leftarrow \mathcal{D}_{\text{unif}}} \left[\text{Span}_{\mathbb{F}}\{1, u_1, \ldots, u_{k-1}\} \neq \mathbb{K}\right] \;\leq\; 2|\mathbb{F}|^{\eta-2} \cdot |\mathbb{F}|^{-(k-1)} \;=\; 2\,|\mathbb{F}|^{-(k-\eta+1)} \,.$$

Combining the three contributions and using $k - 1 \leq \ell$ yields the claim. □

Applicability. The star placement requires $k-1$ distinct coordinates to flip, hence $k \leq \ell + 1$, and the bound of Lemma 2 is only meaningful (negligible) when $k-\eta+1$ is large, i.e. $k > \eta$. The provable regime is therefore $\eta \;<\; k \;\leq\; \ell + 1$, which moreover requires $\eta \leq \ell$. Namely, the extension degree $\eta = [\mathbb{K} : \mathbb{F}]$ must not exceed the log witness size ℓ (with some gap to make $|\mathbb{F}|^{-(k-\eta+1)}$ negligible). While this might not occur for small base fields $\mathbb{F}$ (e.g. $\mathbb{F} = \mathbb{F}_2$), this caveat is mitigated in practice by the fact that we tend to avoid small base fields for our MPCS[3] and we can use packing techniques to group multiple witness entries into elements of a larger base field at the MPCS level (see the *ring-switching* technique in Section 5.2). Moreover, as discussed hereafter, we can live with a non-proven and/or non-negligible hiding error ε_{HID}.

Other placements. Outside this regime – for instance with a very small number of variables ℓ, or when one uses a simple contiguous placement $w^* = (w, m)$ – we do not always have a tight proven bound. Nonetheless, the spanning condition (7) becomes easier to satisfy as k grows, and for any reasonable placement ε_{HID} is expected to decrease quickly as k is increased beyond η; we simply lack a clean closed form away from the star placement. We therefore suggest an alternative approach based on rejection of non-hiding evaluation points.

Hiding by Rejection. An alternative approach enforces perfect hiding by rejection, at the cost of possibly resampling the evaluation point. The key observation is that the spanning condition (7) depends only on the (public) evaluation point r and placement M, not on the witness or the mask: the prover can therefore test it directly, by forming the $\eta \times k$ matrix over $\mathbb{F}$ whose columns are the $\mathbb{F}$-coordinate vectors of $\{\chi_x(r)\}_{x \in M}$ and rejecting r whenever this matrix is not full rank. Conditioned on acceptance, these values span $\mathbb{K}$, so $\widetilde{w}^*(r)$ is *exactly* uniform over $\mathbb{K}$ and the masking is perfectly hiding; since the rejection predicate depends only on public randomness, it also leaks no information about the witness.

In the non-interactive argument, r is derived from the random oracle, so a restart consists of re-randomizing the commitment (fresh mask and fresh commitment randomness), which yields a fresh r; this is exactly the *Fiat–Shamir with aborts* paradigm as used e.g. in Dilithium [Lyu09, DLL+17]. The expected number of repetitions is $1/(1 - \varepsilon_{\text{HID}})$, close to 1 whenever aborts are rare.

However, we can refine the rejection analysis in a particularly useful case. Let us consider $k := \eta$, where $\eta = [\mathbb{K} : \mathbb{F}]$ is the extension degree. As shown in Lemma 1, this is the smallest value of k for which a hiding error strictly smaller than 1 can be achieved. For an integer $0 \leq y \leq 2^\ell$, let us define

$$M_y := \left\{ x = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell : \sum_{j=1}^{\ell} x_j 2^{j-1} < y \right\}.$$

For $i \in \{1, \ldots, \ell\}$ and an integer $0 \leq y \leq 2^i$, let $r^{(i)} := (r_1, \ldots, r_i)$ and define

$$V_i^{[y]} := \text{Span}_{\mathbb{F}} \left\{ \chi_x^{(i)}(r^{(i)}) : x \in M_y \right\}, \qquad \chi_x^{(i)}(r^{(i)}) := \prod_{j=1}^{i} \big(x_j r_j + (1 - x_j)(1 - r_j)\big) \,.$$

Let us take the placement $M := M_\eta$. The spanning condition is thus $V_\ell^{[\eta]} = \mathbb{K}$, *i.e.* $\dim V_\ell^{[\eta]} = \eta$. Instead of only checking that $\dim V_\ell^{[\eta]} = \eta$, we check the spanning condition progressively. The following theorem describe how to obtain a k-hiding point with selective rejection at each sampled challenges $(r_1, \ldots, r_\ell)$.

[3] Our DECS-based MPCS is based on Merkle trees and the base field $\mathbb{F}$ is used to encode the leaves of the Merkle tree (see Section 4.2).

Theorem 6. *Denoting $\eta := [\mathbb{K} : \mathbb{F}]$ and $\sigma := \lceil \log_2(\eta) \rceil$, we have*

- *For all $1 \leq i < \sigma$,*

$$\Pr_{r_i \leftarrow \mathbb{K}}[\dim V_i^{[2^i]} = 2^i \mid \dim V_{i-1}^{[2^{i-1}]} = 2^{i-1}] \geq 1 - \frac{1}{|\mathbb{F}|} - \frac{2}{|\mathbb{K}|} \ ;$$

- *For all $i := \sigma$,*

$$\Pr_{r_i \leftarrow \mathbb{K}}[\dim V_i^{[\eta]} = \eta \mid \dim V_{i-1}^{[2^{i-1}]} = 2^{i-1}] \geq 1 - \frac{(|\mathbb{F}|^{2^{i-1}} - 1) \cdot (|\mathbb{F}|^{\eta - 2^{i-1}} - 1)}{(|\mathbb{F}| - 1) \cdot |\mathbb{K}|} - \frac{2}{|\mathbb{K}|} \approx 1 - \frac{1}{|\mathbb{F}| - 1} \ ;$$

- *For all $\sigma < i \leq \ell$,*

$$\Pr_{r_i \leftarrow \mathbb{K}}[\dim V_i^{[\eta]} = \eta \mid \dim V_{i-1}^{[\eta]} = \eta] = 1 - \frac{1}{|\mathbb{K}|} \ .$$

Proof. Let us first obtain some relations between $V_i^{[y]}$'s:

- Suppose first that $2^{i-1} < y \leq 2^i$. The first 2^{i-1} elements of M_y have $x_i = 0$, while the remaining $y - 2^{i-1}$ elements have $x_i = 1$. Therefore,

$$\begin{aligned} V_i^{[y]} &= \mathrm{Span}_{\mathbb{F}} \left\{ (1 - r_i) \cdot \chi_x^{(i-1)}(r^{(i-1)}) : x \in M_{2^{i-1}} \right\} + \mathrm{Span}_{\mathbb{F}} \left\{ r_i \cdot \chi_x^{(i-1)}(r^{(i-1)}) : x \in M_{y - 2^{i-1}} \right\} \\ &= (1 - r_i) \cdot V_{i-1}^{[2^{i-1}]} + r_i \cdot V_{i-1}^{[y - 2^{i-1}]} \ . \end{aligned} \tag{8}$$

- Suppose now that $0 \leq y \leq 2^{i-1}$. Then $x_i = 0$ for every $x \in M_y$, and hence $\chi_x^{(i)}(r^{(i)}) = (1 - r_i) \cdot \chi_x^{(i-1)}(r^{(i-1)})$. It follows directly that

$$V_i^{[y]} = (1 - r_i) \cdot V_{i-1}^{[y]} \ . \tag{9}$$

The next step consists of establishing the following claim: assuming that $\dim V_{i-1}^{[2^{i-1}]} = 2^{i-1}$, then for all $2^{i-1} < y \leq 2^i$ satisfying $y \leq \eta$,

$$\Pr_{r_i \leftarrow \mathbb{K}}[\dim V_i^{[y]} = y] \geq 1 - \frac{(|\mathbb{F}|^{2^{i-1}} - 1) \cdot (|\mathbb{F}|^{y - 2^{i-1}} - 1)}{(|\mathbb{F}| - 1) \cdot |\mathbb{K}|} + \frac{2}{|\mathbb{K}|} \ .$$

In what follows, we denote $r := r_i$. Since $y - 2^{i-1} \leq 2^{i-1}$, we have that

$$\dim V_{i-1}^{[y - 2^{i-1}]} = y - 2^{i-1}$$

because $V_{i-1}^{[y-2^{i-1}]} \subset V_{i-1}^{[2^{i-1}]}$ (*i.e.* $M_{y-2^{i-1}} \subset M_{2^{i-1}}$).

Let us assume that $\dim V_i^{[y]} < y$. Using Equation (8), we get that there exists $(a, b) \in V_{i-1}^{[2^{i-1}]} \times V_{i-1}^{[y-2^{i-1}]}$ such that $(1 - r) \cdot a = r \cdot b$, with $a \neq 0$ and $b \neq 0$. Denoting

$$R := \left\{ \frac{a}{b} \ : \ a \in V_{i-1}^{[2^{i-1}]}, b \in V_{i-1}^{[y-2^{i-1}]} \text{ with } a \neq 0, b \neq 0 \right\}.$$

We get that either $r \in \{0, 1\}$ which clearly implies $\dim V_i^{[y]} < y$, or $\frac{r}{1-r} \in R$. Since the application $r \mapsto \frac{r}{1-r}$ is a bijective function from $\mathbb{K} \backslash \{1\}$ to $\mathbb{K} \backslash \{-1\}$, we deduce that

$$\Pr_{r \leftarrow \mathbb{K}} \left[\dim V_i^{[y]} < y \right] = \Pr_{r \leftarrow \mathbb{K}} \left[r \in \{0, 1\} \text{ or } \frac{r}{1 - r} \in R \right] \leq \frac{|R| + 2}{|\mathbb{K}|} \ .$$

It remains to estimate the size of R. Given a $r \in R$, the number of couple $(a, b) \in V_{i-1}^{[2^{i-1}]} \times V_{i-1}^{[y-2^{i-1}]}$ such that $(1 - r)a = rb$ is at least $|\mathbb{F}| - 1$, as

$$(1 - r) \underbrace{\gamma \cdot a}_{\in V_{i-1}^{[2^{i-1}]}} = r \underbrace{\gamma \cdot b}_{\in V_{i-1}^{[y-2^{i-1}]}}$$

for all $\gamma \in \mathbb{F}\backslash\{0\}$. Therefore, we get

$$|R| \leq \frac{(|\mathbb{F}|^{2^{i-1}} - 1) \cdot (|\mathbb{F}|^{y-2^{i-1}} - 1)}{|\mathbb{F}| - 1} .$$

As consequence, we get that the desired claim:

$$\Pr_{r \leftarrow \mathbb{K}}[\dim V_i^{[y]} < y] \leq \frac{(|\mathbb{F}|^{2^{i-1}} - 1) \cdot (|\mathbb{F}|^{y-2^{i-1}} - 1)}{(|\mathbb{F}| - 1) \cdot |\mathbb{K}|} + \frac{2}{|\mathbb{K}|}$$

It remains to conclude. When taking $y := 2^i$ with $1 \leq i < \sigma$, the above relation becomes

$$\begin{aligned}\Pr_{r \leftarrow \mathbb{K}}[\dim V_i^{[2^i]} < 2^i \mid \dim V_{i-1}^{[2^{i-1}]} = 2^{i-1}] &\leq \frac{(|\mathbb{F}|^{2^{i-1}} - 1) \cdot (|\mathbb{F}|^{2^{i-1}} - 1)}{(|\mathbb{F}| - 1) \cdot |\mathbb{K}|} + \frac{2}{|\mathbb{K}|} \leq \frac{|\mathbb{F}|^{2^i}}{(|\mathbb{F}| - 1) \cdot |\mathbb{K}|} + \frac{2}{|\mathbb{K}|} \\ &\leq \frac{|\mathbb{F}|^{\eta-1}}{(|\mathbb{F}| - 1) \cdot |\mathbb{K}|} + \frac{2}{|\mathbb{K}|} = \frac{1}{(|\mathbb{F}| - 1) \cdot |\mathbb{F}|} + \frac{2}{|\mathbb{K}|} \leq \frac{1}{|\mathbb{F}|} + \frac{2}{|\mathbb{K}|} ,\end{aligned}$$

leading to the first claim of the theorem. When taking $i := \sigma$ and $y = \eta$, the above relation becomes

$$\Pr_{r \leftarrow \mathbb{K}}[\dim V_i^{[\eta]} < \eta \mid \dim V_{i-1}^{[2^{i-1}]} = 2^{i-1}] \leq \frac{(|\mathbb{F}|^{2^{i-1}} - 1) \cdot (|\mathbb{F}|^{\eta-2^{i-1}} - 1)}{(|\mathbb{F}| - 1) \cdot |\mathbb{K}|} + \frac{2}{|\mathbb{K}|} ,$$

leading to the second claim of the theorem. Finally, using Equation (9), we can observe that the only way that $V_i^{[\eta]} \neq \mathbb{K}$ when $V_{i-1}^{[\eta]} = \mathbb{K}$ is to have $r_i = 1$, occuring with probability $1/|\mathbb{K}|$.

□

Theorem 6 suggests an iterative constructive approach for finding a k-hiding point. For each $1 \leq i < \sigma$, we reject r_i whenever $\dim V_i^{[2^i]} < 2^i$. For $i = \sigma$, we reject r_σ whenever $V_\sigma^{[\eta]} \neq \mathbb{K}$. Finally, for each $\sigma < i \leq \ell$, assuming $V_{i-1}^{[\eta]} = \mathbb{K}$ we have $V_i^{[\eta]} = (1 - r_i) \cdot \mathbb{K}$ (see Equation (9) in the proof of Theorem 6). Hence, it is sufficient to reject $r_i = 1$.

Remark 12. For the binary-field setting, the generic bound of Theorem 6 becomes essentially vacuous at the last doubling step. We therefore additionally evaluate this probability experimentally for $\mathbb{F} = \mathbb{F}_2$ and $\mathbb{K} = \mathbb{F}_{2^{128}}$. More precisely, we independently sample $r_1, \ldots, r_6 \leftarrow \mathbb{K}$ until the conditioning event $\dim V_6^{[64]} = 64$ is satisfied. For each such prefix, we then sample independent $r_7 \leftarrow \mathbb{K}$ and explicitly compute

$$\dim V_7^{[128]} = \dim\big((1 - r_7)V_6^{[64]} + r_7 V_6^{[64]}\big)$$

by expressing the corresponding 128 generators as vectors over $\mathbb{F}_2$ and computing their rank. Repeating this experiment provides a Monte-Carlo estimate of

$$\Pr_{r_7 \leftarrow \mathbb{K}}\left[\dim V_7^{[128]} < 128 \,\middle|\, \dim V_6^{[64]} = 64\right].$$

We performed the experiment for 20 independently sampled prefixes $(r_1, \ldots, r_6)$, using 5 000 independent samples of r_7 for each prefix, for a total of 100 000 trials. The observed rejection probabilities for the individual prefixes range from 0.6980 to 0.7232, and are consistently concentrated around 0.71. Overall, 71 196 out of the 100 000 sampled challenges were rejected, yielding the estimate

$$\widehat{p}_{\text{rej}} = 0.71196.$$

Treating the 100 000 trials as Bernoulli samples gives an approximate 95% confidence interval

$$0.70915 \leq p_{\text{rej}} \leq 0.71477.$$

These experiments therefore indicate that, although the generic upper bound of Theorem 6 is essentially vacuous in the binary-field setting, the actual rejection probability at the last doubling step is approximately 71%. Equivalently, a uniformly sampled r_7 satisfies the spanning condition with probability approximately $1 - \widehat{p}_{\text{rej}} = 0.28804$. Moreover, the relatively small variation observed across the 20 independently sampled prefixes suggests that this probability is not driven by a small number of atypical choices of $V_6^{[64]}$.

The rejection criterion depends only on public values. This observation is essential: in a non-interactive setting where the challenges are derived using a hash function, whenever a rejection occurs, the prover can simply derive a new challenge by calling the hash function with an incremented counter. More precisely, a challenge can be derived as follows:

1: $\texttt{ctr} \leftarrow 0$
2: `restart:`
3: $r_i \leftarrow \mathsf{Hash}_i(\text{previous prover messages} \,\|\, \texttt{ctr})$
4: **if** r_i leads to a rejection **then**
5: $\quad \texttt{ctr} \leftarrow \texttt{ctr} + 1$
6: $\quad$ `goto restart`

The counter `ctr` is then included in the proof transcript. The verifier can therefore rederive r_i from the transcript and check that the resulting challenge does not lead to a rejection.

This rejection mechanism does not provide any advantage to a malicious prover attempting to forge a valid proof for an invalid statement. Indeed, excluding some challenge values does not make cheating any easier. Although the challenge space is reduced by a factor $1 - p_{\text{rej}}$, where p_{rej} denotes the rejection probability, the probability that a Fiat–Shamir hash query yields a challenge that breaks soundness remains unchanged. Consequently, the computational effort required to forge a valid proof is not reduced.

For an honest prover, rejection only introduces a small computational overhead due to the additional hash calls, which is negligible compared with the overall proving time.

Remark 13. A comparable rejection-based strategy is used by NIST candidates based on MPC-in-the-Head [BBB+24, ABB+24b, AAB+24, ABB+24c, ABB+24a] that employ the *one-tree technique* [BBM+24], though there rejection is used as a performance optimization rather than for hiding.

Remark 14. Rejection sampling of leaking evaluation points was also used in [FKR26], but in a setting where the witness lives directly over the cryptographic-size field $\mathbb{K}$, making the rejection probability negligible and the analysis straightforward. In our small-field setting, the rejection probability can instead be significant (*e.g.* 71% over $\mathbb{F}_2$), which is why we resample only the current challenge (rather than aborting the whole protocol) and build the hiding point incrementally, coordinate by coordinate.

For our concrete zero-knowledge argument systems reported in Section 6, we use the above rejection-based approach, as it minimizes the size of the mask and is particularly convenient in a non-interactive setting. Nevertheless, we believe that the alternative approach remains interesting in contexts where rejection is undesirable, for instance in an interactive proof setting.

5.2 Zero-Knowledge Ring-Switching Protocol

Let us assume that the commitment scheme can only commit to field elements over a sufficiently large field. While this is indeed the case for the commitment schemes considered in the next section, the same limitation occurs in many existing commitment schemes. This becomes problematic when the statement is naturally defined over a small field, such as the binary field. One could embed each small-field element into a sufficiently large extension field, but doing so introduces a significant communication overhead and makes the overall proof system less attractive for binary fields.

The *ring-switching protocol* of [DP26] addresses this issue: it reduces an evaluation claim on the MLE of a vector over a small field $\mathbb{F}$ to an evaluation claim on the MLE of its *packed* representation over an extension field $\mathbb{K}$, in which every block of $[\mathbb{K} : \mathbb{F}]$ consecutive coordinates is bundled into a single $\mathbb{K}$-element. This protocol is not zero-knowledge, however: its transcript leaks partial information on the committed vector. We therefore introduce a zero-knowledge variant of ring-switching, which achieves the same reduction while revealing nothing beyond the resulting evaluation claim. We further generalize it to a three-field setting as described below.

Three-Field Ring-Switching Protocol. The original ring-switching protocol of [DP26] considers two fields: a small field $\mathbb{F}$ over which the original witness is defined, and a larger field $\mathbb{K}$ in which both the MLE evaluation point (*i.e.* the sum-check challenges) and the commitment scheme live. In our setting, we separate the two latter roles and introduce an intermediate field $\mathbb{I}$ for the commitment scheme. More precisely, we consider:

– a small field $\mathbb{F}$, over which the original witness is defined (typically $\mathbb{F} = \mathbb{F}_2$ or another small field);
– an intermediate extension field $\mathbb{I} \supseteq \mathbb{F}$, which is the base field of the commitment scheme (in our constructions, $|\mathbb{I}| = N$ or is slightly larger, where N is the size of the Merkle tree underlying the commitment);
– a larger extension field $\mathbb{K} \supseteq \mathbb{I}$, over which the MLE evaluation point is sampled. This field is of cryptographic size in order to make the soundness error negligible (i.e. $|\mathbb{K}| = 2^\lambda$ or slightly larger where λ is the security parameter).

Let us denote by ι and κ the integers such that $[\mathbb{I} : \mathbb{F}] = 2^\iota$ and $[\mathbb{K} : \mathbb{F}] = 2^\kappa$, assuming power-of-2 extension degrees. We refer to Remark 15 for the general case of arbitrary extension degrees. Let $(\beta_v^{\mathbb{I}})_{v\in\{0,1\}^\iota}$ be an $\mathbb{F}$-basis of $\mathbb{I}$, and let $(\beta_u^{\mathbb{K}})_{u\in\{0,1\}^\kappa}$ be an $\mathbb{F}$-basis of $\mathbb{K}$. Consider a witness vector $w \in \mathbb{F}^\omega$ of length ω and denote $\ell = \lceil \log_2 \omega \rceil$. Let $\omega' := \lceil \omega/2^\iota \rceil$ and $\ell' := \lceil \log_2 \omega' \rceil = \ell - \iota$.

The goal of the ring-switching protocol is to reduce an evaluation claim

$$\widetilde{w}(r) = s \quad \text{for} \quad w \in \mathbb{F}^\omega,\, r \in \mathbb{K}^\ell \text{ and } s \in \mathbb{K}$$

into an evaluation claim on a *packed witness*, namely a claim of the form

$$\widetilde{w}_{\text{packed}}(r') = s' \quad \text{for} \quad w_{\text{packed}} \in \mathbb{I}^{\omega'},\, r' \in \mathbb{K}^{\ell'} \text{ and } s' \in \mathbb{K}\ .$$

For all $v \in \{0,1\}^\iota$, let $w_v \in \mathbb{F}^{\omega'}$ denote the subvector of w, such that $\widetilde{w}(v, X) = \widetilde{w}_v(X)$. The packed witness $w_{\text{packed}} \in \mathbb{I}^{\omega'}$ is defined as

$$w_{\text{packed}} := \sum_{v\in\{0,1\}^\iota} w_v \cdot \beta_v^{\mathbb{I}}\ .$$

Namely, each coordinate of w_{packed} is obtained by packing of 2^ι coordinates of w into an element of $\mathbb{I}$. In our constructions, such a reduction enables to apply the commitment to the packed witness w_{packed} over the intermediate field $\mathbb{I}$, rather than to the witness over the small field $\mathbb{F}$.

Let us decompose the evaluation point r as:

$$r = (r_{\text{head}}, r_{\text{tail}}) \qquad r_{\text{head}} \in \mathbb{K}^\iota, \quad r_{\text{tail}} \in \mathbb{K}^{\ell'}\ .$$

For each $x \in \{0,1\}^{\ell'}$, we further decompose the equality MLE evaluation $\widetilde{\text{eq}}(r_{\text{tail}}, x)$ in the $\mathbb{F}$-basis of $\mathbb{K}$ as

$$\widetilde{\text{eq}}(r_{\text{tail}}, x) = \sum_{u\in\{0,1\}^\kappa} A_{x,u}\beta_u^{\mathbb{K}}, \qquad A_{x,u} \in \mathbb{F}.$$

For each $u \in \{0,1\}^\kappa$, let $\widetilde{A}_u \in \mathbb{F}[X_1, \ldots, X_{\ell'}]$ denote the MLE of the vector $(A_{x,u})_{x\in\{0,1\}^{\ell'}}$.

The three-field ring-switching protocol is described in Figure 11. The prover first provides partial evaluations $(\hat{s}_v)_v$, corresponding to the MLE evaluations $\widetilde{w}(v, r_{\text{tail}})$ of the 2^ι subvectors of w for $v \in \{0,1\}^\iota$. This reduces the main evaluation claim to evaluation claims on these partial evaluations. Since the latter are elements of $\mathbb{K}$, they can be decomposed over $\mathbb{F}$ using the basis $(\beta_u^{\mathbb{K}})_{u\in\{0,1\}^\kappa}$, yielding a $2^\kappa \times 2^\iota$ matrix of $\mathbb{F}$-coordinates. Reading this matrix row-wise and packing each row over $\mathbb{I}$ transforms these claims into linear claims directly on $\widetilde{w}_{\text{packed}}$. Finally, these linear claims are randomly batched over $\mathbb{K}$ and reduced, using the sum-check protocol, to a single evaluation claim on $\widetilde{w}_{\text{packed}}$. This evaluation claim involves Λ^{-1} and therefore requires the protocol to abort whenever $\Lambda = 0$, unless the relation can instead be propagated in the form $s' \stackrel{?}{=} \Lambda \cdot \widetilde{w}_{\text{packed}}(r')$. This event occurs with probability at most $\ell'/|\mathbb{K}|$. The soundness analysis readily extends from the two-field case.

Remark 15. For simplicity, we assumed above that the extension degrees were powers of two. The protocol naturally generalizes to arbitrary extension degrees. Let $[\mathbb{I} : \mathbb{F}] \le 2^\iota$ and $[\mathbb{K} : \mathbb{F}] \le 2^\kappa$. Instead of packing 2^ι subvectors, we only pack $[\mathbb{I} : \mathbb{F}]$ subvectors and define $\widetilde{w}(z, X)$ to be zero for the remaining indices $v \in \{0,1\}^\iota$. More precisely, only the indices $v = (v_1, \ldots, v_\iota) \in \{0,1\}^\iota$ such that $\sum_j v_j \cdot 2^{j-1} < [\mathbb{I} : \mathbb{F}]$ correspond to actual subvectors. Similarly, when indexing the $\mathbb{F}$-basis of $\mathbb{K}$ by $\{0,1\}^\kappa$, only the indices $u = (u_1, \ldots, u_\kappa) \in \{0,1\}^\kappa$ such that $\sum_{j=1} u_j \cdot 2^{j-1} < [\mathbb{K} : \mathbb{F}]$ correspond to basis elements. The protocol naturally extends by treating the remaining indices as zero.

$\underline{\mathcal{P}(w, r, s)}$ $\underline{\mathcal{V}(r, s)}$

For each $v \in \{0,1\}^\iota$,

set $\hat{s}_v := \widetilde{w}(v, r_{\text{tail}}) \in \mathbb{K}$. $\xrightarrow{(\hat{s}_v)_{v\in\{0,1\}^\iota}}$ Check $s \overset{?}{=} \sum_{v\in\{0,1\}^\iota} \widetilde{\text{eq}}(v, r_{\text{head}})\, \hat{s}_v$.

