Spartan and Bulletproofs are simulation-extractable (for free!)

Quang Dao[1] and Paul Grubbs[2]

[1] Carnegie Mellon University
qvd@andrew.cmu.edu **
[2] University of Michigan
paulgrub@umich.edu

Abstract. Increasing deployment of advanced zero-knowledge proof systems, especially zkSNARKs, has raised critical questions about their security against real-world attacks. Two classes of attacks of concern in practice are *adaptive soundness* attacks, where an attacker can prove false statements by choosing its public input after generating a proof, and *malleability* attacks, where an attacker can use a valid proof to create another valid proof it could not have created itself. Prior work has shown that *simulation-extractability* (SIM-EXT), a strong notion of security for proof systems, rules out these attacks.

In this paper, we prove that two transparent, discrete-log-based zkSNARKs, Spartan and Bulletproofs, are simulation-extractable (SIM-EXT) in the random oracle model if the discrete logarithm assumption holds in the underlying group. Since these assumptions are required to prove standard security properties for Spartan and Bulletproofs, our results show that SIM-EXT is, surprisingly, "for free" with these schemes. Our result is the first SIM-EXT proof for Spartan and encompasses both linear- and sublinear-verifier variants. Our result for Bulletproofs encompasses both the aggregate range proof and arithmetic circuit variants, and is the first to not rely on the algebraic group model (AGM), resolving an open question posed by Ganesh et al. (EUROCRYPT '22). As part of our analysis, we develop a generalization of the tree-builder extraction theorem of Attema et al. (TCC '22), which may be of independent interest.

1 Introduction

Zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) allow a computationally-bounded prover to produce a proof about a NP statement *without* revealing anything other than its validity, and with proof size *sublinear* in the size of the witness [12,32,34]. An important line of recent works [7,9,13,15,17,20,32,35,38,44,48,60,64] has produced concretely efficient constructions of zkSNARKs for range proofs (e.g., Bulletproofs [16]) and general arithmetic circuit satisfiability (e.g., Spartan [55]) that have seen widespread deployment, especially in blockchains and cryptocurrencies [8,59,1,54,52,21,65], along with potential deployment in other areas of interests [40].

As zkSNARKs are deployed in practice, it is important to understand whether they are actually secure against the kinds of attacks they are likely to face in real systems. Two security properties in particular give us pause: first, *adaptive soundness*, where a malicious prover must be unable to prove false statements even if it chooses the input after generating a proof; a related notion, adaptive knowledge soundness, guarantees extraction is possible against such an adaptive prover. The second property is *non-malleability*, where an accepting proof cannot be modified into a different one without knowing the witness. Neither property is implied by standard security definitions like non-adaptive (knowledge) soundness and zero knowledge, and schemes lacking these properties have been attacked in practice. For example, the voting system Helios was broken by an adaptive soundness attack on a zero-knowledge proof [11]; subsequent work found similar issues with the SwissPost voting system [41] for government elections. Though not against zero-knowledge proofs directly, malleability attacks are common in cryptocurrencies: for example, a malleability attack was allegedly used[3] to steal hundreds of millions of dollars from MtGox [47].

** Part of the work was done while the first author was at the University of Michigan.

[3] A later study [22] cast some doubt on these claims, but did find evidence that over three hundred thousand Bitcoins had been involved in malleability attacks.

Fortunately, a security property called *simulation extractability* (SIM-EXT) implies adaptive (knowledge) soundness and non-malleability for zkSNARKs. Intuitively, SIM-EXT requires that the knowledge extractor succeeds even when the malicious prover can request *simulated* proofs for *arbitrary* statements. If we could prove zkSNARKs that are already used (or are likely to be used) in practice are SIM-EXT, we could be more confident they would resist advanced attacks that use adaptivity or malleability. Ideally, we could prove SIM-EXT in idealized models (e.g, the random oracle model, or ROM) and using assumptions (e.g. discrete-log), which are sufficient to prove standard security guarantees for zkSNARKs; this would indicate SIM-EXT comes (roughly) "for free".

A pair [29,30] of beautiful recent works by Ganesh et al. on SIM-EXT for zkSNARKs lays a path towards this goal. In [29], the authors give a general SIM-EXT theorem for zkSNARKs with updatable SRS, and use it to show PlonK [28], Marlin [20], and Sonic [45] are all SIM-EXT. In [30], the authors show SIM-EXT for Bulletproofs. Unfortunately, these works do not get us all the way towards our goal: first, because their techniques do not extend to *transparent* zkSNARKs like Spartan, which use different building blocks; second, because their results rely on the algebraic group model (AGM) [27] and are not currently known to hold from discrete log in the ROM.

1.1 Our Results

In this paper we prove that Spartan and Bulletproofs, two state-of-the-art transparent zkSNARKs, satisfy SIM-EXT in the ROM assuming only that the discrete log assumption holds. Our analyses required developing some new technical tools which may be of independent interest. Since Spartan and Bulletproofs were originally analyzed in the ROM and rely on the discrete log assumption, our results imply these protocols are SIM-EXT "for free"—unmodified and without additional assumptions or stronger idealized models. More precisely, we prove SIM-EXT for two variants of Spartan—Spartan-NIZK, which has linear verifier time, and Spartan-SNARK, which has sublinear verifier time—instantiated with the default Hyrax-based polynomial commitment scheme [60]. These are the first proofs of SIM-EXT for any Spartan variant; we believe the Spartan-SNARK result is also the first proof of SIM-EXT for *any* transparent zkSNARK with sublinear verifier time. Similarly, we prove SIM-EXT for two versions of Bulletproofs—the aggregate range proof protocol BP-ARP used in several cryptocurrencies [36,46] and the arithmetic circuit satisfiability proof BP-ACSPf. Our proofs for these protocols are the first that do not rely on the algebraic group model.

Our results help to build confidence that state-of-the-art and deployed zkSNARKs resist the kinds of attacks these protocols will face as they see wider deployment in the future. Of more theoretical interest, they also imply the surprising fact that, in the ROM, a powerful primitive like a SIM-EXT zkSNARK can be built from a very weak assumption like discrete log.

The proofs of these four theorems are nontrivial; to prove them we built several new technical tools that may be of independent interest for future SIM-EXT analyses. We extended prior security notions for SIM-EXT to the transparent NIZK setting. We also needed to develop a nontrivial generalization of the tree extractor of Attema et al. [2].

Our analyses are also done with an emphasis on concrete security. Where possible we try to explicitly measure adversarial runtime and success probability. We also evaluate our bounds to estimate bit security for typical parameters for Spartan and Bulletproofs, and compare the bit security we obtain against other analyses where possible. Our bounds inherit the non-tightness common to most rewinding-based knowledge soundness analyses of NIZKs, and so the provable SIM-EXT security we get (in terms of bits) is quite low. Nevertheless, we believe our results can be improved by future work, and hope they eventually inform future parameter selection processes for zkSNARK standards [66].

1.2 Technical Overview

We follow the high-level approach to proving SIM-EXT developed by [24] and further generalized in [29,30]: for a Fiat-Shamir-compiled argument Π_{FS}, SIM-EXT is implied by three other properties: (1) adaptive knowledge soundness, (2) a form of zero knowledge, and (3) a unique-response property. Since the results in [24] are specific to Σ-protocols and those in [30] are specific to the AGM, we take the SIM-EXT theorem of [29] as our starting point. After suitable adaptations to the transparent setting—we give these in Section 3—this theorem says that Π_{FS} is $\mathsf{SIM\text{-}EXT}$ in the ROM if:

1. it is adaptively knowledge sound (hereafter we will omit "adaptive" if it is clear from context),
2. it is perfect k-ZK, meaning that there exists a simulator that perfectly simulates honest proofs, but only programs the RO when generating the k-th challenge,
3. it is k-UR for the *same* round k, meaning no adversary can produce two accepting proofs that are identical up to the k-th round, *even if* it can program that round's challenge.

Proving these three properties is challenging, and required us to develop novel techniques which we summarize below.

Knowledge Soundness. We prove knowledge soundness for non-interactive versions of Spartan and Bulletproofs using a standard chain of reductions: namely, we reduce to the *special soundness* of the underlying interactive argument. Intuitively, special soundness of a proof system refers to the ability of an extractor to extract a witness from a *tree* of accepting transcripts with suitable structure. For multi-round protocols, special soundness is parameterized by a vector $(n_1, n_2, \ldots, n_r)$ describing the needed structure: each node at level one must have n_1 outgoing edges, level two nodes have n_2 edges, etc. Recently, Attema et al. [2] proved that knowledge soundness of the Fiat-Shamir-compiled argument Π_{FS} follows from special soundness of Π. We take it as our starting point; unfortunately, we cannot apply it directly to either Spartan or Bulletproofs. There are two main reasons for this: first, Attema et al. only consider perfect special soundness, but both Spartan and Bulletproofs only satisfy *computational* special soundness—roughly, because an extractor could fail to extract a witness from a tree of transcripts if a malicious prover finds a nontrivial discrete log relation.

The second reason is more subtle, and has to do with ensuring the tree has the right structure for extraction to be possible. In Attema et al., each node of the transcript tree is a prover message whose outgoing edges are labeled with distinct verifier challenges. For certain rounds in both Spartan and Bulletproofs, these verifier challenges must satisfy an extra predicate (beyond distinctness) for extraction to be possible. The tree-builder by Attema et al. does not support outputting such trees with extra structure.

To address these limitations, in Section 4 we give a generalization of Attema et al.'s tree-builder that has the desired properties. Our generalization captures other predicates on verifier challenges using the notion of an *efficiently-decidable partition* of the space of challenges. Intuitively, we build a wrapper algorithm that that sits between the prover and the Attema et al. tree-builder, and ensures the tree has the right structure by enforcing a partition of the challenge space.

Armed with this generalization, we prove computational special soundness for all variants of $\mathsf{Spartan}$ and $\mathsf{Bulletproofs}$, which in turn implies knowledge soundness for their Fiat-Shamir-compiled versions. In both cases, our generalized tree-builder is a crucial component: for example, special soundness of Bulletproofs requires verifier challenges to be distinct modulo ± 1, and Spartan requires linear independence for batching challenges sent during the sumcheck subprotocol.

Building k-ZK Simulators. For $\mathsf{SIM\text{-}EXT}$, we must prove that Spartan and Bulletproofs are perfect k-ZK, meaning their proofs can be simulated by a simulator that can only program the RO in a single round. This is a departure from the typical way to build NIZK simulators, which typically reprogram the RO in every round; in particular, doing this for Spartan and Bulletproofs requires giving entirely new simulators for these constructions.

We build our k-ZK simulator for Bulletproofs using an approach similar to [29]. Our k-ZK simulator construction for $\mathsf{Spartan\text{-}NIZK}$ uses a novel strategy that is worth highlighting here: it delays the round at which the RO is reprogrammed as late as possible in the protocol (in fact, our simulator only needs to reprogram the very last verifier challenge). Another interesting aspect of our k-ZK simulator for Spartan is that the same simulator works for both $\mathsf{Spartan\text{-}NIZK}$ and $\mathsf{Spartan\text{-}SNARK}$—though the two protocols have major differences, we observe that the parts of $\mathsf{Spartan\text{-}SNARK}$ that work differently than $\mathsf{Spartan\text{-}NIZK}$ consist entirely of evaluating (extensions of) public matrices at a public point, and so are trivially simulatable.

k-Unique Response. To finish, we need to show Spartan and Bulletproofs are k-UR for the same k as their respective k-ZK simulators. For Spartan variants, this is straightforward—we need only reprogram the RO during the final Σ-subprotocol, and it is well known [24] that Σ-protocols satisfy unique response.

For $\mathsf{BP\text{-}ARP}$ and $\mathsf{BP\text{-}ACSPf}$, proving k-UR is more challenging. Indeed, prior work relied heavily on the AGM for analyzing unique response—for example, [30] observe that proving their version of unique response is the only part of their analysis that seems to actually rely on the AGM, and [29] need the AGM to show that KZG polynomial commitments are unique response.

We prove k-UR for Bulletproofs using a new proof strategy that, intuitively, replaces the AGM with extraction. In more detail, we extract witnesses from both proofs output by the k-UR adversary, then argue that *either* the witnesses are the same *or* the adversary has found a discrete log relation. To finish, we use the (novel) result that the Bulletproofs inner-product argument has unique proofs. Thus, if the witnesses are the same, the proofs must be the same as well.

Limitations and open questions. Our results do have some important limitations. Notably, our emphasis on removing the AGM means that the tightness of our Bulletproofs results is worse than the comparable result of [30]. While this is inherent in some sense because our extractors use rewinding instead of straight-line extraction, it means that the bit security of Bulletproofs and Spartan we could prove with typical parameters would come out to be quite poor. We discuss this in Section 7.

An interesting open problem we leave to future work is generalizing our techniques to other transparent zkSNARKs. In particular, there is a great deal of commonality between our proofs for Spartan and Bulletproofs which could be abstracted out and proven more generally. As many later works [44,64,35,56] have built on Spartan viewed as a *polynomial IOP* [17,20], it would be interesting to generalize our analyses into a $\mathsf{SIM\text{-}EXT}$ framework for polynomial IOPs.

1.3 Related Work

Simulation-extractability ($\mathsf{SIM\text{-}EXT}$) for NIZKs was first defined in [53] (using different terminology). Thereafter, a long line of work refined and studied $\mathsf{SIM\text{-}EXT}$ [24,50], built $\mathsf{SIM\text{-}EXT}$ NIZKs [37], and showed that $\mathsf{SIM\text{-}EXT}$ is sufficient for other primitives like signatures of knowledge [19]. Other concurrent works attacked security of NIZKs in deployed systems, such as the voting system Helios, showing the importance of adaptive soundness [11] which is implied by $\mathsf{SIM\text{-}EXT}$. Other work has looked at UC security for NIZKs [18] and given results on $\mathsf{SIM\text{-}EXT}$ in the QROM [23]. These works are not relevant to our results, since $\mathsf{SIM\text{-}EXT}$ does not imply UC security in the ROM; further, we study zkSNARKs built from discrete log, which is broken by quantum attacks.

The simulation-extractability of zkSNARKs is comparatively less well-studied. Two important prior works [29,30] which rely on the algebraic group model [27] (AGM) are described above; [30] proves $\mathsf{SIM\text{-}EXT}$ of Bulletproofs, and [29] proves $\mathsf{SIM\text{-}EXT}$ of Plonk [28], Marlin [20], and Sonic [45].

Other work has investigated generic transforms for achieving $\mathsf{SIM\text{-}EXT}$ from any zkSNARK [5], particularly focused on $\mathsf{SIM\text{-}EXT}$ transforms for the Groth16 zkSNARK [3,4]. Since Groth16 [38] is built using a different approach than either Spartan or Bulletproofs, and relies on non-falsifiable knowledge assumptions or the AGM, our results are incomparable to theirs.

Our paper analyzes $\mathsf{SIM\text{-}EXT}$ for Bulletproofs [16] and Spartan [55], two transparent zkSNARKs built from discrete-log assumptions. There is a line of related work building similar SNARKs, such as Hyrax [60], and extensions to recursive composition like Halo [15] and Nova [44]. We suspect our techniques would extend to these constructions, and leave extending them to future work.

A key technical tool our results rely on is a "tree-builder" for proving knowledge soundness of NIZKs built from multi-round interactive arguments. As described above, our approach is a generalization of a beautiful recent work by Attema et al. [2]. This work develops a tree-builder for *perfect* special sound protocols which are extractable given a tree of *distinct* verifier challenges; we generalize their result to support *computational* special soundness and to allow different conditions on verifier challenges. Wikstrom [63] gives an alternate construction and analysis of a tree-builder which could have served as a starting point for us; however, their extractor has a worse concrete running time and tightness than Attema et al. In a revision of [17], the authors generalize Attema et al.'s tree builder to handle general predicates on prover messages; since we need more general predicates on verifier challenges, their generalization is not directly useful to us. Other recent works [33,42] analyze the knowledge soundness of Bulletproofs in the AGM/GGM without using an explicit tree-builder by, for example, going through the notion of round-by-round soundness [10].

Concurrent work. After the acceptance of this paper, Ganesh et al. posted a full and revised version [31] of their conference version [30]. Their revised version proves that Bulletproofs satisfy SIM-EXT in the ROM only, removing the need for the AGM. We note that their technique is somewhat different from ours: in particular, they prove that Bulletproofs satisfy a different notion of *weak unique response* (FS-WUR), which turns out to be enough for their version of the SIM-EXT theorem (cf. our Theorem 3.4). Their proof of FS-WUR shares some similarity with ours, however, namely in the use of rewinding to extract witnesses from non-simulated proofs. Nevertheless, we note that our results additionally include proving that Spartan satisfies SIM-EXT.

2 Preliminaries

We use $\mathbb{F}$ to denote a finite field with $\mathbb{F}^* = \mathbb{F} - \{0\}$, and λ to denote the security parameter. For $k, n \in \mathbb{N}$, we denote $[k, n] = \{k, k+1, \ldots, n\}$, and $[n] = [1, n]$. We denote uniform sampling from a set S by $a \xleftarrow{\$} S$. We denote vectors by boldface, e.g. $\mathbf{g} = (g_1, \ldots, g_n)$, and write $\mathbf{g}^{\mathbf{a}}$ to mean $g_1^{a_1} \cdot \cdots \cdot g_n^{a_n}$. We denote the length of a vector $\mathbf{a}$ by $|\mathbf{a}|$, the inner product between two vectors $\mathbf{a}, \mathbf{b}$ by $\mathbf{a} \cdot \mathbf{b}$ or $\langle \mathbf{a}, \mathbf{b} \rangle$, the Hadamard (entry-wise) product by $\mathbf{a} \circ \mathbf{b}$, and the tensor product by $\mathbf{a} \otimes \mathbf{b} = (a_1 b_1, \ldots, a_1 b_m, \ldots, a_n b_1, \ldots, a_n b_m)$.

Our relations are of the form $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$ and are efficiently decidable, e.g. there exists a deterministic polynomial time algorithm that given (pp, x, w) outputs whether $(\mathsf{pp}, x, w) \in \mathcal{R}$. We abbreviate PPT for probabilistic polynomial time, and EPT for expected (probabilistic) polynomial time.

We use *code-based games* [6] to define many of our security notions. A game $\mathsf{G}_{\mathsf{S}}^{\mathcal{A}_1, \ldots, \mathcal{A}_n}$ denotes a run of parties $\mathcal{A}_1, \ldots, \mathcal{A}_n$ on a pre-specified set of procedures given by S, returning a bit $b \in \{0,1\}$. We denote $\Pr[\mathsf{G}_{\mathsf{S}}^{\mathcal{A}_1, \ldots, \mathcal{A}_n}]$ the probability over the random coins used by S and all adversaries that the game's output is 1.

Lemma 2.1 (Schwartz-Zippel Lemma). *Let $\mathbb{F}$ be a finite field and $f \in \mathbb{F}^{\leq d}[X_1, \ldots, X_n]$ be a non-zero multivariate polynomial with total degree at most d. Let S be a subset of $\mathbb{F}$. Then*

$$\Pr[f(x_1, \ldots, x_n) = 0] \leq d/|S|,$$

where the probability is taken over the choice of $x_i \xleftarrow{\$} S$ for all $i = 1, \ldots, n$.

2.1 Assumptions

We assume the existence of a group generator generating *global public parameters* $\mathsf{pp}_{\mathcal{G}} := (\mathbb{G}, \mathbb{F}) \leftarrow \mathsf{GroupGen}(1^\lambda)$, where $\mathbb{G}$ is a group of prime order, with $\mathbb{F}$ as the corresponding field. These global parameters are used in the setup phase of every protocol we consider. We also assume a generator sampling procedure $g_1, \ldots, g_n \xleftarrow{\$} \mathsf{GenSamp}(\mathbb{G}, n)$.

We define the following assumptions with respect to an adversary $\mathcal{A}$ running in *expected polynomial time.*[4]

Definition 2.2 (Discrete Log [33]). *We say that the* discrete log (DL) *assumption holds for the tuple of algorithms ($\mathsf{GroupGen}$, $\mathsf{GenSamp}$) (see Section 2.1) if for all EPT adversaries $\mathcal{A}$, the following probability is negligible in λ:*

$$\mathbf{Adv}^{\mathsf{DL}}_{\mathsf{GroupGen}, \mathsf{GenSamp}}(\mathcal{A}) := \Pr\left[\mathsf{DL}^{\mathcal{A}}_{\mathbb{G}}(\lambda)\right].$$

[4] This is because we can only reduce breaking the security of the protocols we consider to such adversaries breaking DL or DL-REL.

Game $\mathsf{DL}^{\mathcal{A}}_{\mathsf{GroupGen},\mathsf{GenSamp}}(\lambda)$	**Game** $\mathsf{DL\text{-}REL}^{\mathcal{A}}_{\mathsf{GroupGen},\mathsf{GenSamp},n}(\lambda)$
$(\mathbb{G},\mathbb{F}) \leftarrow \mathsf{GroupGen}(1^\lambda)$	$(\mathbb{G},\mathbb{F}) \leftarrow \mathsf{GroupGen}(1^\lambda)$
$g,h \xleftarrow{\$} \mathsf{GenSamp}(\mathbb{G},2)$	$g_1,\ldots,g_n \xleftarrow{\$} \mathsf{GenSamp}(\mathbb{G},n)$
$a \leftarrow \mathcal{A}(g,h)$	$(a_1,\ldots,a_n) \leftarrow \mathcal{A}(g_1,\ldots,g_n)$
return $(g^a = h)$	**return** $\left(\prod_{i=1}^{n} g_i^{a_i} = 1\right) \wedge ((a_1,\ldots,a_n) \neq 0^n)$

Fig. 1: Games for Discrete Log

Definition 2.3 (Discrete Log Relation [33]). *We say that the* discrete log relation ($\mathsf{DL\text{-}REL}$) *assumption holds for* ($\mathit{GroupGen}$, $\mathit{GenSamp}$) *if for all EPT adversaries $\mathcal{A}$ and all $n \in \mathbb{N}$, the following probability is negligible in λ:*

$$\mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathit{GroupGen},\mathit{GenSamp},n}(\mathcal{A},\lambda) := \Pr\left[\mathsf{DL\text{-}REL}^{\mathcal{A}}_{\mathit{GroupGen},\mathit{GenSamp},n}(\lambda)\right].$$

The two discrete log assumptions are tightly related.

Lemma 2.4 (Lemma 1 [33]). *For every EPT adversary $\mathcal{A}$ against* $\mathsf{DL\text{-}REL}$*, there exists a EPT adversary $\mathcal{B}$ against* DL*, nearly as efficient as $\mathcal{A}$, such that*

$$\mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathit{GroupGen},\mathit{GenSamp},n}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{DL}}_{\mathit{GroupGen},\mathit{GenSamp}}(\mathcal{B}) + \frac{1}{|\mathbb{F}|}.$$

By a slight abuse of notation, we will abbreviate $\mathbb{G}$ for ($\mathsf{GroupGen}$, $\mathsf{GenSamp}$) in the rest of the paper.

2.2 Interactive Arguments

We define an interactive argument for relation $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$.

Definition 2.5. *An* interactive argument *for a relation $\mathcal{R}$ is a tuple of PPT algorithms $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ with the following syntax:*

- $\mathsf{Setup}(\mathsf{pp}_{\mathcal{G}}) \to \mathsf{pp}$: *outputs public parameters* pp *given global parameters* $\mathsf{pp}_{\mathcal{G}}$*,*
- $\langle \mathcal{P}(w), \mathcal{V} \rangle(\mathsf{pp}, x) \to \{0,1\}$: *an interactive protocol whereby the prover $\mathcal{P}$, holding a witness w, interacts with the verifier $\mathcal{V}$ on common input* (pp, x) *to convince $\mathcal{V}$ that* $(\mathsf{pp}, x, w) \in \mathcal{R}$*. At the end, $\mathcal{V}$ outputs a bit for accept/reject.*

We define the following properties for interactive arguments:

- **Completeness.** For any adversary $\mathcal{A}$,

$$\Pr\left[\begin{matrix}(\mathsf{pp},x,w) \notin \mathcal{R} \vee \\ \langle \mathcal{P}(w),\mathcal{V}\rangle(\mathsf{pp},x) = 1\end{matrix} \quad : \quad \begin{matrix}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{pp}_{\mathcal{G}}) \\ (x,w) \leftarrow \mathcal{A}(\mathsf{pp})\end{matrix}\right] = 1.$$

- **Knowledge Soundness.** There exists a expected polynomial time extractor $\mathcal{E}$ such that for any stateful PPT adversary $\mathcal{P}^*$,

$$\Pr\left[b = 1 \ \wedge \ (\mathsf{pp},x,w) \notin \mathcal{R} \quad : \quad \begin{matrix}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{pp}_{\mathcal{G}}) \\ (x, \mathsf{st}_{\mathcal{P}^*}) \leftarrow \mathcal{P}^*(\mathsf{pp}) \\ b \leftarrow \langle \mathcal{P}^*(\mathsf{st}_{\mathcal{P}^*}), \mathcal{V}\rangle(\mathsf{pp},x) \\ w \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{pp},x)\end{matrix}\right] \leq \mathsf{negl}(\lambda).[5]$$

Here $\mathcal{E}$ gets black-box access to each of the next-message functions of $\mathcal{P}^*$ in the interactive protocol, and can rewind $\mathcal{P}^*$ to any point in the interaction.

[5] We do not define soundness, but this implies soundness with the same advantage.

Definition 2.6 (Honest-Verifier Zero-Knowledge). *A public-coin interactive argument $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R}$ is* ϵ-honest-verifier zero-knowledge (ϵ-HVZK) *if there exists a PPT simulator $\mathcal{S}$ such that for all $\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$ and $(\mathsf{pp}, x, w) \in \mathcal{R}$, the following distributions are ϵ-statistically indistinguishable*

$$\{\mathsf{View}_{\mathcal{V}}\langle \mathcal{P}(\mathsf{pp}, x, w), \mathcal{V}(\mathsf{pp}, x)\rangle\} \approx_s \{\mathcal{S}(\mathsf{pp}, x)\}.$$

Here $\mathsf{View}_{\mathcal{V}}\langle \mathcal{P}(\mathsf{pp}, x, w), \mathcal{V}(\mathsf{pp}, x)\rangle$ denotes the view of the verifier, consisting of the transcript and its own randomness. Π satisfies perfect HVZK if it is 0-HVZK.

Definition 2.7 (Public-Coin). *An interactive argument $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is* public-coin *if in each round i the verifier $\mathcal{V}$ samples its message uniformly at random from some challenge space Ch_i, and uses no other randomness.*

Any public-coin interactive argument has a general $(2r + 2)$-message, or equivalently, $(r + 1)$-round format where the verifier sends the 0-th message, and the prover sends the last message. In particular, the transcript is of the form $\mathsf{tr} = (c_0, a_1, c_1, \ldots, a_r, c_r, a_{r+1})$, where $(a_1, \ldots, a_{r+1})$ are the prover's messages and $(c_0, \ldots, c_r)$ are the verifier's messages. Additionally, we have $c_0 = \emptyset$ in all protocols we consider, so that we will only consider $(2r+1)$-message protocols (where the prover sends the first and last message).