For each $v \in \{0,1\}^\iota$, decompose

$$\hat{s}_v = \sum_{u\in\{0,1\}^\kappa} \hat{s}_{u,v}\beta^{\mathbb{K}}_u, \qquad \hat{s}_{u,v} \in \mathbb{F}.$$

For each $u \in \{0,1\}^\kappa$, combine

$$\hat{\hat{s}}_u := \sum_{v\in\{0,1\}^\iota} \hat{s}_{u,v}\beta^{\mathbb{I}}_v \in \mathbb{I}.$$

$\xleftarrow{r''}$ Sample $r'' \leftarrow \mathbb{K}^\kappa$.

$\mathcal{P}$ and $\mathcal{V}$ run the sum-check protocol on the claim:

$$\sum_{u\in\{0,1\}^\kappa} \widetilde{\text{eq}}(u, r'')\, \hat{\hat{s}}_u \overset{?}{=} \sum_{x\in\{0,1\}^{\ell'}} \left(\sum_{u\in\{0,1\}^\kappa} \widetilde{\text{eq}}(u, r'')\, \widetilde{A}_u(x) \right) \widetilde{w}_{\text{packed}}(x)\ .$$

At the end this sum-check protocol, the claim is reduced to the following evaluation claim:

$$s' \cdot \Lambda^{-1} \overset{?}{=} \widetilde{w}_{\text{packed}}(r')\ , \qquad \text{with}\ \ \Lambda := \sum_{u\in\{0,1\}^\kappa} \widetilde{\text{eq}}(u, r'')\, \widetilde{A}_u(r') \in \mathbb{K}\ , \tag{10}$$

where $r' \in \mathbb{K}^{\ell'}$ is the sum-check challenge evaluation point and s' is the final value claimed by the $\mathcal{P}$. We note that $\mathcal{P}$ and $\mathcal{V}$ can compute the Λ term efficiently so that the claim is a standard evaluation claim on $\widetilde{w}_{\text{packed}}(r')$.

Fig. 11: Three-field ring-switching protocol.

Soundness of the three-field ring-switching protocol. The security proof of [DP26] readily extends to the three-field setting, as its security arguments are agnostic to the particular fields involved. The soundness of the construction is established in [DP26] by showing that combining the ring-switching protocol with a secure extractable MPCS Com over the large field yields a secure extractable MPCS Com' over the small field. In the terminology of Definition 9, Com' inherits the state function, the extractor (whose output is unpacked) and the error $\varepsilon_{\mathcal{H}}$ of Com, and its per-round error is the maximum of that of Com, of the per-round soundness error of the sum-check and of the batching error $\kappa/|\mathbb{K}|$ appearing below; we only sketch the argument in the extractable-MPCS language of [DP26].

The extractor of Com' first invokes the extractor of Com and then returns the unpacked version of the extracted MLE. Hence, the extraction step fails with the same probability as for Com, which is negligible by the extractability of Com. It remains to show that an accepted Com' opening proof is consistent with the extracted MLE.

Let (r, s) denote the opening claim, and let $\widetilde{w}_{\text{packed}}$ and $\widetilde{w}$ denote, respectively, the extracted packed MLE and its unpacked version. We denote by $(r'', r', s', (\hat{s}_v)_v, (\hat{\hat{s}}_u)_u)$ the quantities appearing in the

opening transcript. By the security of the underlying MPCS, we have

$$\widetilde{w}_{\text{packed}}(r') = s' \cdot \left(\sum_{u \in \{0,1\}^\kappa} \widetilde{\text{eq}}(u, r'')\, \widetilde{A}_u(r') \right)^{-1} \tag{11}$$

except with negligible probability. We now bound the probability that an opening proof is accepted while $\widetilde{w}(r) \neq s$.

Let $\bar{s}_v := \widetilde{w}(v, r_{\text{tail}})$ for every $v \in \{0,1\}^\iota$. If the first prover message $(\hat{s}_v)_v$ were consistent with $\widetilde{w}$, namely if $\hat{s}_v = \bar{s}_v$ for every v, then $\widetilde{w}(r) = \sum_{v \in \{0,1\}^\iota} \widetilde{\text{eq}}(v, r_{\text{head}})\, \hat{s}_v$. By the verifier's consistency check, the right-hand side is equal to s, contradicting our assumption that $\widetilde{w}(r) \neq s$. Therefore, there exists at least one v such that $\hat{s}_v \neq \bar{s}_v$.

Let $\bar{s}_v = \sum_{u \in \{0,1\}^\kappa} \bar{s}_{u,v} \beta_u^{\mathbb{K}}$ denote the $\mathbb{F}$-decomposition of $\bar{s}_v$, and define its recombination over the $\mathbb{I}$-basis as $\bar{\bar{s}}_u := \sum_{v \in \{0,1\}^\iota} \bar{s}_{u,v} \beta_v^{\mathbb{I}}$. Since there exists a v such that $\bar{s}_v \neq \hat{s}_v$, uniqueness of the $\mathbb{F}_0$-decomposition implies that the corresponding coordinate matrices differ. Since recombination over the $\mathbb{F}_0$-basis $(\beta_v^{\mathbb{I}})_v$ is injective, there consequently exists a u such that $\bar{\bar{s}}_u \neq \hat{\bar{s}}_u$. Therefore, by the Schwartz–Zippel lemma applied to the corresponding multilinear polynomials,

$$\sum_{u \in \{0,1\}^\kappa} \hat{\bar{s}}_u\, \widetilde{\text{eq}}(u, r'') \neq \sum_{u \in \{0,1\}^\kappa} \bar{\bar{s}}_u\, \widetilde{\text{eq}}(u, r'') \tag{12}$$

with probability at least $1 - \kappa/|\mathbb{K}|$ over the random choice of r''.

It remains to express $(\bar{\bar{s}}_u)_u$ in terms of the extracted packed MLE $\widetilde{w}_{\text{packed}}$. For every $v \in \{0,1\}^\iota$, we have

$$\begin{aligned}
\bar{s}_v = \widetilde{w}(v, r_{\text{tail}}) &= \sum_{x \in \{0,1\}^{\ell'}} \widetilde{\text{eq}}(x, r_{\text{tail}}) \cdot \widetilde{w}(v, x) \\
&= \sum_{x \in \{0,1\}^{\ell'}} \left(\sum_{u \in \{0,1\}^\kappa} A_{x,u} \beta_u^{\mathbb{K}} \right) \cdot \widetilde{w}(v, x) = \sum_{u \in \{0,1\}^\kappa} \underbrace{\left(\sum_{x \in \{0,1\}^{\ell'}} A_{x,u} \cdot \widetilde{w}(v, x) \right)}_{\in \mathbb{F}} \cdot \beta_u^{\mathbb{K}}
\end{aligned}$$

By uniqueness of the $\mathbb{F}$-decomposition, it follows that

$$\bar{s}_{u,v} = \sum_{x \in \{0,1\}^{\ell'}} A_{x,u} \cdot \widetilde{w}(v, x)$$

for every $u \in \{0,1\}^\kappa$ and $v \in \{0,1\}^\iota$. Hence, for every $u \in \{0,1\}^\kappa$,

$$\begin{aligned}
\bar{\bar{s}}_u = \sum_{v \in \{0,1\}^\iota} \bar{s}_{u,v} \beta_v^{\mathbb{I}} &= \sum_{v \in \{0,1\}^\iota} \sum_{x \in \{0,1\}^{\ell'}} A_{x,u} \cdot \widetilde{w}(v, x) \cdot \beta_v^{\mathbb{I}} \\
&= \sum_{x \in \{0,1\}^{\ell'}} A_{x,u} \cdot \left(\sum_{v \in \{0,1\}^\iota} \widetilde{w}(v, x) \cdot \beta_v^{\mathbb{I}} \right) = \sum_{x \in \{0,1\}^{\ell'}} A_{x,u} \cdot \widetilde{w}_{\text{packed}}(x)
\end{aligned}$$

Combining this identity with Equation (12), we obtain

$$\sum_{u \in \{0,1\}^\kappa} \hat{\bar{s}}_u\, \widetilde{\text{eq}}(u, r'') \neq \sum_{x \in \{0,1\}^{\ell'}} \left(\sum_{u \in \{0,1\}^\kappa} \widetilde{A}_u(x)\, \widetilde{\text{eq}}(u, r'') \right) \widetilde{w}_{\text{packed}}(x).$$

since A_u is the multilinear extension of $x \mapsto A_{x,u}$. Thus, the claim given as input to the underlying sum-check protocol is false. By the soundness of the sum-check protocol, except with its soundness error, the resulting reduced evaluation claim cannot be consistent with the actual evaluation of the extracted polynomial $\widetilde{w}_{\text{packed}}$. In particular, it cannot lead to Equation (11), except with negligible probability.

Therefore, an accepted opening proof with $\widetilde{w}(r) \neq s$ occurs only with the sum of the negligible extraction and MPCS errors, the sum-check soundness error, and the batching error at most $\kappa/|\mathbb{K}|$. This proves the soundness of the three-field construction.

Zero-Knowledge Ring-Switching. The ring-switching protocol described above is not zero-knowledge. It reveals two kinds of witness-dependent information: the partial evaluations $(\hat{s}_v)_{v\in\{0,1\}^\iota}$ sent to the verifier, and the transcript of the sum-check protocol.

A natural idea would be to apply the CBIP paradigm formalized in Section 3 to this protocol to achieve zero-knowledge. Unfortunately, this protocol cannot be directly translated into the CBIP paradigm. This is because of the $\mathbb{F}$-decomposition of each partial evaluation $\hat{s}_v \in \mathbb{K}$ which must be performed by the verifier. Expressed as a polynomial function over $\mathbb{K}$, such a decomposition is of high degree (specifically of degree $|\mathbb{F}|^{2^\kappa-1}$) and hence the trace constraints would inherit this high degree, making the construction impractical.

To overcome this issue, we do not mask the trace messages consisting of the partial evaluations $(\hat{s}_v)_v$ but we reveal them in plain. To avoid any information leakage, we rely on the hiding MLE construction introduced in the previous section. Because we must reveal $\hat{s}_v = \widetilde{w}(v, r_{\text{tail}})$ for each $v \in \{0,1\}^\iota$, we use one mask $m_v \in \mathbb{F}^{2^\kappa}$ yielding one hiding MLE $\widetilde{w}^*(v, X) = \widetilde{w}^*_v(X)$ for each v, where w^*_v is the extension of w_v with the mask m_v. We use a fixed mask placement M within each hiding MLE. Namely, M is the set of hypercube points at which the coefficients of m_v are encoded in $\widetilde{w}^*(v, X)$, for every $v \in \{0,1\}^\iota$. In this context, if an evaluation point r_{tail} is 2^κ-hiding with respect to M for the hiding MLE $\widetilde{w}^*(v, X)$ for some $v \in \{0,1\}^\iota$, then it is 2^κ-hiding for all the other $v \in \{0,1\}^\iota$ as well.

Formally, our zero-knowledge ring-switching protocol reduces an SVCS instance

$$\mathbb{x}_{\mathbb{F}} = (\mathbb{F}, \mathbb{K}, \omega + 2^\iota \cdot 2^\kappa, \tau, d, \mathcal{E}, \mathcal{F})$$

with witness $(w_v, m_v)_{v\in\{0,1\}^\iota} \in \mathbb{F}^{2^\iota(2^{\ell'}+2^\kappa)}$ into an SVCS instance

$$\mathbb{x}_{\mathbb{I}} = (\mathbb{I}, \mathbb{K}, \omega' + 2^\kappa, \tau, d, \mathcal{E}', \mathcal{F}')$$

with witness $(w_{\text{packed}}, m_{\text{packed}}) \in \mathbb{I}^{\omega'} \times \mathbb{I}^{2^\kappa}$, where

$$w_{\text{packed}} = \sum_{v\in\{0,1\}^\iota} \beta^{\mathbb{I}}_v \cdot w_v \quad \textit{and} \quad m_{\text{packed}} = \sum_{v\in\{0,1\}^\iota} \beta^{\mathbb{I}}_v \cdot m_v \ .$$

Let $\ell^* := \lceil \log_2(\omega' + 2^\kappa) \rceil$. We assume that the incoming SVCS instance over $\mathbb{F}$ has a single multilinear constraint $\widetilde{w}^*(r) = \psi(t)$, namely $\mathcal{E} = \{(\psi, r)\}$. This is without loss of generality as we can always combine multiple multilinear constraints into a single one as described in Section 3.2. We moreover assume that, writing $r = (r_{\text{head}}, r_{\text{tail}})$ with $r_{\text{head}} \in \{0,1\}^\iota$ and $r_{\text{tail}} \in \{0,1\}^{\ell^*}$, the point r_{tail} is hiding for the hiding MLE $\widetilde{w}^*(v, X)$ for every $v \in \{0,1\}^\iota$. This either holds with overwhelming probability over the random choice of r or is achieved by construction using the rejection approach described in Section 5.1.

The prover and the verifier execute the ring-switching protocol on the claim $\tilde{w}^*(r) = \psi(t)$. The partial evaluations $(\hat{s}_v)_{v\in\{0,1\}^\iota}$ are revealed in the clear, while the final sum-check protocol of the ring-switching protocol is executed in the CBIP paradigm. In the original setting, the verifier starts by verifying that the partial evaluations $(\hat{s}_v)_{v\in\{0,1\}^\iota}$ match with the claimed value s. Here this translates to recomputing s from the $(\hat{s}_v)_v$ on the verifier side by

$$s = \sum_{v\in\{0,1\}^\iota} \widetilde{\text{eq}}(v, r_{\text{head}})\, \hat{s}_v$$

and adding the linear trace constraint $\psi(t) - s = 0$ to the SVCS instance. Then, because the partial evaluations $(\hat{s}_v)_{v\in\{0,1\}^\iota}$ are sent in clear, the values $\{\hat{s}_u\}_u$ defining the sum-check statement are public. The execution of the sum-check protocol in the CBIP paradigm results in a new evaluation claim of the form

$$\Lambda \cdot \widetilde{w}^*_{\text{packed}}(r') = \psi'(t)$$

where r' is the evaluation point drawn by the verifier during this execution, ψ' is the linear function defining the new evaluation claim, and Λ is the related public value defined in Figure 11. This execution of the sum-check protocol consumes $2\ell^*$ elements of the trace mask. The constraints of the obtained SVCS instance are then simply defined as:

$$\mathcal{E}' = \{(r', \Lambda^{-1} \cdot \psi')\} \ , \quad \mathcal{F}' = \mathcal{F} \cup \{t \mapsto \psi(t) - s\} \ .$$

The overall zero-knowledge ring-switching protocol is described in Figure 12.

Context:

- $\mathcal{P}$ has previously committed to the packed witness $w^*_{\text{packed}} = (w_{\text{packed}}, m_{\text{packed}}) \in \mathbb{I}^{\omega' + 2^\kappa}$ and to a trace mask $t \in \mathbb{K}^\tau$, and holds the underlying small-field witness $(w_v, m_v)_{v \in \{0,1\}^\iota}$ over $\mathbb{F}$.
- $\mathcal{P}$ and $\mathcal{V}$ hold an SVCS instance $\mathbb{x}_{\mathbb{F}} = (\mathbb{F}, \mathbb{K}, \omega + 2^\iota \cdot 2^\kappa, \tau, d, \mathcal{E}, \mathcal{F})$ whose single evaluation constraint is $\widetilde{w}^*(r) = \psi(t)$, with $r = (r_{\text{head}}, r_{\text{tail}}) \in \mathbb{K}^\iota \times \mathbb{K}^{\ell^*}$ and $\ell^* := \lceil \log_2(\omega' + 2^\kappa) \rceil$ the number of variables of each masked subvector $\widetilde{w}^*(v, X)$.
- The point r_{tail} is a 2^κ-hiding point with respect to the placement M (see Section 5.1).
- For $x \in \{0,1\}^{\ell^*}$ and $u \in \{0,1\}^\kappa$, the coefficients $A_{x,u} \in \mathbb{F}$ are defined by $\widetilde{\text{eq}}(r_{\text{tail}}, x) = \sum_u A_{x,u}\, \beta^{\mathbb{K}}_u$, and $\widetilde{A}_u$ denotes the MLE of $(A_{x,u})_{x \in \{0,1\}^{\ell^*}}$.
- The trace mask t includes $2\,\ell^*$ elements dedicated to the sum-check execution below, so that the outgoing instance has trace size τ as well.
- At the end of the protocol, the claim is reduced to an SVCS instance $\mathbb{x}_{\mathbb{I}} = (\mathbb{I}, \mathbb{K}, \omega' + 2^\kappa, \tau, d, \mathcal{E}', \mathcal{F}')$ on the witness (w^*_{packed}, t), as defined below.

Protocol:

- $\mathcal{P}$ sends the partial evaluations $\hat{s}_v = \widetilde{w}^*(v, r_{\text{tail}}) \in \mathbb{K}$, for $v \in \{0,1\}^\iota$, in the clear.
- $\mathcal{V}$ computes
$$s = \sum_{v \in \{0,1\}^\iota} \widetilde{\text{eq}}(v, r_{\text{head}})\, \hat{s}_v$$
and $\mathcal{P}$ and $\mathcal{V}$ add the linear trace constraint $t \mapsto \psi(t) - s$ to $\mathcal{F}'$, which enforces the consistency of the revealed partial evaluations with the incoming claim.
- $\mathcal{P}$ and $\mathcal{V}$ decompose each partial evaluation over $\mathbb{F}$ and repack the resulting coefficients over $\mathbb{I}$:
$$\hat{s}_v = \sum_{u \in \{0,1\}^\kappa} \hat{s}_{u,v}\, \beta^{\mathbb{K}}_u \;\text{ with }\; \hat{s}_{u,v} \in \mathbb{F}\,, \qquad \hat{\hat{s}}_u := \sum_{v \in \{0,1\}^\iota} \hat{s}_{u,v}\, \beta^{\mathbb{I}}_v \;\in\; \mathbb{I}\,.$$
- $\mathcal{V}$ samples $r'' \leftarrow \mathbb{K}^\kappa$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ and $\mathcal{V}$ run the sum-check protocol in the CBIP paradigm (Figure 5) on the claim
$$\sum_{x \in \{0,1\}^{\ell^*}} g(x) \;=\; \hat{c}\,, \quad \text{where} \quad g(x) := \Big(\sum_{u \in \{0,1\}^\kappa} \widetilde{\text{eq}}(u, r'')\, \widetilde{A}_u(x) \Big) \cdot \widetilde{w}^*_{\text{packed}}(x)\,,$$
the claimed value $\hat{c} := \sum_u \widetilde{\text{eq}}(u, r'')\, \hat{\hat{s}}_u$ being public – so that the initial affine map of the execution is the constant map $t \mapsto \hat{c}$. At the end of this execution, the claim is reduced to $g(r') = \psi'(t)$ for a random $r' \in \mathbb{K}^{\ell^*}$ chosen by the verifier and a public affine map $\psi' : \mathbb{K}^\tau \to \mathbb{K}$.
- $\mathcal{P}$ and $\mathcal{V}$ compute the public scalar
$$\Lambda \;:=\; \sum_{u \in \{0,1\}^\kappa} \widetilde{\text{eq}}(u, r'')\, \widetilde{A}_u(r') \;\in\; \mathbb{K}$$
and, provided $\Lambda \neq 0$ (otherwise the protocol aborts), set $\psi'' := \Lambda^{-1} \cdot \psi'$, so that the claim $g(r') = \psi'(t)$ rewrites as the multilinear evaluation constraint $\widetilde{w}^*_{\text{packed}}(r') = \psi''(t)$.
- $\mathcal{P}$ and $\mathcal{V}$ output the SVCS instance $\mathbb{x}_{\mathbb{I}}$ with
$$\mathcal{E}' = \big\{(r', \psi'')\big\} \qquad \text{and} \qquad \mathcal{F}' = \mathcal{F} \cup \big\{t \mapsto \psi(t) - s\big\}\,.$$

Fig. 12: Zero-knowledge ring-switching protocol, from an SVCS instance over $\mathbb{F}$ to an SVCS instance over $\mathbb{I}$.

Remark 16. To avoid inverting Λ and the need to abort when $\Lambda = 0$, the prover can instead reveal a masked version of $\widetilde{w}^*_{\text{packed}}(r')$, prove in zero knowledge the Λ-dependent linear relation between s' and $\widetilde{w}^*_{\text{packed}}(r')$, and define $\mathcal{E}'$ to enforce consistency with the publicly revealed masked evaluation. This comes at the cost of only one additional $\mathbb{K}$-element in the trace.

Theorem 7. *Assume that each prover message in the CBIP sum-check protocol is masked using distinct elements of the trace mask, that these elements are sampled uniformly at random, and that the partial evaluation point r_{tail} is a 2^κ-hiding point with respect to the placement M. Then, the protocol of Figure 12 is perfectly zero-knowledge.*

Proof. The sum-check prover messages are perfectly one-time-pad masked by independent uniformly random elements of the trace mask and therefore leak no information about the witness. The only other possible source of leakage consists of the values $\{\hat{s}_v\}_{v\in\{0,1\}^\iota}$ revealed *in the clear* during the ring-switching protocol.

For every $v \in \{0,1\}^\iota$, $\widetilde{w}^*(v, X)$ is a 2^κ-hiding multilinear extension of w_v, with mask m_v. For any point $r \in \mathbb{K}^{\ell^*}$, let $\mathcal{D}_r(w_v)$ denote the distribution of $\widetilde{w}^*(v, r)$ when m_v is sampled uniformly at random from $\mathbb{F}^{2^\kappa}$. Since r_{tail} is a 2^κ-hiding point with respect to the placement M, we have $\mathcal{D}_{r_{\text{tail}}}(w_v) = \mathcal{U}(\mathbb{K})$ for every $w_v \in \mathbb{F}^{\omega'}$ and every $v \in \{0,1\}^\iota$, where $\mathcal{U}(\mathbb{K})$ denotes the uniform distribution over $\mathbb{K}$. Consequently, for every $v \in \{0,1\}^\iota$, the partial evaluation $\hat{s}_v := \widetilde{w}^*(v, r_{\text{tail}})$ is uniformly distributed over $\mathbb{K}$, independently of w_v. Moreover, since the masks $(m_v)_{v\in\{0,1\}^\iota}$ are sampled independently, the partial evaluations $(\hat{s}_v)_{v\in\{0,1\}^\iota}$ are jointly uniform and independent of the witness. Revealing them therefore leaks no information about the witness.

In conclusion, neither the masked sum-check transcripts nor the revealed partial evaluations leak any information about the witness, which proves the claim. □

5.3 Circuit Flattening for GKR

The GKR protocol (see Section 2.2) provides particularly small proofs for low-depth circuits. Specifically, the proof size is logarithmic in the circuit width while being linear in the circuit depth. For layered circuits with a large depth, the proof size can become significant. To address this issue, a folklore idea is to flatten the circuit by committing to intermediate layers: given a circuit with d layers, split them into K blocks, each of depth at most $\lceil d/K \rceil$, where K is a design parameter trading circuit depth against witness size. A concrete instantiation of this idea is presented in [Lon26]. We adapt this idea in our setting and further improve it.

Consider a layered circuit C with d layers, and split it into K blocks of layers, as:

$$y = C(x) = C_K \circ \ldots \circ C_2 \circ C_1(x) \ .$$

Let us further denote by $z_i = C_i(z_{i-1})$ the output of the i-th block, with $z_0 = x$. Also, let $x = (w_x, x')$ and $y = (w_y, y')$ where w_x and w_y are the witness (private) parts of the input and output, while x' and y' are the public parts. For a standard application of GKR in our context, one commits to $w = (w_x, w_y)$ and applies GKR to reduce the layered circuit claim to evaluation claims on $\widetilde{x}$ and $\widetilde{y}$. In our first compiler, the latter claims are directly proven through the LCS as linear constraints on w while in our second compiler, they are merged into a single MLE constraint on $\widetilde{w}$ (see Section 3.2) which is then proven through the MPCS.

The flattened version of C is the circuit C' defined as:

$$(z_1, \ldots, z_{K-1}, y) = C'(x, z_1, \ldots, z_{K-1}) = (C_1(x), C_2(z_1), \ldots, C_K(z_{K-1})) \ ,$$

which is of depth $\lceil d/K \rceil$ and of width at most $K \cdot \omega$, where ω is the maximum width of the original circuit C. The input of C' is $x' = (x, z_1, \ldots, z_{K-1})$ and its output is $y' = (z_1, \ldots, z_{K-1}, y)$. In this context, the witness is extended with the intermediate circuit outputs as $w = (w_x, w_y, z_1, \ldots, z_{K-1})$. The application of GKR yields evaluation claims on $\widetilde{x}'$ and $\widetilde{y}'$, which are then proven through the LCS or MPCS as before. A notable difference with the proposition of [Lon26] is that we don't need to include subtraction gates to prove that the input of C_{i+1} equals the output of C_i, this is directly enforced by the constraints on w, which can be roughly summed up as "the z_i part of x and the z_i part of y are both taken from the same z_i part from w".

Another improvement is to exploit the structure of the flattened circuit in the sum-check of GKR. Indeed, the flattened circuit is composed of K independent blocks of layers, which allows some improved sum-check application compared to the standard GKR. Concretely, assume homogeneous layers of degree d and width 2^ℓ and denote

$$V_i^{(s)}(k) = \sum_{j_1,\ldots,j_d \in \{0,1\}^\ell} F_i^{(s)}(k, j_1, \ldots, j_d) \cdot \prod_{u=1}^{d} V_{i-1}(j_u) \ ,$$

the output of the i-th layer of the sub-circuit C_s, indexed by $s \in \{0,1\}^\iota$ with $\iota = \log K$. We can then rewrite the layer-i sum-check of GKR as:

$$V_i'(s,k) = \sum_{z \in \{0,1\}^\iota} \sum_{j_1,\ldots,j_d \in \{0,1\}^\ell} \widetilde{\mathrm{eq}}(z,s) \cdot F_i'(z, k, j_1, \ldots, j_d) \cdot \prod_{u=1}^{d} V_{i-1}'(z, j_u) \ ,$$

where $V_i'(s,k) = V_i^{(s)}(k)$ and $F_i'(s,k,j_1,\ldots,j_d) = F_i^{(s)}(k,j_1,\ldots,j_d)$ for all $s \in \{0,1\}^\iota$. The above sum-check is of degree $d+2$ in the z variable and of degree 2 in the $j_1,\ldots,j_d$ variables, which results in communication of $(d+2)\iota + 2d\ell$ field elements. On the other hand, applying standard GKR to the flattened circuit would result in a communication of $2d(\ell+\iota)$ field elements (since the state width would scale from 2^ℓ to $K \cdot 2^\ell = 2^{\ell+\iota}$). Therefore, the above tweak is more efficient than standard GKR applied to the flattened circuit whenever $d \geq 2$, which is the case for most practical circuits.