2.3 Non-Interactive Arguments in the ROM

In practice, we often use the *Fiat-Shamir transform* (see Section 2.4) to compile public-coin interactive arguments into their *non-interactive* versions, in a model where both parties have black-box access to a *random oracle*, i.e. a uniformly sampled function $\mathsf{H} : \{0, 1\}^* \to \{0, 1\}^\lambda$. For public-coin $(2r + 1)$-message interactive arguments with challenge spaces $\mathsf{Ch}_1, \ldots, \mathsf{Ch}_r$, we will actually need r independent random oracles $\mathsf{H}_i : \{0, 1\}^* \to \mathsf{Ch}_i$ with $i \in [1, r]$. For simplicity, we will denote these by a single random oracle H, and it will be clear from context which random oracle is being used in a given round.

Definition 2.8. *A* non-interactive argument (NARG) *in the ROM for a relation $\mathcal{R} \subseteq \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^*$ is a tuple of algorithms $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$, with $\mathcal{P}, \mathcal{V}$ having black-box access to a random oracle H, with the following syntax:*

- $\mathsf{Setup}(\mathsf{pp}_{\mathcal{G}}) \to \mathsf{pp}$ *generates the public parameters,*
- $\mathcal{P}^{\mathsf{H}}(\mathsf{pp}, x, w) \to \pi$ *generates a proof given* pp *and an input-witness pair (x, w),*
- $\mathcal{V}^{\mathsf{H}}(\mathsf{pp}, x, \pi) \to \{0, 1\}$ *checks if proof π is valid for* pp *and input x.*

We define the following properties of NARGs:

- **Completeness.** For every adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l}(\mathsf{pp}, x, w) \notin \mathcal{R} \vee \\ \mathcal{V}^{\mathsf{H}}(\mathsf{pp}, x, \pi) = 1\end{array} \quad : \quad \begin{array}{c}\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}}) \\ (x, w) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathsf{pp}) \\ \pi \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{pp}, x, w)\end{array}\right] = 1.$$

- **Knowledge Soundness.** Π is *(adaptively) knowledge sound (*KS*)* if there exists an extractor $\mathcal{E}$ running in expected polynomial time such that for every PPT adversary $\mathcal{P}^*$, the following probability is negligible in λ:
$$\mathbf{Adv}^{\mathsf{KS}}_{\Pi_{\mathsf{FS}}, \mathcal{R}}(\mathcal{E}, \mathcal{P}^*) := \left|\Pr[\mathsf{KS}^{\mathcal{P}^*}_{0, \Pi_{\mathsf{FS}}}(\lambda)] - \Pr[\mathsf{KS}^{\mathcal{E}, \mathcal{P}^*}_{1, \Pi_{\mathsf{FS}}, \mathcal{R}}(\lambda)]\right|.$$
The knowledge soundness games are defined in Figure 2.

We define zero-knowledge in a model where the random oracle is *explicitly-programmable* [61] by the simulator. Here, the simulator $\mathcal{S}$ can reprogram the random oracle H, and this modified oracle is provided to the distinguisher.

Definition 2.9 (Zero-Knowledge). *Π satisfies* (statistical) unbounded non-interactive zero-knowledge *(NIZK) if there exists a PPT simulator $\mathcal{S}$ such that for $\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$ and any unbounded distinguisher $\mathcal{D}$, the following probability is negligible in λ:*
$$\mathbf{Adv}^{ZK}_{\Pi_{\mathsf{FS}}, \mathcal{R}}(\mathcal{S}, \mathcal{D}) := \left|\Pr\left[ZK^{\mathcal{D}, \mathcal{P}}_{0, \Pi_{\mathsf{FS}}, \mathcal{R}}(\lambda)\right] - \Pr\left[ZK^{\mathcal{D}, \mathcal{S}}_{1, \Pi_{\mathsf{FS}}, \mathcal{R}}(\lambda)\right]\right|.$$
The zero-knowledge games are defined in Figure 3.

Game $\mathsf{KS}^{\mathcal{P}^*}_{0,\Pi_{\mathsf{FS}}}(\lambda)$	Game $\mathsf{KS}^{\mathcal{E},\mathcal{P}^*}_{1,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$	$\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$
$(x, \pi) \leftarrow (\mathcal{P}^*)^{\mathsf{H}}(\mathsf{pp})$	$(x, \pi) \leftarrow (\mathcal{P}^*)^{\mathsf{H}}(\mathsf{pp})$
$b \leftarrow \mathcal{V}^{\mathsf{H}}_{\mathsf{FS}}(\mathsf{pp}, x, \pi)$	$b \leftarrow \mathcal{V}^{\mathsf{H}}_{\mathsf{FS}}(\mathsf{pp}, x, \pi)$
return b	$w \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{pp}, x, \pi)$
	return $b \wedge (\mathsf{pp}, x, w) \in \mathcal{R}$

Fig. 2: Knowledge soundness security games. Here the extractor $\mathcal{E}$ is given black-box access to $\mathcal{P}^*$. In particular, $\mathcal{E}$ implements H for $\mathcal{P}^*$ and can rewind $\mathcal{P}^*$ to any point.

Game $\mathsf{ZK}^{\mathcal{D},\mathcal{P}}_{0,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)$	Game $\mathsf{ZK}^{\mathcal{D},\mathcal{S}}_{1,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$	$\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$
$b \leftarrow \mathcal{D}^{\mathsf{H}(\cdot),\mathcal{P}'(\mathsf{pp},\cdot,\cdot)}(1^\lambda)$	$b \leftarrow \mathcal{D}^{\mathsf{H}(\cdot),\mathcal{S}'(\mathsf{pp},\cdot,\cdot)}(1^\lambda)$
return b	**return** b
$\mathcal{P}'(\mathsf{pp}, x, w)$	$\mathcal{S}'(\mathsf{pp}, x, w)$
if $(\mathsf{pp}, x, w) \notin \mathcal{R}$ **then return** $\perp$	**if** $(\mathsf{pp}, x, w) \notin \mathcal{R}$ **then return** $\perp$
else return $\mathcal{P}(\mathsf{pp}, x, w)$	**else return** $\mathcal{S}^{\mathsf{RePro}}(\mathsf{pp}, x)$

Fig. 3: Zero-knowledge security games. Here the simulator $\mathcal{S}$ gets access to a RePro oracle that on input (a, b) reprograms $\mathsf{H}(a) := b$.

2.4 The Fiat-Shamir Transformation

We define the Fiat-Shamir transform [25], which removes interaction from any public-coin interactive argument.

Definition 2.10 (Fiat-Shamir Transformation). *Let $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be a public-coin $(2r + 1)$-message interactive argument of knowledge. Denote the transcript as $\mathsf{tr} = (a_1, c_1, \ldots, a_r, c_r, a_{r+1})$. The* Fiat-Shamir transformation *turns Π into a non-interactive protocol Π_{FS} in the ROM, where:*

- $\mathsf{Setup}_{\mathsf{FS}}(\mathsf{pp}_{\mathcal{G}})$ *is the same as* $\mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$,
- *the prover $\mathcal{P}_{\mathsf{FS}}$, on input (pp, x, w), invokes $\mathcal{P}(x, w)$, and instead of asking the verifier for challenge c_i in round i, queries the random oracle to get*
$$c_i = \mathsf{H}(\mathsf{pp}, x, a_1, \ldots, a_i) \text{ for all } i = 1, \ldots, r.$$
$\mathcal{P}_{\mathsf{FS}}$ then outputs a non-interactive proof $\pi = (a_1, \ldots, a_r, a_{r+1})$.
- *the verifier $\mathcal{V}_{\mathsf{FS}}$, on input (pp, x, π), derives challenges c_i's by querying the random oracle as $\mathcal{P}_{\mathsf{FS}}$ does, then runs $\mathcal{V}(\mathsf{pp}, x, (a_1, c_1, \ldots, a_r, c_r, a_{r+1}))$ and outputs what $\mathcal{V}$ outputs.*

For all protocols Π considered in this paper, it is clear that both Π and Π_{FS} satisfy (perfect) completeness. Furthermore, Π_{FS} satisfies knowledge soundness if Π is *(computationally) special sound* (see Section 4). For zero-knowledge, we have a *canonical simulator* $\mathcal{S}_{\mathsf{FS}}$ for Π_{FS} based on any HVZK simulator $\mathcal{S}$ for Π.

Definition 2.11 (Canonical Simulator). *Let Π be a public-coin interactive argument with HVZK simulator $\mathcal{S}$. Define the* canonical simulator *$\mathcal{S}_{\mathsf{FS}}$ for Π_{FS} to be an algorithm that on input (pp, x) runs $\mathcal{S}(\mathsf{pp}, x)$ to get a transcript $\mathsf{tr} = (a_1, c_1, \ldots, a_r, c_r, a_{r+1})$, then reprogram $\mathsf{H}(\mathsf{pp}, x, a_1, \ldots, a_i) := c_i$ for all $i \in [r]$.*

Game $\mathsf{SIM\text{-}EXT}^{\mathcal{S},\mathcal{P}^*}_{0,\Pi_{\mathsf{FS}}}(\lambda)$	**Game** $\mathsf{SIM\text{-}EXT}^{\mathcal{E},\mathcal{S},\mathcal{P}^*}_{1,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$	$\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$
$(x, \pi) \leftarrow (\mathcal{P}^*)^{\mathsf{H},\mathcal{S}}(\mathsf{pp})$	$(x, \pi) \leftarrow (\mathcal{P}^*)^{\mathsf{H},\mathcal{S}}(\mathsf{pp})$
$b \leftarrow \mathcal{V}^{\mathsf{H}'}_{\mathsf{FS}}(\mathsf{pp}, x, \pi)$	$b \leftarrow \mathcal{V}^{\mathsf{H}'}_{\mathsf{FS}}(\mathsf{pp}, x, \pi)$
return $b \wedge (x, \pi) \notin \mathcal{Q}_{\mathsf{Sim}}$	$w \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{pp}, x, \pi)$
	return $b \wedge (x, \pi) \notin \mathcal{Q}_{\mathsf{Sim}} \wedge (\mathsf{pp}, x, w) \in \mathcal{R}$

Fig. 4: $\mathsf{SIM\text{-}EXT}$ security games. In both games, $\mathcal{S}$ returns a proof π upon an input x (and may reprogram the random oracle), while $\mathcal{Q}_{\mathsf{Sim}}$ records all pairs (x, π) queried by $\mathcal{P}^*$. H' denotes the modified RO after all proof simulation queries. $\mathcal{E}$ is given black-box access to $\mathcal{P}^*$; in particular, it implements H and $\mathcal{S}$ for $\mathcal{P}^*$ and can rewind $\mathcal{P}^*$ to any point in its execution (with same initial randomness).

Remark 2.12. It can be shown that $\mathcal{S}_{\mathsf{FS}}$ is a NIZK simulator for Π_{FS} if $\mathcal{S}$ is an HVZK simulator and the fact that the first message a_1 has sufficient min-entropy [24,30]. Looking ahead, given *any* simulator $\mathcal{S}$ for Π_{FS}, to show that it is a NIZK simulator, it suffices to show that $\mathcal{S}$ produces indistinguishable transcripts $\mathsf{tr} = (a_1, c_1, \ldots, a_{r+1})$ from honestly generated transcripts, and that the first message a_1 has sufficient min-entropy.

3 Simulation Extractability

We define the central notion of our work, simulation extractability ($\mathsf{SIM\text{-}EXT}$), which requires that extractability holds even when the malicious prover is given access to simulated proofs. $\mathsf{SIM\text{-}EXT}$ implies adaptive (knowledge) soundness and non-malleability for the proof system [30,43,51], and allows building secure *signatures of knowledge* via standard transforms [19,39].

Definition 3.1 (Simulation Extractability). *Let $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be a public-coin zero-knowledge interactive argument for relation $\mathcal{R}$ with associated NIZK $\Pi_{\mathsf{FS}} = (\mathsf{Setup}, \mathcal{P}_{\mathsf{FS}}, \mathcal{V}_{\mathsf{FS}})$. We say Π_{FS} satisfies* simulation extractability ($\mathsf{SIM\text{-}EXT}$) *with respect to a simulator $\mathcal{S}$ if there exists an efficient simulator-extractor $\mathcal{E}$ such that for every PPT adversary $\mathcal{P}^*$, the following probability is negligible in λ:*

$$\mathbf{Adv}^{\mathsf{SIM\text{-}EXT}}_{\Pi_{\mathsf{FS}},\mathcal{R}}(\mathcal{S}, \mathcal{E}, \mathcal{P}^*, \lambda) := \left| \Pr[\mathsf{SIM\text{-}EXT}^{\mathcal{S},\mathcal{P}^*}_{0,\Pi_{\mathsf{FS}}}(\lambda)] - \Pr[\mathsf{SIM\text{-}EXT}^{\mathcal{E},\mathcal{S},\mathcal{P}^*}_{1,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)] \right|.$$

Games $\mathsf{SIM\text{-}EXT}_0$ *and* $\mathsf{SIM\text{-}EXT}_1$ *are defined in Figure 4.*

We will state an adaptation of the results in [29], which establishes a general theorem about simulation extractability. In particular, the authors of [29] define the notion of a *k-zero-knowledge* simulator that only needs to reprogram the random oracle in round k. Similarly, they define a property of *k-unique response*, which roughly states that the malicious prover's responses are uniquely determined after round k. Together, these two properties (for the same k) along with knowledge soundness will be enough to show simulation extractability.

Definition 3.2 (k-Zero-Knowledge). *Let $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be a $(2r + 1)$-message public-coin interactive argument with HVZK simulator $\mathcal{S}$, and $k \in [1, r]$. Let Π_{FS} be its associated FS-transformed NIZK. We say Π_{FS} satisfies* (perfect) k-zero-knowledge (k-ZK) *if there exists a zero-knowledge simulator $\mathcal{S}_{\mathsf{FS},k}$ that only needs to program the random oracle in round k, and whose output is identically distributed to that of honestly generated proofs.*

Definition 3.3 (k-Unique Response). *Let $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be a $(2r + 1)$-message public-coin interactive argument, with Π_{FS} its associated FS-transformed* NARG *and $k \in [0, r]$. We say Π_{FS} satisfies* k-unique response (k-UR) *if for all PPT adversaries $\mathcal{A}$, the following probability (defined with respect to the game in Figure 5) is negligible in λ:*

$$\begin{array}{|l|}\hline \textbf{Game } k\text{-}\mathsf{UR}^{\mathcal{A}}_{\Pi_{\mathsf{FS}}}(\lambda) \\ \hline \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{pp}_{\mathcal{G}}) \\ (x, \pi, \pi', c) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathsf{pp}) \\ b \leftarrow \mathcal{V}_{\mathsf{FS}}^{\mathsf{H}[(\mathsf{pp}, x, \pi|_k) \mapsto c]}(\mathsf{pp}, x, \pi) = 1 \\ b' \leftarrow \mathcal{V}_{\mathsf{FS}}^{\mathsf{H}[(\mathsf{pp}, x, \pi'|_k) \mapsto c]}(\mathsf{pp}, x, \pi') = 1 \\ \textbf{return } b \wedge b' \wedge \pi \neq \pi' \wedge \pi|_k = \pi'|_k \\ \hline \end{array}$$

Fig. 5: Security game for k-unique response. Here $\mathsf{H}[(\mathsf{pp}, x, \pi|_k) \mapsto c]$ denotes the random oracle where the input $(\mathsf{pp}, x, \pi|_k)$ is reprogrammed to output c.

$$\mathbf{Adv}^{k\text{-}\mathsf{UR}}_{\Pi_{\mathsf{FS}}}(\mathcal{A}) := \Pr\left[k\text{-}\mathsf{UR}^{\mathcal{A}}_{\Pi_{\mathsf{FS}}}(\lambda)\right].$$

When $k = 0$, we say that Π_{FS} has (computationally) unique proofs*.*

We now state a key theorem that relates SIM-EXT to these properties; it is similar to the SIM-EXT theorem given in [29], with SRS update oracles removed.

Theorem 3.4. *Let Π_{FS} be a Fiat-Shamir compiled non-interactive argument for relation $\mathcal{R}$ from a $(2r+1)$-message public-coin interactive argument Π. Assume Π_{FS} satisfies KS, has a perfect k-ZK simulator $\mathcal{S}_{\mathsf{FS},k}$ for some $k \in [1, r]$, and satisfies k-UR (for the same k). Then Π_{FS} satisfies SIM-EXT.*

Concretely, let $\mathcal{E}$ be a KS extractor for Π_{FS}. There exists a SIM-EXT simulator-extractor $\mathcal{E}_{\mathsf{SE}}$ for Π_{FS} such that, for every PPT prover $\mathcal{P}^*$ against SIM-EXT of Π_{FS} that makes at most q_{H} random oracle queries and q_{Sim} simulation queries, there exists another PPT prover $\mathcal{P}^*_{\mathsf{KS}}$ against KS and PPT adversary $\mathcal{A}$ against k-UR such that

$$\mathbf{Adv}^{\mathsf{SIM\text{-}EXT}}_{\Pi_{\mathsf{FS}},\mathcal{R}}(\mathcal{S}_{\mathsf{FS},k}, \mathcal{E}_{\mathsf{SE}}, \mathcal{P}^*) \le \mathbf{Adv}^{\mathsf{KS}}_{\Pi_{\mathsf{FS}},\mathcal{R}}(\mathcal{E}, \mathcal{P}^*_{\mathsf{KS}}) + \mathbf{Adv}^{k\text{-}\mathsf{UR}}_{\Pi_{\mathsf{FS}}}(\mathcal{A}).$$

Furthermore, both $\mathcal{P}^*_{\mathsf{KS}}$ and $\mathcal{A}$ make at most q_{H} random oracle queries; their runtime is roughly equal to $\mathcal{P}^*$'s runtime plus q_{Sim} invocations of $\mathcal{S}_{\mathsf{FS},k}$. $\mathcal{E}_{\mathsf{SE}}$ invoked on $\mathcal{P}^*$ is nearly as efficient as $\mathcal{E}$ invoked on $\mathcal{P}^*_{\mathsf{KS}}$.

We first give the high-level intuition for our proof before presenting the formal details. Our proof proceeds in a few steps (following the strategy of [29]): first, we show how to build a KS adversary $\mathcal{P}^*_{\mathsf{KS}}$ from any SIM-EXT adversary $\mathcal{P}^*$ using a "wrapper" that takes care of the proof simulation oracle and the associated reprogramming of the random oracle. Then, we show how to build a SIM-EXT extractor $\mathcal{E}_{\mathsf{SE}}$ from an arbitrary KS extractor $\mathcal{E}$—this works by using the wrapper to turn the SIM-EXT adversary $\mathcal{P}^*$ into a KS adversary $\mathcal{P}^*_{\mathsf{KS}}$, then running the KS extractor on this adversary. Finally, we relate the advantages of $\mathcal{P}^*$ and $\mathcal{P}^*_{\mathsf{KS}}$. This steps turns out to be subtle because of how the wrapper emulates the proof simulation oracle.

In more detail, the wrapper must allow the simulator to "reprogram" the random oracle. (Recall that by assumption, our simulator only needs to reprogram in the kth round.) However, the wrapper has no power to program its own random oracle—it can only keep a table of which points the simulator has reprogrammed, and use this table to answer random oracle queries instead of querying its own random oracle. Thus, there is some "bad" event where $\mathcal{P}^*$'s output does not verify against the random oracle of $\mathcal{P}^*_{\mathsf{KS}}$. To finish the proof, we need to upper-bound the probability of this happening. We show that this probability is upper-bounded by the k-UR advantage of an adversary built from $\mathcal{P}^*$. Conditioned on this bad event not happening, we show that the KS advantage of $\mathcal{P}^*_{\mathsf{KS}}$ against $\mathcal{E}$ is an upper bound on the SE advantage of $\mathcal{P}^*$ against $\mathcal{E}_{\mathsf{SE}}$.

Proof. Below, we abbreviate $\mathsf{SIM\text{-}EXT}^{\mathcal{S}_{\mathsf{FS},k},\mathcal{P}^*}_{0,\Pi_{\mathsf{FS}}}$ as Hyb_0 and $\mathsf{SIM\text{-}EXT}^{\mathcal{E}_{\mathsf{SE}},\mathcal{S}_{\mathsf{FS},k},\mathcal{P}^*}_{1,\Pi_{\mathsf{FS}},\mathcal{R}}$ as Hyb_1. We start by defining $\mathcal{E}_{\mathsf{SE}}$. However, in order to define $\mathcal{E}_{\mathsf{SE}}$, we first describe the following "adversary wrapper" $\mathcal{W}$ that takes in an adversary $\mathcal{P}^*$ against SIM-EXT and returns an adversary $\mathcal{P}^*_{\mathsf{KS}}$ against KS. This wrapper $\mathcal{W}$ transforms $\mathcal{P}^*$ into $\mathcal{P}^*_{\mathsf{KS}}$ as follows:

- $\mathcal{P}^*_{\mathsf{KS}}$ gets $\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$ and runs $\mathcal{P}^*$ on input pp.
- $\mathcal{P}^*_{\mathsf{KS}}$ keeps a local table $\mathcal{T}$ of programmed random oracle queries, and a local table $\mathcal{Q}_{\mathsf{Sim}}$ of simulated proof queries.
- To answer $\mathcal{P}^*$'s proof simulation queries, $\mathcal{P}^*_{\mathsf{KS}}$ internally runs $\mathcal{S}_{\mathsf{FS},k}$, adds any programmed oracle queries to $\mathcal{T}$, and returns the resulting simulated proof.
- To answer $\mathcal{P}^*$'s random oracle queries, if the query is not in $\mathcal{T}$, $\mathcal{P}^*_{\mathsf{KS}}$ passes to its own RO H and relays back the result; if the query is in $\mathcal{T}$, $\mathcal{P}^*_{\mathsf{KS}}$ answers with the programmed result in $\mathcal{T}$.
- When $\mathcal{P}^*$ outputs (x, π), $\mathcal{P}^*_{\mathsf{KS}}$ aborts if $(\mathsf{pp}, x, \pi|_k) \in \mathcal{T}$, i.e. H' has been programmed on the k-th prefix of π, and otherwise outputs (x, π).

From the description of $\mathcal{P}^*_{\mathsf{KS}}$, it is clear that it provides an identical view of the oracles $\mathsf{H}, \mathcal{S}_{\mathsf{FS},k}$ to $\mathcal{P}^*$. In particular, the output of $\mathcal{P}^*$ when simulated by $\mathcal{P}^*_{\mathsf{KS}}$ is identically distributed to the case when $\mathcal{P}^*$ is run with actual oracles $\mathsf{H}, \mathcal{S}_{\mathsf{FS},k}$. We now define the simulator-extractor $\mathcal{E}_{\mathsf{SE}}$ based on $\mathcal{E}$ and $\mathcal{W}$ as follows:

– Given an adversary $\mathcal{P}^*$, $\mathcal{E}_{\mathsf{SE}}$ invokes the wrapper $\mathcal{W}$ to emulate an adversary $\mathcal{P}^*_{\mathsf{KS}}$ against KS from its access to $\mathcal{P}^*$.
– $\mathcal{E}_{\mathsf{SE}}$ then runs $\mathcal{E}$ on $\mathcal{P}^*_{\mathsf{KS}}$ and outputs what $\mathcal{E}$ outputs.

We observe the following about the efficiency of $\mathcal{P}^*_{\mathsf{KS}}$ and $\mathcal{E}_{\mathsf{SE}}$. Note that $\mathcal{P}^*_{\mathsf{KS}}$ makes at most the same number of RO queries as $\mathcal{P}^*$, and the runtime of $\mathcal{P}^*_{\mathsf{KS}}$ is roughly the runtime of $\mathcal{P}^*$ plus the runtime of q_{Sim} many invocations of the simulator $\mathcal{S}_{\mathsf{FS},k}$. The runtime of $\mathcal{E}_{\mathsf{SE}}$ is equal to the runtime of $\mathcal{E}$ on $\mathcal{P}^*_{\mathsf{KS}}$.

Going back to the analysis, let $\mathcal{P}^*$ be an arbitrary PPT adversary against $\mathsf{SIM\text{-}EXT}$. We define $\mathcal{P}^*_{\mathsf{KS}}$ to be the corresponding adversary against KS, constructed from $\mathcal{P}^*$ via the wrapper $\mathcal{W}$. We also define bad to be the event, in either hybrids, that the malicious prover $\mathcal{P}^*$ outputs (x, π) such that:

– H' has been programmed on the k-th prefix of π, i.e. $(\mathsf{pp}, x, \pi|_k) \in \mathcal{T}$,
– $\mathcal{V}^{\mathsf{H}'}_{\mathsf{FS}}(\mathsf{pp}, x, \pi) \to 1$ and $(x, \pi) \notin \mathcal{Q}_{\mathsf{Sim}}$.

Note that the second condition is identical to the winning condition of Hyb_0. We observe that bad is well-defined since the two hybrids are identical up until running $b \leftarrow \mathcal{V}^{\mathsf{H}'}_{\mathsf{FS}}(\mathsf{pp}, x, \pi)$, which is what bad depends on.

We claim that there exists an adversary $\mathcal{A}$ against k-UR, nearly as efficient as $\mathcal{P}^*_{\mathsf{KS}}$, such that $\Pr[\mathsf{bad}] \le \mathbf{Adv}^{k\text{-}\mathsf{UR}}_{\Pi_{\mathsf{FS}}}(\mathcal{A})$. We construct $\mathcal{A}$ as follows.

- $\mathcal{A}$ gets $\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$ and runs $\mathcal{P}^*$ on input pp.
- $\mathcal{A}$ emulates $\mathcal{P}^*_{\mathsf{KS}}$ in answering $\mathcal{P}^*$'s queries. In particular, it keeps track of the table $\mathcal{T}$ for programmed RO queries, and the table $\mathcal{Q}_{\mathsf{Sim}}$ for simulated proofs.
- When $\mathcal{P}^*$ outputs (x, π), $\mathcal{A}$ searches through the simulation queries to find $(x, \pi') \in \mathcal{Q}_{\mathsf{Sim}}$ that satisfies $\pi \neq \pi'$ and $\pi|_k = \pi'|_k$, aborting if no such query exists; $\mathcal{A}$ then looks up the programmed challenge $\mathsf{H}(\mathsf{pp}, x, \pi|_k) := c$ in $\mathcal{T}$ and returns (x, π, π', c).

It is clear that $\mathcal{A}$ is nearly as efficient as $\mathcal{P}^*_{\mathsf{KS}}$. It remains to argue the following points:

– First, we show that $\mathcal{A}$ does not abort if bad happens, i.e. we need to show that there exists $(x, \pi') \in \mathcal{Q}_{\mathsf{Sim}}$ such that $\pi \neq \pi'$ and $\pi|_k = \pi'_k$. From the first condition of bad, it follows that there must be a programmed oracle query for the k-th prefix $\pi|_k$ of π. Such RO reprogramming only happens during proof simulation using $\mathcal{S}_{\mathsf{FS},k}$. Since $\mathcal{S}_{\mathsf{FS},k}$ only programs the RO in the k-th round, there exists a simulation query $(x, \pi') \in \mathcal{Q}_{\mathsf{Sim}}$ such that $\pi'|_k = \pi|_k$. The second condition of bad then implies that $\pi \neq \pi'$, since $(x, \pi) \notin \mathcal{Q}_{\mathsf{Sim}}$.
– Next, we show that $\mathcal{A}$ wins the k-UR game, i.e. that $\mathcal{V}^{\mathsf{H}^\star}_{\mathsf{FS}}(\mathsf{pp}, x, \pi) = \mathcal{V}^{\mathsf{H}^\star}_{\mathsf{FS}}(\mathsf{pp}, x, \pi') = 1$, $\pi \neq \pi'$ and $\pi|_k = \pi'|_k$, where $\mathsf{H}^\star = \mathsf{H}[(\mathsf{pp}, x, \pi'|_k) \mapsto c]$. The last two conditions ($\pi \neq \pi'$ and $\pi|_k = \pi'|_k$) are satisfied by the construction of $\mathcal{A}$. Next, we note that relative to the modified random oracle H', we have $\mathcal{V}^{\mathsf{H}'}_{\mathsf{FS}}(\mathsf{pp}, x, \pi) \to 1$ by the second condition of bad, and $\mathcal{V}^{\mathsf{H}'}_{\mathsf{FS}}(\mathsf{pp}, x, \pi') \to 1$ by the guarantee of the simulator $\mathcal{S}_{\mathsf{FS},k}$. We finish by noting that H' and H^* give identical answers for the prefixes of π

and π' queried by the verifier $\mathcal{V}_{\mathsf{FS}}$, i.e. that $\mathcal{V}_{\mathsf{FS}}^{\mathsf{H}^*}(\mathsf{pp}, x, \pi) \to 1$ and $\mathcal{V}_{\mathsf{FS}}^{\mathsf{H}^*}(\mathsf{pp}, x, \pi') \to 1$ as well. This is because the prefix for the k-th round $(\mathsf{pp}, x, \pi|_k) = (\mathsf{pp}, x, \pi'|_k)$ is programmed to c in both H' and H^*, and neither H' nor H^* reprograms H on any other prefix.

We now show that conditioned on bad not happening, for the adversary $\mathcal{P}^*_{\mathsf{KS}}$ constructed above, we have the following equalities:

$$\Pr[\mathsf{Hyb}_0 \wedge \neg\mathsf{bad}] = \Pr[\mathsf{KS}_{0,\Pi_{\mathsf{FS}}}^{\mathcal{P}^*_{\mathsf{KS}}}(\lambda)], \qquad \Pr[\mathsf{Hyb}_1 \wedge \neg\mathsf{bad}] = \Pr[\mathsf{KS}_{1,\Pi_{\mathsf{FS}},\mathcal{R}}^{\mathcal{E},\mathcal{P}^*_{\mathsf{KS}}}(\lambda)].$$

For the first equality, note that $\mathsf{Hyb}_0 \wedge \neg\mathsf{bad}$ is equivalent to the following conditions:

- H' is *not* programmed on the k-th prefix of π, i.e. $(\mathsf{pp}, x, \pi|_k) \notin \mathcal{T}$,
- $\mathcal{V}_{\mathsf{FS}}^{\mathsf{H}'}(\mathsf{pp}, x, \pi) \to 1$ and $(x, \pi) \notin \mathcal{Q}_{\mathsf{Sim}}$.

These conditions are also equivalent to $\mathsf{KS}_{0,\Pi_{\mathsf{FS}}}^{\mathcal{P}^*_{\mathsf{KS}}}(\lambda)$, since it means that:

- $\mathcal{P}^*_{\mathsf{KS}}$ does not abort, which implies $(\mathsf{pp}, x, \pi|_k) \notin \mathcal{T}$,
- $\mathcal{V}_{\mathsf{FS}}^{\mathsf{H}}(\mathsf{pp}, x, \pi) \to 1$, which by the previous fact, also implies $\mathcal{V}_{\mathsf{FS}}^{\mathsf{H}'}(\mathsf{pp}, x, \pi) \to 1$ (since H and H' give the same answers to queries made by $\mathcal{V}_{\mathsf{FS}}$). Furthermore, since $(\mathsf{pp}, x, \pi|_k)$ is not programmed, π cannot be a simulated proof, i.e. $(x, \pi) \notin \mathcal{Q}_{\mathsf{Sim}}$.

A similar argument also establishes the second equality. Namely, $\mathsf{Hyb}_1 \wedge \neg\mathsf{bad}$ has the same conditions as $\mathsf{Hyb}_0 \wedge \neg\mathsf{bad}$, plus the condition that the witness $w \leftarrow \mathcal{E}_{\mathsf{SE}}^{\mathcal{P}^*}(\mathsf{pp}, x, \pi)$ satisfies $(\mathsf{pp}, x, w) \in \mathcal{R}$. Similarly, $\mathsf{KS}_{1,\Pi_{\mathsf{FS}},\mathcal{R}}^{\mathcal{E},\mathcal{P}^*_{\mathsf{KS}}}(\lambda)$ is the same as $\mathsf{KS}_{0,\Pi_{\mathsf{FS}}}^{\mathcal{P}^*_{\mathsf{KS}}}(\lambda)$ plus the condition that $w \leftarrow \mathcal{E}^{\mathcal{P}^*_{\mathsf{KS}}}(\mathsf{pp}, x, \pi)$ satisfies $(\mathsf{pp}, x, w) \in \mathcal{R}$. By the construction of $\mathcal{E}_{\mathsf{SE}}$, we see that $\mathcal{E}_{\mathsf{SE}}^{\mathcal{P}^*}$ is identical to $\mathcal{E}^{\mathcal{P}^*_{\mathsf{KS}}}$, which means the witness w produced is the same.

Putting everything together, we get the desired bound:

$$\begin{aligned}\mathbf{Adv}_{\Pi_{\mathsf{FS}},\mathcal{R}}^{\mathsf{SIM\text{-}EXT}}(\mathcal{S}_{\mathsf{FS}}, \mathcal{E}_{\mathsf{SE}}, \mathcal{P}^*) &\leq |\Pr[\mathsf{Hyb}_0] - \Pr[\mathsf{Hyb}_1]| \\ &\leq |\Pr[\mathsf{Hyb}_0 \wedge \neg\mathsf{bad}] - \Pr[\mathsf{Hyb}_1 \wedge \neg\mathsf{bad}]| + \Pr[\mathsf{bad}] \\ &\leq \mathbf{Adv}_{\Pi_{\mathsf{FS}},\mathcal{R}}^{\mathsf{KS}}(\mathcal{E}, \mathcal{P}^*_{\mathsf{KS}}) + \mathbf{Adv}_{\Pi_{\mathsf{FS}}}^{k\text{-}\mathsf{UR}}(\mathcal{A}).\end{aligned}$$

□

4 Tree of Transcripts and Special Soundness

In this section, we show how to establish knowledge soundness (KS) of a FS-transformed protocol Π_{FS} based on the *computational special soundness* of the interactive protocol Π. The key is to construct an efficient *tree builder* $\mathcal{TB}$ that, given oracle access to a malicious prover $\mathcal{P}^*$ for Π_{FS}, outputs a suitable *tree of accepting transcripts*, upon which a valid witness can be extracted.

Definition 4.1 (Tree of Transcripts). *Let Π be a $(2r+1)$-message public-coin interactive argument for a relation $\mathcal{R}$, with challenge spaces $\mathsf{Ch}_1, \ldots, \mathsf{Ch}_r$. Given $\mathbf{n} = (n_1, \ldots, n_r) \in \mathbb{N}^r$ and $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_r)$ with $\phi_i : \mathsf{Ch}_i^{n_i} \to \{0,1\}$ for $i \in [r]$, we say that $\mathscr{T}$ is a $(\boldsymbol{\phi}, \mathbf{n})$-*tree of accepting transcripts *for* pp *if:*

1. *$\mathscr{T}$ is a tree of depth $r+1$,*
2. *For each $i \in [r+1]$, each vertex at depth i is labeled with a prover's i-th message a_i, and if $i \leq r$, has exactly n_i outgoing edges to its children, with each edge labeled with a verifier's i-th challenge $c_{i,1}, \ldots, c_{i,n_i}$ satisfying $\phi_i(c_{i,1}, \ldots, c_{i,n_i}) = 1$. Additionally, the root's label is prepended with x (so the label becomes (x, a_1)),*
3. *The labels on any root-to-leaf path form a valid input-transcript pair (x, tr).*

We additionally define $\mathscr{T}$ to be accepting with respect to a input-transcript pair (x, tr) if (x, tr) corresponds to the left-most path of $\mathscr{T}$. We define a predicate $\mathsf{IsAccepting}((\boldsymbol{\phi}, \mathbf{n}), \mathsf{pp}, x, (\pi,)\mathscr{T})$ to check whether $\mathscr{T}$ is a $(\boldsymbol{\phi}, \mathbf{n})$-tree of accepting transcripts for pp and x, and optionally π.

Game $\mathsf{TreeBuild}^{\mathcal{TB},\mathcal{P}^*}_{\Pi_{\mathsf{FS}},(\boldsymbol{\phi},\mathbf{n})}(\lambda)$	**Game** $\mathsf{SS}^{\mathcal{TE},\mathcal{A}}_{\Pi,\mathcal{R},(\boldsymbol{\phi},\mathbf{n})}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$	$\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$
$(x, \pi) \leftarrow (\mathcal{P}^*)^{\mathsf{H}}(\mathsf{pp})$	$(x, \mathscr{T}) \leftarrow \mathcal{A}(\mathsf{pp})$
$\mathscr{T} \leftarrow \mathcal{TB}^{\mathcal{P}^*}(\mathsf{pp}, x, \pi)$	$w \leftarrow \mathcal{TE}(\mathsf{pp}, x, \mathscr{T})$
return $(\mathcal{V}^{\mathsf{H}}(\mathsf{pp}, x, \pi) = 1) \wedge$ $\mathsf{IsAccepting}((\boldsymbol{\phi}, \mathbf{n}), \mathsf{pp}, x, \pi, \mathscr{T})$	**return** $(\mathsf{pp}, x, w) \notin \mathcal{R} \wedge$ $\mathsf{IsAccepting}((\boldsymbol{\phi}, \mathbf{n}), \mathsf{pp}, x, \mathscr{T})$

Fig. 6: Games for tree-building and special soundness. Here the tree-builder $\mathcal{TB}$ is given black-box access to $\mathcal{P}^*$. In particular, $\mathcal{TB}$ implements H for $\mathcal{P}^*$ and can rewind $\mathcal{P}^*$ to any point in its execution.

The usual definition of a tree of accepting transcripts [2,14] has ϕ_i be the predicate that the i-th challenges $c_{i,1}, \ldots, c_{i,n_i}$, coming from a vertex at depth i, are distinct (we call this the *distinctness predicate*). In that case, we will also abbreviate $\mathscr{T}$ as a $\mathbf{n}$-tree of accepting transcripts. However, we will need to consider more general *partition predicates* in our proofs of knowledge soundness for Spartan and Bulletproofs.

Definition 4.2 (Partition Predicate). *Let* $\mathsf{Ch} = \mathsf{Ch}^{(1)} \sqcup \mathsf{Ch}^{(2)} \cdots \sqcup \mathsf{Ch}^{(C)}$ *be a partition* $\mathscr{P}$ *of a set* Ch *into* C *blocks. We assume the partition is* efficient*, i.e. given an index* $i \in [C]$*, we can enumerate the set* $\mathsf{Ch}^{(i)}$ *in polynomial time. For* $n \in \mathbb{N}$*, we define the corresponding* partition predicate $\phi_{\mathscr{P},n} : \mathsf{Ch}^n \to \{0,1\}$ *to be* $\phi_{\mathscr{P},n}(c_1, \ldots, c_n) = 1$ *if and only if* $c_1, \ldots, c_n$ *belong in distinct blocks of* Ch*.*

Remark 4.3. Looking ahead, we will consider the following partition predicates:

- When $\mathsf{Ch} = \mathbb{F}^*$ is partitioned into $\{x, -x\}$ for all x. We abbreviate this predicate into the number n of challenges as $n_{\pm}$.
- When $\mathsf{Ch} = \mathbb{F}^2$ is partitioned into $\{c \cdot x \mid c \in \mathbb{F}^*\}$ for all $x \in \{(0,0),(0,1)\} \cup \{(1,a) \mid a \in \mathbb{F}\}$ (this implies linear independence between two vectors). We abbreviate this predicate into the number n of challenges as n_{li}.

We now state a theorem asserting the existence of an efficient tree-builder that can generate $(\boldsymbol{\phi}, \mathbf{n})$-trees of accepting transcripts, where $\boldsymbol{\phi}$ consists of partition predicates as defined above. We give the proof in Section 4.1; our proof relies on the tree-builder constructed in the work of Attema et al. [2]. We give a comparison of our tree-builder with that of Wikström in Appendix A.

Theorem 4.4 (Efficient Tree Builder). *Let* Π *be a* $(2r+1)$*-message public-coin interactive argument with challenge spaces* $\mathsf{Ch}_1, \ldots, \mathsf{Ch}_r$*. Consider any efficiently decidable partition* $\mathsf{Ch}_i = \sqcup_{j=1}^{C_i} \mathsf{Ch}_{i,j}$ *with minimum partition size* $C = \min_i C_i$*, and let* $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_r)$ *be the corresponding partition predicate. Consider any* $\mathbf{n} = (n_1, \ldots, n_r) \in \mathbb{N}^r$ *with* $N = \prod_{i=1}^r n_i$*.*

There exists a probabilistic algorithm $\mathcal{TB}$ *for* Π_{FS} *with the following guarantees: given oracle access to a malicious prover* $\mathcal{P}^*$ *for* Π_{FS} *with success probability* $\epsilon(\mathcal{P}^*) := \Pr[\mathsf{KS}^{\mathcal{P}^*}_{0,\Pi_{\mathsf{FS}}}]$*,* $\mathcal{TB}$ *wins the tree-building game* $\mathsf{TreeBuild}^{\mathcal{TB},\mathcal{P}^*}_{\Pi_{\mathsf{FS}},(\boldsymbol{\phi},\mathbf{n})}$ *(shown in Figure 6) with probability at least*

$$\Pr\left[\mathsf{TreeBuild}^{\mathcal{TB},\mathcal{P}^*}_{\Pi_{\mathsf{FS}},(\boldsymbol{\phi},\mathbf{n})}\right] \geq \epsilon(\mathcal{P}^*) - \frac{Q(Q-1)/2 + (Q+1)\left(\sum_{i=1}^r n_i - r\right)}{C}.$$

Furthermore, $\mathcal{TB}$ *makes in expectation at most* $(Q+1)(N-1)+1$ *rewinding calls to* $\mathcal{P}^*$*, where* Q *is an upper bound on the number of RO queries of* $\mathcal{P}^*$*.*

Remark 4.5. We note the quadratic dependence on the number of queries Q in our bound. This seems to be an inherent limitation of our proof technique, which stems from a birthday bound (see Section 4.1), and we leave achieving a tighter bound to future work.

We now define computational special soundness, which stipulates the existence of a tree-extraction procedure $\mathcal{TE}$ that, given an appropriate tree of accepting transcripts produced by an efficient adversary, outputs a witness with high probability.

Definition 4.6 (Special Soundness). *Let Π be a $(2r+1)$-message public-coin interactive argument for a relation $\mathcal{R}$ with challenge spaces $\mathsf{Ch}_1, \ldots, \mathsf{Ch}_r$. For any $\mathbf{n} = (n_1, \ldots, n_r) \in \mathbb{N}^r$ and $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_r)$ with $\phi_i : \mathsf{Ch}_i^{n_i} \to \{0,1\}$, we say Π is* $(\boldsymbol{\phi}, \mathbf{n})$-computational special sound *if there exists a PPT tree-extraction algorithm $\mathcal{TE}$ such that for all EPT adversary $\mathcal{A}$, the following probability is negligible in λ:*

$$\mathbf{Adv}^{\mathsf{SS}}_{\Pi,\mathcal{R},(\boldsymbol{\phi},\mathbf{n})}(\mathcal{TE}, \mathcal{A}) := \Pr\left[\mathsf{SS}^{\mathcal{TE},\mathcal{A}}_{\Pi,\mathcal{R},(\boldsymbol{\phi},\mathbf{n})}(\lambda)\right].$$

The special soundness game is shown in Figure 6. We say Π is computational special sound (SS) *if it is $(\boldsymbol{\phi}, \mathbf{n})$-computational special sound for some $\boldsymbol{\phi}$ and $\mathbf{n}$.*

Using Theorem 4.4 and Definition 4.6, we get the following consequence that computational special soundness for an interactive protocol implies knowledge soundness for its non-interactive version.

Lemma 4.7. *Let Π be a $(2r+1)$-message public-coin interactive argument that is $(\boldsymbol{\phi}, \mathbf{n})$-computational special sound with tree extractor $\mathcal{TE}$, where $\mathbf{n} = (n_1, \ldots, n_r) \in \mathbb{N}^r$ and $\boldsymbol{\phi}$ is a partition predicate with minimum partition size C. Then Π_{FS} satisfies knowledge soundness. Concretely, there exists an EPT extractor $\mathcal{E}$ such that for every PPT adversary $\mathcal{P}^*$ against* KS *making at most Q random oracle calls, there exists an EPT adversary $\mathcal{A}$ against* SS *such that*

$$\mathbf{Adv}^{\mathsf{KS}}_{\Pi_{\mathsf{FS}},\mathcal{R}}(\mathcal{E}, \mathcal{P}^*) \leq \frac{Q(Q-1)/2 + (Q+1)\left(\sum_{i=1}^{r} n_i - r\right)}{C} + \mathbf{Adv}^{\mathsf{SS}}_{\Pi,(\boldsymbol{\phi},\mathbf{n})}(\mathcal{TE}, \mathcal{A}).$$

Both $\mathcal{E}$ and $\mathcal{A}$ runs in expected time that is at most $O(Q \cdot N)$ the runtime of $\mathcal{P}^*$.

Proof. Our proof goes through a sequence of hybrids. Hyb_0 is the game $\mathsf{KS}^{\mathcal{P}^*}_{0,\Pi_{\mathsf{FS}}}$. Hyb_1 is the same as Hyb_0, except we also run $\mathcal{TB}^{\mathcal{P}^*}(\mathsf{pp}, x, \pi) \to \mathscr{T}$ and output 0 if $\mathscr{T}$ is not a $(\boldsymbol{\phi}, \mathbf{n})$-tree of accepting transcripts with respect to (pp, x, π). Note that Hyb_1 is the same as the game $\mathsf{TreeBuild}^{\mathcal{TB},\mathcal{P}^*}_{\Pi_{\mathsf{FS}},(\boldsymbol{\phi},\mathbf{n})}$. Using Theorem 4.4, we get

$$|\Pr[\mathsf{Hyb}_0] - \Pr[\mathsf{Hyb}_1]| \leq \frac{Q(Q-1)/2 + (Q+1)\left(\sum_{i=1}^{r} n_i - r\right)}{C}.$$

We define Hyb_2 to be the same as Hyb_1, except we also run $\mathcal{TE}(\mathsf{pp}, x, \mathscr{T}) \to w$ and output 0 if $(\mathsf{pp}, x, w) \notin \mathcal{R}$. We define the extractor $\mathcal{E}$ to be as follows: run $\mathcal{TB}^{\mathcal{P}^*}(\mathsf{pp}, x, \pi) \to \mathscr{T}$ to obtain a tree of accepting transcripts, then run $\mathcal{TE}(\mathsf{pp}, x, \mathscr{T}) \to w$ to obtain a witness. By definition of $\mathcal{E}$, we can see that Hyb_2 is the same as the game $\mathsf{KS}^{\mathcal{E},\mathcal{P}^*}_{1,\Pi_{\mathsf{FS}},\mathcal{R}}$.

We now claim that there exists an adversary $\mathcal{A}$ against SS such that

$$|\Pr[\mathsf{Hyb}_1] - \Pr[\mathsf{Hyb}_2]| \leq \mathbf{Adv}^{\mathsf{SS}}_{\Pi,(\boldsymbol{\phi},\mathbf{n})}(\mathcal{TE}, \mathcal{A}).$$

We define $\mathcal{A}$ to be as follows: given oracle access to $\mathcal{P}^*$, $\mathcal{A}$ runs $(\mathcal{P}^*)^{\mathsf{H}}(\mathsf{pp}) \to (x, \pi)$ by simulating H for $\mathcal{P}^*$, then runs $\mathcal{TB}^{\mathcal{P}^*}(\mathsf{pp}, x, \pi) \to \mathscr{T}$, and outputs $(x, \mathscr{T})$. It is then straightforward to argue that Hyb_2 returns 0 while Hyb_1 returns 1 precisely when $\mathcal{A}$ wins in SS. $\square$

4.1 Proof of Theorem 4.4

We first introduce the notions of an *abstract adversary* and an *abstract tree of transcripts*, which can be defined independently of any interactive argument Π. Such notions were also considered in [2] in terms of *abstract sampling games*, and we will state the tree-builder $\mathcal{TB}_{\mathsf{AFK}}$ of [2] in these terms.

Definition 4.8 (Abstract Adversary). *Let $S_1, \ldots, S_r$ be finite sets. Denote by $\mathsf{H} = (\mathsf{H}_1, \ldots, \mathsf{H}_r)$ be a collection of random oracles $\mathsf{H}_i : \{0,1\}^* \to S_i$. A* r-round, Q-query random oracle adversary $\mathcal{A}$ *against $S_1, \ldots, S_r$ is a* deterministic *adversary having oracle access to $\mathsf{H}_1, \ldots, \mathsf{H}_r$, making at most Q total accesses* *to these random oracles, and returning* $((a_1, \ldots, a_{r+1}), v)$ *where* a_i*'s are strings and* $v \in \{0, 1\}$*. The* success probability *of* $\mathcal{A}$ *is defined to be*

$$\epsilon(\mathcal{A}) := \Pr[v = 1 \mid \mathcal{A}^{\mathsf{H}} \to ((a_1, \ldots, a_{r+1}), v)],$$

where the probability is defined over the randomness of choosing H.

Definition 4.9 (Abstract Tree of Transcripts). *Let* $S_1, \ldots, S_r$ *be any finite sets,* $\mathcal{A}$ *be any abstract adversary against* $S_1, \ldots, S_r$*, and* $\mathbf{n} = (n_1, \ldots, n_r) \in \mathbb{N}^r$*. A* $\mathbf{n}$-abstract tree of transcripts $\mathscr{T}$ *for* $\mathcal{A}$ *and* $\mathsf{H} = (\mathsf{H}_1, \ldots, \mathsf{H}_r)$ *is a labeled* $\mathbf{n}$*-tree where:*

- *Each vertex at depth* $i \in [r+1]$ *is labeled with a message* a_i*,*
- *Each of the* n_i *edges coming from a vertex at depth* $i \in [r]$ *is labeled with a different element* $s \in S_i$*,*
- *For any root-to-leaf path, if the edges are labeled* $(s_1, \ldots, s_r)$ *and the vertices are labeled* $(a_1, \ldots, a_{r+1})$*, then* $\mathcal{A}^{\mathsf{H}'} \to ((a_1, \ldots, a_{r+1}), 1)$ *where* $\mathsf{H}' = (\mathsf{H}_1[a_1 \mapsto s_1], \ldots, \mathsf{H}_r[(a_1, \ldots, a_r) \mapsto s_r])$*.*

Remark 4.10. Let Π be a $(2r+1)$-message public-coin interactive argument with challenge sets $\mathsf{Ch}_1, \ldots, \mathsf{Ch}_r$. From any *deterministic* adversary $\mathcal{P}^*$ against KS of Π_{FS}, we can build an abstract adversary $\mathcal{A}$ against the set $\mathsf{Ch}_1, \ldots, \mathsf{Ch}_r$ by running $(x, \pi = (a_1, \ldots, a_{r+1})) \leftarrow (\mathcal{P}^*)^{\mathsf{H}}(\mathsf{pp})$ (with pp hard-coded) and also $v \leftarrow \mathcal{V}_{\mathsf{FS}}^{\mathsf{H}}(\mathsf{pp}, x, \pi)$; $\mathcal{A}$ then outputs $(((x, a_1), a_2, \ldots, a_{r+1}), v)$. A $\mathbf{n}$-tree of accepting transcripts for (pp, x, π) can be seen as a $\mathbf{n}$-abstract tree of transcripts for $\mathcal{A}$.

We now state the guarantees of the tree-builder $\mathcal{TB}_{\mathsf{AFK}}$ in [2].

Theorem 4.11 ([2], adapted). *Consider any sets* $S_1, \ldots, S_r$ *and any* $\mathbf{n} = (n_1, \ldots, n_r) \in \mathbb{N}^r$ *with* $N = \prod_{i=1}^r n_i$*. There exists a probabilistic algorithm* $\mathcal{TB}_{\mathsf{AFK}}$ *with the following guarantees: given oracle access to any* Q*-query abstract adversary* $\mathcal{A}$ *against* $S_1, \ldots, S_r$ *with success probability* $\epsilon(\mathcal{A})$*,* $\mathcal{TB}_{\mathsf{AFK}}$ *outputs a* $\mathbf{n}$*-abstract tree of transcript* $\mathscr{T}$ *with probability at least*

$$\frac{\epsilon(\mathcal{A}) - (Q+1)\kappa}{1 - \kappa}, \text{ where } \kappa = 1 - \prod_{i=1}^{r} \left(1 - \frac{n_i - 1}{|S_i|}\right).$$

Furthermore, $\mathcal{TB}_{\mathsf{AFK}}$ *makes in expectation at most* $(Q+1)(N-1)+1$ *oracle calls to* $\mathcal{A}$*.*

Remark 4.12. We simplify the bounds of Theorem 4.11 in two ways: first, we have

$$\kappa \le \kappa' := \sum_{i=1}^{r} \frac{n_i - 1}{C_i} \le \frac{\sum_{i=1}^r n_i - r}{C}, \quad \text{where } C = \min_{i=1,\ldots,r} C_i,$$

and secondly, we simplify the success probability of $\mathcal{TB}$ to be at least

$$\frac{\epsilon(\mathcal{A}) - (Q+1)\kappa}{1 - \kappa} \ge \epsilon(\mathcal{A}) - (Q+1)\kappa' = \epsilon(\mathcal{A}) - \frac{(Q+1)(\sum_{i=1}^r n_i - r)}{C}.$$

This leads to a cleaner expression while not sacrificing any tightness, as both κ and κ' will be negligible for our abstract adversaries.

We now give a proof of Theorem 4.4, using the tree-builder $\mathcal{TB}_{\mathsf{AFK}}$ in Theorem 4.11 as a black box.

Proof (Proof of Theorem 4.4). Without loss of generality, we assume that $\mathcal{P}^*$ is deterministic; this is because if we can prove the theorem for every choice of $\mathcal{P}^*$'s randomness, then by averaging we also prove the theorem for arbitrary $\mathcal{P}^*$. Thus, the only source of randomness in the game $\mathsf{KS}_{0,\Pi_{\mathsf{FS}}}^{\mathcal{P}^*}$, and hence of the success probability $\epsilon(\mathcal{P}^*) := \Pr[\mathsf{KS}_{0,\Pi_{\mathsf{FS}}}^{\mathcal{P}^*}]$, is the choice of the random oracle H.