For the particular case of an iterated layered circuit where $C_1 = C_2 = \ldots = C_K$, the above sum-check can be further improved by dropping the z variable from F_i' and defining $F_i' = F_i^{(s)}$ for any $s \in \{0,1\}^\iota$ (since they are all equal). This reduces the communication to $(d+1)\iota + 2d\ell$ field elements.

6 Applications

This section presents several applications of our zero-knowledge argument systems, including arithmetic circuits, hashing based on Keccak and Poseidon, and lattice-based relations. We compare the resulting proof sizes with state-of-the-art proof systems that exhibit similar characteristics and rely on comparable assumptions. Since our proof system relies only on symmetric cryptography, and does not require computational assumptions (e.g., the hardness of the discrete logarithm or lattice problems), we restrict our comparison to proof systems with a similar assumption profile, namely MPC-in-the-Head and hash-based proof systems.[4]

For the first family, we consider the VOLE-in-the-Head framework [BBD+23], which underlies several recent signature schemes [BBB+24, ABB+24b, BBFR24]. This framework relies on GGM (seed) trees to commit to degree-1 polynomials (*i.e.* lines). The proof sizes reported in our comparison incorporate the size optimization of [BBM+24]. For hash-based proof systems, namely those relying on Merkle-tree commitments, we consider Ligero [AHIV17, AHIV23], SmallWood [FR26], and Longfellow ZK [Fas26, FS26] as representative of the current state of the art for the parameter ranges considered here. Ligero relies on proximity tests to ensure that the polynomials committed through the Merkle trees are close to low-degree polynomials. SmallWood instead relies on the degree-enforcing test introduced in [FR25]. Although structurally similar to Ligero's proximity test, the degree-enforcing test allows fewer Merkle leaves to be opened at the cost of computing more linear combinations, resulting in shorter proof transcripts over a wider range of parameters. Finally, Longfellow ZK can be viewed as an instantiation of the GKR protocol in the CBIP paradigm, where the witness and trace mask, and their related constraints, are committed and proved using Ligero.

Other hash-based proof systems have been proposed. Some, such as Brakedown [GLS+23], build upon Ligero and primarily improve the asymptotic computational complexity of the prover rather than reducing proof sizes. There are also proof systems based on the FRI protocol [BBHR18a], such as Aurora [BCR+19] and STARK [BBHR18b], further recently improved by such as STIR [ACFY24a] and WHIR [ACFY24b], but their proof sizes are not competitive for the range of statement sizes considered here. For instance, [FR26] showed that Aurora outperforms SmallWood in terms of proof size only once

[4] Depending on the precise definition of the MPC-in-the-Head paradigm, these two families may overlap.

proofs exceed approximately 500 kilobytes, whereas all the proof sizes reported hereafter remain below this threshold.

While we have a proof-of-concept implementation of our scheme in SageMath, which allows us to validate the reported proof sizes, we do not yet have an optimized implementation from which representative running times could be obtained. Nevertheless, our design is closely related to Longfellow ZK, as both rely on similar commit-and-prove systems based on Merkle trees[5] and make use of sum-check protocols. Longfellow ZK reports running times ranging from a few tens to a few hundreds of milliseconds. We therefore expect the computational cost of our construction to be of a similar order of magnitude.

6.1 Arithmetic Circuits

In what follows, we apply our proof system to proving the correctness of arithmetic circuits. We first consider general unstructured circuits, before turning to structured circuits.

General Arithmetic Circuits via R1CS. The sum-check protocol can be used to prove the satisfiability of an R1CS instance, as shown in [Set20]. An R1CS instance $\mathbb{x} = (\mathbb{F}, \omega, A, B, C)$ is defined by a finite field $\mathbb{F}$, an integer ω, and three matrices $A, B, C \in \mathbb{F}^{\omega \times \omega}$. A witness for $\mathbb{x}$ is a vector $w \in \mathbb{F}^{\omega}$ satisfying

$$(A \cdot w) \circ (B \cdot w) = C \cdot w \ ,$$

where $\circ$ denotes the Hadamard product. It is well known that R1CS naturally captures the satisfiability of general arithmetic circuits: the witness w contains the private inputs and outputs of the circuit, together with the outputs of its multiplication gates.[6]

Let $\ell := \lceil \log_2(\omega) \rceil$, let $\tilde{w} : \mathbb{K}^\ell \to \mathbb{K}$ be the multilinear extension of w, and let $\tilde{A}, \tilde{B}, \tilde{C} : \mathbb{K}^\ell \times \mathbb{K}^\ell \to \mathbb{K}$ be the bivariate multilinear extensions of A, B, C, respectively. We define

$$h(X) := \left(\sum_{j \in \{0,1\}^\ell} \tilde{A}(X, j)\tilde{w}(j) \right) \left(\sum_{j \in \{0,1\}^\ell} \tilde{B}(X, j)\tilde{w}(j) \right) - \left(\sum_{j \in \{0,1\}^\ell} \tilde{C}(X, j)\tilde{w}(j) \right) \ . \tag{13}$$

The R1CS instance is satisfied if and only if $h(x) = 0$ for every $x \in \{0,1\}^\ell$. Define

$$q(\gamma) := \sum_{x \in \{0,1\}^\ell} \widetilde{\mathrm{eq}}(\gamma, x)\, h(x) \ ,$$

where $\widetilde{\mathrm{eq}}$ is the multilinear extension of the equality function. Since q is the multilinear extension of the evaluations of h over the Boolean hypercube, $h(x) = 0$ for every $x \in \{0,1\}^\ell$ if and only if q is the zero polynomial. Consequently, if the R1CS relation is not satisfied, then q is a nonzero multilinear polynomial of total degree at most ℓ. In that case, by the Schwartz-Zippel lemma, for a uniformly random $\gamma \in \mathbb{K}^\ell$,

$$\Pr[q(\gamma) = 0] \leq \frac{\ell}{|\mathbb{K}|} \ .$$

Observing $q(\gamma) = 0$ thus mean that the R1CS relation is satisfied with overwhelming probability. As described in [Set20], the claim $q(\gamma) = 0$ can be reduced to a claim of the form $\tilde{w}(r) = t$ for some (r, t) by using Equation (13) and two successive sum-check protocols. The first sum-check reduces the original claim to three claims of the form

$$\tilde{a}(r') = t_a, \qquad \tilde{b}(r') = t_b, \qquad \tilde{c}(r') = t_c,$$

for some (r', t_a, t_b, t_c), where $\tilde{a}$, $\tilde{b}$, and $\tilde{c}$ are the multilinear extensions of $A \cdot w$, $B \cdot w$, and $C \cdot w$, respectively. A second sum-check can then reduce these claims to an evaluation claim on $\tilde{w}$.

[5] Specifically, Longfellow ZK uses Ligero as its underlying commit-and-prove system, whereas we use our own tailored commit-and-prove systems based on SmallWood. These can be viewed as variants of Ligero built from the degree-enforcing commitment scheme (DECS), which, for a comparable Merkle-tree size parameter, achieves similar efficiency.

[6] The witness w may additionally contain a coordinate fixed to 1 to represent constant terms.

While this two-step protocol fits the format of an SVCS instance, we can obtain a more efficient protocol in our setting. Indeed, the second sum-check only verifies *linear* relations in the committed witness. We can therefore omit it and verify these linear relations directly through the commitment scheme (see the extended version of an SVCS instance). The resulting protocol is described in Figure 13.

Context:

- $\mathcal{P}$ has previously committed to a witness $w \in \mathbb{F}^\omega$.
- $\mathcal{P}$ claims that $(A \cdot w) \circ (B \cdot w) = C \cdot w$, where $A, B, C \in \mathbb{F}^{\omega \times \omega}$ are public.
- ℓ is defined as $\ell := \lceil \log_2(\omega) \rceil$.

Protocol:

- $\mathcal{V}$ chooses a random $\gamma \in \mathbb{K}^\ell$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ and $\mathcal{V}$ run the sum-check protocol on the claim

$$0 = \sum_x \widetilde{\text{eq}}(\gamma, x) \cdot (\tilde{a}(x) \cdot \tilde{b}(x) - \tilde{c}(x)),$$

  where $\tilde{a}(x), \tilde{b}(x), \tilde{c}(x) \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_\ell]$ are the dimension-ℓ multilinear extensions of respectively $A \cdot w$, $B \cdot w$, and $C \cdot w$. At the end of the protocol, the claim is reduced to checking claims $\tilde{a}(r) = t_a$, $\tilde{b}(r) = t_b$, and $\tilde{c}(r) = t_c$ for some $r \in \mathbb{K}^\ell$ and $t_a, t_b, t_c \in \mathbb{K}$.
- $\mathcal{P}$ convinces $\mathcal{V}$ that the committed witness w satisfies $\tilde{a}(r) = t_a$, $\tilde{b}(r) = t_b$, and $\tilde{c}(r) = t_c$. Since $\tilde{a}(r)$, $\tilde{b}(r)$, and $\tilde{c}(r)$ are linear functions of w, these claims can be checked directly using the commitment scheme.

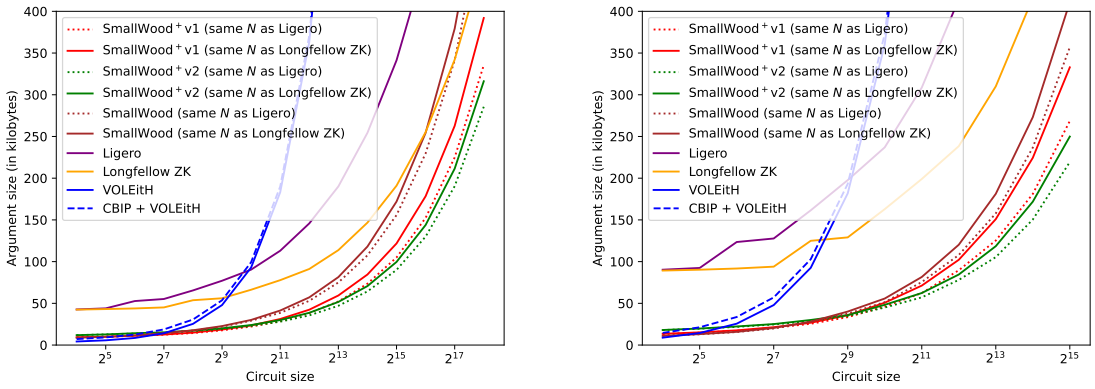
Fig. 13: Sumcheck-based protocol to prove knowledge of an R1CS satisfiability.

We report in Figure 14 the proof sizes obtained for general arithmetic circuits over 64-bit and 256-bit fields. We compare our results with VOLE-in-the-Head, SmallWood, Ligero, and Longfellow ZK, targeting 128-bit security for all proof systems. We additionally consider a VOLEitH-based construction following the CBIP paradigm, but using VOLEitH as the underlying commitment scheme. For all constructions relying on sum-check protocols (our proof system, Longfellow ZK, and "CBIP + VOLEitH"), we use the R1CS protocol described above. For Longfellow ZK, rather than using the underlying GKR protocol directly, we adapt it by replacing its sum-check protocol with the R1CS sum-check protocol described in Figure 13.

To provide a reasonably fair comparison, we instantiate schemes based on Merkle trees using the same tree size N as Ligero and Longfellow ZK. For the VOLEitH-based constructions, which instead rely on GGM trees, we use 11 GGM trees with 2048 leaves each (classical parameter sets in the literature). We make the following observations:

1. Combining CBIP with VOLEitH does not provide any benefit in this setting, since the sum-check protocol does not reduce the size of the committed witness: one field element still needs to be committed per multiplication gate. As expected, VOLEitH-based proof systems therefore quickly lead to large proof sizes because their communication grows linearly with the size of the committed data.
2. Combining sum-check with Ligero (*i.e.* Longfellow ZK) or SmallWood (*i.e.* our scheme) is more beneficial than Ligero or SmallWood alone (although we note that our scheme further imrpoves SmallWood-style commitments to fit this setting). Without sum-check, Ligero and SmallWood require committing to approximately three and two field elements per multiplication gate, respectively, whereas Longfellow ZK and our scheme require only one. The resulting reduction in proof size nevertheless remains moderate, since the sum-check trace mask must additionally be committed.
3. Our scheme achieves smaller proof sizes than Longfellow ZK. This improvement comes from the underlying commitment scheme, which provides shorter opening proofs, including when both schemes use trees of the same size.

To summarize, for general unstructured arithmetic circuits, our proof system improves upon the considered state-of-the-art constructions, although the resulting savings remain moderate. In the next section, we show that substantially larger improvements can be obtained when the structure of the circuit can be exploited.



(a) Argument sizes for 64-bit fields

(b) Argument sizes for 256-bit fields

Fig. 14: Argument sizes for proving the correctness of general arithmetic circuits.

Layered Arithmetic Circuits. Let us now consider layered arithmetic circuits, namely circuits whose gates can be partitioned into layers such that the inputs to each layer are outputs of the preceding layer. In this setting, expressing circuit satisfiability through R1CS is generally suboptimal, as it does not exploit the layered structure of the computation. One can instead rely on the GKR protocol [GKR08].

As described in Section 2.2, a layered circuit instance $\mathbb{x} = (\mathbb{F}, n, \{\ell_0, \ldots, \ell_n\}, d, \{F_i^{(m)}\}_{i,m})$ is defined by a finite field $\mathbb{F}$, an integer n denoting the number of circuit layers, integers $\ell_0, \ldots, \ell_n$, where $\omega_i = 2^{\ell_i}$ is the width of the i-th state, and a collection of $(\omega_i + m \cdot \omega_{i-1})$-variate functions $F_i^{(m)} : \mathbb{F}^{\omega_i + m \cdot \omega_{i-1}} \to \mathbb{F}$, for $i = 1, \ldots, n$ and $m = 1, \ldots, d$, defining the computation performed by each layer. A witness for $\mathbb{x}$ is a pair (x, y), with $x \in \mathbb{F}^{\omega_0}$ and $y \in \mathbb{F}^{\omega_n}$, such that $y = C(x)$, where C denotes the circuit defined by the functions $F_i^{(m)}$.

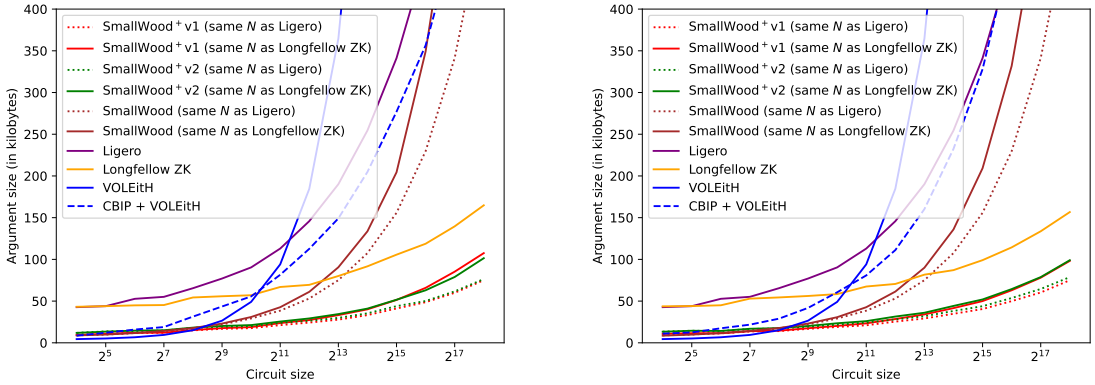
The GKR protocol proceeds layer by layer, successively reducing a claim about an evaluation of the multilinear extension of an intermediate state to a claim about an evaluation of the multilinear extension of the preceding state. After processing all the layers, the resulting claim only involves the committed circuit inputs. Since each layer reduction is performed using a sum-check protocol, GKR naturally fits the format of an SVCS instance.

We report in Figures 15 and 16 the proof sizes obtained for layered arithmetic circuits over 64-bit and 256-bit fields, respectively. We consider two families of circuits, parameterized by their total size $|C|$. The first family consists of relatively flat circuits, whose depth scales as $\sqrt[3]{|C|}$. The second consists of relatively deep circuits, whose layer width scales as $\sqrt[3]{|C|}$. In both families, the inputs to the first layer can be expressed as linear combinations of a fixed number of circuit inputs.[7] To reduce the communication due to the circuit depth, we use the flattening technique described in Section 5.3 (where we select the flattening parameter K minimising the proof size).

We compare our results with the same proof systems as in the previous section on general arithmetic circuits. The only difference is that, for constructions following the CBIP paradigm, we now exploit the layered structure through the GKR protocol.

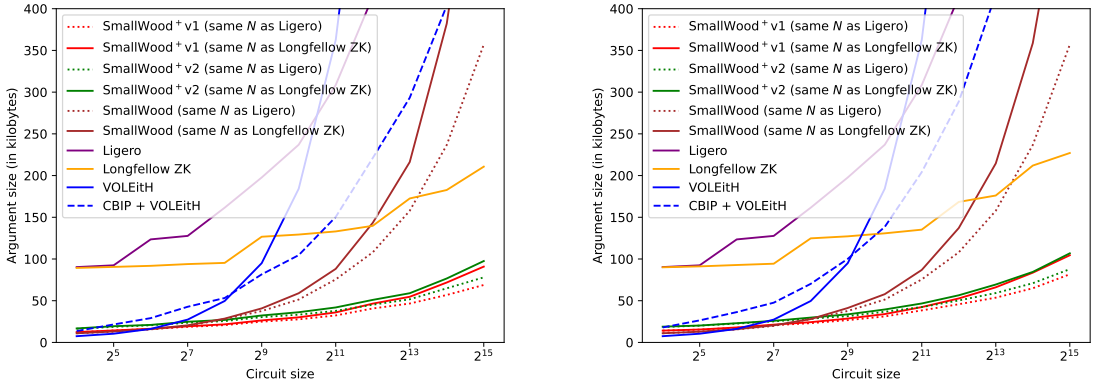
We make the following observations:

[7] This choice allows us to illustrate how the proof size scales with the circuit size independently of the size of the circuit inputs.



(a) Flat circuit (b) Deep circuit

Fig. 15: Argument sizes for proving the correctness of layered arithmetic circuits over a 64-bit field.



(a) Flat circuit (b) Deep circuit

Fig. 16: Argument sizes for proving the correctness of layered arithmetic circuits over a 256-bit field.

1. Even when using flattening, the GKR protocol allows us to commit to fewer field elements than there are multiplication gates in the circuit. Consequently, the proof size is significantly reduced compared to the constructions that do not rely on GKR. As expected, exploiting the layered structure results in substantially smaller proofs than treating a circuit of the same size over the same field as a general unstructured arithmetic circuit.
2. Our scheme achieves highly competitive proof sizes already for circuits of size 2^9. In particular, proof sizes below 100 KB can be achieved for circuits as large as 2^{15} over $\mathbb{F}_{2^{256}}$ and 2^{18} over $\mathbb{F}_{2^{64}}$.
3. Interestingly, the proof sizes obtained for flat and deep circuits are very similar.[8] This is mainly due to the flattening technique, which tends to smooth out the differences between the two circuit structures. More significant differences might arise for irregular circuits (*i.e.* circuits whose layer widths vary across the computation). However, assessing such differences would likely require a dedicated analysis tailored to the specific circuit structure and application.

[8] We initially aimed to provide results for *squared circuits* (with width $\sqrt{|C|}$ and depth $\sqrt{|C|}$) as well, but this intermediate setting falls between the flat and deep cases, which is irrelevant given how close the corresponding results are.

6.2 Hashing

In what follows, we apply our proof system to hash functions. We will first investigate Keccak-based hashing, then we will study the case of Poseidon-based hashing.

Keccak-Based Hashing. To illustrate the performance of $\mathsf{SmallWood}^+$ to prove Keccak permutations, we consider the following statement: given two public bitstrings $y_0, y_1 \in \{0,1\}^{256}$, we know $x \in \{0,1\}^{1600-256}$ such that $\mathsf{Keccak}(y_0 \parallel x) = (y_1 \parallel z)$ for some $z \in \{0,1\}^{1600-256}$. This corresponds to a variant of the *Constrained-Input Constrained-Output* (CICO) problem for the Keccak permutation, which is hard to solve.

To prove the correct execution of a Keccak permutation, the most direct approach is to represent the permutation as a layered binary circuit of width 1600, with one circuit layer per Keccak round, and to apply the GKR protocol. While this approach leads to the shortest achievable proof sizes, the underlying sum-check protocols are computationally expensive, as GKR does not fully exploit the structure of the considered circuit layers.

Each Keccak round involves 1600 multiplications, one per state element, where each operand is a linear combination of 11 elements of the previous state. A generic GKR treatment does not preserve this bilinear structure: after expanding the product of the two linear forms, each output is handled as a quadratic polynomial containing $11^2 = 121$ monomials. Similarly to the R1CS case, we can design a more computationally efficient protocol by explicitly preserving this structure, at the cost of a slightly longer trace.

Let us assume that we want to prove the correct execution of at most 2^{ℓ_Z} Keccak permutations, and let $\ell_X = \lceil \log_2(1600) \rceil$. We denote by $\widetilde{\mathsf{state}}^{(i)}(X, Z)$ the multilinear extension of the states after round i, such that $\widetilde{\mathsf{state}}^{(i)}(X, z)$ for $z \in \{0,1\}^{\ell_Z}$ is the MLE of the ith-round state of the jth Keccak permutation with $\langle z \rangle := j$, where $\langle z \rangle$ corresponds to the integer with binary representation z.

The transition between two consecutive round states can then be written as

$$\widetilde{\mathsf{state}}^{(i+1)}(X, Z) = \widetilde{c}^{(i)}(X, Z) + \sum_{x,z} \widetilde{\mathrm{eq}}(X, x)\widetilde{\mathrm{eq}}(Z, z) \left(\widetilde{\mathsf{offset}}^{(i)}(x, z) + (1 - \widetilde{\mathsf{left}}^{(i)}(x, z))\widetilde{\mathsf{right}}^{(i)}(x, z) \right),$$

where

$$\begin{cases} \widetilde{\mathsf{offset}}^{(i)}(x, z) & = \sum_y \widetilde{A}_{\mathsf{offset}}(x, y)\widetilde{\mathsf{state}}^{(i)}(y, z), \\ \widetilde{\mathsf{left}}^{(i)}(x, z) & = \sum_y \widetilde{A}_{\mathsf{left}}(x, y)\widetilde{\mathsf{state}}^{(i)}(y, z), \\ \widetilde{\mathsf{right}}^{(i)}(x, z) & = \sum_y \widetilde{A}_{\mathsf{right}}(x, y)\widetilde{\mathsf{state}}^{(i)}(y, z). \end{cases}$$

Here, $\widetilde{A}_{\mathsf{offset}}$, $\widetilde{A}_{\mathsf{left}}$, and $\widetilde{A}_{\mathsf{right}}$ are the bivariate multilinear extensions of the linear maps deriving, from an intermediate state, the three inputs involved in the nonlinear round operation. Moreover, $\widetilde{c}^{(i)}$ denotes the multilinear extension of the round constants. We can then use a two-level sum-check protocol to reduce a claim of the form $\widetilde{\mathsf{state}}^{(i+1)}(r) = v$ to a claim of the form $\widetilde{\mathsf{state}}^{(i)}(r') = v'$. To optimize the proof size, we use the flattening technique described in Section 5.3 to reduce the depth of the proved computation at the cost of increasing the amount of committed data. A Keccak permutation consists of 24 rounds. We partition these rounds into K groups of $24/K$ consecutive rounds. The resulting computation therefore has a depth of $24/K$ round transition, while each layer contains the intermediate states corresponding to K rounds for each proved permutation. Rather than applying the flattening technique of Section 5.3 directly, we pack the flattened rounds along the variable Z in the relations above. Consequently, 2^{ℓ_Z} is chosen as an upper bound on the number of proved permutations multiplied by the flattening factor K. We thus commit jointly to the inputs of all the permutations (1344 bits per permutation) and to the intermediate Keccak states introduced by the flattening procedure.

Committing these binary elements using the LCS or MPCS commitment schemes would lead to suboptimal proof sizes. Instead, we rely on the zero-knowledge ring-switching protocol described in Section 5.2. This allows us to pack 16 bits into a single $\mathbb{F}_{2^{16}}$ element. We report in Table 1 the resulting proof sizes for different numbers of Keccak permutations. We observe that our scheme achieves significantly smaller proof sizes than the other proof systems considered in the comparison. SmallWood and our scheme use Merkle trees with 2048 leaves, which is comparable to the tree sizes used by Longfellow ZK. Smaller trees

can be used to reduce the proving time at the cost of a moderate increase in proof size (*e.g.* 20.1–74.5 KB with 1024 leaves, 23.6–97.2 KB with 512 leaves), while still remaining smaller than Longfellow ZK. Note that the Longfellow ZK numbers already use our ring-switching technique to pack its committed binary data (Ligero, which underlies Longfellow ZK, has no native way to handle $\mathbb{F}_2$ elements efficiently); the reported gap (e.g., 127.0 KB down to 63.1 KB for 100 permutations) therefore understates our actual contribution.

# Perm.	Ligero	VOLEitH	SmallWood	Longfellow ZK	CBIP+VOLEitH	SmallWood$^+$	
						Compiler 1	Compiler 2
1	182.0 KB	29.4 KB	89.9 KB	59.8 KB (12)	51.9 KB (12)	**18.1 KB** (12)	21.2 KB (24)
2	241.0 KB	55.8 KB	135.9 KB	65.6 KB (12)	74.3 KB (8)	**20.7 KB** (12)	23.9 KB (12)
5	358.3 KB	135.0 KB	243.4 KB	73.7 KB (8)	121.0 KB (6)	**25.8 KB** (8)	29.0 KB (8)
10	489.1 KB	267.0 KB	389.0 KB	82.5 KB (4)	171.5 KB (4)	**31.1 KB** (6)	34.4 KB (8)
20	673.2 KB	531.0 KB	641.4 KB	92.8 KB (3)	248.5 KB (3)	**37.9 KB** (4)	41.3 KB (6)
50	1034.4 KB	1323.0 KB	1313.7 KB	110.0 KB (3)	402.5 KB (2)	**49.3 KB** (3)	54.0 KB (3)
100	1441.7 KB	2643.0 KB	2367.8 KB	127.0 KB (2)	586.8 KB (1)	**63.1 KB** (2)	66.4 KB (3)

Table 1: Proof sizes for proving the correct execution of Keccak permutations. The values in parentheses indicate the flattening factor. SmallWood and SmallWood$^+$ use Merkle trees with 2048 leaves. VOLEitH uses degree-4 constraints, whereas CBIP+VOLEitH uses only degree-2 constraints. The tree sizes for Ligero and Longfellow ZK follow the parameter-selection methodology of Ligero. They range from 4 600 to 40 000 leaves for Ligero and from 1 140 to 2 320 leaves for Longfellow ZK.

Poseidon-Based Hashing. We now investigate the case of proving Poseidon-based hashing. We consider several approaches. The first one consists of expressing the Poseidon statement under an R1CS-like constraint system, where the non-linear operation is the fan-in-1 S-box gate rather than a fan-in-2 multiplication gate. The second approach simply consists in expressing the statement as a layered circuit and relying on a GKR-like protocol, like we did for Keccak. Our third approach consists in a hybrid strategy, mixing the two previous approaches depending on the Poseidon permutation rounds.

Protocol for S-box circuits. We consider a variant of R1CS to capture arithmetic circuits made of linear gates (additions and multiplications by constants) and fan-in-1 *S-box* gates. We assume that the S-box relation $y = S(x)$ can be checked using a bivariate degree-d polynomial function f_S such that

$$y = S(x) \Leftrightarrow f_S(x, y) = 0 \ .$$

An *S-box circuit* instance $\mathbb{x}_{\text{SBOX}} = (\mathbb{F}, \omega, A, B, f_S)$ is defined by a finite field $\mathbb{F}$, an integer ω, two $\omega \times \omega$ matrices A, B over $\mathbb{F}$ and a degree-d polynomial function $f_S : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$. A witness for $\mathbb{x}_{\text{SBOX}}$ is a vector $w \in \mathbb{F}^\omega$ such that:

$$\boldsymbol{f_S}(A \cdot w, B \cdot w) = 0 \ ,$$

where $\boldsymbol{f_S}$ denotes the application of f_S coordinate-wise. The satisfiability of an S-box circuit instance can be proven using a sum-check-based protocol in the same way as for R1CS, with the main difference that the polynomial h defined in Equation (13) is replaced by the following polynomial:

$$h(X) := f_S\Big(\sum_{j \in \{0,1\}^\ell} \tilde{A}(X, j) \cdot \tilde{w}(j), \sum_{j \in \{0,1\}^\ell} \tilde{B}(X, j) \cdot \tilde{w}(j) \Big) \ . \tag{14}$$

Similarly to the R1CS case, one can simply use a single execution of the sum-check protocol to reduce the satisfiability claim into evaluation claims of the form

$$\tilde{a}(r') = t_a, \qquad \tilde{b}(r') = t_b,$$

for some (r', t_a, t_b), where $\tilde{a}$ and $\tilde{b}$ are the multilinear extensions of $A \cdot w$ and $B \cdot w$, respectively. Those evaluation claims can then be directly verify through the commitment scheme.

In Poseidon, the S-box operation is a low-degree power of the inputs: $S(x) = x^\alpha$ where α is usually 3, 5 or 7. Therefore, the verification polynomial is simply $f_S : (x, y) \mapsto x^\alpha - y$.

GKR-like protocol. Let us assume that we want to check at most 2^{ℓ_Z} Poseidon permutations, which has a state size of a most 2^{ℓ_X} field elements, for some (ℓ_X, ℓ_Z).

For $z \in \{0,1\}^{\ell_Z}$, we denote $\mathsf{state}^{(i,z)}$ the intermediary state just before the ith S-box layer in the jth Poseidon permutation with $\langle z \rangle := j$. We further denote

- $\widetilde{\mathsf{state}}^{(i)}(X, z)$ its multilinear extension, when the corresponding S-box layer is full;
- $\widetilde{\mathsf{state}}^{(i)}_{\text{left}}(X, z)$ the multilinear extension of the coordinates of $\mathsf{state}^{(i,z)}$ for which no S-box operation are applied, when the corresponding S-box layer is partial;
- $\widetilde{\mathsf{state}}^{(i)}_{\text{right}}(z)$ the last coordinate of $\mathsf{state}^{(i,z)}$, where the S-box is applied, when the corresponding S-box layer is partial.

To compute $\widetilde{\mathsf{state}}^{(i+1)}(X, Z)$ from $\widetilde{\mathsf{state}}^{(i)}(X, Z)$ when considering a full round, we can use the following relations:

$$\begin{cases} \widetilde{\mathsf{inter}}^{(i)}(Y, Z) = \sum_{y,z} \widetilde{\mathrm{eq}}(Y, y) \cdot \widetilde{\mathrm{eq}}(Z, z) \cdot S(\widetilde{\mathsf{state}}^{(i)}(y, z)) \\ \widetilde{\mathsf{state}}^{(i+1)}(X, Z) = \widetilde{c}^{(i)}(X, Z) + \sum_y \widetilde{M}(X, y) \cdot \widetilde{\mathsf{inter}}^{(i)}(y, Z) \end{cases}$$

$\widetilde{\mathsf{inter}}^{(i)}(x, z)$ is the multilinear extension of the intermediary state just after applying the S-box layer to the state represented by $\widetilde{\mathsf{state}}^{(i)}$. Moreover $\widetilde{M}$ is the bivariate MLE of the MDS matrix M defining the linear layer, while $\widetilde{c}^{(i)}$ is the MLE of the round constants.

To compute $\widetilde{\mathsf{state}}^{(i+1)}_{\text{left}}(X, Z)$ and $\widetilde{\mathsf{state}}^{(i+1)}_{\text{right}}(Z)$ from $\widetilde{\mathsf{state}}^{(i)}_{\text{left}}(X, Z)$ and $\widetilde{\mathsf{state}}^{(i)}_{\text{right}}(Z)$ when considering a *partial* round, we can use the following relations:

$$\begin{cases} \widetilde{\mathsf{inter}}^{(i)}(Z) = \sum_z \widetilde{\mathrm{eq}}(Z, z) \cdot S(\widetilde{\mathsf{state}}^{(i)}_{\text{right}}(z)) \\ \widetilde{\mathsf{state}}^{(i+1)}_{\text{left}}(X, Z) = \widetilde{c}^{(i)}_{\text{left}}(X, Z) + \sum_y \widetilde{B_0}(X, y) \cdot \widetilde{\mathsf{state}}^{(i)}_{\text{left}}(y, Z) + \widetilde{b_1}(X) \cdot \widetilde{\mathsf{inter}}^{(i)}(Z) \\ \widetilde{\mathsf{state}}^{(i+1)}_{\text{right}}(Z) = \widetilde{c}^{(i)}_{\text{right}}(Z) + \sum_y \widetilde{c_0}(y) \cdot \widetilde{\mathsf{state}}^{(i)}_{\text{left}}(y, Z) + b_2 \cdot \widetilde{\mathsf{inter}}^{(i)}(Z) \end{cases}$$

where $\widetilde{B_0}$, $\widetilde{b_1}$ and $\widetilde{b_0}$ are the MLE respectively of B_0, b_1 and b_0 such that

$$\begin{pmatrix} \mathsf{state}^{(i+1,z)}_{\text{left}} \\ \mathsf{state}^{(i+1,z)}_{\text{right}} \end{pmatrix} = \begin{pmatrix} B_0 & b_1 \\ b_0^\top & b_2 \end{pmatrix} \begin{pmatrix} \mathsf{state}^{(i,z)}_{\text{left}} \\ \mathsf{inter}^{(i,z)}_{\text{right}} \end{pmatrix} = \begin{pmatrix} B_0 & b_1 \\ b_0^\top & b_2 \end{pmatrix} \begin{pmatrix} \mathsf{state}^{(i,z)}_{\text{left}} \\ S(\mathsf{state}^{(i,z)}_{\text{right}}) \end{pmatrix} ,$$

while $\widetilde{c}^{(i)}_{\text{left}}$ and $\widetilde{c}^{(i)}_{\text{right}}$ are the MLE of the round constants. The transition between a full layer and a partial layer can be easily handled by tweaking the above linear relations.

Regarding flattening, we consider two options: (i) we do not rely on flattening at all, meaning that we just commit to the permutation inputs; (ii) we split the layers between full rounds and partial rounds, which are checked independently, and we apply independent flattening on each side.

Hybrid method. The hybrid method consists of combining the two previous approaches. The key observation is that one might estimate that using a GKR-protocol could be suboptimal for partial rounds. Indeed, when relying on flattening, one needs to commit to entire states of the partial rounds, even if they only involve a single S-box gate per round. Therefore, in the hybrid method, we use the GKR-like strategy for full round, while using the R1CS-like strategy for partial rounds.

Analysis. We report in Table 2 the proof sizes obtained when proving either a single or 100 Poseidon permutation(s), for different field sizes and S-box degrees. For all instances, we consider a state size of 512 bits. We additionally report the proof sizes obtained with SmallWood.

In the regime where only a small number of permutations must be proved, the S-box arithmetization yields the shortest proofs among the three strategies considered above. Indeed, in this setting, the trace overhead introduced by the GKR-like and hybrid approaches is not negligible compared to the size of the committed witness. However, over large fields, SmallWood still achieves smaller proof sizes. This is consistent with our observations for unstructured arithmetic circuits based on R1CS where the gain over SmallWood is limited for small instances. Although the S-box approach combined with sum-check reduces

Instance				SmallWood	S-box circuits		GKR-like protocol		Hybrid protocol	
$\log_2 q$	α	t	R_P		Compiler 1	Compiler 2	Compiler 1	Compiler 2	Compiler 1	Compiler 2
Statement made of a single permutation										
256	3	2	57	**13.9 KB**	14.4 KB	17.7 KB	17.2 KB	21.0 KB	15.5 KB	19.3 KB
256	5	2	57	**14.5 KB**	15.5 KB	19.5 KB	18.6 KB	22.9 KB	16.9 KB	21.1 KB
64	7	8	22	12.5 KB	**11.5 KB**	14.1 KB	14.8 KB	17.6 KB	13.3 KB	15.8 KB
32	7	16	14	11.8 KB	**11.3 KB**	13.9 KB	13.6 KB	16.3 KB	12.6 KB	15.2 KB
Statement made of 100 permutations										
256	3	2	57	131.1 KB	101.7 KB	91.4 KB	**70.0 KB**	76.9 KB	98.0 KB	91.4 KB
256	5	2	57	135.4 KB	102.8 KB	93.2 KB	**81.3 KB**	88.2 KB	101.5 KB	95.7 KB
64	7	8	22	77.8 KB	51.4 KB	49.9 KB	55.3 KB	59.3 KB	**45.1 KB**	47.2 KB
32	7	16	14	71.8 KB	45.0 KB	44.9 KB	49.6 KB	54.1 KB	**39.0 KB**	41.8 KB

Table 2: Proof sizes when proving Poseidon permutation. q: field size, α: sbox degree, t: state size. Merkle tree of 4096 leaves.

the size of the committed witness by roughly a factor of two compared with SmallWood, it additionally requires committing to a trace mask. Consequently, the reduction in witness size only translates into an overall reduction in proof size once the statement is sufficiently large for the fixed overhead of the sum-check protocol to be amortized.

The situation changes when proving a large number of permutations. The S-box arithmetization does not scale well because the witness size grows linearly with the number of permutations. In contrast, a GKR protocol reduces the size of the committed witness at the cost of a moderately larger execution trace. Overall, this trade-off becomes favorable as the number of permutations increases. However, a pure GKR approach is not always optimal. Owing to the sparsity of the partial rounds, it can be more efficient to prove the partial rounds using the S-box arithmetization while retaining the GKR protocol for the full rounds.

To understand this trade-off, consider the use of flattening over the partial rounds to minimize the proof size. In practice, the optimal flattening factor K typically results in a relatively small circuit depth d. When using GKR for the partial rounds, the commitment essentially contains the intermediate states, requiring approximately $t \cdot K$ field elements. Since the flattened circuit satisfies $R_P \approx d \cdot K$, proving the correctness of the R_P S-boxes (one S-box per partial round) requires committing to approximately $t \cdot K \approx \frac{t}{d} \cdot R_P$ field elements. By contrast, with the S-box arithmetization, the commitment contains the input state of the first partial round together with the outputs of all partial-round S-boxes, for a total of approximately $t + R_P$ field elements. Consequently, the optimal strategy depends primarily on the ratio t/d. If $t/d < 1$, the GKR commitment is smaller, whereas if $t/d > 1$, the S-box arithmetization becomes more efficient. This behavior is reflected in Table 2. For the instances defined over 256-bit fields, the state width t is relatively small, yielding $t/d < 1$ for the considered parameters. As a result, the GKR protocol achieves the shortest proofs. Conversely, for instances over smaller fields, the state width must increase in order to maintain a comparable security level (roughly preserving $t \log_2 q$). Consequently, $t/d > 1$ for the considered parameters, making the hybrid protocol more efficient.

Finally, when proving a very large number of permutations, the optimal depth d increases. Eventually, d becomes larger than t, making the GKR commitment asymptotically smaller than that of the hybrid approach. This explains why the GKR protocol is asymptotically the most efficient solution.

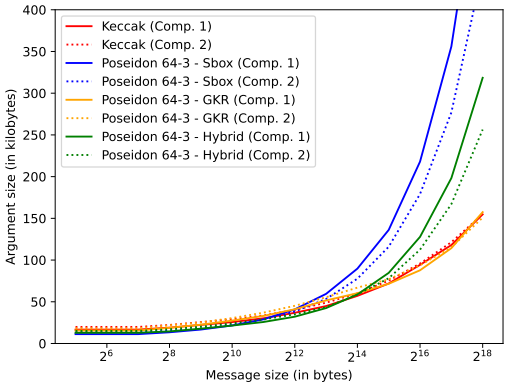
Let us mention that we omit Ligero and Longfellow ZK from this comparison to limit the number of reported results. Similarly to the case of arithmetic circuits, neither proof system achieves proof sizes as small as those considered here (below 100 KB over a 256-bit field or below 50 KB over a 64-bit field). Consequently, our proof system outperforms both schemes in the considered use cases.

Message Hashing. We report in Figure 17 the proof sizes obtained for message hashing, namely for proving knowledge of a message m of a given length that is a preimage of a public digest. We consider both Keccak and Poseidon, each based on sponge mode, with a 256-bit output. Each subfigure corresponds to a different set of Poseidon parameters and also includes the Keccak results, which are identical across subfigures.

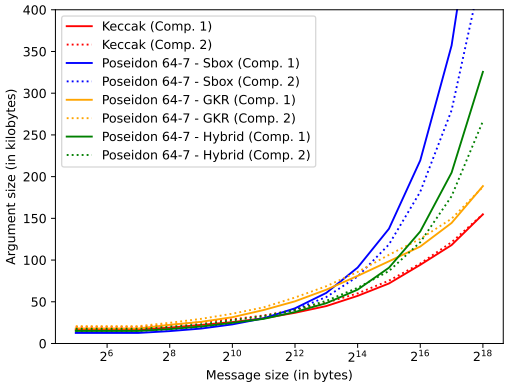
Interestingly, although Poseidon is specifically designed to be arithmetization-friendly while Keccak is not, our proof system generally achieves smaller proof sizes with Keccak than with Poseidon. This is

largely enabled by the ring-switching technique, which allows us to efficiently handle statements defined over $\mathbb{F}_2$ without incurring the proof size overhead that would occur otherwise (i.e. with an MPCS handling binary values).

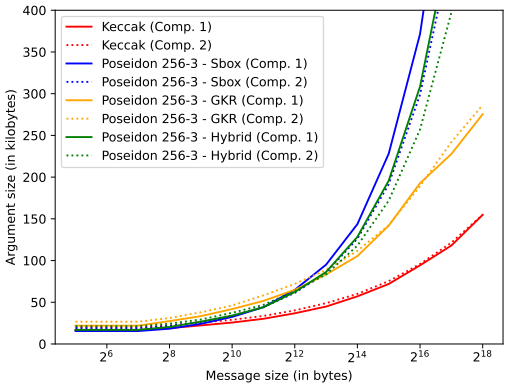
Nevertheless, proof size does not necessarily reflect proving efficiency. Although we do not yet have an optimized implementation of our proof system, we expect proving Keccak-based statements to be more computationally expensive than their Poseidon-based counterparts, since the cost of GKR scales with the number of multiplications per layer – 1 600 for Keccak, versus only 2 (partial round) to 80 (full round, degree-7 S-box) for Poseidon, i.e. 38 400 multiplications per Keccak permutation against 200–728 for Poseidon (a factor of more than 50). Keccak and Poseidon thus exhibit a trade-off between proof size and proving time.



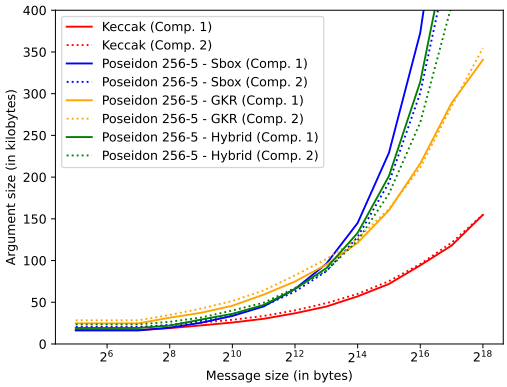
(a) Poseidon over a 64-bit field and S-box degree 3 ($t = 20$, $R_P = 22$)



(b) Poseidon over a 64-bit field and S-box degree 7 ($t = 20$, $R_P = 22$)



(c) Poseidon over a 256-bit field and S-box degree 3 ($t = 5$, $R_P = 60$)



(d) Poseidon over a 256-bit field and S-box degree 5 ($t = 5$, $R_P = 60$)

Fig. 17: Argument sizes for proving knowledge of a sponge-hash preimage (Merkle tree with 4096 leaves).

6.3 Lattices

We now apply SmallWood$^+$ to lattice problems, following the approach of SmallWood [FR26]. More specifically, we consider proving knowledge of a solution to a Learning With Errors (LWE) instance, namely two vectors $s \in \mathbb{F}_q^n$ and $e \in \mathbb{F}_q^m$ satisfying $\|s\|_\infty, \|e\|_\infty \leq \beta$ and $t = A \cdot s + e$, where $A \in \mathbb{F}_q^{m \times n}$ and $t \in \mathbb{F}_q^m$ are public.

The underlying sum-check-based protocol relies on the multilinear extensions $\hat{s}$ and $\hat{e}$ of the secret vector s and the error vector e, respectively. The first step is to prove that $\hat{s}$ and $\hat{e}$ encode values in the

interval $-\beta, \ldots, \beta$. This is achieved by using a batched sum-check protocol to verify that

$$\begin{cases} 0 = \sum_Y \widetilde{\mathrm{eq}}(X, Y) \cdot V_{\{-\beta,\ldots,+\beta\}}(\hat{s}(Y)), \\ 0 = \sum_Y \widetilde{\mathrm{eq}}(X, Y) \cdot V_{\{-\beta,\ldots,+\beta\}}(\hat{e}(Y)), \end{cases} \quad \text{where} \quad V_{\{-\beta,\ldots,+\beta\}}(X) := \prod_{i=-\beta}^{\beta} (X - i). \tag{15}$$

Let r denote the random evaluation point sampled by the sum-check protocol. At the end of this phase, the verifier must check that $v_s = \hat{s}(r)$ and $v_e = \hat{e}(r)$, and that the relation $t = A \cdot s + e$ holds. Assuming that a commitment to s is available, the verification can be delegated to the commitment scheme using the extended version of the SVCS instances. We formalize the protocol in Figure 18.

Similarly to SmallWood, we can use coefficient packing to reduce the size of the committed witness, at the cost of increasing the sum-check trace. We describe the entire protocol in Figure 19. In that case, the previous claims ($v_s = \hat{s}(r)$, $v_e = \hat{e}(r)$, and $t = A \cdot s + e$) will be checked through a batched sum-check protocol verifying

$$\begin{cases} v_e = \sum_Y \widetilde{A}(r, Y) \cdot \hat{s}(Y), \\ v_s = \sum_Y \widetilde{\mathrm{eq}}(r, Y) \cdot \hat{s}(Y), \end{cases}$$

where $\widetilde{A}$ denotes the bivariate multilinear extension of the matrix A. The second equation is included to merge the consistency check $v_s = \hat{s}(r)$ with the verification resulting from this second sum-check protocol. Let r' denote the random point generated by this second sum-check protocol. The verifier is then left with the task of checking that $v'_s = \hat{s}(r')$ for some claimed value v'_s. One can directly rely on the commitment scheme to perform those checks if s were committed without packing, but when using packing, we need to use the third sum-check protocol to transform the claim on the unpacked MLE into a claim on the corresponding packed MLE. The packing/unpacking strategy follows exactly the one described in SmallWood, which only uses polynomial relations (and not advanced techniques such as the ring-switching technique [DP26]).

LWE Instance					Proof Size		
Name	q	n	m	β	SmallWood	Compiler 1	Compiler 2
Toy Example 1	$\approx 2^{32}$	2048	512	1	**17.0 KB**	**17.0 KB** (2)	18.9 KB
Toy Example 2	$\approx 2^{61}$	4096	1024	Bin.	**21.4 KB**	22.1 KB (3)	25.3 KB
Kyber512	3329	2×256	2×256	3	14.1 KB	**12.1 KB** (1)	14.9 KB
Kyber768	3329	3×256	3×256	2	15.0 KB	**11.9 KB** (1)	14.9 KB
Kyber1024	3329	4×256	4×256	2	16.5 KB	**12.4 KB** (1)	15.5 KB
Dilithium2	8380417	4×256	4×256	2	17.5 KB	**13.2 KB** (1)	16.6 KB
Dilithium3	8380417	5×256	6×256	4	22.1 KB	**15.0 KB** (1)	18.7 KB
Dilithium5	8380417	7×256	8×256	2	22.7 KB	**15.5 KB** (1)	18.8 KB
Subset-Sum	$\approx 2^{256}$	256	1	Bin.	**12.6 KB**	18.4 KB (2)	22.6 KB

Table 3: Proof sizes when applied on some LWE instances. "Bin." means that the secret is a binary vector. The number in parenthesis correspond to the packing factor. The given sizes are the average measured sizes over 10 executions.

We apply SmallWood$^+$ 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+21a] instances. We additionally apply it 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 3.

For each instance, Table 3 reports the proof sizes obtained with our construction and compares them to those of SmallWood. While SmallWood$^+$ outperforms SmallWood for the Kyber and Dilithium parameter sets, this is not the case for the other instances. The reason is as follows. In both SmallWood and SmallWood$^+$, the main bottleneck in proof size is the commitment-opening phase, whose cost scales with the amount of committed data. In SmallWood, the prover commits to the LWE secret and the noise vectors. In contrast, our construction commits to the LWE secret together with the sum-check trace. Therefore, the comparison roughly amounts to comparing the size of the sum-check trace with the size of the noise vector. In many practical instances, the sum-check trace remains relatively small, which

explains the advantage of our approach for Kyber and Dilithium. However, when the noise vector is itself small, its size becomes comparable to that of the sum-check trace. This is the case for the toy examples considered in Table 3, where the secret vector s is large while the noise vector e is relatively small. In such settings, the amount of committed data is similar in both approaches, resulting in comparable proof sizes. Finally, for the subset-sum problem, there is no noise term at all. As a consequence, SmallWood becomes particularly efficient, whereas SmallWood$^+$ still incurs the cost of committing to the sum-check trace, leading to larger proofs.

Context:

- $\mathcal{P}$ has previously committed to a witness $s \in \mathbb{F}^n$.
- $\mathcal{P}$ claims that there exists $e \in \mathbb{F}^m$ such that $t = As + e$ with $\|s\|_\infty \leq \beta$ and $\|s\|_\infty \leq \beta$, where $A \in \mathbb{F}^{m \times n}$ and $b \in \mathbb{F}^n$ are public.
- ℓ and ℓ_s denotes respectively $\ell := \lceil \log_2(\max(n, m)) \rceil$ and $\ell_s := \lceil \log_2(n) \rceil$

Protocol:

- $\mathcal{V}$ chooses a random $r_0 \in \mathbb{K}^\ell$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ and $\mathcal{V}$ run the batched sum-check protocol on the two claim

$$\begin{cases} 0 = \sum_y \widetilde{\mathrm{eq}}(r_0, y) \cdot V_{\{-\beta,\ldots,+\beta\}}(\hat{s}(y)), \\ 0 = \sum_y \widetilde{\mathrm{eq}}(r_0, y) \cdot V_{\{-\beta,\ldots,+\beta\}}(\hat{e}(y)), \end{cases}$$

  where $V_{-\beta,\ldots,+\beta}$ is the vanishing polynomial defined in Equation (15), and where $\hat{s} \in \mathbb{F}^{\leq 1}[Y_1, \ldots, Y_\ell]$ and $\hat{e} \in \mathbb{F}^{\leq 1}[Y_1, \ldots, Y_\ell]$ are the dimension-ℓ multilinear extensions of s and e. At the end of the protocol, the claim is reduced to checking claims $\hat{s}(r) = v_s$ and $\hat{e}(r) = v_e$ for some $r \in \mathbb{K}^\ell$, $v_s \in \mathbb{K}$, and $v_e \in \mathbb{K}$.
- $\mathcal{P}$ convinces $\mathcal{V}$ that the committed witness satisfies $\hat{s}(r) = v_s$ and $\hat{e}(r) = v_e$, where $\hat{s}$ is the multilinear extension of s and $\hat{e}$ is the multilinear extension of $e = t - As$.

Fig. 18: Sum-check-based protocol to prove knowledge of a LWE solution (without packing).