For $i \in [r]$, define $\mathsf{H}_i^\star : \{0,1\}^* \to [C_i]$, where we recall that $\mathsf{Ch}_i = \sqcup_{j=1}^{C_i} \mathsf{Ch}_{i,j}$. We will construct an abstract adversary $\mathcal{A}$ against the sets $[C_1], \ldots, [C_r]$, having access to random oracles $\mathsf{H}^\star = (\mathsf{H}_1^\star, \ldots, \mathsf{H}_r^\star)$ and to the malicious prover $\mathcal{P}^*$. It does the following:

– Get $\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$ and run $\mathcal{P}^*$ on pp.
– Initialize an empty table $\mathcal{T}$ of triples $((i, a, (c, j))$, where $c \in \mathsf{Ch}_{i,j}$, denoting the result $\mathsf{H}_i(a) = c$.
– When $\mathcal{P}^*$ makes an oracle query to H_i on input a, search through $\mathcal{T}$ for an entry $(i, a, (c, \cdot))$, and return c. If no such entry exists, query $\mathsf{H}_i^\star(a) \to j$, then sample $c \xleftarrow{\$} \mathsf{Ch}_{i,j}$ uniformly at random, add $(i, a, (c, j))$ to $\mathcal{T}$, and return c as the answer to $\mathcal{P}^*$.
– When $\mathcal{P}^*$ outputs $(x, \pi = (a_1, \dots, a_{r+1}))$, run $v \leftarrow \mathcal{V}^{\mathsf{H}}(\mathsf{pp}, x, \pi)$ (where H is determined by $\mathcal{T}$), and output $(((x, a_1), a_2, \dots, a_{r+1}), v)$.

We now define our tree-builder $\mathcal{TB}$. Given oracle access to $\mathcal{P}^*$, it emulates the abstract adversary $\mathcal{A}$, then run the tree-builder $\mathcal{TB}_{\mathsf{AFK}}$ on $\mathcal{A}$. If $\mathcal{TB}_{\mathsf{AFK}}$ returns a $\mathbf{n}$-abstract tree of transcripts $\mathscr{T}_{abs}$, then $\mathcal{TB}$ returns a $(\boldsymbol{\phi}, \mathbf{n})$-tree of accepting transcripts $\mathscr{T}$ for Π_{FS} as follows:

– For each vertex at depth $i \in [r+1]$ of $\mathscr{T}_{abs}$ with label a_i, the same vertex for $\mathscr{T}$ has label a_i as well,
– For each edge labeled j going from a vertex labeled a at depth $i \in [r]$, the same edge for $\mathscr{T}$ has label c, where c is the unique challenge such that $(i, a, (c, j)) \in \mathcal{T}$.

We argue that $\mathscr{T}$ is indeed a $(\boldsymbol{\phi}, \mathbf{n})$-tree of accepting transcripts. It is clear that $\mathscr{T}$ is of the right arity. For any vertex v at depth $i \in [r]$, we know that the edges coming from v are labeled with different $(j_{i,1}, \dots, j_{i,n_i})$ in $\mathscr{T}_{abs}$. This implies that for $\mathscr{T}$, the edges coming from the corresponding vertex v has challenges $(c_{i,1}, \dots, c_{i,n_i})$ satisfying $c_{i,k} \in \mathsf{Ch}_{i,j_{i,k}}$ for all $k \in [n_i]$. Hence $\mathscr{T}$ satisfies the partition predicate $\boldsymbol{\phi}$.

It remains to analyze the success probability and expected running time of our tree-builder. The expected running time is easier to analyze. First, the abstract adversary $\mathcal{A}$ is nearly as efficient as $\mathcal{P}^*$, as it runs $\mathcal{P}^*$ once, and does some other tasks in comparable time (managing table $\mathcal{T}$, running Setup and $\mathcal{V}_{\mathsf{FS}}$). Our tree-builder $\mathcal{TB}$ then invokes $\mathcal{TB}_{\mathsf{AFK}}$ once on $\mathcal{A}$, hence inheriting the expected running time of $\mathcal{TB}_{\mathsf{AFK}}$. Concretely, the expected running time of $\mathcal{TB}$ is at most $(Q-1) \cdot (N+1) + 1$ times the running time of $\mathcal{P}^*$.

Next, we analyze the success probability of the abstract adversary $\mathcal{A}$. If $\mathcal{A}$ can perfectly simulate the random oracles $\mathsf{H} = (\mathsf{H}_1, \dots, \mathsf{H}_r)$ for $\mathcal{P}^*$, then it is clear that its winning probability is the same as $\mathcal{P}^*$, i.e. we would have $\epsilon(\mathcal{A}) = \epsilon(\mathcal{P}^*)$.

Our central observation is that perfect simulation occurs when *no two queries to* $\mathsf{H}_i^\star$*, for any* $i \in [r]$*, result in the same index* j *where the block* $\mathsf{Ch}_{i,j}$ *has size* > 1. We call this event bad. When bad happens, simulation would not be perfect as we would return the same challenge $c \in \mathsf{Ch}_{i,j}$ to two different queries made by $\mathcal{P}^*$, whereas a truly random oracle would return *independent random* challenges $c, c' \in \mathsf{Ch}_{i,j}$.

We can analyze the probability that bad happens using a standard analysis of the birthday attack. Assume $\mathcal{P}^*$ makes Q_i queries to H_i for $i \in [r]$, so that $Q = Q_1 + \cdots + Q_r$. The probability that bad happens for a given round i can be bounded by at most $\frac{1}{C_i}\binom{Q_i}{2}$. By an union bound, we have

$$\epsilon(\mathcal{P}^*) - \epsilon(\mathcal{A}) \le \Pr[\mathsf{bad}] \le \sum_{i=1}^{r} \frac{1}{C_i}\binom{Q_i}{2} \le \frac{1}{C}\binom{Q}{2},$$

where $C = \min_{i \in [r]} C_i$. We now apply the guarantee on the success probability of $\mathcal{TB}_{\mathsf{AFK}}$ (see Remark 4.12) to conclude that

$$\begin{aligned}\Pr[\mathcal{TB} \text{ succeeds}] &\ge \epsilon(\mathcal{A}) - \frac{(Q+1)(\sum_{i=1}^{r} n_i - r)}{C} \\ &\ge \epsilon(\mathcal{P}^*) - \frac{Q(Q-1)/2 + (Q+1)(\sum_{i=1}^{r} n_i - r)}{C}.\end{aligned}$$

□

5 Simulation Extractability of Spartan

In this section, we use our general theorems to prove SIM-EXT of Spartan [55], a transparent zkSNARKs with security based on the discrete log assumption. [55] presents two version of Spartan, one with a linear verifier (called Spartan-NIZK) and one with a sublinear verifier (called Spartan-SNARK) achieved via encoding the R1CS matrices with a *sparse multilinear polynomial commitment*.

5.1 Spartan Preliminaries

Multilinear Polynomials. Let $\mathbb{F}$ be a finite field and $\mu \in \mathbb{N}$. A multivariate polynomial $p(X_1, \dots, X_\mu) \in \mathbb{F}[X_1, \dots, X_\mu]$ is *multilinear* if its individual degree in each variable X_i is at most 1. Equivalently, it is a linear combination of 2^μ monomials of the form $\{X_1^{\delta_1} \dots X_\mu^{\delta_\mu}\}_{\delta_1,\dots,\delta_\mu \in \{0,1\}}$.

Evaluations on Boolean Hypercube. We can also represent a multilinear polynomial $p(X_1, \dots, X_\mu)$ by its evaluations on the Boolean hypercube $\{p(x)\}_{x \in \{0,1\}^\mu}$. These evaluations determine the polynomial uniquely via the following interpolation formula:

$$p(X_1, \dots, X_\mu) = \sum_{y \in \{0,1\}^\mu} p(y) \cdot \widetilde{\mathrm{eq}}(X, y),$$

where

$$\widetilde{\mathrm{eq}}(X, y) = \prod_{i=1}^{\mu} \widetilde{\mathrm{eq}}(X_i, y_i), \qquad \widetilde{\mathrm{eq}}(X_i, y_i) = X_i \cdot y_i + (1 - X_i) \cdot (1 - y_i).$$

Note that for $x, y \in \{0,1\}^\mu$, we have $\widetilde{\mathrm{eq}}(x, y) = 0$ if $x \neq y$, and $\widetilde{\mathrm{eq}}(x, x) = 1$.

Multilinear Extension. Given $g : \{0,1\}^\mu \to \mathbb{F}$, we define its *multilinear extension* $\widetilde{g}(X_1, \dots, X_\mu) \in \mathbb{F}[X_1, \dots, X_\mu]$ to be the unique multilinear polynomial with evaluation $\widetilde{g}(x) = g(x)$ for all $x \in \{0,1\}^\mu$.

Dense Representation of Sparse Matrices. Let $M \in \mathbb{F}^{m \times m}$ be a matrix with $n = O(m)$ non-zero entries. We can pick some canonical ordering of these non-zero entries, and represent M as three vectors $(\mathsf{row}, \mathsf{col}, \mathsf{val}) \in (\mathbb{F}^n)^3$ such that $M(\mathsf{row}_i, \mathsf{col}_i) = \mathsf{val}_i$ is the i-th non-zero entry of M. We can also compute the multilinear extension $\widetilde{M}$ by the following formula:

$$\widetilde{M}(X, Y) = \sum_{j=1}^{n} \mathsf{val}_j \cdot \widetilde{\mathrm{eq}}(\mathsf{row}_j, X) \cdot \widetilde{\mathrm{eq}}(\mathsf{col}_j, Y). \tag{1}$$

Multilinear PCS. A *multilinear polynomial commitment scheme* PC is a tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{Commit})$ along with an interactive protocol Open, where:

- $\mathsf{Setup}(\mu, \mathsf{pp}_\mathcal{G}) \to \mathsf{pp}$: on input number of variables μ and global parameters $\mathsf{pp}_\mathcal{G}$, outputs public parameters pp.
- $\mathsf{Commit}(\mathsf{pp}, p; \omega) \to C$: on input public parameters pp and a multilinear polynomial $p \in \mathbb{F}[X_1, \dots, X_\mu]$, samples randomness ω and outputs a commitment C.
- $\mathsf{Open}\langle \mathcal{P}, \mathcal{V} \rangle \to \{0,1\}$: a public-coin interactive argument for the relation

$$\mathcal{R}_{\mathsf{PC.Open}} = \left\{ \begin{matrix} (\mathsf{pp}, (C, x, v), (p, \omega)) : \\ C = \mathsf{PC.Commit}(\mathsf{pp}, p; \omega) \ \wedge \ p \text{ is multilinear } \wedge \ p(x) = v \end{matrix} \right\}.$$

We say that PC satisfies $X \in \{$*completeness*, *knowledge soundness*, *computational special soundness*, *honest-verifier zero-knowledge*$\}$ if and only if $\mathsf{PC.Open}$ satisfies X.

We also define a multilinear PCS *for random openings* by changing the syntax of Open to require evaluations only on random points x sent as challenge by the verifier. This allows for the extractor to rewind the evaluation point as well as the other parts of the transcript. Looking ahead, this notion is useful as one of our subprotocols, $\mathsf{PC}_{\mathsf{Multi}}$, is only a multilinear PCS in this weak sense.

5.2 Spartan Protocols

We first describe the two variants of $\mathsf{Spartan}$. Note that in a slight abuse of terminology, we will use $\mathsf{Spartan\text{-}NIZK}$ and $\mathsf{Spartan\text{-}SNARK}$ to refer to the interactive versions of their respective protocols. When we wish to refer specifically to the non-interactive versions, we will write $\mathsf{Spartan\text{-}NIZK}_{\mathsf{FS}}$ and $\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}$.

Definition 5.1 ($\mathsf{R1CS}$). *A* $\mathsf{R1CS}$ instance *is a tuple* $(\mathbb{F}, A, B, C, m, n, \mathsf{io})$ *where* $A, B, C \in \mathbb{F}^{m \times m}$ *each with at most* $n = \Omega(m)$ *non-zero entries, and* $m \geq |\mathsf{io}| + 1$. *A* $\mathsf{R1CS}$ witness *is a vector* $w \in \mathbb{F}^{m - |\mathsf{io}| - 1}$ *such that if* $Z = (\mathsf{io}, 1, w)$, *then* $(A \cdot Z) \circ (B \cdot Z) = C \cdot Z$.

$\mathsf{Spartan}$ *makes further assumptions on the* $\mathsf{R1CS}$ *instances, namely that* $m = 2^\mu, n = 2^\nu$ *are powers of two, and* $|\mathsf{io}| + 1 = |w| = m/2$.

Key ideas. Both the NIZK and SNARK variants of Spartan prove satisfiability of R1CS instances using roughly the same ideas we now outline. See Figure 7 for a protocol description. It uses the following sub-protocols (q.v. Appendix B.1):

1. The Pedersen commitment scheme $\mathbf{g}^{\mathbf{a}} \cdot h^{\omega} \leftarrow \mathsf{Commit}((n, \mathbf{g}, h), \mathbf{a}; \omega)$.
2. Four Σ-protocols sharing the same setup:
   (a) OpenPf to prove knowledge of a commitment $C = g^x \cdot h^{\omega}$,
   (b) EqPf to prove equality of two commitments $C_1 = g^x \cdot h^{\omega_1}, C_2 = g^x \cdot h^{\omega_2}$,
   (c) ProdPf to prove that three commitments $C_{v_1}, C_{v_2}, C_{v_3}$ satisfy $v_1 \cdot v_2 = v_3$,
   (d) DotProdPf to prove that a multi-commitment $C_{\mathbf{x}}$ and a commitment C_y satisfy $y = \langle \mathbf{x}, \mathbf{a} \rangle$ for a public vector $\mathbf{a}$,
3. A $(\mu + 1)$-round public-coin interactive protocol $\mathsf{PC}_{\mathsf{Multi}}.\mathsf{Open}$ for proving polynomial evaluations of any multilinear polynomial $p(X_1, \ldots, X_\mu)$.
4. Additionally, in the case of Spartan-SNARK, we also need $\mathsf{PC}_{\mathsf{SparseMulti}}.\mathsf{Open}$ for proving evaluations of *sparse* multilinear polynomials $\widetilde{A}, \widetilde{B}, \widetilde{C}$.

At a high level, the main idea of Spartan is to reduce the satisfiability of the given R1CS instance to a claim that can be verified via sumcheck. To do this, the matrices A, B, C are interpreted as functions $\{0,1\}^\mu \times \{0,1\}^\mu \to \mathbb{F}$, and similarly $Z : \{0,1\}^\mu \to \mathbb{F}$, by writing the indices as their binary representations. We then take the *multilinear extension* $\widetilde{A}, \widetilde{B}, \widetilde{C}, \widetilde{Z}$ of these functions, and define the polynomial

$$\widetilde{\mathcal{F}_{\mathsf{io}}}(X) = \left(\sum_{y \leftarrow \{0,1\}^\mu} \widetilde{A}(X, y) \cdot \widetilde{Z}(y) \right) \cdot \left(\sum_{y \leftarrow \{0,1\}^\mu} \widetilde{B}(X, y) \cdot \widetilde{Z}(y) \right) - \left(\sum_{y \leftarrow \{0,1\}^\mu} \widetilde{C}(X, y) \cdot \widetilde{Z}(y) \right).$$

Note that $\widetilde{\mathcal{F}_{\mathsf{io}}}(X)$ vanishes on $\{0,1\}^\mu$ if and only if the R1CS constraint is satisfied. Finally, we turn this vanishing condition into a sumcheck instance by defining $\mathcal{G}_{\mathsf{io},\tau}(X) = \widetilde{\mathcal{F}_{\mathsf{io}}}(X) \cdot \widetilde{\mathsf{eq}}(X, \tau)$ for a random $\tau \in \mathbb{F}^\mu$, supplied by the verifier. The goal is then to prove that $\sum_{y \in \{0,1\}^\mu} \mathcal{G}_{\mathsf{io},\tau}(y) = 0$. The prover and verifier engage in sumcheck for this claim. The final step of sumcheck requires the verifier to evaluate $\mathcal{G}_{\mathsf{io},\tau}$ at a random point r_x, but the verifier cannot do this itself; thus, the prover and verifier engage in another run of sumcheck (more precisely, three runs batched together with verifier randomness) to reduce the task of evaluating $\mathcal{G}_{\mathsf{io},\tau}(r_x)$ to evaluating $\widetilde{A}, \widetilde{B}, \widetilde{C}$ all at (r_x, r_y), and $\widetilde{Z}$ at r_y. In both Spartan-NIZK and Spartan-SNARK, the verifier gets a commitment to the evaluation of the witness, and is convinced the committed evaluation is correct via $\mathsf{PC}_{\mathsf{Multi}}.\mathsf{Open}$. (Our analyses below assume $\mathsf{PC}_{\mathsf{Multi}}$ is instantiated with HyraxPC [60].) In Spartan-NIZK, the verifier evaluates $\widetilde{A}, \widetilde{B}, \widetilde{C}$ itself; in Spartan-SNARK, the prover sends the verifier the evaluations and uses $\mathsf{PC}_{\mathsf{SparseMulti}}.\mathsf{Open}$, a secondary proof protocol, to convince the verifier of their correctness.

For completeness, we list the full content of a Spartan-NIZK transcript in Appendix B.1. We can compute the number of rounds of Spartan-NIZK to be $\mathsf{r} = 7\mu + 11$. For Spartan-SNARK, the transcript is the same except for the verifier sending its commitments to $\widetilde{A}, \widetilde{B}, \widetilde{C}$ to the prover, the evaluations v_1, v_2, v_3, and the $\mathcal{O}(\mu)$-round transcript of $\mathsf{PC}_{\mathsf{SparseMulti}}.\mathsf{Open}$. Thus, the transcript of Spartan-SNARK has $O(\mu)$ more rounds for evaluating $\widetilde{A}(r_x, r_y), \widetilde{B}(r_x, r_y), \widetilde{C}(r_x, r_y)$.

5.3 SIM-EXT Analysis of Spartan-NIZK

Following Theorem 3.4, to prove that $\mathsf{Spartan\text{-}NIZK}_{\mathsf{FS}}$ satisfies SIM-EXT, we will need to show that it satisfies knowledge soundness (KS) along with k-ZK and k-UR for the same round k. By Lemma 4.7, knowledge soundness in turn depends on computational special soundness (SS) of the interactive protocol Spartan-NIZK. Our first set of results will be to establish SS of Spartan-NIZK through the following steps: (1) We first analyze the information-theoretic core of Spartan-NIZK, which is obtained from the protocol by sending all polynomials and evaluations in the clear, and checking the equalities directly. We call this variant Spartan-Core. (2) We then analyze how to extract from the various commitments and subprotocols in Spartan-NIZK to recover Spartan-Core.

The soundness of Spartan-Core has been analyzed in [55].

R1CS Relation.

$$\mathcal{R}_{\mathsf{R1CS}} = \left\{ \begin{array}{c} ((\mathbb{F}, m, n, A, B, C), \mathsf{io}, w) : \\ (A \cdot Z) \circ (B \cdot Z) = (C \cdot Z), \text{ where } Z = (\mathsf{io}, 1, w) \end{array} \right\}.$$

Setup Phase. Let $\mu = \log m$. Run $\mathsf{pp}_{\mathcal{G}} = (\mathbb{G}, \mathbb{F}) \leftarrow \mathsf{GroupGen}(1^\lambda)$, $\mathsf{pp}_{\mathsf{Multi}} \leftarrow \mathsf{PC}_{\mathsf{Multi}}.\mathsf{Setup}(\mu, \mathsf{pp}_{\mathcal{G}})$ and $\mathsf{pp}_{\Sigma} \leftarrow \Sigma.\mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$. Run $\mathsf{pp}_{\mathsf{Sparse}} \leftarrow \mathsf{PC}_{\mathsf{Multi}}.\mathsf{Setup}(\mu, n, \mathsf{pp}_{\mathcal{G}})$. Return $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Multi}}, \mathsf{pp}_{\Sigma}, \mathsf{pp}_{\mathsf{Sparse}})$.

Interaction Phase.

0. $\mathcal{V}$ computes $C_{\widetilde{X}} \leftarrow \mathsf{PC}_{\mathsf{SparseMulti}}.\mathsf{Commit}(\mathsf{pp}, \widetilde{X})$ for $X \in \{A, B, C\}$.

$\mathcal{V}$ then sends the coins used in this step to $\mathcal{P}$.

1. $\mathcal{P}$ computes $C_{\widetilde{w}} \leftarrow \mathsf{PC}_{\mathsf{Multi}}.\mathsf{Commit}(\mathsf{pp}_{\mathsf{PC}}, \widetilde{w})$ and sends $C_{\widetilde{w}}$ to $\mathcal{V}$.

2. $\mathcal{V}$ responds with challenge $\tau \xleftarrow{\$} \mathbb{F}^\mu$.

3. $\mathcal{P}, \mathcal{V}$ engage in sumcheck for $\sum_{x \in \{0,1\}^\mu} \mathcal{G}_{\mathsf{io},\tau}(x) \stackrel{?}{=} 0$.[a]

At the end, $\mathcal{P}$ sends $C_{e_x} \leftarrow \mathsf{Commit}(\mathsf{pp}, e_x)$ supposedly containing $e_x = \mathcal{G}_{\mathsf{io},\tau}(r_x)$ for $r_x \xleftarrow{\$} \mathbb{F}^\mu$ sent by $\mathcal{V}$.

4. $\mathcal{P}$ computes $v_M = \sum_{y \in \{0,1\}^\mu} \widetilde{M}(r_x, y) \cdot \widetilde{Z}(y)$ for $M \in \{A, B, C\}$. $\mathcal{P}$ then computes $C_{v_M} \leftarrow \mathsf{Commit}(\mathsf{pp}, v_M)$ for $M \in \{A, B, C\}$ and $C_{v_{AB}} \leftarrow \mathsf{Commit}(\mathsf{pp}, v_A \cdot v_B)$.
$\mathcal{P}$ sends $C_{v_A}, C_{v_B}, C_{v_C}, C_{v_{AB}}$ to $\mathcal{V}$.

5. $\mathcal{P}, \mathcal{V}$ engage in ProdPf to show that $v_{AB} = v_A \cdot v_B$.

6. $\mathcal{P}, \mathcal{V}$ engage in OpenPf to show that C_{v_C} is indeed a commitment to v_C.

7. $\mathcal{P}, \mathcal{V}$ engage in EqPf to show that $e_x = (v_{AB} - v_C) \cdot \widetilde{\mathrm{eq}}(r_x, \tau)$.

8. $\mathcal{V}$ responds with challenges $r_A, r_B, r_C \xleftarrow{\$} \mathbb{F}$.

9. Let $\mathcal{H}_{r_x}(Y) = \sum_{M \in \{A,B,C\}} r_M \cdot \widetilde{M}(r_x, Y) \cdot \widetilde{Z}(Y)$ and $T = \sum_{M \in \{A,B,C\}} r_M \cdot v_M$.
$\mathcal{P}, \mathcal{V}$ engage in sumcheck for $\sum_{y \in \{0,1\}^\mu} \mathcal{H}_{r_x}(y) = T$.

At the end, $\mathcal{P}$ sends a commitment C_{e_y} supposedly containing $e_y = \mathcal{H}_{r_x}(r_y)$ for $r_y \xleftarrow{\$} \mathbb{F}^\mu$ sent by $\mathcal{V}$.

10. $\mathcal{P}, \mathcal{V}$ engage in $\mathsf{PC}_{\mathsf{Multi}}.\mathsf{Open}$ for $\widetilde{w}((r_y)_{[1:]}) \to v_w$. At the end, both parties get C_{v_w} and compute

$$C_{v_Z} = C_{v_w}^{1-(r_y)_0} \cdot C_{v_{\mathsf{io}}}^{(r_y)_0},$$

where $v_{\mathsf{io}} \leftarrow \widetilde{(\mathsf{io}, 1)}((r_y)_{[1:]})$ and $C_{v_{\mathsf{io}}} \leftarrow \mathsf{Commit}(\mathsf{pp}, v_{\mathsf{io}}; 0)$.

11. $\mathcal{V}$ computes $v_1 = \widetilde{A}(r_x, r_y)$, $\quad v_2 = \widetilde{B}(r_x, r_y)$, $\quad v_3 = \widetilde{C}(r_x, r_y)$.

Instead $\mathcal{V}$ receives v_1, v_2, v_3 from $\mathcal{P}$.

Then $\mathcal{P}, \mathcal{V}$ engage in $\mathsf{PC}_{\mathsf{Multi}}.\mathsf{Open}$ to check that v_1, v_2, v_3 are correct.

12. $\mathcal{P}, \mathcal{V}$ engage in EqPf to check that $e_y = (r_A \cdot v_1 + r_B \cdot v_2 + r_C \cdot v_3) \cdot v_Z$.

[a] The sumcheck subroutine is described in Figure 8.

Fig. 7: Spartan-NIZK, with modifications for Spartan-SNARK in red.

Sumcheck Sub-Protocol. The sumcheck relation is $\sum_{x\in\{0,1\}^\mu} p(x) = T$, where p is a multivariate polynomial of individual degree at most d. $\mathcal{V}$ is given a commitment C_p and a commitment C_T. The sumcheck subprotocol reduces this claim to the claim that $p(r_x) \stackrel{?}{=} e_x$ for a random $r_x \stackrel{\$}{\leftarrow} \mathbb{F}^\mu$ sampled randomly by $\mathcal{V}$, and some claimed value $e_x \in \mathbb{F}$ available as a commitment C_{e_x} to $\mathcal{V}$.

Let $e_0 = T$. For $i = 1, \ldots, \mu$:

1. $\mathcal{P}$ computes the polynomial $p_i(X) = \sum_{x\in\{0,1\}^{\mu-i}} P(r_1, \ldots, r_{i-1}, X, x)$, parse it as a vector of coefficients, then sends $C_{p_i} \leftarrow \mathsf{Commit}(\mathsf{pp}, p_i; \omega_{p_i})$ to $\mathcal{V}$.
2. $\mathcal{V}$ responds with challenge $r_i \stackrel{\$}{\leftarrow} \mathbb{F}$.
3. $\mathcal{P}$ computes $e_i = p_i(r_i)$, then sends $C_{e_i} \leftarrow \mathsf{Commit}(\mathsf{pp}, e_i; \omega_{e_i})$ to $\mathcal{V}$.
4. $\mathcal{V}$ responds with challenges $w_{i,1}, w_{i,2} \stackrel{\$}{\leftarrow} \mathbb{F}$.
5. $\mathcal{P}, \mathcal{V}$ compute $\mathbf{a} = w_{i,1} \cdot (\mathbf{0}^k + \mathbf{1}^k) + w_{i,2} \cdot \mathbf{r_i}^k$ and $C_{y_i} = C_{e_{i-1}}^{w_{i,1}} \cdot C_{e_i}^{w_{i,2}}$. In addition, $\mathcal{P}$ computes $y_i = w_{i,1} \cdot e_{i-1} + w_{i,2} \cdot e_i$ and $\omega_{y_i} = w_{i,1} \cdot \omega_{e_{i-1}} + w_{i,2} \cdot \omega_{e_i}$.
6. $\mathcal{P}, \mathcal{V}$ engage in $\mathsf{DotProdPf}(\mathsf{pp}, (C_{p_i}, C_{y_i}, \mathbf{a}), (p_i, \omega_{p_i}, y_i, \omega_{y_i}))$.

Fig. 8: Sumcheck Sub-Protocol

Lemma 5.2 ([55]). $\mathsf{Spartan\text{-}Core}$ *has soundness error* $\frac{6\mu+1}{|\mathbb{F}|}$.

Special soundness for Σ-protocols was analyzed in another previous work [60].