Context:

- Public data: $A \in \mathbb{F}^{m \times n}$ and $b \in \mathbb{F}^n$
- $\mathcal{P}$ has previously committed to a witness $s_{\text{packed}} \in \mathbb{F}^{\lceil n/p \rceil}$.
- One denotes $s_j := \mathsf{Unpack}_j(s_{\text{packed}})$ for all $j \in \{1, \ldots, p\}$, $s := (s_1 \parallel \ldots \parallel s_p)$ and $e := t - As$.
- $\mathcal{P}$ claims that $t = As + e$ with $\|s\|_\infty \leq \beta$ and $\|s\|_\infty \leq \beta$.
- ℓ and ℓ_s denotes respectively $\ell := \lceil \log_2(\max(n/p, m)) \rceil$ and $\ell_s := \lceil \log_2(n/p) \rceil$

Protocol:

- $\mathcal{V}$ chooses a random $r_0 \in \mathbb{K}^\ell$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ and $\mathcal{V}$ run the batched sum-check protocol on the claims

$$\begin{cases} 0 = \sum_y \widetilde{\text{eq}}(r_0, y) \cdot V_{\{-\beta,\ldots,+\beta\}}(\hat{s}_1^{[\ell]}(y)), \\ \quad \vdots \\ 0 = \sum_y \widetilde{\text{eq}}(r_0, y) \cdot V_{\{-\beta,\ldots,+\beta\}}(\hat{s}_p^{[\ell]}(y)), \\ 0 = \sum_y \widetilde{\text{eq}}(r_0, y) \cdot V_{\{-\beta,\ldots,+\beta\}}(\hat{e}(y)), \end{cases}$$

  where $V_{-\beta,\ldots,+\beta}$ is the vanishing polynomial defined in Equation (15), and where $\hat{s}_j^{[\ell]} \in \mathbb{F}^{\leq 1}[Y_1, \ldots, Y_\ell]$ and $\hat{e} \in \mathbb{F}^{\leq 1}[Y_1, \ldots, Y_\ell]$ are the dimension-ℓ multilinear extensions of s_j and e. At the end of the protocol, the claim is reduced to checking claims $\hat{s}_j^{[\ell]}(r) = v_{s,j}$ and $\hat{e}(r) = v_e$ for some $r \in \mathbb{K}^\ell$, $v_{s,j} \in \mathbb{K}$, and $v_e \in \mathbb{K}$.
- $\mathcal{P}$ and $\mathcal{V}$ run the batched sum-check protocol on the two claim

$$\begin{cases} v_e &= \sum_y \sum_j \widetilde{A}_j(r, y) \cdot \hat{s}_j(y), \\ v_{s,1} &= \sum_y \widetilde{\text{eq}}(r, y) \cdot \hat{s}_1(y), \\ &\vdots \\ v_{s,p} &= \sum_y \widetilde{\text{eq}}(r, y) \cdot \hat{s}_p(y), \end{cases}$$

  where $\widetilde{A}_j \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_\ell, Y_1, \ldots, Y_{\ell_s}]$ and $\hat{s}_j \in \mathbb{F}^{\leq 1}[Y_1, \ldots, Y_{\ell_s}]$ are respectively the multilinear extensions of A_j and s_j. At the end of the protocol, the claim is reduced to checking claims $\hat{s}_j(r') = v'_{s_j}$ for some $r' \in \mathbb{K}^\ell$ and $v'_{s,j} \in \mathbb{K}$.
- $\mathcal{P}$ and $\mathcal{V}$ run the batched sum-check protocol on the two claim

$$\begin{cases} v'_{s,1} &= \sum_y \widetilde{\text{eq}}(r', y) \cdot \mathsf{Unpack}_1(\hat{s}_{\text{packed}}(y)), \\ &\vdots \\ v'_{s,p} &= \sum_y \widetilde{\text{eq}}(r', y) \cdot \mathsf{Unpack}_p(\hat{s}_{\text{packed}}(y)), \end{cases}$$

  where $\hat{s}_{\text{packed}} \in \mathbb{F}^{\leq 1}[Y_1, \ldots, Y_{\ell_s}]$ is the multilinear extension of s_{packed}. At the end of the protocol, the claim is reduced to checking claims $\hat{s}_{\text{packed}}(r'') = v''$ for some $r'' \in \mathbb{K}^\ell$ and $v'' \in \mathbb{K}$.
- $\mathcal{P}$ convinces $\mathcal{V}$ that the committed witness satisfies $\hat{s}_{\text{packed}}(r'') = v''$.

Fig. 19: Sum-check-based protocol to prove knowledge of a LWE solution (with packing).

References

AAB+24. Gor Adj, Nicolas Aragon, Stefano Barbero, Magali Bardet, Emmanuele Bellini, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Andre Esser, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Matthieu Rivain, Luis Rivera-Zamarripa, Carlo Sanna, Jean-Pierre Tillich, Javier Verbel, and Floyd Zweydinger. Mirath (merger of MIRA/MiRitH). Technical report, National Institute of Standards and Technology, 2024. available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures`.

ABB+24a. Najwa Aaraj, Slim Bettaieb, Loïc Bidoux, Alessandro Budroni, Victor Dyseryn, Andre Esser, Thibauld Feneuil, Philippe Gaborit, Mukul Kulkarni, Victor Mateu, Marco Palumbi, Lucas Perin, Matthieu Rivain, Jean-Pierre Tillich, and Keita Xagawa. PERK. Technical report, National Institute of Standards and Technology, 2024. available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures`.

ABB+24b. Carlos Aguilar Melchor, Slim Bettaieb, Loïc Bidoux, Thibauld Feneuil, Philippe Gaborit, Nicolas Gama, Shay Gueron, James Howe, Andreas Hülsing, David Joseph, Antoine Joux, Mukul Kulkarni, Edoardo Persichetti, Tovohery H. Randrianarisoa, Matthieu Rivain, and Dongze Yue. SDitH — Syndrome Decoding in the Head. Technical report, National Institute of Standards and Technology, 2024. available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures`.

ABB+24c. Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Romaric Neveu, Matthieu Rivain, Jean-Pierre Tillich, and Adrien Vinçotte. RYDE. Technical report, National Institute of Standards and Technology, 2024. available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures`.

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`.

ACFY24a. 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.

ACFY24b. Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. STIR: Reed–solomon proximity testing with fewer queries. Cryptology ePrint Archive, Report 2024/390, 2024.

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+24. Carsten Baum, Lennart Braun, Ward Beullens, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. FAEST. Technical report, National Institute of Standards and Technology, 2024. available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures`.

BBC+19. Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 67–97. Springer, Cham, August 2019.

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.

BBFR24. Ryad Benadjila, Charles Bouillaguet, Thibauld Feneuil, and Matthieu Rivain. MQOM — MQ on my Mind. Technical report, National Institute of Standards and Technology, 2024. available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures`.

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.

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.

BCF+17. Eli Ben-Sasson, Alessandro Chiesa, Michael A. Forbes, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Zero knowledge protocols from succinct constraint detection. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 172–206. Springer, Cham, November 2017.

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.

BDK+21a. 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`.

BDK+21b. Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: Short and fast signatures from AES. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 266–297. Springer, Cham, May 2021.

BGG+90. Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 37–56. Springer, New York, August 1990.

BMRS21. Carsten Baum, Alex J. Malozemoff, Marc B. Rosen, and Peter Scholl. Mac'n'cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 92–122, Virtual Event, August 2021. Springer, Cham.

CBBZ23. Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 499–530. Springer, Cham, April 2023.

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

CFS17. Alessandro Chiesa, Michael A. Forbes, and Nicholas Spooner. A zero knowledge sumcheck and its applications. Cryptology ePrint Archive, Report 2017/305, 2017.

CFW26. Alessandro Chiesa, Giacomo Fenzi, and Guy Weissenberg. Zero-knowledge IOPPs for constrained interleaved codes. Cryptology ePrint Archive, Paper 2026/391, 2026.

CMS19. Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 1–29. Springer, Cham, December 2019.

CMT12. Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In Shafi Goldwasser, editor, *ITCS 2012*, pages 90–112. ACM, January 2012.

DHRR26. Rahul Dalal, Tamir Hemo, Eugene Rabinovich, and Ron Rothblum. VEIL: Lightweight zero-knowledge for hash-based multilinear proof systems. Cryptology ePrint Archive, Paper 2026/683, 2026.

DLL+17. Léo Ducas, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS – Dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, Report 2017/633, 2017.

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

DP26. Benjamin E. Diamond and Jim Posen. Polylogarithmic proofs for multilinears over binary towers. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part VII*, volume 16547 of *LNCS*, pages 3–32. Springer, Cham, May 2026.

Fas26. Matteo Frigo and abhi shelat. Anonymous credentials from ECDSA. *IACR Communications in Cryptology*, 3(1), 2026.

FKR26. Eden Florentz Konopnicki and Ron D. Rothblum. Succinct zero-knowledge proofs from one-way functions: The blackbox way. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology – CRYPTO 2026*, pages 157–188, Cham, 2026. Springer Nature Switzerland.

FR25. 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.

FR26. Thibauld Feneuil and Matthieu Rivain. Smallwood: Hash-based polynomial commitments and zero-knowledge arguments for relatively small instances. *IACR Communications in Cryptology*, 3(2), 2026.

FS26. Matteo Frigo and Abhi Shelat. Longfellow zk. Internet-Draft draft-google-cfrg-libzk-02, July 2026. Work in Progress.

GKR08. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 113–122. ACM Press, May 2008.

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.

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.

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

Lon26. Jieyi Long. Interstellar: Efficient GKR-based IVC scheme with privacy-preserving collaborative folding. In Shi Bai and Edoardo Persichetti, editors, *PKC 2026, Part III*, volume 16553 of *LNCS*, pages 159–196. Springer, Cham, May 2026.

Lyu09. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Berlin, Heidelberg, December 2009.

RW24. Noga Ron-Zewi and Mor Weiss. Zero-knowledge IOPs approaching witness length. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 105–137. Springer, Cham, August 2024.

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

Tha13. Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 71–89. Springer, Berlin, Heidelberg, August 2013.

Tha25. Justin Thaler. Sum-check is all you need: An opinionated survey on fast provers in SNARK design. Cryptology ePrint Archive, Report 2025/2041, 2025.

WJB+17. Riad S. Wahby, Ye Ji, Andrew J. Blumberg, abhi shelat, Justin Thaler, Michael Walfish, and Thomas Wies. Full accounting for verifiable outsourcing. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2071–2086. ACM Press, October / November 2017.

WTs+18. Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, May 2018.

XZS22. Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero knowledge proof with linear prover time. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 299–328. Springer, Cham, August 2022.

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

ZXZS20. Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *2020 IEEE Symposium on Security and Privacy*, pages 859–876. IEEE Computer Society Press, May 2020.

A Use of Generative AI

Tools and period. OpenAI ChatGPT (GPT-5.6 Sol) and Anthropic Claude (Opus 5, Fable 5), used between April and November 2026.

Language editing. Used throughout the paper to improve the grammar, clarity and phrasing of author-written text.

Expository drafting. Parts of the introduction (motivation, technical overview, summary of contributions) and of the related-work discussion were drafted by the tools from author-provided content and outlines, and subsequently edited by the authors.

Round-by-round knowledge soundness definition. The adaptation of round-by-round knowledge soundness to commit-and-prove systems in the ROM (Section 2.3) was assisted by Claude. The authors provided the main idea of the definition (distinct states before and after the arrival of the statement, the allowed transitions between them) and Claude drafted its successive formulations, identifying along the way subtleties in the treatment of the prover's hash queries. The final formalism (transcripts extended with hash queries, the state $\perp$ and the stability condition) follows simplifications from the authors. Claude produced an initial draft of the associated discussion and Figure 3 based on the authors guidelines. The authors verified, revised and are responsible for the final definition, its discussion and the associated figure.

Security proofs. Initial drafts of Definition 13, Lemma 3 and of the proofs of Theorems 1, 2, 3, 4 and 5 were produced by Claude from the theorem statements, definitions and proof strategies supplied by the authors. These proofs apply standard techniques – round-by-round soundness analysis, witness extraction from the underlying commitment schemes, and simulation-based zero knowledge – and are provided mostly for completeness; the arguments they formalise are not claimed among the conceptual contributions of this work. Each draft was verified and revised by the authors, who are responsible for the final statements and arguments.

Other technical results. The construction and analysis underlying Lemma 2 were developed through discussion with the tools, and parts of the initial written argument originated from them but were substantially revised by the authors; the statement, the bound and the final proof are the authors'.

Technical review. The tools were used as a reviewing aid on the full LaTeX source, to surface candidate errors, inconsistencies, notational clashes and broken cross-references. All reported issues were adjudicated by the authors, and a number of the reported items concerned author-written proofs, which the authors then corrected.

Prompts. The interaction took the form of an extended, iterative dialogue over the manuscript sources rather than a set of discrete prompts. Representative prompts are reproduced below.

> *(i) Draft a definition of round-by-round knowledge soundness for sum-check-based interactive proofs in the CBIP setting, together with a lemma stating that the CBIP transformation preserves it, and a proof. Use the formalism of the attached sources.*
>
> *(ii) Draft the proof of the theorem on the security of Compiler 1. Strategy: combine the round-by-round knowledge soundness of the underlying SC-IP with the extractability of the linear constraint system, and account separately for the evaluation-merging step. Use the notation of the preceding sections.*
>
> *(iii) Here is the contributions list and technical overview I sketched. Rewrite them as flowing prose for the introduction, preserving my ordering and claims and adding none of your own.*
>
> *(iv) Read the full LaTeX source and report correctness, completeness and consistency issues, prioritising soundness and zero-knowledge problems over typographical ones.*

B Optimized Sum-Check Protocol

For the sake of simplicity, we assume the degree of the intermediate polynomials g_i to be fixed and equal to d for all i. We slightly abuse notations and write $g_i(1), \ldots, g_i(d)$ to refer to the evaluations of g_i at d distinct points of the field $\mathbb{K}$.

The following steps are repeated for $i = 1, \ldots, \ell$:

- $\mathcal{P}$ sends evaluations $g_i(1), \ldots, g_i(d)$ of the polynomial

$$g_i(X_i) = \sum_{x_{i+1}, \ldots, x_\ell} g(r_1, \ldots, r_{i-1}, X_i, x_{i+1}, \ldots, x_\ell) \ .$$

- $\mathcal{V}$ lets

$$g_i(0) = -g_i(1) + \begin{cases} c & \text{if } i = 1 \ , \\ g_{i-1}(r_{i-1}) & \text{if } i > 1 \end{cases}$$

  and retrieves $g_i(X_i)$ from $g_i(0), \ldots, g_i(d)$ by interpolation.
- $\mathcal{V}$ samples a field element r_i and sends it to $\mathcal{P}$.

End of the repetition.

- $\mathcal{V}$ computes $c' = g_\ell(r_\ell)$.
- $\mathcal{P}$ and $\mathcal{V}$ runs a protocol to check that

$$c' = g(r_1, r_2, \ldots, r_\ell) \ .$$

Fig. 20: Optimized sum-check protocol.

C Protocols in the CBIP Paradigm

C.1 From Layered Circuits to SVCS: GKR in the CBIP Paradigm

The GKR protocol with homogeneous degree-d gates (Section 2.2) can be run in the CBIP paradigm and reduced to an extended SVCS instance, as depicted in Figure 21. The prover has previously committed to a witness $w \in \mathbb{F}^\omega$, together with a trace mask t, such that the input x and output y of the circuit can be expressed from w through public affine maps, as in the definition of the extended SVCS (Section 3.1): $x = A_x \cdot w + b_x$ and $y = A_y \cdot w + b_y$. Typically, $w = (w_x, w_y)$ where w_x and w_y are the private parts of the input and output, i.e., $x = (w_x, x')$ and $y = (w_y, y')$ for public parts x' and y': the maps (A_x, b_x) and (A_y, b_y) then simply select the corresponding coordinates of w and inject the public values. More involved relations are also possible: for instance, with the circuit-flattening technique (Section 5.3), the witness contains the intermediate states of the computation, and the affine maps read the same coordinates of w both as the output of one block and as the input of the next one, which enforces their equality at no extra cost. Each iteration of the protocol invokes the masked sum-check protocol of Figure 5 – on $d \cdot \ell_{i-1}$ variables with individual degree 2, hence consuming $2\,d\,\ell_{i-1}$ mask elements – and produces one degree-d trace constraint in $\mathcal{F}$. The initial claim on the output and the final claims on the input translate into multilinear evaluation constraints on $\widetilde{y}$ and $\widetilde{x}$ in $\mathcal{E}$, i.e., on the multilinear extensions of affine functions of the committed witness.

Context:

- $\mathcal{P}$ has previously committed to a witness $w \in \mathbb{F}^\omega$ and a trace mask $t \in \mathbb{K}^\tau$. The input $x \in \mathbb{F}^{2^{\ell_0}}$ and output $y \in \mathbb{F}^{2^{\ell_n}}$ of the circuit are expressed from the witness through public affine maps $x = A_x \cdot w + b_x$ and $y = A_y \cdot w + b_y$ (typically selecting the private coordinates of x and y from w and injecting their public parts).
- The trace mask t is made of $\tau = 1 + n\,d + 2\,d\sum_{i=0}^{n-1} \ell_i$ elements labelled as t_{out}, $\{t_{i,m}\}_{0\le i\le n-1,\,1\le m\le d}$ and $\{t^{(\text{sc},i)}_{k,j}\}_{1\le k\le d\,\ell_{i-1},\,1\le j\le 2}$ for $i = 1, \dots, n$, in the following description.
- $\mathcal{P}$ claims that (x, y) satisfies a layered circuit instance $\mathfrak{x}_{\text{GKR}} = (\mathbb{F}, n, \{\ell_i\}_{0\le i\le n}, d, \{F_i\}_{1\le i\le n})$, i.e. that $y = C(x)$.
- At the end of the protocol the claim is reduced to checking an extended SVCS instance $\mathfrak{x}_{\text{SVCS}} = (\mathbb{F}, \mathbb{K}, \omega, \tau, d, \mathcal{E}, \mathcal{F})$ on the witness (w, t), where $\mathcal{E}$ contains one evaluation constraint on $\widetilde{y}$ (through the map (A_y, b_y)) and d evaluation constraints on $\widetilde{x}$ (through the map (A_x, b_x)), and $\mathcal{F}$ contains n degree-d trace constraints, as defined below.

Protocol:

- $\mathcal{V}$ samples a random point $s_n \in \mathbb{K}^{\ell_n}$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ computes $c_n = \tilde{y}(s_n)$ and sends $m_{\text{out}} = c_n + t_{\text{out}}$ to $\mathcal{V}$, claiming $c_n = \tilde{y}(s_n)$ and $c_n = V_n(s_n)$. Let $\psi_n : \mathbb{K}^\tau \to \mathbb{K}$ be the affine map satisfying $\psi_n(t) = m_{\text{out}} - t_{\text{out}}$.
- $\mathcal{P}$ and $\mathcal{V}$ add the constraint $\tilde{y}(s_n) = \psi_n(t)$ to the SVCS instance $\mathfrak{x}_{\text{SVCS}}$ by adding (s_n, ψ_n) to the set $\mathcal{E}$. The rest of the protocol is dedicated to checking the claim $V_n(s_n) = \psi_n(t)$.

  The following steps are repeated for $i = n$ down to 1. They reduce a claim on the layer-i evaluations to a claim on the layer-$(i-1)$ evaluations.

- $\mathcal{P}$ and $\mathcal{V}$ define the virtual polynomial g_i and the affine map $\hat{\psi}_i : \mathbb{K}^\tau \to \mathbb{K}$ as

$$g_i(j_1, \dots, j_d) := \begin{cases} f_n(s_n, j_1, \dots, j_d) & \text{if } i = n, \\ \sum_{m=1}^{d} \alpha_{i,m} \cdot f_i(r_{i+1,m}, j_1, \dots, j_d) & \text{otherwise,} \end{cases} \qquad \hat{\psi}_i := \begin{cases} \psi_n & \text{if } i = n, \\ \psi_i^{\text{lc}} & \text{otherwise,} \end{cases}$$

  where f_i is defined in Equation (2) (and $\alpha_{i,m}$, $r_{i+1,m}$, ψ_i^{lc} come from the previous iteration). The current claim is

$$\sum_{j_1,\dots,j_d \in \{0,1\}^{\ell_{i-1}}} g_i(j_1, \dots, j_d) = \hat{\psi}_i(t)\ .$$

- $\mathcal{P}$ and $\mathcal{V}$ run the sum-check protocol (Figure 5) on this claim, using the trace mask elements $\{t^{(\text{sc},i)}_{k,j}\}$ to mask the messages. At the end of the sum-check protocol, the claim is reduced to $g_i(r_{i,1}, \dots, r_{i,d}) = \psi_i'(t)$ for random points $r_{i,1}, \dots, r_{i,d} \in \mathbb{K}^{\ell_{i-1}}$ chosen by the verifier and $\psi_i' : \mathbb{K}^\tau \to \mathbb{K}$ an affine map derived from $\hat{\psi}_i$ and the prover messages.
- $\mathcal{P}$ sends masked claimed evaluations $m_{i-1,m} = v_{i-1,m} + t_{i-1,m}$ with $v_{i-1,m} := V_{i-1}(r_{i,m})$ for $m = 1, \dots, d$ to $\mathcal{V}$. Let $\psi_{i-1,m} : \mathbb{K}^\tau \to \mathbb{K}$ be the affine maps satisfying $\psi_{i-1,m}(t) = m_{i-1,m} - t_{i-1,m}$.
- $\mathcal{P}$ and $\mathcal{V}$ compute the public scalar

$$\beta_i := \begin{cases} F_n(s_n, r_{n,1}, \dots, r_{n,d}) & \text{if } i = n, \\ \sum_{m=1}^{d} \alpha_{i,m} \cdot F_i(r_{i+1,m}, r_{i,1}, \dots, r_{i,d}) & \text{otherwise,} \end{cases}$$

  and the function f_i^{chk} defined as

$$f_i^{\text{chk}}(t) := \beta_i \cdot \prod_{u=1}^{d} \psi_{i-1,u}(t) \;-\; \psi_i'(t)\ ,$$

  and add f_i^{chk} to the set $\mathcal{F}$ of $\mathfrak{x}_{\text{SVCS}}$. The claim is now reduced to checking $V_{i-1}(r_{i,m}) = \psi_{i-1,m}(t)$ for all $m = 1, \dots, d$.
- If $i > 1$ (*this is not the last iteration*), $\mathcal{V}$ samples random field elements $\alpha_{i-1,1}, \dots, \alpha_{i-1,d} \in \mathbb{K}$ and sends them to $\mathcal{P}$. $\mathcal{P}$ and $\mathcal{V}$ compute

$$\psi_{i-1}^{\text{lc}} := \sum_{m=1}^{d} \alpha_{i-1,m} \cdot \psi_{i-1,m}\ .$$

  The d claims are reduced to the single claim

$$\sum_{m=1}^{d} \alpha_{i-1,m} \cdot V_{i-1}(r_{i,m}) = \psi_{i-1}^{\text{lc}}(t)\ ,$$

  which implies the d individual claims with probability at least $1 - 1/|\mathbb{K}|$ over the randomness of $\alpha_{i-1,1}, \dots, \alpha_{i-1,d}$. This claim is checked by the next iteration.

  End of the repetition. At this stage, the claim is reduced to checking $V_0(r_{1,m}) = \psi_{0,m}(t)$ for $m = 1, \dots, d$ and some random points $r_{1,1}, \dots, r_{1,d} \in \mathbb{K}^{\ell_0}$.

- Since $V_0 = \tilde{x}$, $\mathcal{P}$ and $\mathcal{V}$ add the constraints $\tilde{x}(r_{1,m}) = \psi_{0,m}(t)$ for $m = 1, \dots, d$ to the SVCS instance $\mathfrak{x}_{\text{SVCS}}$ by adding $(r_{1,m}, \psi_{0,m})$ for $m = 1, \dots, d$ to the set $\mathcal{E}$.

Fig. 21: From layered circuits to SVCS with GKR in the CBIP paradigm.

C.2 R1CS to SVCS Reduction in the CBIP Paradigm

Context:

- $\mathcal{P}$ has previously committed to a witness $w \in \mathbb{F}^\omega$ and a trace mask $t \in \mathbb{K}^\tau$.
- The trace mask t is made of $\tau = 5\ell + 3$ elements labelled as t_A, t_B, t_C, $\{t^{(\mathrm{sc1})}_{i,j}\}_{1\le i\le \ell, 1\le j\le 3}$ and $\{t^{(\mathrm{sc2})}_{i,j}\}_{1\le i\le \ell, 1\le j\le 2}$, in the following description.
- $\mathcal{P}$ claims that w satisfies an R1CS instance $\mathbb{x}_{\mathrm{R1CS}} = (\mathbb{F}, \omega, A, B, C)$.
- At the end of the protocol the claim is reduced to checking an SVCS instance $\mathbb{x}_{\mathrm{SVCS}}$ on the witness (w, t), where $\mathbb{x}_{\mathrm{SVCS}} = (\mathbb{F}, \mathbb{K}, \omega, \tau, d, \mathcal{E}, \mathcal{F})$ with $\mathcal{E} = \{(r_2, \psi'_2)\}$ and $\mathcal{F} = \{f\}$ as defined below.

Protocol:

- $\mathcal{V}$ chooses a random $\gamma \in \mathbb{K}^\ell$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ and $\mathcal{V}$ run the sum-check protocol (Figure 5) on the claim

$$\sum_{x\in\{0,1\}^\ell} g_1(x) = 0 \quad \text{with} \quad g_1(x) = \widetilde{\mathrm{eq}}(\gamma, x) \cdot h(x)$$

  where h is the polynomial defined from the R1CS instance and the witness w in Equation (13). This protocol uses the trace mask elements $\{t^{(\mathrm{sc1})}_{i,j}\}$ to mask the messages. At the end of the protocol, the claim is reduced to checking a claim $g_1(r_1) = \langle \psi_1, t\rangle$ for some $r_1 \in \mathbb{K}^\ell$ and $\psi_1 \in \mathbb{K}^\tau$.
- $\mathcal{P}$ computes

$$c_A = \sum_{j\in\{0,1\}^\ell} \tilde{A}(r_1, j) \cdot \tilde{w}(j)\ , \quad c_B = \sum_{j\in\{0,1\}^\ell} \tilde{B}(r_1, j) \cdot \tilde{w}(j)\ , \quad c_C = \sum_{j\in\{0,1\}^\ell} \tilde{C}(r_1, j) \cdot \tilde{w}(j)\ .$$

  and sends $m_A = c_A + t_A$, $m_B = c_B + t_B$ and $m_C = c_C + t_C$ to $\mathcal{V}$.
- $\mathcal{P}$ and $\mathcal{V}$ compute the function f defined as

$$f(t) := \widetilde{\mathrm{eq}}(\gamma, r_1) \cdot \big((m_A - t_A) \cdot (m_B - t_B) - (m_C - t_C)\big) - \langle \psi_1, t\rangle$$

  and add f to the set $\mathcal{F}$ of the SVCS instance $\mathbb{x}_{\mathrm{SVCS}}$.
- $\mathcal{V}$ chooses random $\gamma_A, \gamma_B, \gamma_C \in \mathbb{K}$ and sends them to $\mathcal{P}$.
- Let g_2 be the polynomial defined as

$$g_2(j) := \big(\gamma_A \cdot \tilde{A}(r_1, j) + \gamma_B \cdot \tilde{B}(r_1, j) + \gamma_C \cdot \tilde{C}(r_1, j)\big) \cdot \tilde{w}(j)\ ,$$

  and let $\hat{\psi} \in \mathbb{K}^\tau$ be the vector satisfying

$$\langle \hat{\psi}, t\rangle = \gamma_A \cdot (m_A - t_A) + \gamma_B \cdot (m_B - t_B) + \gamma_C \cdot (m_C - t_C)\ .$$

  $\mathcal{P}$ and $\mathcal{V}$ run the sum-check protocol on the claim

$$\sum_{j\in\{0,1\}^\ell} g_2(j) = \langle \hat{\psi}, t\rangle$$

  using the trace mask elements $\{t^{(\mathrm{sc2})}_{i,j}\}$ to mask the messages. At the end of the protocol, the claim is reduced to checking $g_2(r_2) = \langle \psi_2, t\rangle$ for some $r_2 \in \mathbb{K}^\ell$ and $\psi_2 \in \mathbb{K}^\tau$.
- By definition of g_2, we have $g_2(r_2) = \langle \psi_2, t\rangle \Leftrightarrow \tilde{w}(r_2) = \langle \psi'_2, t\rangle$ where

$$\psi'_2 = \big(\gamma_A \cdot \tilde{A}(r_1, r_2) + \gamma_B \cdot \tilde{B}(r_1, r_2) + \gamma_C \cdot \tilde{C}(r_1, r_2)\big)^{-1} \cdot \psi_2\ .$$

  $\mathcal{P}$ and $\mathcal{V}$ add the constraint $\tilde{w}(r_2) = \langle \psi'_2, t\rangle$ to the SVCS instance $\mathbb{x}_{\mathrm{SVCS}}$ by adding (r_2, ψ'_2) to the set $\mathcal{E}$ of $\mathbb{x}_{\mathrm{SVCS}}$.

Fig. 22: R1CS to SVCS reduction in the CBIP paradigm.

C.3 S-Box Circuit to SVCS Reduction in the CBIP Paradigm

Context:

- $\mathcal{P}$ has previously committed to a witness $w \in \mathbb{F}^\omega$ and a trace mask $t \in \mathbb{K}^\tau$.
- The trace mask t is made of $\tau = (d+3)\cdot\ell + 2$ elements labelled as t_A, t_B, $\{t_{i,j}^{(\text{sc1})}\}_{1\le i\le \ell, 1\le j\le d+1}$ and $\{t_{i,j}^{(\text{sc2})}\}_{1\le i\le \ell, 1\le j\le 2}$, in the following description.
- $\mathcal{P}$ claims that w satisfies an S-box circuit instance $\mathbb{x}_{\text{S-BOX}} = (\mathbb{F}, \omega, A, B, S)$.
- At the end of the protocol the claim is reduced to checking an SVCS instance $\mathbb{x}_{\text{SVCS}}$ on the witness (w, t), where $\mathbb{x}_{\text{SVCS}} = (\mathbb{F}, \mathbb{K}, \omega, \tau, d, \mathcal{E}, \mathcal{F})$ with $\mathcal{E} = \{(r_2, \psi_2')\}$ and $\mathcal{F} = \{f\}$ as defined below.

Protocol:

- $\mathcal{V}$ chooses a random $\gamma \in \mathbb{K}^\ell$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ and $\mathcal{V}$ run the sum-check protocol (Figure 5) on the claim

$$\sum_{x\in\{0,1\}^\ell} g_1(x) = 0 \quad \text{with} \quad g_1(x) = \widetilde{\text{eq}}(\gamma, x)\cdot h(x)$$

  where h is the polynomial defined from the S-box circuit instance and the witness w in Equation (14). This protocol uses the trace mask elements $\{t_{i,j}^{(\text{sc1})}\}$ to mask the messages. At the end of the protocol, the claim is reduced to checking a claim $g_1(r_1) = \langle \psi_1, t\rangle$ for some $r_1 \in \mathbb{K}^\ell$ and $\psi_1 \in \mathbb{K}^\tau$.
- $\mathcal{P}$ computes

$$c_A = \sum_{j\in\{0,1\}^\ell} \tilde{A}(r_1, j)\cdot\tilde{w}(j)\ , \quad c_B = \sum_{j\in\{0,1\}^\ell} \tilde{B}(r_1, j)\cdot\tilde{w}(j)\ ,$$

  and sends $m_A = c_A + t_A$ and $m_B = c_B + t_B$ to $\mathcal{V}$.
- $\mathcal{P}$ and $\mathcal{V}$ compute the function f defined as

$$f(t) := \widetilde{\text{eq}}(\gamma, r_1)\cdot\big(S(m_A - t_A) - (m_B - t_B)\big) - \langle\psi_1, t\rangle$$

  and add f to the set $\mathcal{F}$ of the SVCS instance $\mathbb{x}_{\text{SVCS}}$.
- $\mathcal{V}$ chooses random $\gamma_A, \gamma_B \in \mathbb{K}$ and sends them to $\mathcal{P}$.
- Let g_2 be the polynomial defined as

$$g_2(j) := \big(\gamma_A\cdot\tilde{A}(r_1, j) + \gamma_B\cdot\tilde{B}(r_1, j)\big)\cdot\tilde{w}(j)$$

  and let $\hat{\psi} \in \mathbb{K}^\tau$ be the vector satisfying

$$\langle\hat{\psi}, t\rangle = \gamma_A\cdot(m_A - t_A) + \gamma_B\cdot(m_B - t_B) + \gamma_C\cdot(m_C - t_C)\ .$$

  $\mathcal{P}$ and $\mathcal{V}$ run the sum-check protocol on the claim

$$\sum_{j\in\{0,1\}^\ell} g_2(j) = \langle\hat{\psi}, t\rangle$$

  using the trace mask elements $\{t_{i,j}^{(\text{sc2})}\}$ to mask the messages. At the end of the protocol, the claim is reduced to checking $g_2(r_2) = \langle\psi_2, t\rangle$ for some $r_2 \in \mathbb{K}^\ell$ and $\psi_2 \in \mathbb{K}^\tau$.
- By definition of g_2, we have $g_2(r_2) = \langle\psi_2, t\rangle \Leftrightarrow \tilde{w}(r_2) = \langle\psi_2', t\rangle$ where

$$\psi_2' = \big(\gamma_A\cdot\tilde{A}(r_1, r_2) + \gamma_B\cdot\tilde{B}(r_1, r_2)\big)^{-1}\cdot\psi_2\ .$$

  $\mathcal{P}$ and $\mathcal{V}$ add the constraint $\tilde{w}(r_2) = \langle\psi_2', t\rangle$ to the SVCS instance $\mathbb{x}_{\text{SVCS}}$ by adding (r_2, ψ_2') to the set $\mathcal{E}$ of $\mathbb{x}_{\text{SVCS}}$.

Fig. 23: S-box circuit to SVCS reduction in the CBIP paradigm.

D Security Proofs

D.1 Round-by-Round Knowledge Soundness of the DECS

We prove Theorem 1. The HVZK part is [FR26, Thm. 2]: the only difference with the scheme analyzed there is that the prover sends $h_R = \mathsf{Hash}(\boldsymbol{R})$ and $\boldsymbol{R}(S)$ instead of $\boldsymbol{R}$, so that the simulator, which samples $\boldsymbol{R}$ itself, computes h_R honestly and needs no further programming. We focus on round-by-round knowledge soundness, which adapts the extractable-binding proof of [FR26, Thm. 1]. The DECS is a CP system for the relation

$$\mathcal{R}_{\mathsf{decs}} \;=\; \big\{(\boldsymbol{P},(E,\boldsymbol{v})) \;:\; \boldsymbol{P}\in(\mathbb{F}[X]^{\le d})^n,\; E\subseteq\mathbb{E},\; \boldsymbol{P}(E)=\boldsymbol{v}\big\}\;.$$

All hash computations (leaf hashes, Merkle nodes, h_R) are queries to the random oracle, with 2λ-bit answers. Given an extended transcript θ, a *preimage* of a digest is a hash query of θ of the expected format answered by that digest: a pair of digests for a Merkle node, a tuple $(\boldsymbol{p},\boldsymbol{m},j,\rho)\in\mathbb{F}^n\times\mathbb{F}^\eta\times[N]\times\{0,1\}^\lambda$ for the leaf at position j, and a vector polynomial in $(\mathbb{F}[X]^{\le d})^\eta$ for h_R.

Proof (Proof of Theorem 1 (RBR knowledge soundness)). Bad transcripts. We let $\mathsf{State}(\theta):=\perp$ whenever some hash query of θ is answered by a digest which already appears earlier in θ: as the answer of an earlier hash query (a collision), in the input of an earlier one, or in an earlier prover message (root or h_R). This is preserved by extension, which gives the first part of condition 6 of Definition 8, and it never happens in an honest execution, where the preimages of root and h_R are queried before these values are sent. In a good transcript, every digest has at most one preimage, and a digest appearing in a prover message or in the input of a query has its preimage, if any, queried *before* that message or query. In particular, along any chain of preimages from root down to a leaf, the queries were made bottom-up and before root was sent, and the preimage of h_R, if any, was queried before h_R was sent.

Extraction data. Let θ be a good commit transcript whose message part contains root. Parsing the hash queries of θ from root downwards (a node without preimage has a $\perp$ subtree), let $J(\theta)\subseteq[N]$ be the positions whose leaf has a preimage $(\boldsymbol{p}_j,\boldsymbol{m}_j,j,\rho_j)$. If θ contains h_R, let $\hat{\boldsymbol{R}}(\theta)$ be its preimage, or $\perp$ if it has none. By the remark above, $J(\theta)$, the extracted leaves $(\boldsymbol{p}_j,\boldsymbol{m}_j)_{j\in J(\theta)}$ and $\hat{\boldsymbol{R}}(\theta)$ only depend on the hash queries of θ made before root, resp. h_R, was sent. Finally, if θ contains Γ, let $\mathsf{Deg}(\Gamma,\theta)$ be the event that there exists $E'\subseteq\{e_j\}_{j\in J(\theta)}$ with $|E'|=d+2$ such that

$$\mathrm{coeff}_{d+1}\big(\boldsymbol{P}^{(E')}\big)\neq 0 \qquad\text{and}\qquad \mathrm{coeff}_{d+1}\big(\boldsymbol{M}^{(E')}+\Gamma\cdot\boldsymbol{P}^{(E')}\big)=0\;,$$

where $\boldsymbol{P}^{(E')}$ and $\boldsymbol{M}^{(E')}$ denote the interpolants (of degree at most $d+1$) of $e_j\mapsto\boldsymbol{p}_j$ and $e_j\mapsto\boldsymbol{m}_j$ over E', and coeff_{d+1} the vector of degree-$(d+1)$ coefficients. This is the event bounded in [FR26, Eq. (1)–(2)]; it depends on Γ and on the extracted leaves only, not on h_R.

State function and extractor. For a good (possibly empty or partial) commit transcript θ, we define

$$\mathsf{State}(\theta)=\begin{cases}\mathsf{cheating} & \text{if } \Gamma\in\theta \text{ and } \mathsf{Deg}(\Gamma,\theta);\\ \mathsf{trashed} & \text{otherwise, if the message part of } \theta \text{ is complete and } \hat{\boldsymbol{R}}(\theta)=\perp;\\ \mathsf{legit} & \text{otherwise.}\end{cases}$$

On a legit complete commit transcript θ, the extractor $\mathsf{Ext}_{\mathsf{com}}$ computes the *openable* set

$$E_{\mathsf{op}}(\theta)\;:=\;\big\{e_j\;:\;j\in J(\theta),\;\hat{\boldsymbol{R}}(e_j)=\boldsymbol{m}_j+\Gamma\cdot\boldsymbol{p}_j\big\}$$

and outputs the interpolant $\hat{\boldsymbol{P}}(\theta)$ of $e_j\mapsto\boldsymbol{p}_j$ over $E_{\mathsf{op}}(\theta)$, of degree less than $|E_{\mathsf{op}}(\theta)|$ (the zero polynomial if this set is empty); on a trashed, cheating or bad transcript, it outputs $\perp$. We claim that *on a legit transcript,* $\deg\hat{\boldsymbol{P}}\le d$. Indeed, assume that some coordinate of $\hat{\boldsymbol{P}}$ has degree greater than d, so that $|E_{\mathsf{op}}|\ge d+2$. Then some $E'\subseteq E_{\mathsf{op}}$ of size $d+2$ satisfies $\mathrm{coeff}_{d+1}(\boldsymbol{P}^{(E')})\neq 0$: otherwise, fixing $E''\subseteq E_{\mathsf{op}}$ of size $d+1$, the interpolants over $E''\cup\{e\}$ for $e\in E_{\mathsf{op}}\setminus E''$ would all have degree at most d, hence all coincide with the interpolant over E'', and so would $\hat{\boldsymbol{P}}$. For such an E', the polynomials $\hat{\boldsymbol{R}}$ and $\boldsymbol{M}^{(E')}+\Gamma\cdot\boldsymbol{P}^{(E')}$ have degree at most $d+1$ and agree on the $d+2$ points of E', hence are equal;

since $\deg \hat{\boldsymbol{R}} \leq d$, this gives $\mathsf{Deg}(\Gamma, \theta)$, contradicting legitimacy. Finally, for a good prove transcript $\theta = \theta_{\mathsf{com}} \| x \| \theta_{\mathsf{prov}}$ with $x = (E, \boldsymbol{v})$, we define $\mathsf{State}(\theta) = \mathsf{alive}$ if and only if

$$\mathsf{State}(\theta_{\mathsf{com}}) = \mathsf{cheating} \quad \text{or} \quad \big(\mathsf{State}(\theta_{\mathsf{com}}) = \mathsf{legit} \ \wedge \ \hat{\boldsymbol{P}}(\theta_{\mathsf{com}})(E) = \boldsymbol{v}\big) \ ,$$

and $\mathsf{State}(\theta) = \mathsf{doomed}$ otherwise. Note that this state depends neither on θ_{prov} nor on the hash queries made after x.

Conditions of Definition 8. (1) The empty transcript is good and legit. (2) In the commit phase, the cheating state depends only on the extracted leaves and on the verifier message Γ, not on the prover messages root and h_R. (3) By definition. (4) The state ignores θ_{prov}. (5) Let θ be a good transcript whose message part is full and accepted by $\mathcal{V}$, and which contains the hash queries of $\mathcal{V}$; we show that θ is alive. If $\mathsf{State}(\theta_{\mathsf{com}}) = \mathsf{cheating}$ this holds by definition, so assume otherwise. The authentication paths accepted by $\mathcal{V}$ are chains of preimages in θ, which, θ being good, were queried before root was sent and are those found when parsing from root: for every $e_j \in E$, $j \in J(\theta_{\mathsf{com}})$ and $(\boldsymbol{p}_j, \boldsymbol{m}_j) = (\boldsymbol{v}_j, \boldsymbol{M}(e_j))$, where $\boldsymbol{v}_j$ is the claimed evaluation and $\boldsymbol{M}(e_j)$ the value sent in θ_{prov}. Likewise, the vector polynomial $\boldsymbol{R}$ interpolated by $\mathcal{V}$ from $\boldsymbol{R}(E \cup S)$ lies in $(\mathbb{F}[X]^{\leq d})^\eta$ and satisfies $\mathsf{Hash}(\boldsymbol{R}) = h_R$, this query being in θ, so that it was queried before h_R was sent: θ_{com} is legit with $\hat{\boldsymbol{R}} = \boldsymbol{R}$, and $\hat{\boldsymbol{R}}(e_j) = \boldsymbol{m}_j + \Gamma \cdot \boldsymbol{p}_j$ for every $e_j \in E$ by construction of $\boldsymbol{R}$. Hence $E \subseteq E_{\mathsf{op}}$ and $\hat{\boldsymbol{P}}(E) = \boldsymbol{v}$: the transcript is alive. (6) Bad transcripts stay bad by construction. For a good $\theta \| (h, a)$, the state and the extractor's output are computed from the same data on $\theta \| (h, a)$ as on θ: in the commit phase, from the extracted leaves and $\hat{\boldsymbol{R}}$, which only depend on the hash queries made before root, resp. h_R, was sent, and from Γ; in the prove phase, from θ_{com} only. Hence $\mathsf{State}(\theta \| (h, a)) = \mathsf{State}(\theta)$, and likewise for $\mathsf{Ext}_{\mathsf{com}}$, which is the stability required by Definition 9.

Conditions of Definition 9. (1) Commit phase. The only verifier message is Γ. Fix a good non-cheating commit transcript θ whose message part is root; then $\theta \| \Gamma$ is good and cheating if and only if $\mathsf{Deg}(\Gamma, \theta)$. The extracted leaves are fixed by θ. For each E' of size $d+2$ with $\boldsymbol{u} := \mathrm{coeff}_{d+1}(\boldsymbol{P}^{(E')}) \neq 0$ and $\boldsymbol{a} := \mathrm{coeff}_{d+1}(\boldsymbol{M}^{(E')})$, the vector $\Gamma \cdot \boldsymbol{u}$ is uniformly distributed in $\mathbb{F}^\eta$ over $\Gamma \leftarrow \mathbb{F}^{\eta \times n}$, so that $\Pr_\Gamma[\Gamma \cdot \boldsymbol{u} + \boldsymbol{a} = 0] = |\mathbb{F}|^{-\eta}$. A union bound over the at most $\binom{N}{d+2}$ sets E' gives $\Pr_\Gamma[\mathsf{Deg}(\Gamma, \theta)] \leq \binom{N}{d+2} / |\mathbb{F}|^\eta = \varepsilon_{\mathrm{RBR}}$, as in [FR26, Eq. (2)]. *(2) Statement arrival.* For a good $\theta \| x$ with θ legit, $\hat{\boldsymbol{P}}(\theta) \in (\mathbb{F}[X]^{\leq d})^n$ by the claim above, so that $(\hat{\boldsymbol{P}}(\theta), (E, \boldsymbol{v})) \in \mathcal{R}_{\mathsf{decs}}$ if and only if $\hat{\boldsymbol{P}}(\theta)(E) = \boldsymbol{v}$, *i.e.* if and only if $\theta \| x$ is alive. *(3) Prove phase.* $\mathsf{DECS.Prove}$ has no verifier message, so the condition is vacuous. *(4) Bad transcripts.* Fix an execution against a prover making at most q queries and choosing the verifier's messages. Say that the i-th query is *lucky* if its answer belongs to the set F_i of 2λ-bit strings fixed before it is answered, namely: the $i-1$ earlier answers, the digests appearing in the inputs of the $i-1$ earlier queries (at most two per query), and the two values root and h_R. The answer is uniformly distributed and independent of F_i, whence

$$\Pr\big[\exists\, i \in [q] : \text{the } i\text{-th query is lucky}\big] \ \leq \ \sum_{i=1}^{q} \frac{3(i-1)+2}{2^{2\lambda}} \ \leq \ \frac{3q(q+1)}{2^{2\lambda+1}} \ = \ \varepsilon_{\mathcal{H}} \ .$$

By definition of the state $\perp$, the final transcript is bad only if some query is lucky. This proves condition 4 and concludes the proof.

D.2 Sum-Check-Based Interactive Proofs and Soundness Preservation Under CBIP

We provide hereafter a formal definition of the round-by-round soundness for a sum-check-based interactive proof (SC-IP), followed by a lemma stating that the CBIP paradigm preserves this soundness. We will use these results in the subsequent security proofs of Compilers 1 and 2.

Definition 13 (Round-by-Round Soundness of SC-IP). *An SC-IP Π for $\mathcal{R}_{\mathrm{IP}} \subseteq \mathbb{F}^\omega \times \mathbb{X}_{\mathrm{IP}}$ is* round-by-round sound relative to the witness *with error $\varepsilon^{\mathsf{ip}}_{\mathrm{RBR}}$ if, for every $\mathbb{x}$ and every $w \in \mathbb{F}^\omega$, there exists a function $\mathsf{State}(\mathbb{x}, w, \cdot)$ mapping partial transcripts to $\{\mathsf{alive}, \mathsf{doomed}\}$ such that*

1. $\mathsf{State}(\mathbb{x}, w, \emptyset) = \mathsf{alive}$ *if and only if* $(w, \mathbb{x}) \in \mathcal{R}_{\mathrm{IP}}$*;*
2. *for every complete transcript π with prover messages m,* $\mathsf{State}(\mathbb{x}, w, \pi) = \mathsf{alive}$ *if and only if* $\mathsf{Chk}_{\mathbb{x}, \pi}(w, m) = 1$*;*

3. *prover messages preserve doom;*
4. *for every partial transcript π that is empty or ends after a prover message, with $\mathsf{State}(\mathbb{x}, w, \pi) = \mathsf{doomed}$, it holds that $\Pr_r[\mathsf{State}(\mathbb{x}, w, \pi \| r) = \mathsf{alive}] \leq \varepsilon^{\mathsf{ip}}_{\mathrm{RBR}}$, where r is the next verifier message.*

Lemma 3 (CBIP Preserves Round-by-Round Soundness). *Let Π be an SC-IP for $\mathcal{R}_{\mathrm{IP}}$, round-by-round sound relative to the witness with error $\varepsilon^{\mathsf{ip}}_{\mathrm{RBR}}$, whose prover messages count ν field elements. Consider its execution in the CBIP paradigm with a pre-committed pair $u = (w, t) \in \mathbb{F}^{\omega} \times \mathbb{K}^{\tau}$, $\tau \geq \nu$, in which the prover sends $\bar{m}_i = m_i + t_i$ instead of m_i, and let the resulting SVCS instance $\mathbb{x}_{\mathrm{SVCS}} = (\mathbb{F}, \mathbb{K}, \omega, \tau, d, \mathcal{E}, \mathcal{F})$ be defined from the residual checks by*

$$\mathcal{E} := \big\{(r_i, \psi_i)\big\}_i \quad \textit{with} \quad \psi_i(t) := \varphi_i(\bar{m} - t) \ , \qquad \mathcal{F} := \big\{f'_j\big\}_j \quad \textit{with} \quad f'_j(t) := f_j(\bar{m} - t) \ ,$$

where $\deg \psi_i = \deg \varphi_i \leq 1$ and $\deg f'_j \leq \deg f_j \leq d$. Then, for every $\mathbb{x}$ and every u, there exists a function $\mathsf{State}_{\mathrm{IP}}(\mathbb{x}, u, \cdot)$ on the partial transcripts of the CBIP execution satisfying conditions 1 to 4 of Definition 13, with condition 2 replaced by: for every complete transcript $\bar{\pi}$, $\mathsf{State}_{\mathrm{IP}}(\mathbb{x}, u, \bar{\pi}) = \mathsf{alive}$ if and only if $\mathsf{Sat}_{\mathrm{SVCS}}(\mathbb{x}_{\mathrm{SVCS}}, u) = 1$; and with the same error $\varepsilon^{\mathsf{ip}}_{\mathrm{RBR}}$.

Proof. Fix $\mathbb{x}$ and $u = (w, t)$. The map Φ_t replacing each prover message m_i by $m_i + t_i$ is a bijection from the partial transcripts of Π onto those of its CBIP execution: it leaves the verifier messages and the round structure unchanged, and is invertible since t is fixed. We define

$$\mathsf{State}_{\mathrm{IP}}(\mathbb{x}, u, \bar{\pi}) \ := \ \mathsf{State}\big(\mathbb{x}, w, \Phi_t^{-1}(\bar{\pi})\big) \ .$$

Conditions 1 and 3 are immediate, as Φ_t maps the empty transcript to the empty transcript and prover messages to prover messages. Condition 4 holds because the verifier is public-coin: its messages are sampled from the same spaces, and independently of t, in both executions, so the probability of a doomed-to-alive transition is unchanged. Finally, for a complete transcript $\bar{\pi}$ with masked messages $\bar{m}$, the unmasked messages are $m = \bar{m} - t$ and, by the definition of $\mathcal{E}$ and $\mathcal{F}$ above,

$$\mathsf{Sat}_{\mathrm{SVCS}}(\mathbb{x}_{\mathrm{SVCS}}, u) = 1 \iff \forall i,\ \widetilde{p}_i(r_i) = \varphi_i(m) \ \text{ and } \ \forall j,\ f_j(m) = 0 \iff \mathsf{Chk}_{\mathbb{x}, \Phi_t^{-1}(\bar{\pi})}(w, m) = 1 \ ,$$

which is condition 2. □

D.3 Security of Compiler 1

By Remark 6, we view the protocol of Figure 6 as a commit-and-prove system for the relation

$$\mathcal{R}_1 \ := \ \big\{\big((w, t), \mathbb{x}\big) \ : \ (w, \mathbb{x}) \in \mathcal{R}_{\mathrm{IP}}\big\} \ ,$$

whose commit protocol is $\mathsf{LCS.Commit}$ and whose prove protocol consists of the CBIP execution of the SC-IP, the trace constraint linearization protocol and $\mathsf{LCS.Prove}$. Following Section 2.3, we write a partial (plain) transcript of this system as

$$\pi \ = \ \pi^{\mathsf{lcs}}_{\mathsf{com}} \,\|\, \mathbb{x} \,\|\, \big(\pi_{\mathrm{IP}} \,\|\, \pi_{\mathrm{LIN}} \,\|\, \pi^{\mathsf{lcs}}_{\mathsf{prov}}\big) \ ,$$

where $\pi^{\mathsf{lcs}}_{\mathsf{com}}$ and $\pi^{\mathsf{lcs}}_{\mathsf{prov}}$ are the transcripts of $\mathsf{LCS.Commit}$ and $\mathsf{LCS.Prove}$, and π_{IP}, π_{LIN} those of the two intermediate phases; an extended transcript θ of the system is such a transcript in which the hash queries of the prover are inserted, and we say that θ stops within one of these parts according to its message part. We write $\mathbb{x}_{\mathrm{SVCS}}(\mathbb{x}, \pi_{\mathrm{IP}})$ for the SVCS instance produced by the SC-IP execution, $\mathsf{State}_{\mathrm{IP}}$ for the state function of that execution given by Lemma 3, and $x_{\mathrm{LCS}}(\mathbb{x}, \pi_{\mathrm{IP}}, \pi_{\mathrm{LIN}})$ for the LCS statement, *i.e.* the set of $\mathbb{K}$-linear constraints of $\mathcal{E} \cup \mathcal{F}'$; these are deterministic functions of the public transcript, the latter being determined at the end of π_{LIN}. We write $\mathcal{F} = \mathcal{F}_{\mathrm{lin}} \cup \mathcal{F}_{\mathrm{mult}}$ for the partition of the trace constraints into the m_{lin} linear ones and the m_{mult} multiplication ones, and g_1, g_2 for the two linear constraints that the linearization protocol substitutes for the latter, so that $\mathcal{F}' = \mathcal{F}_{\mathrm{lin}} \cup \{g_1, g_2\}$ and $x_{\mathrm{LCS}} = \mathcal{E} \cup \mathcal{F}_{\mathrm{lin}} \cup \{g_1, g_2\}$. Finally, we write v_{LIN} for the value revealed by the linearization protocol (denoted v in Section 3.2).