Lemma 5.3 ([60]). *Let* $\Pi \in \{\mathsf{OpenPf}, \mathsf{EqPf}, \mathsf{ProdPf}, \mathsf{DotProdPf}\}$. *Then* Π *is* 2*-perfect special sound. Concretely, there exists a tree-extraction algorithm* $\mathcal{TE}_\Pi$ *that can extract a valid witness for* Π *given any* 2*-tree of accepting transcripts.*

We also need to analyze special soundness of $\mathsf{PC}_{\mathsf{Multi}}.\mathsf{Open}$. Note that while [60] introduced this protocol, they did not provide a concrete soundness result for it. The proof of the lemma below is in Appendix B.2.

Lemma 5.4. $\mathsf{PC}_{\mathsf{Multi}}.\mathsf{Open}$ *is* $\mathbf{n} = (\sqrt{m}, \underbrace{4_\pm, \ldots, 4_\pm}_{\mu/2}, 2)$*-computational special sound. Concretely, there exists a tree-extraction algorithm* $\mathcal{TE}_{\mathsf{PC}_{\mathsf{Multi}}}$ *such that for any EPT adversary* $\mathcal{A}$ *against* SS *of* $\mathsf{PC}_{\mathsf{Multi}}.\mathsf{Open}$*, there exists an EPT adversary* $\mathcal{B}$ *against* $\mathsf{DL\text{-}REL}$*, as efficient as* $\mathcal{A}$ *and* $\mathcal{TE}_{\mathsf{PC}_{\mathsf{Multi}}}$ *combined, such that*

$$\mathbf{Adv}^{\mathsf{SS}}_{\Pi,\mathbf{n}}(\mathcal{TE}_{\mathsf{PC}_{\mathsf{Multi}}}, \mathcal{A}) \leq \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},\sqrt{m}+2}(\mathcal{B}).$$

Our next step is to analyze the computational special soundness of the sumcheck subprotocol in Figure 8. Since it is not strictly an interactive argument, we explicitly state the guarantees of the tree extractor.

Lemma 5.5. *There exists a tree extractor* $\mathcal{TE}_{\mathsf{SC}}$ *such that given a* $(1, 2_{\mathsf{li}}, 2)^\mu$*-tree of accepting transcripts, produced by an adversary* $\mathcal{A}$*, for the sumcheck subprotocol, either outputs polynomials* $p_1(X), \ldots, p_\mu(X)$ *that satisfy the information-theoretic sumcheck protocol, or we can build an adversary* $\mathcal{B}$*, as efficient as* $\mathcal{TE}_{\mathsf{SC}}$ *and* $\mathcal{A}$ *combined, against* $\mathsf{DL\text{-}REL}$*.*

Proof. We will analyze a single iteration $i \in [\mu]$ of the sumcheck subprotocol; all other iterations will follow the same reasoning. We construct a tree extractor $\mathcal{TE}_{\mathsf{SC}}$ that does the following for each iteration $i \in [\mu]$: given a $(1, 2_{\mathsf{li}}, 2)$-tree of transcripts,

1. Run $\mathcal{TE}_{\mathsf{DotProdPf}}$ on each $(1, 1, 2)$-subtree to extract $(p_i, \omega_{p_i}, y_i, \omega_{y_i})$, where

   $$C_{p_i} = \mathsf{PC.Commit}(\mathsf{pp}, p_i; \omega_{p_i}), \quad C_{y_i} = \mathsf{Commit}(\mathsf{pp}, y_i; \omega_{y_i}), \quad \langle p_i, \mathbf{a}_i \rangle = y_i,$$

   and y_i is supposedly equal to $w_{i,1} \cdot e_{i-1} + w_{i,2} \cdot e_i$.

2. Given two pairs of linearly independent challenges $(w_{i,1}, w_{i,2}), (w'_{i,1}, w'_{i,2})$, with extracted witnesses $(p_i, \omega_{p_i}, y_i, \omega_{y_i}), (p'_i, \omega'_{p_i}, y'_i, \omega'_{y_i})$ from the previous step, we first assert that $(p_i, \omega_{p_i}) = (p'_i, \omega'_{p_i})$. If this assertion fails, then we have an adversary $\mathcal{B}$ against $\mathsf{DL\text{-}REL}$ since $C_{p_i} = \mathbf{g}^{p_i} \cdot h^{\omega_{p_i}} = \mathbf{g}^{p'_i} \cdot h^{\omega'_{p_i}}$. Next, we can solve for $e_{i-1}, e_i, \omega_{e_{i-1}}, \omega_{e_i}$ from the linear equations

$$\begin{cases} y_i = w_{i,1} \cdot e_{i-1} + w_{i,2} \cdot e_i \\ y'_i = w'_{i,1} \cdot e_{i-1} + w'_{i,2} \cdot e_i \end{cases} \quad \text{and} \quad \begin{cases} \omega_{y_i} = w_{i,1} \cdot \omega_{e_{i-1}} + w_{i,2} \cdot \omega_{e_i} \\ \omega'_{y_i} = w'_{i,1} \cdot \omega_{e_{i-1}} + w'_{i,2} \cdot \omega_{e_i} \end{cases}.$$

Recall that we also have $\langle p_i, \mathbf{a}_i \rangle = y_i$ and $\langle p_i, \mathbf{a}'_i \rangle = y'_i$; taking the same linear combination used to solve the equations above would give us $p_i(0) + p_i(1) = e_{i-1}$ and $p_i(r_i) = e_i$. Thus, we have extracted valid polynomials for the information-theoretic sumcheck protocol.

□

Putting together the above special soundness results for the subprotocols, we obtain special soundness for Spartan-NIZK.

Lemma 5.6. Spartan-NIZK *satisfies* $\mathbf{n}$*-computational special soundness, where*

$$\mathbf{n} = (1, (1, 2_{\mathsf{li}}, 2)^{\mu}, 2, 2, 2, 1, (2, 2_{\mathsf{li}}, 2)^{\mu}, (\underbrace{4_{\pm}, \ldots, 4_{\pm}}_{\mu/2}, 2), 2).$$

Concretely, there exists a PPT tree extractor $\mathcal{TE}_{\mathsf{Spartan\text{-}NIZK}}$ *such that for every EPT adversary* $\mathcal{A}$ *against* SS *of* Spartan-NIZK*, there exists an EPT adversary* $\mathcal{B}$ *against* DL-REL*, as efficient as* $\mathcal{A}$ *and* $\mathcal{TE}_{\mathsf{Spartan\text{-}NIZK}}$ *combined, such that*

$$\mathbf{Adv}^{\mathsf{SS}}_{\mathsf{Spartan\text{-}NIZK},\mathbf{n}}(\mathcal{TE}_{\mathsf{Spartan\text{-}NIZK}}, \mathcal{A}) \leq \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},\sqrt{m}+2}(\mathcal{B}) + \frac{6\mu + 1}{|\mathbb{F}|}.$$

Proof. We describe the tree extractor $\mathcal{TE}_{\mathsf{Spartan\text{-}NIZK}}$. Given a $\mathbf{n}$-tree of accepting transcripts, it runs the following sub-extractors for the corresponding sub-trees:

1. Run $\mathcal{TE}_{\mathsf{SC}}$ for the first sumcheck subprotocol on each $(1, 2_{\mathsf{li}}, 2)^{\mu}$ sub-tree to extract polynomials $p_i(X)$ for $i \in [\mu]$ that satisfy the information-theoretic sumcheck protocol.
2. Run $\mathcal{TE}_{\mathsf{ProdPf}}, \mathcal{TE}_{\mathsf{OpenPf}}, \mathcal{TE}_{\mathsf{EqPf}}$ on each corresponding 2-subtree to extract claims v_A, v_B, v_C such that $e_x = (v_A \cdot v_B - v_C) \cdot \widetilde{\mathrm{eq}}(r_x, \tau)$.
3. Run $\mathcal{TE}_{\mathsf{SC}}$ for the second sumcheck subprotocol on each $(1, 2_{\mathsf{li}}, 2)^{\mu}$ sub-tree to extract polynomials $p_i(X)$ for $i \in [\mu]$ that satisfy the information-theoretic sumcheck protocol.
4. Run $\mathcal{TE}_{\mathsf{PC_{Multi}}}$ for the opening argument $\mathsf{PC_{Multi}}.\mathsf{Open}$ on the $(\underbrace{4_{\pm}, \ldots, 4_{\pm}}_{\mu/2}, 2)$ sub-tree, and on $2^{\mu/2} = \sqrt{m}$ different challenges r_y provided by the $(2, 2_{\mathsf{li}}, 2)^{\mu}$ sub-tree, to extract a multilinear polynomial $\widetilde{w}(X)$ along with a correct evaluation $\widetilde{w}(r_y) = v_w$.
5. Run $\mathcal{TE}_{\mathsf{EqPf}}$ for the final equality proof to verify the equality $e_y = (r_A \cdot v_A + r_B \cdot v_B + r_C \cdot v_C) \cdot v_Z$.
6. Output the R1CS witness w.

Note that the $(2, 2_{\mathsf{li}}, 2)^{\mu}$ sub-tree in the second sumcheck subprotocol is necessary for extracting both from sumcheck, as well as from $\mathsf{PC_{Multi}}.\mathsf{Open}$. We now consider the following hybrids. Hyb_0 corresponds to the game SS for Spartan-NIZK with the tree extractor constructed above. Hyb_1 is the same as Hyb_0, but we additionally reject if the extracted R1CS witness is not satisfying. Conditioned on the event that none of the sub-extractor fails (and when that happens we get a DL-REL adversary $\mathcal{B}$), Hyb_1 differs from Hyb_0 exactly when the soundness of Spartan-Core is violated, which happens with probability at most $\frac{6\mu+1}{|\mathbb{F}|}$. □

Using Lemma 4.7 with Lemma 5.6, we conclude that $\mathsf{Spartan\text{-}NIZK_{FS}}$ satisfies knowledge soundness. Note that the minimum partition size in the $\mathbf{n}$-tree of transcripts is $C = \frac{|\mathbb{F}|-1}{2}$.

Theorem 5.7. $\mathsf{Spartan\text{-}NIZK_{FS}}$ *satisfies knowledge soundness. In particular, there exists an extractor* $\mathcal{E}_{\mathsf{Spartan\text{-}NIZK_{FS}}}$ *such that for every PPT prover* $\mathcal{P}^*$ *against* KS *of* $\mathsf{Spartan\text{-}NIZK}$ *making at most* Q *random oracle queries, there exists an EPT adversary* $\mathcal{B}$ *against* $\mathsf{DL\text{-}REL}$ *such that*

$$\mathbf{Adv}^{\mathsf{KS}}_{\mathsf{Spartan\text{-}NIZK_{FS}}}(\mathcal{E}_{\mathsf{Spartan\text{-}NIZK_{FS}}}, \mathcal{P}^*) \leq \frac{Q(Q-1) + (Q+1)(13\mu+10) + 2(6\mu+1)}{|\mathbb{F}|-1} + \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},\sqrt{m}+2}(\mathcal{B}).$$

Here $\mu = \log m$*. Both* $\mathcal{B}$ *and the extractor* $\mathcal{E}_{\mathsf{Spartan\text{-}NIZK_{FS}}}$ *runs in expected time that is at most* $O(Q \cdot m^6)$ *the running time of* $\mathcal{P}^*$*.*

Our next task is to exhibit a k-ZK simulator for $\mathsf{Spartan\text{-}NIZK_{FS}}$. The high-level idea is to let the simulator execute all subprotocols except the last with valid witnesses, then only invoke the simulator for the final EqPf.

Theorem 5.8. $\mathsf{Spartan\text{-}NIZK_{FS}}$ *satisfies* $(\mathsf{r}-1)$*-ZK, where* $\mathsf{r} = 7\mu + 11$ *is the number of rounds of* $\mathsf{Spartan\text{-}NIZK}$*.*

Proof. See Figure 9 for a pseudocode description of our simulators. Using the sumcheck sub-simulator $\mathcal{S}_{\mathsf{SC_{FS}}}$ in the top of the figure, we build the full simulator $\mathcal{S}_{\mathsf{FS},\mathsf{r}-1}$ for $\mathsf{Spartan\text{-}NIZK_{FS}}$. From the construction of $\mathcal{S}_{\mathsf{FS},\mathsf{r}-1}$, it is clear that the proofs produced are accepting; this is because all the verifier's checks are done by checking the various proofs, which are either honestly generated, in which case validity follows from completeness, or by invoking the simulator, in which case validity follows from NIZK guarantee. Furthermore, $\mathcal{S}_{\mathsf{Spartan\text{-}NIZK_{FS}},\mathsf{r}-1}$ only makes a *single* RO reprogramming, which when the simulator $\mathcal{S}_{\mathsf{EqPf_{FS}}}$ is invoked.

It remains to show that the output is indistinguishable from that of real transcripts. For the sub-protocols, namely the Σ-protocols along with $\mathsf{PC_{Multi}.Open_{FS}}$, that we generate transcripts by generating honest proofs, we argue that they are indistinguishable. Firstly, the inputs to the arguments are the same (being perfectly blinded commitment). Secondly, the sub-protocols themselves are zero-knowledge, which implies witness indistinguishability. This further implies that the honestly generated proofs made by our simulator are identically distributed as proofs in real transcripts. In the last sub-protocol $\mathsf{EqPf_{FS}}$ for which we use the simulator, we argue indistinguishability using the guarantee of the simulator $\mathcal{S}_{\mathsf{EqPf_{FS}}}$. This concludes the proof of k-ZK. ∎

Lemma 5.9. $\mathsf{Spartan\text{-}NIZK_{FS}}$ *satisfies perfect* $(\mathsf{r}-1)$*-*UR*.*

Proof. The last two rounds of $\mathsf{Spartan\text{-}NIZK_{FS}}$ consists of an instance of the Σ-protocol $\mathsf{EqPf_{FS}}$, which itself satisfies perfect 1-UR. In more detail, the last message in $\mathsf{EqPf_{FS}}$ must be the unique scalar z that satisfies $h^z = (C_1/C_2)^c \cdot \alpha$, where C_1, C_2, α are group elements determined by the previous messages. Hence $\mathsf{Spartan\text{-}NIZK_{FS}}$ satisfies perfect $(\mathsf{r}-1)$-UR. ∎

Combining our results above, we obtain $\mathsf{SIM\text{-}EXT}$ for $\mathsf{Spartan\text{-}NIZK_{FS}}$.

Theorem 5.10. $\mathsf{Spartan\text{-}NIZK_{FS}}$ *is simulation-extractable. Concretely, there exists a simulator-extractor* $\mathcal{E}_{\mathsf{Spartan\text{-}NIZK_{FS}}}$ *such that for every PPT adversary* $\mathcal{P}^*$ *against* $\mathsf{SIM\text{-}EXT}$*, there exists an EPT adversary* $\mathcal{B}$ *against* $\mathsf{DL\text{-}REL}$ *such that*

$$\begin{aligned}&\mathbf{Adv}^{\mathsf{SIM\text{-}EXT}}_{\mathsf{Spartan\text{-}NIZK_{FS}},\mathcal{R}_{\mathsf{R1CS}}}(\mathcal{S}_{\mathsf{Spartan\text{-}NIZK_{FS}},k}, \mathcal{E}_{\mathsf{Spartan\text{-}NIZK_{FS}}}, \mathcal{P}^*)\\ &\qquad\qquad\leq \frac{Q(Q-1) + (Q+1)(13\mu+10) + 2(6\mu+1)}{|\mathbb{F}|-1} + \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},\sqrt{m}+2}(\mathcal{B}).\end{aligned}$$

Both $\mathcal{B}$ *and* $\mathcal{E}_{\mathsf{Spartan\text{-}NIZK_{FS}}}$ *runs in expected time at most* $O(Q \cdot m^6)$ *that of* $\mathcal{P}^*$*.*

Sub-simulator $\mathcal{S}_{\mathsf{SC_{FS}}}$.

Input. Public parameters pp, a commitment C_e to some value e allegedly equal to $\sum_{x \in \{0,1\}^\mu} p(x)$, with p a multivariate polynomial of individual degree at most d, and previous transcript tr (including pp, R1CS input (A, B, C, io), and all prover's messages so far).

Set $e_0 = e$. For $i = 1, \ldots, \mu$:

1. Sample $p_i(X) \xleftarrow{\$} \mathbb{F}^{\leq d}[X]$ randomly conditioned on $p_i(0) + p_i(1) = e_{i-1}$. Compute a commitment $C_{p_i} \leftarrow \mathsf{Commit}(\mathsf{pp}, p_i; \omega_{p_i})$ and append C_{p_i} to tr.
2. Obtain challenge $r_i \leftarrow \mathsf{H}(\mathsf{tr})$.
3. Let $e_i = p_i(r_i)$, compute a commitment $C_{e_i} \leftarrow \mathsf{Commit}(\mathsf{pp}, e_i; \omega_{e_i})$, and append C_{e_i} to tr.
4. Obtain challenges $w_{i,1}, w_{i,2} \leftarrow \mathsf{H}(\mathsf{tr})$.
5. Compute $\mathbf{a}, C_{y_i}, y_i, \omega_{y_i}$ as specified in the sumcheck subprotocol. Generate an honest proof $\pi_i \leftarrow \mathcal{P}_{\mathsf{DotProdPf_{FS}}}(\mathsf{pp}, (C_{p_i}, C_{y_i}, \mathbf{a}_i), (p_i, \omega_{p_i}, y_i, \omega_{y_i}))$. Append π_i to tr.

After μ rounds, return C_{e_μ}.

Simulator $\mathcal{S}_{\mathsf{FS},\mathsf{r}-1}(\mathsf{pp}, (\mathbb{F}, A, B, C, \mathsf{io}))$:

Initialize $\mathsf{tr} = (\mathsf{pp}, A, B, C, \mathsf{io})$.

1. Sample a random multilinear polynomial $\widetilde{w} \xleftarrow{\$} \mathbb{F}[X_1, \ldots, X_\mu]$. Compute $C_{\widetilde{w}} \leftarrow \mathsf{PC_{Multi}.Commit}(\mathsf{pp}, \widetilde{w}; \omega_{\widetilde{w}})$, and append $C_{\widetilde{w}}$ to tr.
2. Obtain challenge $\tau \leftarrow \mathsf{H}(\mathsf{tr})$.
3. Run $\mathcal{S}_{\mathsf{SC_{FS}}}$ on (pp, C_e) with current transcript tr, and get output $C_{e_x} \leftarrow \mathsf{Commit}(\mathsf{pp}, e_x; \omega_{e_x})$ for some scalar $e_x \in \mathbb{F}$.
4. Sample $v_A, v_B, v_C \xleftarrow{\$} \mathbb{F}$ at random conditioned on $(v_A \cdot v_B - v_C) \cdot \widetilde{\mathrm{eq}}(r_x, \tau) = e_x$, and set $v_{AB} = v_A \cdot v_B$. Compute $C_{v_M} \leftarrow \mathsf{Commit}(\mathsf{pp}, v_M; \omega_M)$ for $M \in \{A, B, C\}$ along with $C_{v_{AB}} \leftarrow \mathsf{Commit}(\mathsf{pp}, v_{AB}; \omega_{AB})$, and append them to tr.
5. Generate an honest proof
$$\pi_{\mathsf{ProdPf}} \leftarrow \mathcal{P}_{\mathsf{ProdPf_{FS}}}(\mathsf{pp}, (C_{v_A}, C_{v_B}, C_{v_{AB}}), (v_A, v_B, \omega_{v_A}, \omega_{v_B}, \omega_{v_{AB}})),$$
and append it to tr.
6. Generate an honest proof $\pi_{\mathsf{OpenPf}} \leftarrow \mathcal{P}_{\mathsf{OpenPf_{FS}}}(\mathsf{pp}, (C_{v_C}), (v_C, \omega_{v_C}))$, and append it to tr.
7. Generate an honest proof $\pi_{\mathsf{EqPf},1} \leftarrow \mathcal{P}_{\mathsf{EqPf_{FS}}}(\mathsf{pp}, (C_{e_x}, C_{v'}), (\omega_{e_x} - \omega_{v'}))$, where $v' = (v_A \cdot v_B - v_C) \cdot \widetilde{\mathrm{eq}}(r_x, \tau)$ and $C_{v'} = (C_{v_{AB}} / C_{v_C})^{\widetilde{\mathrm{eq}}(r_x, \tau)}$; then append it to tr.
8. Obtain challenges $r_A, r_B, r_C \leftarrow \mathsf{H}(\mathsf{tr})$.
9. Compute $C_T = r_A \cdot C_{v_A} + r_B \cdot C_{v_B} + r_C \cdot C_{v_C}$. Run $\mathcal{S}_{\mathsf{SC_{FS}}}$ on $(\mathsf{pp}, C_T, \mathsf{tr})$, obtaining output $C_{e_y} \leftarrow \mathsf{Commit}(\mathsf{pp}, e_y; \omega_{e_y})$.
10. Generate opening proof $\pi_{\mathsf{PC_{Multi}.Open}} \leftarrow \mathcal{P}_{\mathsf{PC_{Multi}.Open_{FS}}}(\mathsf{pp}, (C_{\widetilde{w}}, r_y), (\widetilde{w}, \omega_{\widetilde{w}}))$; at the end, get $C_{v_w} = \mathsf{Commit}(\mathsf{pp}, v_w; \omega_{v_w})$, where $v_w \leftarrow \widetilde{w}(r_y[1\ldots])$, and append it to tr.
11. Compute $v_Z = (1 - r_y[0]) \cdot v_w + r_y[0] \cdot \widetilde{(\mathsf{io}, 1)}(r_y[1\ldots])$ and $C_{v_Z} = C_{v_w}^{1-(r_y)_0} \cdot C_{v_{\mathsf{io}}}^{(r_y)_0}$.
12. Compute $v_1 = \widetilde{A}(r_x, r_y), v_2 = \widetilde{B}(r_x, r_y), v_3 = \widetilde{C}(r_x, r_y)$. Generate a *simulated* proof $\pi_{\mathsf{EqPf},2} \leftarrow \mathcal{S}_{\mathsf{EqPf_{FS}}}$ for the equality $e_y = (r_A \cdot v_1 + r_B \cdot v_2 + r_C \cdot v_3) \cdot v_Z$. Append the proof to tr.

Return tr.

Fig. 9: Simulators for proof of k-ZK for $\mathsf{Spartan}$.

5.4 $\mathsf{SIM\text{-}EXT}$ of $\mathsf{Spartan\text{-}SNARK}$

For $\mathsf{Spartan\text{-}SNARK}$, the proof of $\mathsf{SIM\text{-}EXT}$ is similar to that of $\mathsf{Spartan\text{-}NIZK}$. In particular, the proofs of k-ZK and k-UR carries over, and we only need to modify the proof of special soundness to accommodate for the more complex sparse multilinear polynomial commitment scheme. In what follows, we let $\mathsf{r}' = 7\mu + 11 + O(\mu)$ be the round complexity of $\mathsf{Spartan\text{-}SNARK}$.

Lemma 5.11. $\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}$ *satisfies* k-ZK*, where* $k = \mathsf{r}' - 1$.

Proof. We modify the k-ZK simulator of $\mathsf{Spartan\text{-}NIZK}_{\mathsf{FS}}$ to also output opening proofs $\mathsf{PC}_{\mathsf{SparseMulti}}.\mathsf{Open}_{\mathsf{FS}}$ for $\widetilde{M}(r_x, r_y)$ with $M \in \{A, B, C\}$. Since A, B, C are part of the public input, the simulator has full access to the matrices, and hence can produce the proofs honestly. □

Lemma 5.12. $\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}$ *satisfies perfect* k-UR*, where* $k = \mathsf{r}' - 1$.

Proof. Since $\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}$ ends with the same invocation of the equality proof $\mathsf{EqPf}_{\mathsf{FS}}$, we obtain the same result as Lemma 5.9. □

The proof of knowledge soundness for $\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}$ is similar to that of $\mathsf{Spartan\text{-}NIZK}_{\mathsf{FS}}$, except we further need to extract the polynomials involved in $\mathsf{PC}_{\mathsf{SparseMulti}}.\mathsf{Open}$. We give a full proof of the lemma below in Appendix B.2.

Lemma 5.13. $\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}$ *satisfies knowledge soundness. Concretely, there exists an extractor* $\mathcal{E}_{\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}}$ *such that for every PPT prover* $\mathcal{P}^*$ *against* KS *of* $\mathsf{Spartan\text{-}SNARK}$ *making at most* Q *random oracle queries, there exists an EPT adversary* $\mathcal{B}$ *against* $\mathsf{DL\text{-}REL}$ *such that*

$$\begin{aligned}&\mathbf{Adv}^{\mathsf{KS}}_{\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}}(\mathcal{E}_{\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}}, \mathcal{P}^*)\\ &\qquad \le \frac{Q(Q-1) + (Q+1)(25\mu + 9\nu + 16) + 6(m+n) + O(\mu+\nu)}{|\mathbb{F}| - 1} + \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G}, \sqrt{m+n}+2}(\mathcal{B}).\end{aligned}$$

Here $\mu = \log m, \nu = \log n$. *Both* $\mathcal{B}$ *and the extractor* $\mathcal{E}_{\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}}$ *runs in expected time that is at most* $O(Q \cdot m^{7.5} \cdot (m+n)^3)$ *the running time of* $\mathcal{P}^*$.

Combining the results above, we obtain $\mathsf{SIM\text{-}EXT}$ for $\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}$.

Theorem 5.14. $\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}$ *satisfies* $\mathsf{SIM\text{-}EXT}$*. Concretely, there exists a simulator-extractor* $\mathcal{E}_{\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}}$ *such that for every PPT adversary* $\mathcal{P}^*$ *against* $\mathsf{SIM\text{-}EXT}$ *of* $\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}$*, there exists an EPT adversary* $\mathcal{B}$ *against* $\mathsf{DL\text{-}REL}$ *with*

$$\begin{aligned}&\mathbf{Adv}^{\mathsf{SIM\text{-}EXT}}_{\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}, \mathcal{R}_{\mathsf{R1CS}}}(\mathcal{S}_{\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}, k}, \mathcal{E}_{\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}}, \mathcal{P}^*)\\ &\qquad \le \frac{Q(Q-1) + (Q+1)(25\mu + 9\nu + 16) + 6(m+n) + O(\mu+\nu)}{|\mathbb{F}| - 1} + \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G}, \sqrt{m+n}+2}(\mathcal{B}).\end{aligned}$$

$\mathcal{B}$ *and* $\mathcal{E}_{\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}}$ *run in expected time* $O(Q \cdot m^{7.5} \cdot (m+n)^3)$ *that of* $\mathcal{P}^*$.

6 Simulation Extractability of $\mathsf{Bulletproofs}$

In this section, we show that the $\mathsf{Bulletproofs}$ protocols in [16] satisfy $\mathsf{SIM\text{-}EXT}$, without relying on the AGM. The authors of [16] introduced two protocols, an *aggregate range proof* $\mathsf{BP\text{-}ARP}$ and an *arithmetic circuit satisfiability proof* $\mathsf{BP\text{-}ACSPf}$[6], with both building on an *inner product argument* $\mathsf{BP\text{-}IPA}$.

6.1 Inner Product Argument

We describe the Bulletproofs inner product protocol $\mathsf{BP\text{-}IPA}$ in Figure 10, which is parametrized by a number $n \in \mathbb{N}$ that is assumed to be a power of 2. In the protocol, the public input consists of a group element P, and the prover wants to prove knowledge of vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$ such that $P = \mathbf{g}^{\mathbf{a}} \cdot \mathbf{h}^{\mathbf{b}} \cdot u^{\langle \mathbf{a}, \mathbf{b} \rangle}$. The protocol follows a "split-and-fold" approach, where in each round the inner product statement is reduced to a related one of half the dimension. We will show that $\mathsf{BP\text{-}IPA}_{\mathsf{FS}}$ is knowledge sound and satisfies 0-UR, meaning its proofs are computationally unique.

[6] To keep the naming consistent with [16], we refer to them as *proofs* even though they are actually *arguments*.

Inner Product Relation. Given $n = 2^k$ and $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$,

$$\mathcal{R}_{\mathsf{BP\text{-}IPA}} = \left\{((n, \mathbf{g}, \mathbf{h}, u), P, (\mathbf{a}, \mathbf{b})) \mid P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}u^{\langle \mathbf{a}, \mathbf{b}\rangle}\right\}.$$

Interaction Phase.

Set $n_0 \leftarrow n, \mathbf{g}^{(0)} \leftarrow \mathbf{g}, \mathbf{h}^{(0)} \leftarrow \mathbf{h}, P^{(0)} \leftarrow P, \mathbf{a}^{(0)} \leftarrow \mathbf{a}, \mathbf{b}^{(0)} \leftarrow \mathbf{b}$.

For $i = 1, \ldots, k$:

1. $\mathcal{P}$ computes $n_i = n_{i-1}/2,\ c_L = \langle \mathbf{a}^{(i-1)}_{[:n_i]}, \mathbf{b}^{(i-1)}_{[n_i:]}\rangle,\ c_R = \langle \mathbf{a}^{(i-1)}_{[n_i:]}, \mathbf{b}^{(i-1)}_{[:n_i]}\rangle$, and

$$L_i = \left(\mathbf{g}^{(i-1)}_{[n_i:]}\right)^{\mathbf{a}^{(i-1)}_{[:n_i]}} \cdot \left(\mathbf{h}^{(i-1)}_{[:n_i]}\right)^{\mathbf{b}^{(i-1)}_{[n_i:]}} \cdot u^{c_L}, \quad R_i = \left(\mathbf{g}^{(i-1)}_{[:n_i]}\right)^{\mathbf{a}^{(i-1)}_{[n_i:]}} \cdot \left(\mathbf{h}^{(i-1)}_{[n_i:]}\right)^{\mathbf{b}^{(i-1)}_{[:n_i]}} \cdot u^{c_R}.$$

$\mathcal{P}$ sends L_i, R_i to $\mathcal{V}$.

2. $\mathcal{V}$ sends challenge $x_i \xleftarrow{\$} \mathbb{F}^*$.
3. $\mathcal{P}$, $\mathcal{V}$ both compute $P^{(i)} = L_i^{x_i^2} \cdot P^{(i-1)} \cdot R_i^{x_i^{-2}}$, and

$$\mathbf{g}^{(i)} = \left(\mathbf{g}^{(i-1)}_{[:n_i]}\right)^{x_i^{-1}} \circ \left(\mathbf{g}^{(i-1)}_{[n_i:]}\right)^{x_i}, \quad \mathbf{h}^{(i)} = \left(\mathbf{h}^{(i-1)}_{[:n_i]}\right)^{x_i} \circ \left(\mathbf{h}^{(i-1)}_{[n_i:]}\right)^{x_i^{-1}}.$$

4. $\mathcal{P}$ computes $\mathbf{a}^{(i)} = \mathbf{a}^{(i-1)}_{[:n_i]} \cdot x_i^{-1} + \mathbf{a}^{(i-1)}_{[n_i:]} \cdot x_i,\ \ \mathbf{b}^{(i)} = \mathbf{b}^{(i-1)}_{[:n_i]} \cdot x_i + \mathbf{b}^{(i-1)}_{[n_i:]} \cdot x_i^{-1}$.

After k rounds, $\mathcal{P}$ sends $\mathbf{a}^{(k)}, \mathbf{b}^{(k)}$ to $\mathcal{V}$.

Verification. $\mathcal{V}$ checks whether $P^{(k)} \stackrel{?}{=} \left(\mathbf{g}^{(k)}\right)^{\mathbf{a}^{(k)}} \cdot \left(\mathbf{h}^{(k)}\right)^{\mathbf{b}^{(k)}} \cdot u^{\mathbf{a}^{(k)} \cdot \mathbf{b}^{(k)}}$.

Fig. 10: Bulletproofs' Inner Product Argument BP-IPA

Lemma 6.1. $\mathsf{BP\text{-}IPA_{FS}}$ *satisfies knowledge soundness. Concretely, there exists an extractor* $\mathcal{E}_{\mathsf{BP\text{-}IPA_{FS}}}$ *such that for every PPT adversary* $\mathcal{P}^*$ *against* KS *making at most* Q *random oracle queries, there exists an adversary* $\mathcal{B}$ *against* DL-REL *with*

$$\mathbf{Adv}^{\mathsf{KS}}_{\mathsf{BP\text{-}IPA_{FS}}}(\mathcal{E}_{\mathsf{BP\text{-}IPA_{FS}}}, \mathcal{P}^*) \leq \frac{Q(Q-1) + 6(Q+1)\log n}{|\mathbb{F}| - 1} + \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},2n+1}(\mathcal{B}).$$

Both $\mathcal{B}$ *and the extractor* $\mathcal{E}_{\mathsf{BP\text{-}IPA_{FS}}}$ *run in expected time that is at most* $O(Q \cdot n^2)$ *times the runtime of* $\mathcal{P}^*$.

Proof. Using Lemma 4.7, it suffices to show that BP-IPA satisfies computational special soundness. The tree of accepting transcripts is of the form $\mathbf{n} = (\underbrace{4_{\pm}, \ldots, 4_{\pm}}_{\log n})$, and the corresponding tree extractor $\mathcal{TE}_{\mathsf{BP\text{-}IPA}}$ is given in [16]. The guarantee of the tree extractor is that, given the appropriate tree of transcripts, it will either output a witness, or we can use it to build an adversary $\mathcal{B}$ against DL-REL. □

Lemma 6.2. $\mathsf{BP\text{-}IPA_{FS}}$ *satisfies* 0-UR. *Concretely, for every adversary* $\mathcal{A}$ *against* 0-UR *that makes at most* Q *random oracle queries, there exists an adversary* $\mathcal{B}$ *against* DL-REL *such that*

$$\mathbf{Adv}^{\mathsf{0\text{-}UR}}_{\mathsf{BP\text{-}IPA_{FS}}}(\mathcal{A}) \leq 2 \cdot \frac{Q(Q-1) + 6(Q+1)\log n}{|\mathbb{F}| - 1} + 3 \cdot \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},2n+1}(\mathcal{B}).$$

$\mathcal{B}$ *runs in expected time that is at most* $O(Q \cdot n^2)$ *times the running time of* $\mathcal{A}$.

Proof. We define the following sequence of hybrids. Hyb_0 is the game $\mathsf{0\text{-}UR}^{\mathcal{A}}_{\mathsf{BP\text{-}IPA_{FS}}}$. Recall that $\mathcal{A}$ wins in Hyb_0 by producing (P, π, π') such that $\pi \neq \pi'$ are valid proofs for P. We then define Hyb_1 to be the same as Hyb_0, except we additionally run $\mathcal{E}_{\mathsf{BP\text{-}IPA_{FS}}}$ on modified adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively, producing witnesses $w = (\mathbf{a}, \mathbf{b})$ and $w' = (\mathbf{a}', \mathbf{b}')$ for π and π'. Here $\mathcal{A}_1, \mathcal{A}_2$ are wrappers around $\mathcal{A}$; both run $\mathcal{A}$

once to get an output (x, π, π'), then $\mathcal{A}_1$ returns (x, π) and $\mathcal{A}_2$ returns (x, π'). Hyb_1 then outputs 0 if $(\mathsf{pp}, x, w) \notin \mathcal{R}$ or $(\mathsf{pp}, x, w') \notin \mathcal{R}$. Using Lemma 6.1, we can construct EPT adversaries $\mathcal{B}_1, \mathcal{B}_2$ such that

$$|\Pr[\mathsf{Hyb}_0] - \Pr[\mathsf{Hyb}_1]| \leq 2 \cdot \frac{Q(Q-1) + 6(Q+1)\log n}{|\mathbb{F}| - 1} + 2 \cdot \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},2n+1}(\mathcal{B}).$$

This bound follows from invoking the extractor $\mathcal{E}_{\mathsf{BP\text{-}IPA}_{\mathsf{FS}}}$ two times on adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively.

Next, we consider Hyb_2 which is the same as Hyb_1, except we also output 0 if $w \neq w'$. We claim that there exists an adversary $\mathcal{B}_3$ such that

$$|\Pr[\mathsf{Hyb}_1] - \Pr[\mathsf{Hyb}_2]| \leq \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},2n+1}(\mathcal{B}_3).$$

This is because if extraction succeeds, then $w_1 = (\mathbf{a}, \mathbf{b})$ and $w_2 = (\mathbf{a}', \mathbf{b}')$ are witnesses for the same input P. In particular, this implies the equality

$$\mathbf{g}^{\mathbf{a}} \cdot \mathbf{h}^{\mathbf{b}} \cdot u^{\langle \mathbf{a}, \mathbf{b} \rangle} = P = \mathbf{g}^{\mathbf{a}'} \cdot \mathbf{h}^{\mathbf{b}'} \cdot u^{\langle \mathbf{a}', \mathbf{b}' \rangle}.$$

Thus if $w \neq w'$, then we can build a $\mathsf{DL\text{-}REL}$ adversary $\mathcal{B}_3$.

Finally, we argue that $\Pr[\mathsf{Hyb}_2] = 0$. This follows by a closer inspection of the tree-extractor $\mathcal{TE}_{\mathsf{BP\text{-}IPA}}$ given in [16]. In the course of extraction, the tree-extractor $\mathcal{TE}_{\mathsf{BP\text{-}IPA}}$ actually recovers the representation of all group elements $L_1, R_1, \ldots, L_{\log n}, R_{\log n}$ sent in the proof, and these representations are according to how the honest prover would compute them from the witness $w = w' = (\mathbf{a}, \mathbf{b})$ in Figure 10. Thus we can show by induction that $\pi|_i = \pi'|_i$ for all $i \in [\log n]$, and hence $\pi = \pi'$, contradicting the assumption that $\pi \neq \pi'$. □

6.2 Aggregate Range Proof

We give a full description of the aggregate range proof $\mathsf{BP\text{-}ARP}$ in Figure 11. The value m is the number of committed values v_i, and n is the bit length of the upper bound (i.e., we prove $v_i \in [0, 2^n - 1]$ for all $i \in [m]$). Following the same approach as in Section 5 for $\mathsf{Spartan}$, we need to establish three properties of $\mathsf{BP\text{-}ARP}_{\mathsf{FS}}$: (1) knowledge soundness, (2) the existence of a k-ZK simulator, and (3) k-UR for the same round k. We begin with the proof of knowledge soundness, which is essentially a restatement of the original result from [16].

Lemma 6.3. $\mathsf{BP\text{-}ARP}_{\mathsf{FS}}$ *satisfies* $(m \cdot n, m+2, 3, 2, \underbrace{4_{\pm}, \ldots, 4_{\pm}}_{\log(m \cdot n)})$*-computational special soundness, and hence knowledge soundness. Concretely, there exists an extractor* $\mathcal{E}_{\mathsf{BP\text{-}ARP}_{\mathsf{FS}}}$ *such that for every PPT adversary* $\mathcal{P}^*$ *against* KS *making at most* Q *random oracle queries, there exists an adversary* $\mathcal{A}$ *against* $\mathsf{DL\text{-}REL}$ *with*

$$\mathbf{Adv}^{\mathsf{KS}}_{\mathsf{BP\text{-}ARP}_{\mathsf{FS}}}(\mathcal{E}_{\mathsf{BP\text{-}ARP}_{\mathsf{FS}}}, \mathcal{P}^*) \leq \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},2mn+3}(\mathcal{A}) + \frac{Q(Q-1) + 2(Q+1)\left(m(n+1) + 3\log(m \cdot n) + 3\right)}{|\mathbb{F}| - 1}.$$

Both $\mathcal{A}$ *and the extractor* $\mathcal{E}_{\mathsf{BP\text{-}ARP}_{\mathsf{FS}}}$ *run in expected time that is at most* $O(Q \cdot m^4 \cdot n^3)$ *times the runtime of* $\mathcal{P}^*$.

Proof. The description of a tree extractor $\mathcal{TE}_{\mathsf{BP\text{-}ARP}}$, which either outputs a valid witness or a discrete log relation, can be found in [16]. This concludes the proof of computational special soundness. Combining Theorem 4.4 with Lemma 4.7, we conclude knowledge soundness for $\mathsf{BP\text{-}ARP}_{\mathsf{FS}}$. The expected runtime of the extractor $\mathcal{E}_{\mathsf{BP\text{-}ARP}_{\mathsf{FS}}}$, as well as the adversary $\mathcal{A}$, is at most $O(Q \cdot (mn) \cdot (m+2) \cdot 6 \cdot (mn)^2) = O(Q \cdot m^4 \cdot n^3)$ times the runtime of $\mathcal{P}^*$, by Theorem 4.4. □

Lemma 6.4. $\mathsf{BP\text{-}ARP}_{\mathsf{FS}}$ *satisfies perfect* 2*-*ZK*.*

Aggregate Range Proof Relation.

$$\mathcal{R}_{\mathsf{BP\text{-}ARP}} = \left\{ \begin{array}{c} ((m, n, \mathbf{g}, \mathbf{h}, g, h, u), \mathbf{V}, (\mathbf{v}, \boldsymbol{\gamma})) : \\ V_j = g^{v_j} h^{\gamma_j} \ \wedge \ v_j \in [0, 2^n - 1] \ \forall j \in [1, m] \end{array} \right\}.$$

Interaction Phase. Denote $\mathbf{y}^{m \cdot n} = (1, y, \ldots, y^{m \cdot n - 1}) \in \mathbb{F}^{m \cdot n}$.

1. $\mathcal{P}$ samples $\alpha, \rho \xleftarrow{\$} \mathbb{F}$, $\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{F}^{m \cdot n}$ and computes

$$\mathbf{a}_L \in \{0,1\}^{m \cdot n} \text{ such that } \langle (\mathbf{a}_L)_{[(j-1)n, jn-1]}, \mathbf{2}^n \rangle = v_j \ \forall j \in [1, m],$$
$$\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^{m \cdot n},$$
$$A = h^{\alpha} \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R}, \quad S = h^{\rho} \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}.$$

$\mathcal{P}$ sends A, S to $\mathcal{V}$.

2. $\mathcal{V}$ sends challenges $y, z \xleftarrow{\$} \mathbb{F}^*$.
3. $\mathcal{P}$ samples $\beta_1, \beta_2 \xleftarrow{\$} \mathbb{F}$ and computes

$$\ell(X) = (\mathbf{a}_L - z \cdot \mathbf{1}^{m \cdot n}) + \mathbf{s}_L \cdot X,$$
$$r(X) = \mathbf{y}^{m \cdot n} \circ (\mathbf{a}_R + z \cdot \mathbf{1}^{m \cdot n} + \mathbf{s}_R \cdot X) + \sum_{j=1}^{m} z^{j+1} \cdot \left(\mathbf{0}^{(j-1)n} \| \mathbf{2}^n \| \mathbf{0}^{(m-j)n} \right),$$
$$t(X) = \langle \ell(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2, \quad T_1 = g^{t_1} h^{\beta_1}, \quad T_2 = g^{t_2} h^{\beta_2}.$$

$\mathcal{P}$ sends T_1, T_2 to $\mathcal{V}$.

4. $\mathcal{V}$ sends challenge $x \xleftarrow{\$} \mathbb{F}^*$.
5. $\mathcal{P}$ computes

$$\mathbf{l} = \ell(x), \quad \mathbf{r} = r(x), \quad \hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle, \quad \mu = \alpha + \rho \cdot x,$$
$$\beta_x = \beta_2 \cdot x^2 + \beta_1 \cdot x + \sum_{j=1}^{m} z^{j+1} \cdot \gamma_j.$$

$\mathcal{P}$ sends $\hat{t}, \beta_x, \mu$ to $\mathcal{V}$.

6. $\mathcal{V}$ sends challenge $w \xleftarrow{\$} \mathbb{F}^*$.
7. $\mathcal{P}, \mathcal{V}$ both compute

$$\mathbf{h}' = \mathbf{h}^{\mathbf{y}^{-m \cdot n}}, \quad u' = u^w,$$
$$P' = h^{-\mu} \cdot A \cdot S^x \cdot \mathbf{g}^{-z \cdot \mathbf{1}^{m \cdot n}} \cdot (\mathbf{h}')^{z \cdot \mathbf{y}^{m \cdot n}} \cdot \prod_{j=1}^{m} (\mathbf{h}')^{z^{j+1} \cdot \mathbf{2}^n}_{[(j-1)n, jn-1]} \cdot (u')^{\hat{t}}.$$

8. $\mathcal{P}, \mathcal{V}$ engage in $\mathsf{BP\text{-}IPA}$ for the triple $((m \cdot n, \mathbf{g}, \mathbf{h}', u'), P', (\mathbf{l}, \mathbf{r}))$.

Verification.

1. $\mathcal{V}$ rejects if $\mathsf{BP\text{-}IPA}$ fails.
2. $\mathcal{V}$ computes $R = \mathbf{V}^{z^2 \cdot \mathbf{z}^m} \cdot g^{(z - z^2) \cdot \langle \mathbf{1}^{m \cdot n}, \mathbf{y}^{m \cdot n} \rangle - \sum_{j=1}^{m} z^{j+2} \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle} \cdot T_1^x \cdot T_2^{x^2}$.
3. $\mathcal{V}$ checks whether $g^{\hat{t}} h^{\beta_x} \stackrel{?}{=} R$.

Fig. 11: Bulletproofs' Aggregate Range Proof $\mathsf{BP\text{-}ARP}$

Simulator $\mathcal{S}_{\mathsf{BP\text{-}ARP}_{\mathsf{FS}},x}(\mathsf{pp} = (m, n, \mathbf{g}, \mathbf{h}, g, h, u), \mathbf{V})$**:**

1. Initialize $\mathsf{tr} = (\mathsf{pp}, \mathbf{V})$. Sample $\alpha, \rho \xleftarrow{\$} \mathbb{F}$, $\mathbf{a}_L, \mathbf{a}_R, \mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{F}^{m \cdot n}$ and compute $A = h^{\alpha}\mathbf{g}^{\mathbf{a}_L}\mathbf{h}^{\mathbf{a}_R}$, $S = h^{\rho}\mathbf{g}^{\mathbf{s}_L}\mathbf{h}^{\mathbf{s}_R}$. Append A, S to tr.
2. Obtain challenges $y, z \leftarrow \mathsf{H}(\mathsf{tr})$.
3. Sample $x \xleftarrow{\$} \mathbb{F}^*$ and compute

$$\mu = \alpha + \rho \cdot x,$$
$$\mathbf{l} = (\mathbf{a}_L - z \cdot \mathbf{1}^{m \cdot n}) + \mathbf{s}_L \cdot x,$$
$$\mathbf{r} = \mathbf{y}^{m \cdot n} \circ (\mathbf{a}_R + z \cdot \mathbf{1}^{m \cdot n} + \mathbf{s}_R \cdot x) + \sum_{j=1}^{m} z^{j+1} \cdot \left(\mathbf{0}^{(j-1)n} \| \mathbf{2}^n \| \mathbf{0}^{(m-j)n}\right),$$
$$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle.$$

4. Sample $\beta_x \xleftarrow{\$} \mathbb{F}$, $T_1 \xleftarrow{\$} \mathbb{G}$, and compute $T_2 = \left(g^{\hat{t} - \delta(y,z)} \cdot h^{\beta_x} \cdot \mathbf{V}^{-z^2 \cdot \mathbf{z}^m} \cdot T_1^{-x}\right)^{x^{-2}}$, where $\delta(y, z) = (z - z^2) \cdot \langle \mathbf{1}^{m \cdot n}, \mathbf{y}^{m \cdot n} \rangle - \sum_{j=1}^{m} z^{j+2} \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle$. Append T_1, T_2 to tr.
5. Reprogram $\mathsf{H}(\mathsf{tr}) := x$, then append $\hat{t}, \beta_x, \mu$ to tr.
6. Obtain challenge $w \leftarrow \mathsf{H}(\mathsf{tr})$.
7. Compute $\mathbf{h}' = \mathbf{h}^{\mathbf{y}^{-m \cdot n}}$, $u' = u^w$, and

$$P' = h^{-\mu} \cdot A \cdot S^x \cdot \mathbf{g}^{-z \cdot \mathbf{1}^{m \cdot n}} \cdot (\mathbf{h}')^{z \cdot \mathbf{y}^{m \cdot n}} \cdot \prod_{j=1}^{m} (\mathbf{h}')^{z^{j+1} \cdot \mathbf{2}^n}_{[(j-1)n, jn-1]} \cdot (u')^{\hat{t}}.$$

8. Generate an honest proof $\pi_{\mathsf{BP\text{-}IPA}_{\mathsf{FS}}} \leftarrow \mathcal{P}_{\mathsf{BP\text{-}IPA}_{\mathsf{FS}}}((m \cdot n, \mathbf{g}, \mathbf{h}', u'), P', (\mathbf{l}, \mathbf{r}))$.
9. Output $\pi_{\mathsf{BP\text{-}ARP}_{\mathsf{FS}}} = (A, S, T_1, T_2, \hat{t}, \beta_x, \mu, \pi_{\mathsf{BP\text{-}IPA}_{\mathsf{FS}}})$.

Fig. 12: $\mathsf{BP\text{-}ARP}_{\mathsf{FS}}$ k-ZK simulator

Proof. We present the 2-ZK simulator $\mathcal{S}_{\mathsf{BP\text{-}ARP}_{\mathsf{FS}},x}$ in Figure 12, and argue that its output is identically distributed to the output of an honest prover. All the challenges are chosen randomly as with real proofs. Next, in both real and simulated proofs, the proof elements A, T_1, β_x, μ and the underlying vectors $\mathbf{l}, \mathbf{r}$ are distributed uniformly among their respective domains. The proof elements S, T_2 are then uniquely determined from the previous ones from the verification equations that they must satisfy. Finally, both the scalar $\hat{t}$ and the inner product argument $\pi_{\mathsf{BP\text{-}IPA}}$ is generated deterministically from $\mathbf{l}, \mathbf{r}$; this implies identical distributions for those proof elements as well. □

Finally, we show the 2-UR property of $\mathsf{BP\text{-}ARP}$. This result relies on the fact that $\mathsf{BP\text{-}IPA}_{\mathsf{FS}}$ has computationally unique proofs, given in Lemma 6.2.

Lemma 6.5. $\mathsf{BP\text{-}ARP}_{\mathsf{FS}}$ *satisfies* 2-UR. *In particular, for any adversary* $\mathcal{A}$ *against* 2-UR *of* $\mathsf{BP\text{-}ARP}_{\mathsf{FS}}$*, there exists an adversary* $\mathcal{B}$ *against* $\mathsf{DL\text{-}REL}$ *such that*

$$\mathbf{Adv}^{\mathsf{2\text{-}UR}}_{\mathsf{BP\text{-}ARP}_{\mathsf{FS}}}(\mathcal{A}) \le 2 \cdot \frac{Q(Q-1) + 6(Q+1)\log mn}{|\mathbb{F}| - 1} + 3 \cdot \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},2mn+3}(\mathcal{B}).$$

$\mathcal{B}$ *runs in expected time at most* $O(Q \cdot m^2 \cdot n^2)$ *that of* $\mathcal{A}$*'s runtime.*

Proof. We proceed through a sequence of hybrids. The high-level idea is to analyze different cases for where the two proofs π, π' first differ after the x challenge, and reduce each case to breaking $\mathsf{DL\text{-}REL}$ or the unique proof property of $\mathsf{BP\text{-}IPA}$ (which in turn reduces to breaking $\mathsf{DL\text{-}REL}$).

– Hyb_0 is the game $\mathsf{2\text{-}UR}^{\mathcal{A}}_{\mathsf{BP\text{-}ARP}_{\mathsf{FS}}}$. Recall that in this game, an adversary $\mathcal{A}$ outputs an input $\mathbf{V}$, a challenge $x \in \mathbb{F}^*$, and two proofs π, π' that agrees up to the x challenge, i.e. we have $\pi =$

$(A, S, T_1, T_2, \hat{t}, \beta_x, \mu, \pi_{\mathsf{BP\text{-}IPA_{FS}}})$ and $\pi' = (A, S, T_1, T_2, \hat{t}', \beta'_x, \mu', \pi'_{\mathsf{BP\text{-}IPA_{FS}}})$. $\mathcal{A}$ wins if $\pi \neq \pi'$ and both proofs are accepting with respect to the x challenge that it chose.

– Hyb_1 is the same as Hyb_0, except that we also run $\mathcal{E}_{\mathsf{BP\text{-}IPA_{FS}}}$ on the proofs $\pi_{\mathsf{BP\text{-}IPA_{FS}}}, \pi'_{\mathsf{BP\text{-}IPA_{FS}}}$ to extract witnesses $(\mathbf{l}, \mathbf{r})$ and $(\mathbf{l}', \mathbf{r}')$. Hyb_1 returns 0 if the extractor aborts on either proofs, or $\hat{t} \neq \langle \mathbf{l}, \mathbf{r} \rangle$ or $\hat{t}' \neq \langle \mathbf{l}', \mathbf{r}' \rangle$.