Proof of Theorem 2

Proof. Zero-knowledge. Following the definition of HVZK given in Section 2.3, we write

$$\pi^{\mathsf{lcs}} := (\pi^{\mathsf{lcs}}_{\mathsf{com}}, \mathsf{com}, \pi^{\mathsf{lcs}}_{\mathsf{prov}})$$

for the LCS part of the transcript, *i.e.* the commitment together with the transcripts of the $\mathsf{LCS.Commit}$ and $\mathsf{LCS.Prove}$ protocols, and $\mathsf{Sim}_{\mathsf{lcs}}$ for the LCS simulator, which outputs such a tuple together with the associated random oracle programming. The transcript of the protocol of Figure 6 then consists of π^{lcs} and of the masked prover messages of the CBIP execution – including the messages q' and v_{LIN} of the linearization protocol. No value is ever revealed in the clear, so the simulator is obtained in two hybrids from the real distribution H_0:

- H_1: replace π^{lcs} by the output of $\mathsf{Sim}_{\mathsf{lcs}}$ on the statement made of the linear constraints of $\mathcal{E}$ and $\mathcal{F}'$. This statement is a function of the public transcript alone: the evaluation constraints $\widetilde{p}_i(r_i) = \psi_i(t)$ of the (possibly extended) SVCS instance – where each p_i is obtained from the witness w through a public affine map, the standard case corresponding to $p_i = w$ (see Section 3.1) – rewrite as $\mathbb{K}$-linear constraints $\langle a_i, (w,t)\rangle = b_i$ whose coefficients a_i and constant terms b_i are determined by the verifier's challenges, these affine maps and the prover messages. By the $(q, \varepsilon^{\mathsf{lcs}}_{\text{ZK}})$-HVZK of the LCS, H_1 is $\varepsilon^{\mathsf{lcs}}_{\text{ZK}}$-close to H_0.
- H_2: replace every masked prover message by a uniformly random element of $\mathbb{K}$. After H_1, the trace mask t occurs in the transcript only through these messages, each of the form $m_i + t_i$ for a distinct coordinate t_i of t used nowhere else; as t is uniform over $\mathbb{K}^{\tau'}$, this is a one-time-pad masking and $\mathsf{H}_2 \equiv \mathsf{H}_1$. The same holds for the two messages of the linearization protocol: $q' = q + \mu$ is uniform among the polynomials of degree at most $m_{\text{mult}} - 1$ since μ interpolates the uniform coordinates $t_1, \ldots, t_{m_{\text{mult}}}$, and $v_{\text{LIN}} = p_1(r)$ is uniform over $\mathbb{K}$ since the Lagrange coefficient of the uniform coordinate $t_{m_{\text{mult}}+1} = p_1(m_{\text{mult}} + 1)$ in $p_1(r)$ is nonzero for $r \notin [m_{\text{mult}}]$.

The distribution H_2 is independent of w and efficiently samplable from $\mathbb{x}$ alone, which gives $\varepsilon_{\text{ZK}} = \varepsilon^{\mathsf{lcs}}_{\text{ZK}}$.

Round-by-round knowledge soundness. Let $\mathsf{State}_{\mathsf{lcs}}$ and $\mathsf{Ext}^{\mathsf{lcs}}_{\mathsf{com}}$ be the state function and the commitment extractor given by the RBR knowledge soundness of the LCS.

Sub-scheme transcripts. Only some of the messages of the compiled protocol are LCS messages. For an extended transcript θ of the compiled protocol, we write $\theta|_{\mathsf{lcs}}$ for the extended transcript obtained by keeping the LCS messages and *all* the hash queries of θ, and removing the other messages, x_{LCS} being inserted at the point where it becomes determined (as the statement of the LCS, it counts as an LCS prover message). Then $\theta|_{\mathsf{lcs}}$ is an extended transcript of the LCS, in which the prover may have made many hash queries between two messages, and we let $\mathsf{State}_1(\theta) := \bot$ whenever $\mathsf{State}_{\mathsf{lcs}}(\theta|_{\mathsf{lcs}}) = \bot$: a compiled transcript is bad if and only if its LCS part is, which is preserved by extension. We further write $\theta^{\mathsf{lcs}}_{\mathsf{com}}$ for the longest prefix of $\theta|_{\mathsf{lcs}}$ which is a commit transcript (all of $\theta|_{\mathsf{lcs}}$ as long as x_{LCS} is not determined), and, on a good θ with $\pi^{\mathsf{lcs}}_{\mathsf{com}}$ complete, we let

$$\sigma := \mathsf{State}_{\mathsf{lcs}}(\theta^{\mathsf{lcs}}_{\mathsf{com}}) \qquad \text{and} \qquad \hat{u} = (\hat{w}, \hat{t}) := \mathsf{Ext}^{\mathsf{lcs}}_{\mathsf{com}}(\theta^{\mathsf{lcs}}_{\mathsf{com}}) \in \left(\mathbb{F}^{\omega} \times \mathbb{K}^{\tau'}\right) \cup \{\bot\}$$

be the commit state and the extracted value of the LCS, with the convention that $\hat{w} = \hat{t} = \bot$ whenever $\hat{u} = \bot$. The commitment extractor of the compiled system is taken to be $\mathsf{Ext}_{\mathsf{com}}(\theta) := \mathsf{Ext}^{\mathsf{lcs}}_{\mathsf{com}}(\theta|_{\mathsf{lcs}}) = \hat{u}$. Moreover, by stability and by the statement arrival conditions applied to the LCS, on a good θ with π_{LIN} complete, the LCS state at $\theta^{\mathsf{lcs}}_{\mathsf{com}} \| x_{\text{LCS}}$ is alive if and only if $\sigma = \mathsf{cheating}$, or $\sigma = \mathsf{legit}$ and $(\hat{u}, x_{\text{LCS}}) \in \mathcal{R}_{\text{LCS}}$.

State function. We define State_1 on the good transcripts θ of the compiled protocol as follows.

- (S1) if θ is a (possibly partial) commit transcript: $\mathsf{State}_1(\theta) := \mathsf{State}_{\mathsf{lcs}}(\theta|_{\mathsf{lcs}})$.
- (S2–S4) if the message part of θ is $\pi^{\mathsf{lcs}}_{\mathsf{com}} \| \mathbb{x} \| \pi_{\text{IP}} \| \pi_{\text{LIN}}$ with π_{LIN} incomplete (this includes the case where it is $\pi^{\mathsf{lcs}}_{\mathsf{com}} \| \mathbb{x}$): the state is alive if $\sigma = \mathsf{cheating}$, doomed if $\sigma = \mathsf{trashed}$, and otherwise alive if and only if $\hat{u} \neq \bot$ and the following condition holds:
  - (S2) if the message part of θ is $\pi^{\mathsf{lcs}}_{\mathsf{com}} \| \mathbb{x}$: $(\hat{w}, \mathbb{x}) \in \mathcal{R}_{\text{IP}}$;
  - (S3) if θ stops within π_{IP}: $\mathsf{State}_{\text{IP}}(\mathbb{x}, \hat{u}, \pi_{\text{IP}})$ is alive;

- (S4) if θ stops within π_{LIN} (before v_{LIN} is sent): $\hat{u}$ satisfies the constraints of $\mathcal{E} \cup \mathcal{F}_{\text{lin}}$ – which the linearization protocol leaves untouched – and the claim currently at stake in place of $\mathcal{F}_{\text{mult}}$ holds for $\hat{u}$, namely: before the challenge r is sent, $\hat{t}$ satisfies the m_{mult} constraints of $\mathcal{F}_{\text{mult}}$; after r has been sent, $D(r) = 0$, where

$$D(X) \;:=\; p_1(X)\,p_2(X) - p_3(X) - \hat{q}(X) \cdot \prod_{j=1}^{m_{\text{mult}}} (X - j) \;,$$

  with $p_1, p_2, p_3 \in \mathbb{K}[X]$ the interpolation polynomials determined by $\hat{t}$ and the public maps $\psi_i^{(j)}$ – recall that $p_1(m_{\text{mult}}+1) = \hat{t}_{m_{\text{mult}}+1}$ is a committed coordinate – and $\hat{q} := q' - \mu(\hat{t})$ determined by $\hat{t}$ and the prover message q', whose format enforces $\deg q' \leq m_{\text{mult}} - 1$.

– (S5) if π_{LIN} is complete, *i.e.* the message part of θ is $\pi_{\mathsf{com}}^{\mathsf{lcs}} \| \mathbb{x} \| \pi_{\text{IP}} \| \pi_{\text{LIN}} \| \pi_{\mathsf{prov}}^{\mathsf{lcs}}$ with $\pi_{\mathsf{prov}}^{\mathsf{lcs}}$ possibly empty: the state is alive if and only if

$$\mathsf{State}_{\mathsf{lcs}}\big(\theta|_{\mathsf{lcs}}\big) = \mathsf{alive} \quad \text{or} \quad \underbrace{\big(\sigma = \mathsf{legit} \,\wedge\, (\hat{u}, x_{\text{LCS}}) \in \mathcal{R}_{\text{LCS}}\big)}_{\text{the } \textit{guard}} \;.$$

The guard of (S5) only concerns the commit phase of the LCS: it says that the LCS commitment is legit and that the extracted value satisfies the LCS statement. It is implied by the first disjunct of (S5) when $\pi_{\mathsf{prov}}^{\mathsf{lcs}}$ is empty, by the remark above, and it is included so that a doomed state in (S5) always comes with the extraction failure that the prove phase condition of the LCS requires.

State_1 *is a valid state function* in the sense of Definition 8. Conditions 1, 2 and 6 (initial state, prover messages cannot cheat, bad transcripts and hash queries) are inherited from the LCS through (S1) and (S5), σ and $\hat{u}$ being unchanged by hash queries and the other quantities involved in (S2–S4) being determined by $\hat{u}$ and the message part of the transcript; the stability of $\mathsf{Ext}_{\mathsf{com}}$ is that of $\mathsf{Ext}_{\mathsf{com}}^{\mathsf{lcs}}$. Condition 3 (statement arrival) holds by construction of (S2): a trashed commit state gives a doomed state and a cheating one an alive state. For condition 5, a doomed full transcript falls under (S5) with a false guard and a doomed LCS state on the full LCS transcript, which contains the hash queries of the LCS verifier; by condition 5 applied to the LCS, the verifier of $\mathsf{LCS.Prove}$ returns Reject, which is also the output of the verifier of the compiled protocol. There remains condition 4 (prover messages preserve doom). If σ is cheating or trashed, the state is constant along (S2–S4), and at the junction with (S5) a trashed σ makes the guard false and the LCS state at arrival doomed, so that the state stays doomed. Assume now $\sigma = \mathsf{legit}$. Within π_{IP}, doom is preserved by condition 3 of Lemma 3, the conjunct $\hat{u} \neq \bot$ being constant; at the junction between $\mathbb{x}$ and π_{IP}, a doomed state at the arrival of $\mathbb{x}$ means that $\hat{u} = \bot$ – in which case (S3) is doomed – or that $(\hat{w}, \mathbb{x}) \notin \mathcal{R}_{\text{IP}}$, in which case $\mathsf{State}_{\text{IP}}(\mathbb{x}, \hat{u}, \emptyset)$ is doomed by condition 1 of Lemma 3 and remains so after the first prover messages. At the junction between π_{IP} and π_{LIN}, a doomed state at the end of π_{IP} means that $\hat{u} = \bot$ or, by condition 2 of Lemma 3, that $\mathsf{Sat}_{\text{SVCS}}(\mathbb{x}_{\text{SVCS}}, \hat{u}) = 0$, hence that $\hat{u}$ violates one of the constraints of $\mathcal{E} \cup \mathcal{F}_{\text{lin}}$ or that $\hat{t}$ violates one of the constraints of $\mathcal{F}_{\text{mult}}$: the state given by (S4) is doomed as well. Within π_{LIN}, the message q' is sent before the challenge r and the constraint at stake before r does not involve it. Finally, the message v_{LIN} closes π_{LIN} and leads to (S5) with empty $\pi_{\mathsf{prov}}^{\mathsf{lcs}}$: a doomed state before v_{LIN} means that $\hat{u} = \bot$, that $\hat{u}$ violates one of the constraints of $\mathcal{E} \cup \mathcal{F}_{\text{lin}}$, or that $D(r) \neq 0$; in the last case no value of v_{LIN} satisfies both replacement constraints, since the first one, $\psi_1(t) = v_{\text{LIN}}$, forces $v_{\text{LIN}} = \psi_1(\hat{t}) = p_1(r)$, and substituting into the second one yields $D(r) = 0$. In all cases $(\hat{u}, x_{\text{LCS}}) \notin \mathcal{R}_{\text{LCS}}$, so that the guard of (S5) is false and the LCS state at $\theta_{\mathsf{com}}^{\mathsf{lcs}} \| x_{\text{LCS}}$ is doomed, by the remark above: so is the state given by (S5). Within $\pi_{\mathsf{prov}}^{\mathsf{lcs}}$, doom is preserved by the corresponding property of $\mathsf{State}_{\mathsf{lcs}}$, the guard being constant.

Round-by-round knowledge soundness conditions. The commit phase condition is inherited from the LCS through (S1), with error $\varepsilon_{\text{RBR}}^{\mathsf{lcs}}$. The statement arrival condition holds by (S2): on a good transcript $\theta \| \mathbb{x}$ with θ a complete commit transcript and $\sigma = \mathsf{legit}$, the state is alive if and only if $\hat{u} \neq \bot$ and $(\hat{w}, \mathbb{x}) \in \mathcal{R}_{\text{IP}}$, *i.e.* if and only if $(\mathsf{Ext}_{\mathsf{com}}(\theta), \mathbb{x}) \in \mathcal{R}_1$. For the bad transcripts condition, note that a malicious prover of the compiled protocol, run together with the honest verifier, is a malicious prover of the LCS which chooses the LCS verifier's messages and the LCS statement, makes at most q queries, and whose final extended transcript is $\theta|_{\mathsf{lcs}}$ for the final extended transcript θ of the compiled execution; condition 4 applied to the LCS thus gives $\Pr[\mathsf{State}_1(\theta) = \bot] \leq \varepsilon_{\mathcal{H}}^{\mathsf{lcs}}$, *i.e.* $\varepsilon_{\mathcal{H}} \leq \varepsilon_{\mathcal{H}}^{\mathsf{lcs}}$.

It remains to establish the prove phase condition. Let θ be a good doomed transcript which is to be followed by a verifier message r; σ and $\hat{u}$ are the same before and after r. We bound the probability that

the state becomes alive, which upper bounds the probability of that event conjoined with the failure of extraction. If $\sigma = \mathsf{trashed}$ and θ falls under (S2–S4), the state remains doomed. Otherwise:

- *r is a challenge of the SC-IP execution* (including the first one, when the SC-IP opens with a verifier message). Here $\sigma = \mathsf{legit}$: a cheating σ would make the state alive, and the trashed case has just been dealt with. If $\hat{u} = \perp$ the state remains doomed and the probability is 0. Otherwise $\mathsf{State}_{\mathrm{IP}}(\mathbb{x}, \hat{u}, \cdot)$ is doomed and, $\hat{u}$ being fixed, condition 4 of Lemma 3 bounds the probability by $\varepsilon^{\mathsf{ip}}_{\mathrm{RBR}}$.
- *r is the challenge of the linearization protocol.* Here again $\sigma = \mathsf{legit}$, for the same reason. If $\hat{u} = \perp$ or if $\hat{u}$ violates one of the constraints of $\mathcal{E} \cup \mathcal{F}_{\mathrm{lin}}$, these conditions do not depend on r and the state remains doomed. Otherwise $\hat{t}$ violates one of the constraints of $\mathcal{F}_{\mathrm{mult}}$, and the state becomes alive only if $D(r) = 0$. The polynomial D is then nonzero: were it zero, we would have $p_1(j)p_2(j) = p_3(j)$, *i.e.* $\psi_1^{(j)}(\hat{t})\,\psi_2^{(j)}(\hat{t}) = \psi_3^{(j)}(\hat{t})$, for every $j \in [m_{\mathrm{mult}}]$. As $\deg(p_1) \leq m_{\mathrm{mult}}$ and $\deg(p_2), \deg(p_3), \deg(\hat{q}) \leq m_{\mathrm{mult}} - 1$, we have $\deg(D) \leq 2m_{\mathrm{mult}} - 1$, so that, r being uniform over $\mathbb{K} \backslash [m_{\mathrm{mult}}]$, the Schwartz–Zippel lemma gives $\Pr_r[D(r) = 0] \leq (2m_{\mathrm{mult}} - 1)/(|\mathbb{K}| - m_{\mathrm{mult}})$.
- *r is a challenge of* $\mathsf{LCS.Prove}$*, including the first one.* The state is given by (S5) and is doomed, so that the guard is false and $\mathsf{State}_{\mathsf{lcs}}$ is doomed on the good transcript $\theta|_{\mathsf{lcs}}$. The state becomes alive only if the LCS state does, the guard being unchanged, and a false guard means that $\sigma \neq \mathsf{legit}$ or that $(\hat{u}, x_{\mathrm{LCS}}) \notin \mathcal{R}_{\mathrm{LCS}}$, which is the extraction failure of the prove phase condition of Definition 9 applied to the LCS with statement x_{LCS}; this condition bounds the probability by $\varepsilon^{\mathsf{lcs}}_{\mathrm{RBR}}$.

Taking the maximum over the above cases and using $(2m_{\mathrm{mult}} - 1)/(|\mathbb{K}| - m_{\mathrm{mult}}) \leq 2m_{\mathrm{mult}}/|\mathbb{K}|$ for $m_{\mathrm{mult}} \leq \sqrt{|\mathbb{K}|/2}$, we obtain

$$\varepsilon_{\mathrm{RBR}} \;\leq\; \max\left(\varepsilon^{\mathsf{ip}}_{\mathrm{RBR}},\, \varepsilon^{\mathsf{lcs}}_{\mathrm{RBR}},\, \frac{2m_{\mathrm{mult}}}{|\mathbb{K}|}\right) ,$$

which concludes the proof. □

D.4 Security of Compiler 2

As for Compiler 1, we view the protocol of Figure 7 as a commit-and-prove system (Remark 6), here for the relation

$$\mathcal{R}_2 \;:=\; \left\{\left((w^*, t), \mathbb{x}\right) \;:\; (w, \mathbb{x}) \in \mathcal{R}_{\mathrm{IP}}\right\} ,$$

where w denotes the first ω coordinates of $w^* \in \mathbb{F}^{\omega+k}$ – the k mask coordinates are left unconstrained by the statement, as are the coordinates of t. Its commit protocol consists of the two commitments, and its prove protocol of the CBIP execution of the SC-IP, the trace constraint linearization protocol, the message v, $\mathsf{MPCS.Prove}$ and $\mathsf{LCS.Prove}$. We accordingly write a partial (plain) transcript as

$$\pi \;=\; \pi_{\mathsf{com}} \,\|\, \mathbb{x} \,\|\, \left(\pi_{\mathrm{IP}} \,\|\, \pi_{\mathrm{LIN}} \,\|\, v \,\|\, \pi^{\mathsf{mpcs}}_{\mathsf{prov}} \,\|\, \pi^{\mathsf{lcs}}_{\mathsf{prov}}\right) , \qquad \pi_{\mathsf{com}} \;=\; \pi^{\mathsf{lcs}}_{\mathsf{com}} \,\|\, \pi^{\mathsf{mpcs}}_{\mathsf{com}} ,$$

extended transcripts θ being obtained as in the proof of Compiler 1 by inserting the hash queries of the prover. We keep the notation of the proof of Compiler 1 for $\mathbb{x}_{\mathrm{SVCS}}$, $\mathsf{State}_{\mathrm{IP}}$, the partition $\mathcal{F} = \mathcal{F}_{\mathrm{lin}} \cup \mathcal{F}_{\mathrm{mult}}$ and the two replacement constraints g_1, g_2, writing however r_{LIN} for the challenge of the linearization protocol, so as to avoid any confusion with the evaluation point r of the single constraint $\mathcal{E} = \{(r, \psi)\}$. The two statements proven by the commitment schemes are

$$x_{\mathsf{mpcs}} \;:=\; (r, v) \qquad \text{and} \qquad x_{\mathsf{lcs}} \;:=\; \mathcal{F}_{\mathrm{lin}} \cup \{g_1, g_2\} \cup \{\psi(t) - v = 0\} ,$$

both determined as soon as v has been sent.

Proof of Theorem 3

Proof. Zero-knowledge. We write π^{lcs} and π^{mpcs} for the two commitment-scheme parts of the transcript (each made of the commitment together with the transcripts of its commit and prove protocols, as in the proof of Compiler 1), and $\mathsf{Sim}_{\mathsf{lcs}}$, $\mathsf{Sim}_{\mathsf{mpcs}}$ for the corresponding simulators. Besides these two parts, the transcript consists of the masked prover messages of the CBIP execution – including the messages q' and v_{LIN} of the linearization protocol – and of the value v, which is the only value revealed in the clear. The simulator is obtained in four hybrids from the real distribution H_0:

– H_1: replace π^{mpcs} by the output of $\mathsf{Sim}_{\mathsf{mpcs}}$ on the statement $x_{\mathsf{mpcs}} = (r, v)$, which is a function of the public transcript alone. By the $(q, \varepsilon_{\text{ZK}}^{\mathsf{mpcs}})$-HVZK of the MPCS, H_1 is $\varepsilon_{\text{ZK}}^{\mathsf{mpcs}}$-close to H_0.
– H_2: replace π^{lcs} by the output of $\mathsf{Sim}_{\mathsf{lcs}}$ on the statement x_{lcs}, likewise a function of the public transcript alone. By the $(q, \varepsilon_{\text{ZK}}^{\mathsf{lcs}})$-HVZK of the LCS, H_2 is $\varepsilon_{\text{ZK}}^{\mathsf{lcs}}$-close to H_1.
– H_3: replace every masked prover message by a uniformly random element of $\mathbb{K}$. After H_2, the trace mask t occurs in the transcript only through these messages and through v; the former are one-time-pad masked exactly as in the proof of Compiler 1, so that $\mathsf{H}_3 \equiv \mathsf{H}_2$.
– H_4: replace v by a uniformly random element of $\mathbb{K}$. After H_3, the witness w and the mask m occur in the transcript only through $v = \widetilde{w}^*(r)$. The evaluation point r being a vector of verifier challenges, it is uniformly distributed over $\mathbb{K}^\ell$ and independent of (w, m); hence, if r is a k-hiding point, $\widetilde{w}^*(r)$ is exactly uniform over $\mathbb{K}$ for every w (Section 5.1), so that $\mathsf{H}_4 \equiv \mathsf{H}_3$ conditioned on that event, which fails with probability at most ε_{HID}.

The distribution H_4 is independent of w and efficiently samplable from $\mathbb{x}$ alone, which gives $\varepsilon_{\text{ZK}} = \varepsilon_{\text{ZK}}^{\mathsf{lcs}} + \varepsilon_{\text{ZK}}^{\mathsf{mpcs}} + \varepsilon_{\text{HID}}$.