We can see that Hyb_1 is identical to Hyb_0, except when the extractor $\mathcal{E}_{\mathsf{BP\text{-}IPA_{FS}}}$ fails in extracting from either proofs $\pi_{\mathsf{BP\text{-}IPA_{FS}}}, \pi'_{\mathsf{BP\text{-}IPA_{FS}}}$. The probability that this happens is precisely bounded by twice the KS advantage of $\mathsf{BP\text{-}IPA_{FS}}$. Concretely, by Lemma 6.1 there exists an adversary $\mathcal{B}$ against $\mathsf{DL\text{-}REL}$, running in expected time at most $O(Q \cdot m^2 \cdot n^2)$ that of $\mathcal{A}$'s runtime, such that

$$|\Pr[\mathsf{Hyb}_0] - \Pr[\mathsf{Hyb}_1]| \leq 2\frac{Q(Q-1) + 6(Q+1)\log(m \cdot n)}{|\mathbb{F}| - 1} + 2\mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G}, 2mn+3}(\mathcal{B}) .$$

It remains to show that if Hyb_1 returns 1, then there exists an adversary $\mathcal{B}'$ that returns a non-trivial discrete log relation. Adversary $\mathcal{B}'$ is as follows:

- **If $\hat{t} \neq \hat{t}'$ or $\beta_x \neq \beta'_x$:** since both proofs are accepting and are the same up to the x challenge, we have
$$g^{\hat{t}} \cdot h^{\beta_x} = V^{z^2} \cdot g^{\delta(y,z)} \cdot T_1^x \cdot T_2^{x^2} = g^{\hat{t}'} \cdot h^{\beta'_x}.$$
- **If $(\hat{t}, \beta_x) = (\hat{t}', \beta'_x)$ but $\mu \neq \mu'$:** since both proofs $\pi_{\mathsf{BP\text{-}IPA_{FS}}}, \pi'_{\mathsf{BP\text{-}IPA_{FS}}}$ are accepting, we have
$$\begin{aligned}\mathbf{g}^{\mathbf{l}} \cdot \mathbf{h}^{(\mathbf{y}^{-m \cdot n} \circ \mathbf{r})} \cdot h^{\mu} &= A \cdot S^x \cdot \mathbf{g}^{-z \cdot \mathbf{1}^{m \cdot n}} \cdot (\mathbf{h}')^{z \cdot \mathbf{y}^{m \cdot n}} \cdot \prod_{j=1}^{m} (\mathbf{h}')^{z^{j+1} \cdot \mathbf{2}^n}_{[(j-1)n, jn-1]} \cdot u^{w \cdot \hat{t}} \\ &= \mathbf{g}^{\mathbf{l}'} \cdot \mathbf{h}^{(\mathbf{y}^{-m \cdot n} \circ \mathbf{r}')} \cdot h^{\mu'}.\end{aligned}$$
- **If $(\hat{t}, \beta_x, \mu) = (\hat{t}', \beta'_x, \mu')$ but $\pi_{\mathsf{BP\text{-}IPA_{FS}}} \neq \pi'_{\mathsf{BP\text{-}IPA_{FS}}}$:** here, we know that both $\mathsf{BP\text{-}IPA_{FS}}$ proofs are for the same statement P', with extracted witnesses $(\mathbf{l}, \mathbf{r})$, $(\mathbf{l}', \mathbf{r}')$. By Lemma 6.2, we can build an adversary $\mathcal{B}'$ against $\mathsf{DL\text{-}REL}$ given distinct $\mathsf{BP\text{-}IPA_{FS}}$ proofs.

Note that the first two cases above give discrete log relations, and if Hyb_1 returns 1, then $\pi \neq \pi'$, hence at least one of the above cases happens. Putting everything together and unifying $\mathcal{B}, \mathcal{B}'$ we get the desired bound. □

We finally obtain $\mathsf{SIM\text{-}EXT}$ from the previous results and Theorem 3.4.

Theorem 6.6. $\mathsf{BP\text{-}ARP_{FS}}$ *satisfies* $\mathsf{SIM\text{-}EXT}$*. In particular, there exists a simulator-extractor* $\mathcal{E}_{\mathsf{BP\text{-}ARP_{FS}}}$ *such that for any adversary* $\mathcal{P}^*$ *against* $\mathsf{SIM\text{-}EXT}$ *of* $\mathsf{BP\text{-}ARP_{FS}}$*, there exists an adversary* $\mathcal{B}$ *against* $\mathsf{DL\text{-}REL}$ *such that*

$$\mathbf{Adv}^{\mathsf{SIM\text{-}EXT}}_{\mathsf{BP\text{-}ARP_{FS}}}(\mathcal{E}_{\mathsf{BP\text{-}ARP_{FS}}}, \mathcal{P}^*) \leq 4 \cdot \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G}, 2mn+3}(\mathcal{B}) + \frac{3Q(Q-1) + 2(Q+1)\left(m(n+1) + 6\log(mn) + 3\right)}{|\mathbb{F}| - 1} .$$

$\mathcal{B}$ runs in expected time at most $O(Q \cdot m^4 \cdot n^3)$ the runtime of $\mathcal{P}^*$.

6.3 Arithmetic Circuit Satisfiability Proof

We describe $\mathsf{BP\text{-}ACSPf}$ and prove the following theorem in Appendix C.

Theorem 6.7. $\mathsf{BP\text{-}ACSPf_{FS}}$ *satisfies* $\mathsf{SIM\text{-}EXT}$*. Concretely, there exists a simulator-extractor* $\mathcal{E}_{\mathsf{BP\text{-}ACSPf_{FS}}}$ *such that for any adversary* $\mathcal{P}^*$ *against* $\mathsf{SIM\text{-}EXT}$ *of* $\mathsf{BP\text{-}ACSPf_{FS}}$*, there exists an adversary* $\mathcal{B}$ *against* $\mathsf{DL\text{-}REL}$ *such that*

$$\mathbf{Adv}^{\mathsf{SIM\text{-}EXT}}_{\mathsf{BP\text{-}ACSPf_{FS}}}(\mathcal{E}_{\mathsf{BP\text{-}ACSPf_{FS}}}, \mathcal{P}^*) \leq 4 \cdot \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G}, 2n+1}(\mathcal{B}) + \frac{3Q(Q-1) + 2(Q+1)(n + q + 9\log n + 6)}{|\mathbb{F}| - 1} .$$

Here n is the number of multiplication gates, and q is the number of committed inputs. $\mathcal{B}$ runs in expected time at most $O(Q \cdot q \cdot n^3)$ the runtime of $\mathcal{P}^*$.

	Lemma 6.3 ($m = 1$)	[33, **Theorem 4**]
Asymptotic	$O\left(\frac{Q^2+Qn}{\lvert\mathbb{F}\rvert}\right) + \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},2n+3}(\mathcal{A})$ where $\mathbb{E}[t(\mathcal{A})] = O(Q \cdot n^3 \cdot t(\mathcal{P}^*))$	$O\left(\frac{Qn}{\lvert\mathbb{F}\rvert}\right) + \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},2n+3}(\mathcal{A}')$ where $t(\mathcal{A}') = O(Q \cdot n)$
Concrete	≈ 22 bits of security	≈ 164 bits of security

Fig. 13: Comparison of KS advantages, obtained by rewinding (ours) versus AGM [33], for Bulletproofs' single range proof, e.g. BP-ARP with $m = 1$. Here $t(\cdot)$ denotes the running time. For concrete advantage, we take $|\mathbb{F}| \approx 2^{256}$, $n = 64$, $t(\mathcal{P}^*) = 2^{48}$, $Q = 2^{40}$.

7 Quantitative discussion of our SIM-EXT bounds

In this section, we show how to interpret the tightness of our KS bounds for Bulletproofs and Spartan, and compare them with the previous analyses of [30,33] using the Algebraic Group Model (AGM).

For BP-ARP with $m = 1$ (range proof of a single value), we compare our KS bound with the AGM-based bound of [33] in Figure 13. Our approach gives a non-tight bound due to two factors: first, we lose a factor of Q due to rewinding (shown to be somewhat inherent for the similar case of Schnorr signatures [57,26]), and second, our DL-REL adversary is *expected time*, which leads to another "square-root" loss in security [42] (for generic attacks, this means $\mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},2n+3}(\mathcal{A}) \leq \sqrt{t(\mathcal{A})^2/|\mathbb{F}|}$ for an expected poly-time adversary $\mathcal{A}$, compared to $\mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},2n+3}(\mathcal{A}') \leq t(\mathcal{A}')^2/|\mathbb{F}|$ for a *strict* poly time adversary $\mathcal{A}'$). Our concrete KS advantages for Spartan are even lower, due to the bigger tree sizes of Spartan. We leave achieving tighter rewinding-based bounds to future work.

Remark 7.1. We note that Theorem 3.4, which gives SIM-EXT from KS, k-ZK and k-UR, do not rely on the specific method for which these smaller properties are achieved. In particular, plugging in the tight AGM-based analysis [33], our results also imply a *tight* SIM-EXT bound for Bulletproofs—this gives a comparable result to that of Ganesh et al. [30]. For Spartan, a similar tight AGM-based analysis of knowledge soundness (which we leave to future work) would also give a tight SIM-EXT bound.

Acknowledgments

The authors thank Stefano Tessaro, Riad Wahby, Thomas Attema, and the anonymous reviewers at EUROCRYPT 2023 for their helpful comments and suggestions. This research was supported by DARPA under Agreement No. HR00112020022. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Government or DARPA.

References

1. Aleo. https://www.aleo.org/ (2022)
2. Attema, T., Fehr, S., Klooß, M.: Fiat-shamir transformation of multi-round interactive proofs. Cryptology ePrint Archive, Report 2021/1377 (2021), https://eprint.iacr.org/2021/1377
3. Baghery, K., Kohlweiss, M., Siim, J., Volkhov, M.: Another look at extraction and randomization of groth's zk-SNARK. Cryptology ePrint Archive, Report 2020/811 (2020), https://eprint.iacr.org/2020/811
4. Baghery, K., Pindado, Z., Ràfols, C.: Simulation extractable versions of groth's zk-SNARK revisited. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) CANS 20. LNCS, vol. 12579, pp. 453–461. Springer, Heidelberg (Dec 2020)
5. Baghery, K., Sedaghat, M.: Tiramisu: Black-box simulation extractable NIZKs in the updatable CRS model. Cryptology ePrint Archive, Report 2020/474 (2020), https://eprint.iacr.org/2020/474
6. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (May / Jun 2006)

7. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046 (2018), https://eprint.iacr.org/2018/046
8. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE Computer Society Press (May 2014)
9. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 103–128. Springer, Heidelberg (May 2019)
10. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 31–60. Springer, Heidelberg (Oct / Nov 2016)
11. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 626–643. Springer, Heidelberg (Dec 2012)
12. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKs and proof-carrying data. Cryptology ePrint Archive, Report 2012/095 (2012), https://eprint.iacr.org/2012/095
13. Boneh, D., Drake, J., Fisch, B., Gabizon, A.: Halo infinite: Proof-carrying data from additive polynomial commitments. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 649–680. Springer, Heidelberg, Virtual Event (Aug 2021)
14. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg (May 2016)
15. Bowe, S., Grigg, J., Hopwood, D.: Halo: Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021 (2019), https://eprint.iacr.org/2019/1021
16. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society Press (May 2018)
17. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 677–706. Springer, Heidelberg (May 2020)
18. Canetti, R., Sarkar, P., Wang, X.: Triply adaptive UC NIZK. Cryptology ePrint Archive, Report 2020/1212 (2020), https://eprint.iacr.org/2020/1212
19. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Heidelberg (Aug 2006)
20. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.P.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 738–768. Springer, Heidelberg (May 2020)
21. Coda. https://codaprotocol.com/ (2022)
22. Decker, C., Wattenhofer, R.: Bitcoin transaction malleability and MtGox. In: Kutylowski, M., Vaidya, J. (eds.) ESORICS 2014, Part II. LNCS, vol. 8713, pp. 313–326. Springer, Heidelberg (Sep 2014)
23. Don, J., Fehr, S., Majenz, C.: The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 602–631. Springer, Heidelberg (Aug 2020)
24. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat-Shamir transform. In: Galbraith, S.D., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 60–79. Springer, Heidelberg (Dec 2012)
25. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (Aug 1987)
26. Fleischhacker, N., Jager, T., Schröder, D.: On tight security proofs for Schnorr signatures. Journal of Cryptology 32(2), 566–599 (Apr 2019)
27. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Heidelberg (Aug 2018)
28. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019), https://eprint.iacr.org/2019/953
29. Ganesh, C., Khoshakhlagh, H., Kohlweiss, M., Nitulescu, A., Zając, M.: What makes fiat–shamir zksnarks (updatable srs) simulation extractable? In: SCN (2022)
30. Ganesh, C., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 397–426. Springer, Heidelberg (May / Jun 2022)

31. Ganesh, C., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Fiat-shamir bulletproofs are non-malleable (in the random oracle model). Cryptology ePrint Archive (2023)
32. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (May 2013)
33. Ghoshal, A., Tessaro, S.: Tight state-restoration soundness in the algebraic group model. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 64–93. Springer, Heidelberg, Virtual Event (Aug 2021)
34. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th ACM STOC. pp. 291–304. ACM Press (May 1985)
35. Golovnev, A., Lee, J., Setty, S., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and post-quantum SNARKs for R1CS. Cryptology ePrint Archive, Report 2021/1043 (2021), https://eprint.iacr.org/2021/1043
36. Grin: a minimal implementation of mimblewimble. https://github.com/mimblewimble/grin (2022)
37. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (Dec 2006)
38. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (May 2016)
39. Groth, J., Maller, M.: Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 581–612. Springer, Heidelberg (Aug 2017)
40. Grubbs, P., Arun, A., Zhang, Y., Bonneau, J., Walfish, M.: Zero-knowledge middleboxes. In: Butler, K.R.B., Thomas, K. (eds.) USENIX Security 2022. pp. 4255–4272. USENIX Association (Aug 2022)
41. Haines, T., Lewis, S.J., Pereira, O., Teague, V.: How not to prove your election outcome. In: 2020 IEEE Symposium on Security and Privacy. pp. 644–660. IEEE Computer Society Press (May 2020)
42. Jaeger, J., Tessaro, S.: Expected-time cryptography: Generic techniques and applications to concrete soundness. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part III. LNCS, vol. 12552, pp. 414–443. Springer, Heidelberg (Nov 2020)
43. Jain, A., Pandey, O.: Non-malleable zero knowledge: Black-box constructions and definitional relationships. In: Abdalla, M., Prisco, R.D. (eds.) SCN 14. LNCS, vol. 8642, pp. 435–454. Springer, Heidelberg (Sep 2014)
44. Kothapalli, A., Setty, S., Tzialla, I.: Nova: Recursive zero-knowledge arguments from folding schemes. Cryptology ePrint Archive, Report 2021/370 (2021), https://eprint.iacr.org/2021/370
45. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2111–2128. ACM Press (Nov 2019)
46. Bulletproofs+ in monero. https://www.getmonero.org/2020/12/24/Bulletproofs+-in-Monero.html (2020)
47. Mt.Gox press release. https://web.archive.org/web/20140214041924/https://www.mtgox.com/press_release_20140210.html (2014)
48. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy. pp. 238–252. IEEE Computer Society Press (May 2013)
49. Partition matroid. https://en.wikipedia.org/wiki/Partition_matroid (2022)
50. Pass, R.: On deniability in the common reference string and random oracle model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (Aug 2003)
51. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 533–542. ACM Press (May 2005)
52. Polygon. https://polygon.technology/ (2022)
53. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS. pp. 543–553. IEEE Computer Society Press (Oct 1999)
54. Scroll. https://scroll.io/ (2022)
55. Setty, S.: Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 704–737. Springer, Heidelberg (Aug 2020)
56. Setty, S., Lee, J.: Quarks: Quadruple-efficient transparent zkSNARKs. Cryptology ePrint Archive, Report 2020/1275 (2020), https://eprint.iacr.org/2020/1275
57. Seurin, Y.: On the exact security of Schnorr-type signatures in the random oracle model. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 554–571. Springer, Heidelberg (Apr 2012)
58. Spartan: High-speed zksnarks without trusted setup. https://github.com/microsoft/Spartan (2022)
59. Starkware. https://starkware.co/ (2022)

60. Wahby, R.S., Tzialla, I., shelat, a., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: 2018 IEEE Symposium on Security and Privacy. pp. 926–943. IEEE Computer Society Press (May 2018)
61. Wee, H.: Zero knowledge in the random oracle model, revisited. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 417–434. Springer, Heidelberg (Dec 2009)
62. Wikström, D.: Special soundness revisited. Cryptology ePrint Archive, Report 2018/1157 (2018), https://eprint.iacr.org/2018/1157
63. Wikström, D.: Special soundness in the random oracle model. Cryptology ePrint Archive, Report 2021/1265 (2021), https://eprint.iacr.org/2021/1265
64. Xie, T., Zhang, Y., Song, D.: Orion: Zero knowledge proof with linear prover time. Cryptology ePrint Archive, Report 2022/1010 (2022), https://eprint.iacr.org/2022/1010
65. Zhang, F., Maram, D., Malvai, H., Goldfeder, S., Juels, A.: DECO: Liberating web data using decentralized oracles for TLS. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1919–1938. ACM Press (Nov 2020)
66. ZKProofs Standards. https://zkproof.org/ (2022)

A Comparison with Wikström's Tree-Builder

In this section, we give comparisons between our tree-builder with the one by Wikström in [63]. We begin by stating the definition of a *matroid* and its associated *matroid predicate*.

Definition A.1 (Matroid). *A* matroid *is a pair* $(S, \mathcal{I})$ *of a* ground set S *and a set* $\mathcal{I} \subset 2^S$ *of* independent sets *such that:*

1. $\mathcal{I}$ *is non-empty,*
2. *If* $A \in \mathcal{I}$ *and* $B \subset A$*, then* $B \in \mathcal{I}$*,*
3. *If* $A, B \in \mathcal{I}$ *and* $|A| > |B|$*, then there exists an element* $a \in A \setminus B$ *such that* $\{a\} \cup B \in \mathcal{I}$*.*

A basis *of* $\mathbb{M}$ *is a set* $B \in \mathcal{I}$ *such that* $B \cup \{x\} \notin \mathcal{I}$ *for every* $x \in S \setminus B$*. The* rank *of* $\mathbb{M}$ *is the unique cardinality of each basis in* $\mathcal{I}$*.*

Definition A.2. *Given a matroid* $\mathbb{M} = (S, \mathcal{I})$ *and a subset* $A \subset S$*, the* submatroid *induced by* A *is the pair* $(A, \mathcal{I} \cap 2^A)$*, and its* rank $\mathsf{rank}(A)$ *is the rank of the submatroid induced by* A*. The* span *of* A *is defined by* $\mathsf{Span}(A) = \{x \in S \mid \mathsf{rank}(A \cup \{x\}) = \mathsf{rank}(A)\}$*.*

A subset $A \subset S$ *is a* flat *of* $\mathbb{M}$ *if* $\mathsf{Span}(A) = A$*. It is a* hyperplane *if it is a flat of rank* $\mathsf{rank}(\mathbb{M}) - 1$*. The* subdensity *of a matroid* $\mathbb{M}$ *is defined to be* $\omega_{\mathbb{M}} = \max_A \frac{|A|}{|S|}$*, where* A *ranges over all hyperplanes of* $\mathbb{M}$*.*

Definition A.3 (Matroid Predicate). *Let* $\mathbb{M} = (\mathsf{Ch}, \mathcal{I})$ *be a matroid of rank* n *with underlying set* Ch*. We define the corresponding* matroid predicate $\phi_{\mathbb{M}} : \mathsf{Ch}^n \to \{0,1\}$ *to be* $\phi_{\mathbb{M}}(c_1, \ldots, c_n) = 1$ *if and only if* $\{c_1, \ldots, c_n\}$ *is a basis of* $\mathbb{M}$*.*

When $\mathbb{M}$ is the *uniform matroid*, with the independent sets being all subsets of size at most n, we recover the usual predicate of distinct elements, and the corresponding subdensity is $\omega_{\mathbb{M}} = \frac{n-1}{|\mathsf{Ch}|}$. The partition predicates $\phi_{\mathscr{P},n}$ we consider in this work correspond to *partition matroids* [49], with subdensity $\omega_{\mathbb{M}} = \frac{|\mathsf{Ch}^{(1)}| + \cdots + |\mathsf{Ch}^{(n-1)}|}{|\mathsf{Ch}|}$, where without loss of generality we assume $\mathsf{Ch}^{(1)}, \ldots, \mathsf{Ch}^{(n-1)}$ are the blocks of the largest sizes.

Theorem A.4 ([63], adapted). *Let* Π *be a* $(2r+1)$*-message public-coin interactive argument with challenge spaces* $\mathsf{Ch}_1, \ldots, \mathsf{Ch}_r$*. Consider matroids* $\mathbb{M}_1 = (\mathsf{Ch}_1, \mathcal{I}_1), \ldots, \mathbb{M}_r = (\mathsf{Ch}_r, \mathcal{I}_r)$ *of rank* $\mathbf{n} = (n_1, \ldots, n_r)$*, respectively. Denote by* $\boldsymbol{\phi}_{\mathbb{M}}$ *the corresponding matroid predicate.*

For any $\nu_1 \geq \cdots \geq \nu_{r-1} > 1$*, there exists a probabilistic algorithm* $\mathcal{TB}_{\mathsf{Wik}}$ *for* Π_{FS} *such that, given oracle access to a malicious prover* $\mathcal{P}^*$ *for* Π_{FS} *making at most* Q *RO queries with success probability* $\epsilon(\mathcal{P}^*) := \Pr[\mathsf{KS}_{0,\Pi_{\mathsf{FS}}}^{\mathcal{P}^*}]$*, has the following characteristics:*

– success probability: $\Pr\left[\mathsf{TreeBuild}_{\Pi_{\mathsf{FS}},(\boldsymbol{\phi}_{\mathbb{M}},\mathbf{n})}^{\mathcal{TB}_{\mathsf{Wik}},\mathcal{P}^*}\right] \geq \epsilon(\mathcal{P}^*) - Q \sum_{i=1}^{r} \omega_{\mathbb{M}_i} \prod_{j=1}^{i-1} \nu_j$,

– expected number of queries: *at most* $Q \cdot \prod_{j=1}^{r} n_j \cdot c_0$.

Here c_0 *is a constant equal to*

$$c_0 = 3^{r+1} \cdot \frac{\nu_{r-1}^2}{(\nu_{r-1}-1)^2} \cdot \prod_{i=1}^{r-1} \frac{\nu_i^2}{\nu_i - 1} \cdot \min_{k_i \in (0,1)} \frac{k_i}{h_{n_i}^{CG}(k_i)},$$

where $h_{n_i}^{CG}(k_i)$ *is the* head bound *of the* compounded geometric (CG) *distribution,*[7] *i.e.* $\Pr_{X\sim CG(n,\Delta)}[X < k \cdot \mathbb{E}[CG(n,\Delta)]] \geq h_n^{CG}(k)$ *for every* $\Delta \in [0,1]$.

Comparison with Attema et al's tree-builder $\mathcal{TB}_{\mathsf{AFK}}$*.* Note that in order to match the success probability of $\mathcal{TB}_{\mathsf{AFK}}$, we need to set $\nu_1, \ldots, \nu_{r-1} \to 1$, which would make the expected number of queries goes to infinity. For values of ν_i that are bounded from 1, say $\nu_1 = \cdots = \nu_{r-1} = 2$ as suggested in [63], and for logarithmic-round protocols such as Spartan and Bulletproofs, the success probability is worse by a factor of $2^{O(r)} = n^{O(1)}$, where n is the size of the instance. Accordingly, the expected number of queries is also worse by a factor of $O(3^r \cdot 4^r) = n^{O(1)}$ as well.

Comparison with our tree builder $\mathcal{TB}$*.* Since we build on $\mathcal{TB}_{\mathsf{AFK}}$, we do not suffer from large constants as mentioned above. However, our tree builder suffers from a *quadratic* dependence on Q, which for large values of Q would become worse than the bound for $\mathcal{TB}_{\mathsf{Wik}}$. Nevertheless, our tree builder has the same expected running time as $\mathcal{TB}_{\mathsf{AFK}}$, which is faster than that of $\mathcal{TB}_{\mathsf{Wik}}$. Finally, our tree builder can only handle *partition predicates*, and some relevant matroid predicates do not lie in this class, i.e. linear independence of more than 2 challenges.

B Omitted details for Spartan

B.1 Descriptions of Subprotocols

Transcript contents. A transcript for Spartan-NIZK (respectively Spartan-SNARK) consists of the following:

- a commitment $C_{\widetilde{w}}$ to the multilinear extension of the witness,
- the verifier randomness $\tau \in \mathbb{F}^\mu$,
- μ sumcheck rounds, each consisting of two commitments, two verifier challenges, and a transcript of DotProdPf (for a total of 3 inner rounds),
- a commitment C_{e_x} to the claimed evaluation of $\mathcal{G}_{\mathsf{io},\tau}$
- four commitments, one to each of the evaluations v_A, v_B, v_C and one to the product $v_A \cdot v_B$,
- a transcript of ProdPf, to verify the product commitment
- a transcript of EqPf and OpenPf to verify the claimed relationship (line 6 of Figure 7) holds in the exponent,
- three verifier challenges $r_A, r_B, r_C \in \mathbb{F}$,
- μ sumcheck rounds to verify $\mathcal{H}_{r_x}(Y)$ has the claimed sum over the hypercube,
- a commitment C_{e_y} to the evaluation $\mathcal{H}_{r_x}(r_y)$,
- a commitment C_{v_w} to the evaluation of $\widetilde{w}$ at $(r_y)_{[1:]}$,
- a transcript of $\mathsf{PC}_{\mathsf{Multi}}.\mathsf{Open}$ showing that the commitment C_{v_w} contains the correct evaluation,
- for Spartan-SNARK only, a transcript of $\mathsf{PC}_{\mathsf{SparseMulti}}.\mathsf{Open}$ proving correct evaluations of $\widetilde{A}(r_x, r_y) = v_1$, $\widetilde{B}(r_x, r_y) = v_2$, $\widetilde{C}(r_x, r_y) = v_3$,
- a transcript of EqPf showing that the required relationship (line 11) holds in the exponent of the two commitments.

Using the listing above, we can compute the number of rounds of Spartan-NIZK to be $r = 1 + 3\mu + (2 + 2 + 2) + 1 + 3\mu + (\mu + 1) + 2 = 7\mu + 11$.

Description of Σ*-protocols.* We present the Σ-protocols used by Spartan in Figure 14. We note that the equality proof EqPf only proves that the two commitments are to the same equal value, but we cannot

Opening Relation.

$$\mathcal{R}_{\mathsf{Open}} = \left\{ \begin{matrix} ((g,h), C, (x,r)) : \\ C = g^x \cdot h^r \end{matrix} \right\}$$

$\mathcal{P} \to \mathcal{V} :\ \alpha \leftarrow g^{t_1} \cdot h^{t_2}$, with $t_1, t_2 \xleftarrow{\$} \mathbb{F}$.
$\mathcal{V} \to \mathcal{P} :\ c \xleftarrow{\$} \mathbb{F}^*$.
$\mathcal{P} \to \mathcal{V} :\ z_1 \leftarrow xc + t_1,\ z_2 \leftarrow rc + t_2$.
$\mathcal{V} :$ checks that $g^{z_1} \cdot h^{z_2} \stackrel{?}{=} C^c \cdot \alpha$.

(a) Proof of Opening OpenPf

Equality Relation.

$$\mathcal{R}_{\mathsf{Eq}} = \left\{ \begin{matrix} ((g,h), (C_1, C_2), (r)) : \\ C_1 = C_2 \cdot h^r \end{matrix} \right\}$$

$\mathcal{P} \to \mathcal{V} :\ \alpha \leftarrow h^t$, where $t \xleftarrow{\$} \mathbb{F}$.
$\mathcal{V} \to \mathcal{P} :\ c \xleftarrow{\$} \mathbb{F}^*$.
$\mathcal{P} \to \mathcal{V} :\ z \leftarrow c \cdot (r_1 - r_2) + t$.
$\mathcal{V} :$ checks that $h^z \stackrel{?}{=} (C_1/C_2)^c \cdot \alpha$.