Round-by-round knowledge soundness. Let $(\mathsf{State}_{\mathsf{lcs}}, \mathsf{Ext}_{\mathsf{com}}^{\mathsf{lcs}})$ and $(\mathsf{State}_{\mathsf{mpcs}}, \mathsf{Ext}_{\mathsf{com}}^{\mathsf{mpcs}})$ be given by the RBR knowledge soundness of the two commitment schemes. With the sub-scheme transcripts $\theta|_{\mathsf{lcs}}$ and $\theta|_{\mathsf{mpcs}}$ and their commit prefixes $\theta_{\mathsf{com}}^{\mathsf{lcs}}$ and $\theta_{\mathsf{com}}^{\mathsf{mpcs}}$ defined as in the proof of Theorem 2 (the two statements x_{lcs} and x_{mpcs} being inserted when v is sent), we let $\mathsf{State}_2(\theta) := \bot$ whenever $\mathsf{State}_{\mathsf{lcs}}(\theta|_{\mathsf{lcs}}) = \bot$ or $\mathsf{State}_{\mathsf{mpcs}}(\theta|_{\mathsf{mpcs}}) = \bot$, which is preserved by extension, and on a good θ we let

$$\sigma_{\mathsf{lcs}} := \mathsf{State}_{\mathsf{lcs}}(\theta_{\mathsf{com}}^{\mathsf{lcs}}) \ , \qquad \sigma_{\mathsf{mpcs}} := \mathsf{State}_{\mathsf{mpcs}}(\theta_{\mathsf{com}}^{\mathsf{mpcs}}) \ ,$$

and, once both commit transcripts are complete,

$$\hat{u} \ = \ (\hat{w}^*, \hat{t}) \ := \ \Big(\mathsf{Ext}_{\mathsf{com}}^{\mathsf{mpcs}}\big(\theta_{\mathsf{com}}^{\mathsf{mpcs}}\big),\ \mathsf{Ext}_{\mathsf{com}}^{\mathsf{lcs}}\big(\theta_{\mathsf{com}}^{\mathsf{lcs}}\big)\Big) \ ,$$

with the convention that $\hat{u} = \bot$ as soon as one of the two components is $\bot$, and that $\hat{w}$ denotes the first ω coordinates of $\hat{w}^*$. By stability, these quantities do not depend on the point of the execution at which they are evaluated; in particular, while $\mathsf{LCS.Commit}$ is running, $\theta|_{\mathsf{mpcs}}$ consists of hash queries only and $\sigma_{\mathsf{mpcs}} = \mathsf{State}_{\mathsf{mpcs}}(\emptyset) = \mathsf{legit}$. We write σ for the combination of the two commit states: cheating if one of them is cheating, otherwise trashed if one of them is trashed, and legit otherwise. The commitment extractor of the compiled protocol is $\mathsf{Ext}_{\mathsf{com}}(\theta) := \hat{u}$. Note that the two commitments are on disjoint witnesses, so that no consistency is required of them: their only interaction is through the value v. Finally, as in the proof of Theorem 2, on a good transcript where v has been sent, the MPCS state at $\theta_{\mathsf{com}}^{\mathsf{mpcs}} \| x_{\mathsf{mpcs}}$ is alive if and only if the *guard*

$$g_{\mathsf{mpcs}} \ :\equiv \ \sigma_{\mathsf{mpcs}} = \mathsf{cheating} \ \vee \ \big(\sigma_{\mathsf{mpcs}} = \mathsf{legit} \wedge (\hat{w}^*, x_{\mathsf{mpcs}}) \in \mathcal{R}_{\text{MPCS}}\big)$$

holds, and likewise for the LCS with

$$g_{\mathsf{lcs}} \ :\equiv \ \sigma_{\mathsf{lcs}} = \mathsf{cheating} \ \vee \ \big(\sigma_{\mathsf{lcs}} = \mathsf{legit} \wedge (\hat{t}, x_{\mathsf{lcs}}) \in \mathcal{R}_{\text{LCS}}\big) \ .$$

State function. We define State_2 on good transcripts as follows:

– (S1) if θ is a (possibly partial) commit transcript: $\mathsf{State}_2(\theta) := \sigma$;
– (S2–S4) if the message part of θ is $\pi_{\mathsf{com}} \| \mathbb{x} \| \pi_{\text{IP}} \| \pi_{\text{LIN}}$ and v has not been sent yet: the state is alive if $\sigma = \mathsf{cheating}$, doomed if $\sigma = \mathsf{trashed}$, and otherwise alive if and only if $\hat{u} \neq \bot$ and: (S2) if $\theta = \pi_{\mathsf{com}} \| \mathbb{x}$, $(\hat{w}, \mathbb{x}) \in \mathcal{R}_{\text{IP}}$; (S3) if θ stops within π_{IP}, $\mathsf{State}_{\text{IP}}(\mathbb{x}, \hat{u}, \pi_{\text{IP}})$ is alive; (S4) if θ stops within or at the end of π_{LIN}, $\hat{u}$ satisfies the evaluation constraint of $\mathcal{E}$ and the constraints of $\mathcal{F}_{\text{lin}}$, and the claim currently at stake in place of $\mathcal{F}_{\text{mult}}$ holds for $\hat{u}$ – namely the m_{mult} constraints of $\mathcal{F}_{\text{mult}}$ before r_{LIN} is sent, then $D(r_{\text{LIN}}) = 0$, then g_1 and g_2 once v_{LIN} has been sent – with D, p_1, p_2, p_3 and $\hat{q}$ as in the proof of Theorem 2;
– (S5) if v has been sent, *i.e.* the message part of θ is $\pi_{\mathsf{com}} \| \mathbb{x} \| \pi_{\text{IP}} \| \pi_{\text{LIN}} \| v \| \pi_{\mathsf{prov}}^{\mathsf{mpcs}} \| \pi_{\mathsf{prov}}^{\mathsf{lcs}}$ with the two prove transcripts possibly empty or partial: the state is alive if and only if both

$$\mathsf{State}_{\mathsf{mpcs}}\big(\theta|_{\mathsf{mpcs}}\big) = \mathsf{alive} \ \ \vee \ \ g_{\mathsf{mpcs}} \qquad \text{and} \qquad \mathsf{State}_{\mathsf{lcs}}\big(\theta|_{\mathsf{lcs}}\big) = \mathsf{alive} \ \ \vee \ \ g_{\mathsf{lcs}}$$

hold.

State_2 *is a valid state function.* The initial state is legit, as both commit states are, and condition 6 (as well as the stability of the extractor) is inherited from the two schemes, all the quantities involved being unchanged by hash queries or determined by the message part of the transcript. A prover message of the commit phase extends one of the two commit transcripts and leaves the other unchanged; if the compiled state is not cheating then neither commit state is, and the extended one remains so by condition 2 of Definition 8 applied to the corresponding scheme: prover messages cannot cheat. Statement arrival holds by construction of (S2). For doom preservation, the arguments of the proof of Theorem 2 apply verbatim within π_{IP} and π_{LIN} and at the junctions between $\mathbb{x}$ and π_{IP} and between π_{IP} and π_{LIN}; within $\pi^{\mathsf{mpcs}}_{\mathsf{prov}}$ and $\pi^{\mathsf{lcs}}_{\mathsf{prov}}$, doom is preserved by the corresponding property of $\mathsf{State}_{\mathsf{mpcs}}$ and $\mathsf{State}_{\mathsf{lcs}}$, the other conjunct of (S5) being constant along the phase (the other scheme's transcript does not change). There remains the message v, which closes (S4) and leads to (S5) with empty prove transcripts, where the two conjuncts of (S5) reduce to the guards g_{mpcs} and g_{lcs}. Assume the state is doomed at the end of π_{LIN}. If $\sigma = \mathsf{trashed}$, one of the two commit states is trashed and the corresponding guard is false: (S5) is doomed. Otherwise both commit states are legit, and by (S4) either $\hat{u} = \bot$, or $\hat{t}$ violates one of the constraints of $\mathcal{F}_{\text{lin}} \cup \{g_1, g_2\}$ – in both cases g_{lcs} is false, whatever v – or $\hat{u}$ violates the evaluation constraint, *i.e.* $\widetilde{\hat{w}}^*(r) \neq \psi(\hat{t})$, in which case no value of v can satisfy both $\widetilde{\hat{w}}^*(r) = v$ and $\psi(\hat{t}) = v$, so that g_{mpcs} or g_{lcs} is false. Doom is therefore preserved. Finally, a doomed full transcript makes one of the two conjuncts of (S5) false, *i.e.* the full MPCS transcript or the full LCS transcript is doomed on a transcript containing the hash queries of the corresponding verifier, so that this verifier returns Reject by condition 5 applied to that scheme – and the verifier of the compiled protocol accepts only if both accept.

Round-by-round knowledge soundness conditions. For the commit phase condition, a verifier message of π_{com} belongs to one of the two commit protocols and can only change the corresponding commit state; if the compiled state is not cheating, neither commit state is, and the probability that the concerned one becomes cheating is at most its own error: the bound is $\max(\varepsilon^{\mathsf{lcs}}_{\text{RBR}}, \varepsilon^{\mathsf{mpcs}}_{\text{RBR}})$. The statement arrival condition holds by (S2): on a good transcript $\theta \| \mathbb{x}$ with θ a complete commit transcript and $\sigma = \mathsf{legit}$, both commit states are legit and the state is alive if and only if $\hat{u} \neq \bot$ and $(\hat{w}, \mathbb{x}) \in \mathcal{R}_{\text{IP}}$, *i.e.* if and only if $(\mathsf{Ext}_{\mathsf{com}}(\theta), \mathbb{x}) \in \mathcal{R}_2$. For the bad transcripts condition, a malicious prover of the compiled protocol run together with the honest verifier is a malicious prover of each of the two schemes, as in the proof of Theorem 2, so that condition 4 applied to each scheme and a union bound give $\varepsilon_{\mathcal{H}} \leq \varepsilon^{\mathsf{lcs}}_{\mathcal{H}} + \varepsilon^{\mathsf{mpcs}}_{\mathcal{H}}$.

For the prove phase condition, let θ be a good doomed transcript which is to be followed by a verifier message r'. The cases where r' is a challenge of the SC-IP execution or the challenge r_{LIN} of the linearization protocol are treated exactly as in the proof of Theorem 2, and yield the bounds $\varepsilon^{\mathsf{ip}}_{\text{RBR}}$ and $(2m_{\text{mult}} - 1)/(|\mathbb{K}| - m_{\text{mult}})$ respectively. The two remaining cases are:

- r' *is a challenge of* MPCS.Prove. The LCS conjunct of (S5) does not depend on r'; if it is false, the state remains doomed. Otherwise the MPCS conjunct is false: g_{mpcs} is false and $\mathsf{State}_{\mathsf{mpcs}}$ is doomed on the good transcript $\theta|_{\mathsf{mpcs}}$. A false guard means that $\sigma_{\mathsf{mpcs}} \neq \mathsf{legit}$ or that $(\hat{w}^*, x_{\mathsf{mpcs}}) \notin \mathcal{R}_{\text{MPCS}}$, which is the extraction failure of the prove phase condition applied to the MPCS; the state becomes alive only if the MPCS state does, so that the probability is at most $\varepsilon^{\mathsf{mpcs}}_{\text{RBR}}$.
- r' *is a challenge of* LCS.Prove. Symmetrically, the MPCS conjunct does not depend on r' (the MPCS transcript is complete), and if the LCS conjunct is false the prove phase condition applied to the LCS bounds the probability by $\varepsilon^{\mathsf{lcs}}_{\text{RBR}}$.

Taking the maximum over all the cases, and rounding the linearization term as in the proof of Theorem 2, we obtain

$$\varepsilon_{\text{RBR}} \;\leq\; \max\left(\varepsilon^{\mathsf{ip}}_{\text{RBR}},\, \varepsilon^{\mathsf{lcs}}_{\text{RBR}},\, \varepsilon^{\mathsf{mpcs}}_{\text{RBR}},\, \frac{2m_{\text{mult}}}{|\mathbb{K}|}\right),$$

which concludes the proof. □

D.5 Security of SmallWood-LCS and SmallWood-MPCS

Both schemes share the same commit protocol and the same extractor, which we describe once. The commitment consists of a DECS commitment to the vector polynomial $\boldsymbol{P}^+ := (\boldsymbol{P}, M)$ – with the convention of Remark 7 for encoding $M \in \mathbb{K}[X]^{\leq d_Q}$ over $\mathbb{F}$ – so that the commit transcript of either scheme is the commit transcript $\pi^{\mathsf{decs}}_{\mathsf{com}}$ of the DECS. Let $\mathsf{Ext}^{\mathsf{decs}}_{\mathsf{com}}$ and $\mathsf{State}_{\mathsf{decs}}$ be the commitment extractor and the state function given by the RBR knowledge soundness of the DECS. We denote the sub-scheme

transcript $\theta|_{\mathsf{decs}}$ and its commit prefix $\theta^{\mathsf{decs}}_{\mathsf{com}}$, and the DECS statement x_{decs} defined below being inserted when the evaluations are sent. The state of either scheme is $\perp$ whenever $\mathsf{State}_{\mathsf{decs}}(\theta|_{\mathsf{decs}}) = \perp$, and on a good transcript θ we let

$$\sigma \;:=\; \mathsf{State}_{\mathsf{decs}}(\theta^{\mathsf{decs}}_{\mathsf{com}}) \qquad \text{and} \qquad (\hat{\boldsymbol{P}}, \hat{M}) \;:=\; \mathsf{Ext}^{\mathsf{decs}}_{\mathsf{com}}\big(\theta^{\mathsf{decs}}_{\mathsf{com}}\big)$$

be the commit state and the extracted polynomials of the DECS, which by stability do not depend on the point of the execution at which they are evaluated. We take for the commitment extractor of either scheme

$$\mathsf{Ext}_{\mathsf{com}} \;:\; \theta_{\mathsf{com}} \longmapsto \hat{w} \;:=\; \text{the first } \omega \text{ entries of } \hat{W} := \hat{\boldsymbol{P}}(\Omega) \in \mathbb{F}^{n\times s} \;,$$

with $\hat{w} = \perp$ whenever $(\hat{\boldsymbol{P}}, \hat{M}) = \perp$, and we let $\hat{Q}_i := \langle \boldsymbol{A}_i, \hat{\boldsymbol{P}}\rangle$ and $\hat{Q}' := \langle v_{\mathrm{L}}, \hat{\boldsymbol{P}}\rangle$ denote the polynomials that the extracted witness polynomial induces in the LCS and in the MPCS respectively, so that

$$\sum_{x\in\Omega} \hat{Q}_i(x) \;=\; \langle a_i, \hat{w}\rangle \qquad \text{and} \qquad \big\langle \hat{Q}'(\Omega), v_{\mathrm{R}}\big\rangle \;=\; v_{\mathrm{L}}^{\top}\cdot \hat{W}\cdot v_{\mathrm{R}} \;=\; \widetilde{\hat{w}}(r)\;. \tag{16}$$

Finally, in both proofs, $x_{\mathsf{decs}} := \big(E, \boldsymbol{P}^{+}(E)\big)$ denotes the DECS statement, namely the opening set E together with the evaluations $\boldsymbol{P}(E)$ and $M(E)$ *as sent by the prover*; it is determined at that point of the execution. On a good transcript where x_{decs} is determined, by the statement arrival conditions applied to the DECS, the DECS state at $\pi^{\mathsf{decs}}_{\mathsf{com}}\|x_{\mathsf{decs}}$ is alive if and only if $\sigma = \mathsf{cheating}$, or $\sigma = \mathsf{legit}$ and the claimed evaluations are the true evaluations $\hat{\boldsymbol{P}}(E)$ and $\hat{M}(E)$ of the extracted polynomials. And since prover messages preserve doom, the DECS state on the full DECS transcript is alive only if this holds.

Proof of Theorem 4 (SmallWood-LCS)

Proof. Zero-knowledge. The transcript consists of the DECS part – the commitment and the transcript of $\mathsf{DECS.Prove}$ – together with the challenges $\gamma = (\gamma_1, \ldots, \gamma_m)$ and E, the polynomial Q and the evaluations $\boldsymbol{P}(E), M(E)$. The simulator is obtained in three hybrids from the real distribution H_0:

- H_1: replace the DECS part by the output of $\mathsf{Sim}_{\mathsf{decs}}$ on the statement x_{decs}, which is $\varepsilon^{\mathsf{decs}}_{\mathrm{ZK}}$-close to H_0 by the HVZK of the DECS;
- H_2: replace $\boldsymbol{P}(E)$ by a uniform element of $\mathbb{F}^{n\times s'}$. This is perfect: for a fixed witness matrix $W = \boldsymbol{P}(\Omega)$, the vector polynomial $\boldsymbol{P}$ is an affine function of the random matrix $R = \boldsymbol{P}(\Omega')$, and the map $R \mapsto \boldsymbol{P}(E)$ is a bijection – a polynomial of degree at most $s+s'-1$ vanishing on $\Omega\cup E$, a set of $s+s'$ points since $E \subseteq \mathbb{E}$ and $\Omega \subseteq \mathbb{F}\setminus\mathbb{E}$, is identically zero – so that $\boldsymbol{P}(E)$ is uniform and independent of W;
- H_3: replace Q by a uniformly random polynomial of $\mathbb{K}[X]^{\le d_Q}$ subject to $\sum_{x\in\Omega} Q(x) = \sum_i \gamma_i b_i$, and set $M(E) := Q(E) - \sum_i \gamma_i Q_i(E)$, the values $Q_i(E)$ being computed from the simulated $\boldsymbol{P}(E)$. This is perfect as well: the masking polynomial M is uniform in $\{M \in \mathbb{K}[X]^{\le d_Q} : \sum_{x\in\Omega} M(x) = 0\}$ and independent of R, so that $Q = M + \sum_i \gamma_i Q_i$ is uniform subject to $\sum_{x\in\Omega} Q(x) = \sum_i \gamma_i \langle a_i, w\rangle = \sum_i \gamma_i b_i$, and $M(E)$ is then determined by Q and $\boldsymbol{P}(E)$.

The resulting distribution is independent of w and efficiently samplable from the statement alone, which gives $\varepsilon_{\mathrm{ZK}} = \varepsilon^{\mathsf{decs}}_{\mathrm{ZK}}$.

Round-by-round knowledge soundness. Write $\hat{Q} := \hat{M} + \sum_i \gamma_i \hat{Q}_i$ for the polynomial that the extracted pair and the challenge γ determine. We define the state function on good transcripts as follows. On a (possibly partial) commit transcript θ, (T1) $\mathsf{State}(\theta) := \mathsf{State}_{\mathsf{decs}}(\theta|_{\mathsf{decs}})$. On a transcript of the prove phase, the state is alive if $\sigma = \mathsf{cheating}$, doomed if $\sigma = \mathsf{trashed}$, and otherwise alive if and only if $\hat{w} \neq \perp$ and the following condition holds:

- (T2) on $\pi^{\mathsf{decs}}_{\mathsf{com}}\|x$, where $x = ((a_1, b_1), \ldots, (a_m, b_m))$: $\langle a_i, \hat{w}\rangle = b_i$ for all $i \in [m]$;
- (T3) after the challenge γ has been sent:

$$\sum_{x\in\Omega} \hat{M}(x) \;+\; \sum_{i=1}^{m} \gamma_i\big(\langle a_i, \hat{w}\rangle - b_i\big) \;=\; 0\;;$$

- (T4) after Q has been sent: $\sum_{x\in\Omega} Q(x) = \sum_i \gamma_i b_i$ and $Q = \hat{Q}$;

– (T5) after the challenge E has been sent: $\sum_{x\in\Omega} Q(x) = \sum_i \gamma_i b_i$ and $Q(E) = \hat{Q}(E)$;
– (T6) after the evaluations $\boldsymbol{P}(E), M(E)$ have been sent, and throughout DECS.Prove: the conditions of (T5) hold, the verifier's check $Q(e) = M(e) + \sum_i \gamma_i Q_i(e)$ passes for all $e \in E$, and $\mathsf{State}_{\mathsf{decs}}(\theta|_{\mathsf{decs}})$ is alive, which, as noted above, implies on a legit commit state that the claimed evaluations $\boldsymbol{P}(E), M(E)$ are $\hat{\boldsymbol{P}}(E), \hat{M}(E)$.

Note that (T3) is exactly the verifier's check $\sum_{x\in\Omega} Q(x) = \sum_i \gamma_i b_i$ read on the extracted polynomials, by (16); for an honest prover the first term vanishes and the second one is zero termwise.

This is a valid state function. The initial state, the fact that prover messages cannot cheat, and condition 6 (as well as the stability of the extractor) are inherited from the DECS, all the quantities involved being unchanged by queries or determined by the message part of the transcript; statement arrival holds by construction of (T2). Doom is preserved by prover messages: this is trivial when σ is cheating or trashed, so assume it is legit. The message Q preserves doom: if (T3) is doomed then $\sum_\Omega \hat{Q} \neq \sum_i \gamma_i b_i$, so that either $Q = \hat{Q}$ and the first condition of (T4) fails, or $Q \neq \hat{Q}$ and the second one does. So do the evaluations sent at the last prover message of the scheme and the opening message of DECS.Prove, since (T6) requires the conditions of (T5) and the DECS state is preserved by the latter. Finally, a doomed full transcript either has a trashed commit state – in which case the DECS state at $\pi^{\mathsf{decs}}_{\mathsf{com}} \| x_{\mathsf{decs}}$ is doomed, hence also on the full DECS transcript, and the DECS verifier rejects – or has a legit one and fails one of the conditions of (T6): the verifier's own checks fail, or the DECS state is doomed and the DECS verifier rejects, or the claimed evaluations are the extracted ones and the check $Q(e) = M(e) + \sum_i \gamma_i Q_i(e)$ passes while $Q(E) \neq \hat{Q}(E)$, which is contradictory. In all cases the verifier returns Reject.

The commit phase condition is inherited from the DECS through (T1), with error $\varepsilon^{\mathsf{decs}}_{\mathrm{RBR}}$. The statement arrival condition holds by (T2), since on a legit commit transcript we have $(\hat{w}, x) \in \mathcal{R}$ if and only if $\hat{w} \neq \perp$ and $\langle a_i, \hat{w}\rangle = b_i$ for all i. For the bad transcripts condition, a malicious prover of the scheme run together with the honest verifier is a malicious prover of the DECS, as in the proof of Theorem 2, whence $\varepsilon_{\mathcal{H}} \leq \varepsilon^{\mathsf{decs}}_{\mathcal{H}}$. There remain the doomed-to-alive transitions on verifier messages; if σ is trashed the state remains doomed, so assume it is legit. The transitions occur on three kinds of verifier messages (DECS.Prove has none):

– *a challenge of* DECS.Commit, bounded by $\varepsilon^{\mathsf{decs}}_{\mathrm{RBR}}$ by (T1) and the commit phase condition applied to the DECS;
– *the challenge* γ: if (T2) is doomed, then $\hat{w} = \perp$ – and the state remains doomed – or $\langle a_i, \hat{w}\rangle \neq b_i$ for some i, so that the left-hand side of (T3) is a non-constant affine function of γ – its coefficient in γ_i being nonzero – and vanishes with probability at most $1/|\mathbb{K}|$ over $\gamma \leftarrow \mathbb{K}^m$;
– *the challenge* E: if (T4) is doomed because $\sum_\Omega Q \neq \sum_i \gamma_i b_i$, this does not depend on E and the state remains doomed; otherwise $Q \neq \hat{Q}$, and since both have degree at most d_Q, the nonzero polynomial $Q - \hat{Q}$ has at most d_Q roots in $\mathbb{E}$, so that $Q(E) = \hat{Q}(E)$ holds for at most $\binom{d_Q}{s'}$ of the $\binom{|\mathbb{E}|}{s'}$ possible sets E.

Taking the maximum over these cases gives $\varepsilon_{\mathrm{RBR}} \leq \max\left(\varepsilon^{\mathsf{decs}}_{\mathrm{RBR}}, 1/|\mathbb{K}|, \binom{d_Q}{s'}/\binom{|\mathbb{E}|}{s'}\right)$. □

Proof of Theorem 5 (SmallWood-MCS)

Proof. The proof follows the same lines as that of Theorem 4, the role played there by the constraint $\sum_{x\in\Omega} M(x) = 0$ on the masking polynomial being played here by the value δ revealed by the prover.

Zero-knowledge. The transcript consists of the DECS part, the value δ, the challenges γ and E, the polynomial Q and the evaluations $\boldsymbol{P}(E), M(E)$. Hybrids H_1 and H_2 are as in the previous proof – the argument for $\boldsymbol{P}(E)$ being unchanged – and give $\varepsilon^{\mathsf{decs}}_{\mathrm{ZK}}$. For H_3, we replace δ by a uniform element of $\mathbb{K}$, then, once γ is known, Q by a uniformly random polynomial of $\mathbb{K}[X]^{\leq d_Q}$ subject to $\langle Q(\Omega), v_{\mathrm{R}}\rangle = \delta + \gamma v$, and finally set $M(E) := Q(E) - \gamma Q'(E)$ from the simulated $\boldsymbol{P}(E)$. This is perfect: M is uniform in $\mathbb{K}[X]^{\leq d_Q}$, so $\delta = \langle M(\Omega), v_{\mathrm{R}}\rangle$ is uniform over $\mathbb{K}$ – the vector v_{R} being a tensor product of nonzero vectors, hence nonzero – and, conditioned on δ, the polynomial $Q = M + \gamma Q'$ is uniform subject to $\langle Q(\Omega), v_{\mathrm{R}}\rangle = \delta + \gamma\langle Q'(\Omega), v_{\mathrm{R}}\rangle = \delta + \gamma v$, while $M(E)$ is determined by Q and $\boldsymbol{P}(E)$. Hence $\varepsilon_{\mathrm{ZK}} = \varepsilon^{\mathsf{decs}}_{\mathrm{ZK}}$.

Round-by-round knowledge soundness. Write $\hat{Q} := \hat{M} + \gamma\hat{Q}'$. The state function is defined as in the previous proof – (T1) on commit transcripts and, on the prove phase, alive if σ is cheating, doomed if it

is trashed, and otherwise alive if and only if $\hat{w} \neq \perp$ and the current condition holds – with (T2)–(T4) replaced by:

- (U2) on $\pi_{\mathsf{com}}^{\mathsf{decs}}\|(r, v)$: $\widetilde{\hat{w}}(r) = v$;
- (U3) after δ has been sent: $\widetilde{\hat{w}}(r) = v$ and $\langle \hat{M}(\Omega), v_{\mathrm{R}} \rangle = \delta$;
- (U4) after the challenge γ has been sent:

$$\big(\langle \hat{M}(\Omega), v_{\mathrm{R}} \rangle - \delta\big) \; + \; \gamma \, \big(\widetilde{\hat{w}}(r) - v\big) \; = \; 0 \; ,$$

  which is the verifier's check $\langle Q(\Omega), v_{\mathrm{R}} \rangle = \delta + \gamma v$ read on the extracted polynomials, by (16);
- (U5)–(U7) as (T4)–(T6), with $\sum_{x \in \Omega} Q(x) = \sum_i \gamma_i b_i$ replaced by $\langle Q(\Omega), v_{\mathrm{R}} \rangle = \delta + \gamma v$ and with the verifier's check of (T6) replaced by $Q(e) = M(e) + \gamma Q'(e)$ for all $e \in E$.

The verification that this is a valid state function, the statement arrival and bad transcripts conditions, and the transitions on the challenges of the DECS and on E are as in the previous proof, the message δ preserving doom because the first conjunct of (U3) is that of (U2). The challenge γ is treated as follows, for a legit commit state: if (U3) is doomed because $\hat{w} = \perp$, the state remains doomed; if it is doomed because $\widetilde{\hat{w}}(r) \neq v$, the left-hand side of (U4) is a non-constant affine function of γ and vanishes with probability at most $1/|\mathbb{K}|$ over $\gamma \leftarrow \mathbb{K}$; and if it is doomed only because $\langle \hat{M}(\Omega), v_{\mathrm{R}} \rangle \neq \delta$, that left-hand side is a nonzero constant and the state remains doomed. We conclude as before that $\varepsilon_{\mathcal{H}} \leq \varepsilon_{\mathcal{H}}^{\mathsf{decs}}$ and $\varepsilon_{\mathrm{RBR}} \leq \max\big(\varepsilon_{\mathrm{RBR}}^{\mathsf{decs}}, 1/|\mathbb{K}|, \binom{d_Q}{s'}/\binom{|\mathbb{E}|}{s'}\big)$. □