(b) Proof of Equality EqPf

Product Relation.

$$\mathcal{R}_{\mathsf{Prod}} = \left\{ \begin{matrix} ((g,h), (X,Y,Z), (x,y,r_x,r_y,r_z)) : \\ X = g^x \cdot h^{r_x}, Y = g^y \cdot h^{r_y}, Z = g^{x \cdot y} \cdot h^{r_z} \end{matrix} \right\}$$

$\mathcal{P} \to \mathcal{V} :\ \alpha \leftarrow g^{b_1} \cdot h^{b_2},\ \beta \leftarrow g^{b_3} \cdot h^{b_4},\ \gamma \leftarrow X^{b_3} \cdot h^{b_5}$, where $b_1, \ldots, b_5 \xleftarrow{\$} \mathbb{F}$.
$\mathcal{V} \to \mathcal{P} :\ c \xleftarrow{\$} \mathbb{F}^*$.
$$\mathcal{P} \to \mathcal{V} :\ \begin{cases} z_1 \leftarrow b_1 + c \cdot x,\ z_2 \leftarrow b_2 + c \cdot r_x, \\ z_3 \leftarrow b_3 + c \cdot y,\ z_4 \leftarrow b_4 + c \cdot r_y,\ z_5 = b_5 + c \cdot (r_z - r_x y). \end{cases}$$
$\mathcal{V} :$ checks that $\alpha \cdot X^c \stackrel{?}{=} g^{z_1} \cdot h^{z_2}$, $\beta \cdot Y^c \stackrel{?}{=} g^{z_3} \cdot h^{z_4}$, $\delta \cdot Z^c \stackrel{?}{=} X^{z_3} \cdot h^{z_5}$.

(c) Proof of Product ProdPf

Dot Product Relation.

$$\mathcal{R}_{\mathsf{DotProd}} = \left\{ \begin{matrix} ((n, g, \mathbf{g}, h), (X, Y, \mathbf{a}), (\mathbf{x}, y, r_{\mathbf{x}}, r_y)) : \\ X = \mathbf{g}^{\mathbf{x}} \cdot h^{r_{\mathbf{x}}}, Y = g^y \cdot h^{r_y}, y = \langle \mathbf{x}, \mathbf{a} \rangle \end{matrix} \right\}$$

$\mathcal{P} \to \mathcal{V} :\ \beta \leftarrow \mathbf{g}^{\mathbf{b}} \cdot h^{r_\beta}$, $\delta \leftarrow g^{\langle \mathbf{a}, \mathbf{b} \rangle} \cdot h^{r_\delta}$, where $\mathbf{b} \xleftarrow{\$} \mathbb{F}^n$ and $r_\beta, r_\delta \xleftarrow{\$} \mathbb{F}$.
$\mathcal{V} \to \mathcal{P} :\ c \xleftarrow{\$} \mathbb{F}^*$.
$\mathcal{P} \to \mathcal{V} :\ \mathbf{z} \leftarrow c \cdot \mathbf{x} + \mathbf{b}$, $z_\beta \leftarrow c \cdot r_{\mathbf{x}} + r_\beta$, $z_\delta \leftarrow c \cdot r_y + r_\delta$.
$\mathcal{V} :$ checks that $X^c \cdot \beta \stackrel{?}{=} \mathbf{g}^{\mathbf{z}} \cdot h^{z_\beta}$ and $Y^c \cdot \delta \stackrel{?}{=} g^{\langle \mathbf{a}, \mathbf{z} \rangle} \cdot h^{z_\delta}$.

(d) Proof of Dot Product $\mathsf{DotProdPf}$

Fig. 14: Σ-protocols used in $\mathsf{Spartan}$. In $\mathsf{Spartan}$, the same group elements g, h are used across these protocols.

extract the value. Therefore, it is necessary for us to invoke OpenPf in line 6 of Figure 7 to extract the underlying value v_C from its commitment C_{v_C}.

Description of $\mathsf{PC}_{\mathsf{Multi}}$. We give some intuition on the multilinear polynomial commitment $\mathsf{PC}_{\mathsf{Multi}}$ (see Figure 15) used in $\mathsf{Spartan}$, which was first introduced in [60]. Recall that to evaluate a multilinear polynomial $p(X_1, \ldots, X_\mu)$ given in its evaluation form $(p(0), \ldots, p(2^\mu - 1))$ at a point $x = (x_1, \ldots, x_\mu)$, we use the following formula:

$$\begin{aligned} p(x) &= \sum_{k \in \{0,1\}^\mu} p(k) \cdot \prod_{i=1}^{\mu} \widetilde{\mathrm{eq}}(x_i, k_i) \\ &= \sum_{k \in \{0,1\}^{\mu/2}} \sum_{\ell \in \{0,1\}^{\mu/2}} \prod_{i=1}^{\mu/2} \widetilde{\mathrm{eq}}(x_i, k_i) \cdot p(k + 2^{\mu/2} \cdot \ell) \cdot \prod_{i=1}^{\mu/2} \widetilde{\mathrm{eq}}(x_{\mu/2+i}, \ell_i). \end{aligned}$$

If we denote

$$L = \left(\prod_{i=1}^{\mu/2} \widetilde{\mathrm{eq}}(x_i, k_i) \right)_{k \in \{0,1\}^{\mu/2}}, \quad R = \left(\prod_{i=1}^{\mu/2} \widetilde{\mathrm{eq}}(x_{\mu/2+i}, \ell_i) \right)_{\ell \in \{0,1\}^{\mu/2}}, \quad T = \Big(p(k + 2^{\mu/2}\ell) \Big)_{k,\ell \in \{0,1\}^{\mu/2}},$$

then we have $p(x) = L \cdot T \cdot R^T$. In the commitment phase of $\mathsf{PC}_{\mathsf{Multi}}$, the prover $\mathcal{P}$ commit to each row of T, giving a commitment $C_T = (C_0, \ldots, C_{2^{\mu/2}-1})$. In the opening phase, the verifier $\mathcal{V}$ can compute a commitment to $L \cdot T$ as $C_{L \cdot T} = \prod_{k=0}^{2^{\mu/2}-1} C_k^{L_k}$. We now reduce to proving an inner product argument that $(L \cdot T) \cdot R^T = p(x)$, with $L \cdot T$ given by a commitment, and R is public. This is handled by a logarithmic-sized dot product proof $\mathsf{LogDotProdPf}$, which shares many similarities with the inner product argument $\mathsf{BP\text{-}IPA}$ of Bulletproofs, but differs in the fact that it is zero-knowledge (while $\mathsf{BP\text{-}IPA}$ is not).

Description of $\mathsf{PC}_{\mathsf{SparseMulti}}$. We give the description of the sparse multilinear polynomial commitment scheme $\mathsf{PC}_{\mathsf{SparseMulti}}$ in Figure 16, and provide the high-level intuition here. Recall that from Equation 1, given a matrix $M \in \mathbb{F}^{m \times m}$ with n non-zero entries, we can calculate the value of $\widetilde{M}(r_{\mathsf{row}}, r_{\mathsf{col}})$ for any $r_{\mathsf{row}}, r_{\mathsf{col}} \in \mathbb{F}^\mu$ in $O(n)$ operations:

$$\widetilde{M}(X, Y) = \sum_{j=1}^{n} \mathsf{val}_i \cdot \widetilde{\mathrm{eq}}(\mathsf{row}_j, X) \cdot \widetilde{\mathrm{eq}}(\mathsf{col}_j, Y).$$

To achieve a sublinear verifier, the prover in $\mathsf{Spartan\text{-}SNARK}$ will use a *secondary* proof system Π for circuit satisfiability, then compute a proof of correct evaluation for $\widetilde{M}(r_{\mathsf{row}}, r_{\mathsf{col}})$. The circuit $\mathsf{CircuitSparseEval}$ for correct evaluation is described in Figure 17, and utilizes an auxiliary memory-checking procedure $\mathsf{MemoryInTheHead}$ to guarantee soundness.

In the commitment phase for $\mathsf{PC}_{\mathsf{SparseMulti}}$, we will commit to the dense representation $(\mathsf{row}, \mathsf{col}, \mathsf{val})$ of M, along with the auxiliary *time-stamp sequences* $\mathsf{read\text{-}ts}_X, \mathsf{write\text{-}ts}_X, \mathsf{audit\text{-}ts}_X$ for $X \in \{\mathsf{row}, \mathsf{col}\}$. These sequences help ensure the correct evaluation of M according to Equation 1. In the evaluation phase, the prover $\mathcal{P}$ will commit to vectors of evaluations

$$e_{\mathsf{row}} = (\widetilde{\mathrm{eq}}(\mathsf{row}_0, r_{\mathsf{row}}), \ldots, \widetilde{\mathrm{eq}}(\mathsf{row}_{n-1}, r_{\mathsf{row}})),$$

and similarly for e_{col}. We can then re-write the evaluation formula as $\widetilde{M}(r_{\mathsf{row}}, r_{\mathsf{col}}) = \sum_{j=1}^{n} \mathsf{val}_j \cdot (e_{\mathsf{row}})_j \cdot (e_{\mathsf{col}})_j$. The evaluation circuit $\mathsf{CircuitSparseEval}$ would then compute $\widetilde{M}(r_{\mathsf{row}}, r_{\mathsf{col}})$ according to this formula, and additionally prove that e_{row} is consistent with row (similarly e_{col} is consistent with col) via asserting the equality of two multi-sets

$$\mathsf{ISet}_{\mathsf{row}} \sqcup \mathsf{WSet}_{\mathsf{row}} = \mathsf{RSet}_{\mathsf{row}} \sqcup \mathsf{ASet}_{\mathsf{row}},$$

[7] We refer to [62] for more details on the distribution.

Description of $\mathsf{PC}_{\mathsf{Multi}}$:

- $\mathsf{Setup}(\mu, \mathsf{pp}_{\mathcal{G}})$: abort if μ is odd. Parse $\mathsf{pp}_{\mathcal{G}}$ as a group description $(\mathbb{G}, \mathbb{F})$. Sample $g_1, \ldots, g_{\mu/2}, h \xleftarrow{\$} \mathbb{G}$ and output $\mathsf{pp} = (\mathbb{F}, \mathbb{G}, g, g_1, \ldots, g_{\mu/2}, h)$.

- $\mathsf{Commit}(\mathsf{pp}, p(X_1, \ldots, X_\mu); \boldsymbol{\omega})$: parse the evaluations $\{p(0), \ldots, p(2^\mu - 1)\}$ as a $2^{\mu/2} \times 2^{\mu/2}$ matrix $\mathbf{T}$ in column-major order, i.e. $T_{i,j} = p(i + 2^{\mu/2} j)$. For each $i \in [2^{\mu/2}]$, sample $\omega_i \xleftarrow{\$} \mathbb{F}$ and compute $C_i = \prod_{j=1}^{2^{\mu/2}} g_j^{T_{i,j}} h^{\omega_i}$. Let $\boldsymbol{\omega} = (\omega_i)_{i \in [2^{\mu/2}]}$.
Output $\mathbf{C} = (C_1, \ldots, C_{2^{\mu/2}})$.

- $\mathsf{Open}\langle \mathcal{P}(p, \boldsymbol{\omega}, v, \omega_v), \mathcal{V} \rangle(\mathsf{pp}, \mathbf{C}, x, C_v)$: given a commitment C_v as public input, with a random evaluation point $x \xleftarrow{\$} \mathbb{F}^\mu$ sent as challenge by $\mathcal{V}$ in previous rounds,

1. Let $\widetilde{\mathrm{eq}}_L(Y) = \prod_{i=1}^{\mu/2} \widetilde{\mathrm{eq}}(x_i, Y_i)$ and $\widetilde{\mathrm{eq}}_R(Y) = \prod_{i=\mu/2+1}^{\mu} \widetilde{\mathrm{eq}}(x_i, Y_i)$.
2. $\mathcal{P}, \mathcal{V}$ both compute $P = C_v \cdot \prod_{k \in \{0,1\}^{\mu/2}} C_k^{\widetilde{\mathrm{eq}}_L(k)}$ and $\mathbf{r} = (\widetilde{\mathrm{eq}}_R(k))_{k \in \{0,1\}^{\mu/2}}$.
3. $\mathcal{P}$ also computes
$$\mathbf{l} = \left(\textstyle\sum_{k \in \{0,1\}^{\mu/2}} T_{k,j} \cdot \widetilde{\mathrm{eq}}_L(k)\right)_{j \in \{0,1\}^{\mu/2}}, \quad \omega_P = \omega_v + \textstyle\sum_{k \in \{0,1\}^{\mu/2}} \omega_k \cdot \widetilde{\mathrm{eq}}_L(k).$$
4. $\mathcal{P}, \mathcal{V}$ engage in $\mathsf{LogDotProdPf}((2^{\mu/2}, g, \mathbf{g}, h), (P, \mathbf{r}), (\mathbf{l}, v, \omega_P)$.

Logarithmic Dot Product Proof. For $n = 2^k$,
$$\mathcal{R}_{\mathsf{LogDotProd}} = \left\{((n, g, \mathbf{g}, h), (P, \mathbf{a}), (\mathbf{x}, y, r_P) : P = g^y \cdot \mathbf{g}^{\mathbf{x}} \cdot h^{r_P}, y = \langle \mathbf{x}, \mathbf{a} \rangle \right\}.$$
1. Set $n_0 \leftarrow n, \mathbf{g}^{(0)} \leftarrow \mathbf{g}, P^{(0)} \leftarrow P, \mathbf{a}^{(0)} \leftarrow \mathbf{a}, \mathbf{x}^{(0)} \leftarrow \mathbf{x}, y^{(0)} \leftarrow y, r_P^{(0)} \leftarrow r_P$.
For $i = 1, \ldots, k$:
(a) $\mathcal{P}$ computes $y_L^{(i)} \leftarrow \left\langle \mathbf{x}_{[:n_i]}^{(i-1)}, \mathbf{a}_{[n_i:]}^{(i-1)} \right\rangle$, $y_R^{(i)} \leftarrow \left\langle \mathbf{x}_{[n_i:]}^{(i-1)}, \mathbf{a}_{[:n_i]}^{(i-1)} \right\rangle$, then samples $r_L^{(i)}, r_R^{(i)} \xleftarrow{\$} \mathbb{F}$ and sends
$$L_i \leftarrow g^{y_L^{(i)}} \cdot \left(\mathbf{g}^{(i-1)}\right)^{\mathbf{x}_{[:n_i]}^{(i-1)}} \cdot h^{r_L^{(i)}}, \; R_i \leftarrow g^{y_R^{(i)}} \cdot \left(\mathbf{g}^{(i-1)}\right)^{\mathbf{x}_{[n_i:]}^{(i-1)}} \cdot h^{r_R^{(i)}}.$$
(b) $\mathcal{V}$ sends challenge $c_i \xleftarrow{\$} \mathbb{F}$.
(c) $\mathcal{P}$ and $\mathcal{V}$ both compute $P^{(i)} \leftarrow L_i^{c_i^2} \cdot P^{(i-1)} \cdot R_i^{c_i^{-2}}$ and
$$\mathbf{a}^{(i)} \leftarrow c_i^{-1} \cdot \mathbf{a}_{[:n_i]}^{(i-1)} + c_i \cdot \mathbf{a}_{[n_i:]}^{(i-1)}, \quad \mathbf{g}^{(i)} \leftarrow \left(\mathbf{g}_{[:n_i]}^{(i-1)}\right)^{c_i^{-1}} \circ \left(\mathbf{g}_{[n_i:]}^{(i-1)}\right)^{c_i}.$$
(d) $\mathcal{P}$ computes $\mathbf{x}^{(i)} \leftarrow c_i \cdot \mathbf{x}_{[:n_i]}^{(i-1)} + c_i^{-1} \cdot \mathbf{x}_{[n_i:]}^{(i-1)}$ and
$$y^{(i)} \leftarrow c_i^2 \cdot y_L^{(i)} + y^{(i-1)} + c_i^{-2} \cdot y_R^{(i)}, \quad r_P^{(i)} \leftarrow c_i^2 \cdot r_L^{(i)} + r_P^{(i-1)} + c_i^{-2} \cdot r_R^{(i)}.$$
2. Set $\hat{g} \leftarrow \mathbf{g}^{(k)}, \hat{P} \leftarrow P^{(k)}, \hat{a} \leftarrow \mathbf{a}^{(k)}, \hat{x} \leftarrow \mathbf{x}^{(k)}, \hat{y} \leftarrow y^{(k)}, \hat{r_P} \leftarrow r_P^{(k)}$.
$\mathcal{P}$ samples $d, r_\beta, r_\delta \xleftarrow{\$} \mathbb{F}$ and sends $\beta \leftarrow g^d \cdot h^{r_\beta}$, $\delta \leftarrow \hat{g}^d \cdot h^{r_\delta}$.
3. $\mathcal{V}$ sends challenge $c \xleftarrow{\$} \mathbb{F}$.
4. $\mathcal{P}$ sends $z_1 \leftarrow d + c \cdot \hat{y}$ and $z_2 \leftarrow \hat{a} \cdot (c \cdot \hat{r_P} + r_\beta) + r_\delta$.
5. $\mathcal{V}$ checks that $\left(\hat{P}^c \cdot \beta\right)^{\hat{a}} \cdot \delta \stackrel{?}{=} \left(\hat{g} \cdot g^{\hat{a}}\right)^{z_1} \cdot h^{z_2}$.

Fig. 15: Multilinear Polynomial Commitment Scheme $\mathsf{PC}_{\mathsf{Multi}}$

where

$$\begin{aligned}
\mathsf{ISet}_{\mathsf{row}} &= \{(i, (\mathsf{mem}_{\mathsf{row}})_i, 0)\}_{i\in[0,m-1]},\\
\mathsf{RSet}_{\mathsf{row}} &= \{(\mathsf{row}_i, (e_{\mathsf{row}})_i, (\mathsf{read\text{-}ts}_{\mathsf{row}})_i)\}_{i\in[0,n-1]},\\
\mathsf{WSet}_{\mathsf{row}} &= \{(\mathsf{row}_i, (e_{\mathsf{row}})_i, (\mathsf{write\text{-}ts}_{\mathsf{row}})_i)\}_{i\in[0,n-1]},\\
\mathsf{ASet}_{\mathsf{row}} &= \{(i, (\mathsf{mem}_{\mathsf{row}})_i, (\mathsf{audit\text{-}ts}_{\mathsf{row}})_i)\}_{i\in[0,m-1]}.
\end{aligned}$$

We can verify this equivalence by showing that their hashes are the same; in particular, [55] proposes to use the following algebraic hash functions (defined for any $\gamma_1, \gamma_2 \in \mathbb{F}$ and $\ell \in \mathbb{N}$):

1. $h_{\gamma_1} : \mathbb{F}^3 \to \mathbb{F}$ defined by $h_{\gamma_1}(a, v, t) = a \cdot \gamma^2 + v \cdot \gamma + t$.
2. $\mathcal{H}_{\gamma_1,\gamma_2} : (\mathbb{F}^\ell)^3 \to \mathbb{F}$ defined by $\mathcal{H}_{\gamma_1,\gamma_2}(A, V, T) = \prod_{i=1}^{\ell}(h_{\gamma_1}(A_i, V_i, T_i) - \gamma_2)$.

The soundness loss for these hash functions, as they are used in CircuitSparseEval, are summarized in Lemma B.2.

Optimizations for $\mathsf{PC}_{\mathsf{SparseMulti}}$. In the reference implementation [58], the proof system Π used to prove CircuitSparseEval is a non-zero-knowledge variant of Hyrax [60], with optimizations for batching three evaluation checks for $\widetilde{A}, \widetilde{B}, \widetilde{C}$ into one. Due to these optimizations, the commitment and opening procedure of $\mathsf{PC}_{\mathsf{SparseMulti}}$ are changed to accomodate three matrices A, B, C at once. In particular, in the commitment phase, we concatenate the vectors

$$\mathsf{ops}_{\mathsf{batch}} = (\mathsf{row}, \mathsf{read\text{-}ts}_{\mathsf{row}}, \mathsf{write\text{-}ts}_{\mathsf{row}}, \mathsf{col}, \mathsf{read\text{-}ts}_{\mathsf{col}}, \mathsf{write\text{-}ts}_{\mathsf{col}})$$

for three matrices A, B, C, and produce a single polynomial commitment $C_{\widetilde{\mathsf{ops}}}$ for them. We also produce a single polynomial commitment to the vectors $(\mathsf{audit\text{-}ts}_{\mathsf{row}}, \mathsf{audit\text{-}ts}_{\mathsf{col}})$ for all three of A, B, C as $C_{\widetilde{\mathsf{mem}}}$. In the opening phase, a transcript for $\mathsf{PC}_{\mathsf{SparseMulti}}.\mathsf{Open}$ consists of the following:

1. A single polynomial commitment $C_{\widetilde{e_{\mathsf{batch}}}}$ to the vectors $(e_{\mathsf{row}}, e_{\mathsf{col}})$ for all three of A, B, C.
2. Challenges $\gamma_1, \gamma_2 \in \mathbb{F}^*$.
3. A proof that step 1 in CircuitSparseEval is computed correctly for $M \in \{A, B, C\}$. We can use a sumcheck protocol (with all polynomials sent in the clear, since we do not need zero-knowledge) to verify this part of the circuit.
4. A proof that step 2 in CircuitSparseEval is computed correctly for $M \in \{A, B, C\}$. We arrange the product circuit in a binary tree and invoke $O(\log(m + n))$ sumcheck protocols to check this part of the circuit.
5. At the end of the sumchecks, we need to do three opening arguments for the polynomials we committed.

We refer to the implementation [58] for full details regarding these optimizations. The crucial point is that in the proof of knowledge soundness (see Lemma B.3), the extractor only needs to extract from these three opening proofs, then we reduce to the soundness of the underlying information-theoretic protocol.

B.2 Omitted Proofs

We first prove Lemma 5.4, restated below.

Lemma B.1 (Restatement of Lemma 5.4). $\mathsf{PC}_{\mathsf{Multi}}.\mathsf{Open}$ *is* $\mathbf{n}$*-computational special sound, where* $\mathbf{n} = (\sqrt{m}, \underbrace{4_\pm, \ldots, 4_\pm}_{\mu/2}, 2)$*. Concretely, there exists a tree-extraction algorithm* $\mathcal{TE}_{\mathsf{PC}_{\mathsf{Multi}}}$ *such that for any PPT adversary* $\mathcal{A}$ *against* SS *of* $\mathsf{PC}_{\mathsf{Multi}}.\mathsf{Open}$*, there exists an EPT adversary* $\mathcal{B}$ *against* DL-REL *such that*

$$\mathbf{Adv}^{\mathsf{SS}}_{\Pi,\mathbf{n}}(\mathcal{TE}_{\mathsf{PC}_{\mathsf{Multi}}}, \mathcal{A}) \leq \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},\sqrt{m}+2}(\mathcal{B}).$$

Requirements for $\mathsf{PC}_{\mathsf{SparseMulti}}$**.** A polynomial commitment scheme PC and an interactive argument Π for circuit satisfiability.

- $\mathsf{Setup}(\mu, n, \mathsf{pp}_{\mathcal{G}})$: run $\mathsf{pp}_1 \leftarrow \mathsf{PC.Setup}(\mu, \mathsf{pp}_{\mathcal{G}})$ and $\mathsf{pp}_2 \leftarrow \mathsf{PC.Setup}(\log n, \mathsf{pp}_{\mathcal{G}})$. Output $\mathsf{pp} = (\mathsf{pp}_1, \mathsf{pp}_2)$.

- $\mathsf{Commit}(\mathsf{pp}, \widetilde{M})$:

1. Let $(\mathsf{row}, \mathsf{col}, \mathsf{val})$ be the dense representation of $\widetilde{M}$.
Compute $C_X \leftarrow \mathsf{PC.Commit}(\mathsf{pp}, \widetilde{X})$ for $X \in \{\mathsf{row}, \mathsf{col}, \mathsf{val}\}$.
2. Let $(\mathsf{read\text{-}ts}_{\mathsf{row}}, \mathsf{write\text{-}ts}_{\mathsf{row}}, \mathsf{audit\text{-}ts}_{\mathsf{row}}) \leftarrow \mathsf{MemoryInTheHead}(2^{\mu/2}, n, \mathsf{row})$.
Compute $C_{X_{\mathsf{row}}} \leftarrow \mathsf{PC.Commit}(\mathsf{pp}, \widetilde{X_{\mathsf{row}}})$ for $X \in \{\mathsf{read\text{-}ts}, \mathsf{write\text{-}ts}, \mathsf{audit\text{-}ts}\}$.
3. Let $(\mathsf{read\text{-}ts}_{\mathsf{col}}, \mathsf{write\text{-}ts}_{\mathsf{col}}, \mathsf{audit\text{-}ts}_{\mathsf{col}}) \leftarrow \mathsf{MemoryInTheHead}(2^{\mu/2}, n, \mathsf{col})$.
Compute $C_{X_{\mathsf{col}}} \leftarrow \mathsf{PC.Commit}(\mathsf{pp}, \widetilde{X_{\mathsf{col}}})$ for $X \in \{\mathsf{read\text{-}ts}, \mathsf{write\text{-}ts}, \mathsf{audit\text{-}ts}\}$.
4. Output

$$C = (C_{\mathsf{row}}, C_{\mathsf{col}}, C_{\mathsf{val}}, C_{\mathsf{read\text{-}ts_{row}}}, C_{\mathsf{write\text{-}ts_{row}}}, C_{\mathsf{audit\text{-}ts_{row}}}, C_{\mathsf{read\text{-}ts_{col}}}, C_{\mathsf{write\text{-}ts_{col}}}, C_{\mathsf{write\text{-}ts_{val}}}).$$

- $\mathsf{Open}\langle \mathcal{P}(\widetilde{M}, \omega), \mathcal{V}\rangle(\mathsf{pp}, C, (r_{\mathsf{row}}, r_{\mathsf{col}}), v)$:

1. $\mathcal{P}$ computes $e_X = (\widetilde{\mathrm{eq}}(X_0, r_X), \ldots, \widetilde{\mathrm{eq}}(X_{n-1}, r_X))$ for $X \in \{\mathsf{row}, \mathsf{col}\}$, then sends $C_{e_X} \leftarrow \mathsf{PC.Commit}(\mathsf{pp}, \widetilde{e_X})$ for $X \in \{\mathsf{row}, \mathsf{col}\}$.
2. $\mathcal{V}$ sends challenges $\gamma_1, \gamma_2 \xleftarrow{\$} \mathbb{F}^*$.
3. $\mathcal{P}, \mathcal{V}$ engage in Π to verify that $\widetilde{M}(r_{\mathsf{row}}, r_{\mathsf{col}}) = v$, computed using $\mathsf{CircuitSparseEval}$.

Memory checking procedure $\mathsf{MemoryInTheHead}(m, n, \mathsf{addr} \in \mathbb{F}^m)$**:**

1. Initialize empty $\mathsf{read\text{-}ts}_{\mathsf{addr}}, \mathsf{write\text{-}ts}_{\mathsf{addr}} \in \mathbb{F}^n$, $\mathsf{audit\text{-}ts}_{\mathsf{addr}} \in \mathbb{F}^m$. For $i = 1, \ldots, m$:

(a) $a \leftarrow \mathsf{addr}_i$,
(b) $\mathsf{ts} \leftarrow (\mathsf{audit\text{-}ts}_{\mathsf{addr}})_a$,
(c) $\mathsf{read\text{-}ts}_i \leftarrow \mathsf{ts}$,
(d) $\mathsf{ts} \leftarrow \mathsf{ts} + 1$,
(e) $\mathsf{write\text{-}ts}_i \leftarrow \mathsf{ts}$,
(f) $(\mathsf{audit\text{-}ts}_{\mathsf{addr}})_a \leftarrow \mathsf{ts}$.

2. Output $(\mathsf{read\text{-}ts}_{\mathsf{addr}}, \mathsf{write\text{-}ts}_{\mathsf{addr}}, \mathsf{audit\text{-}ts}_{\mathsf{addr}})$.

Fig. 16: Sparse Multilinear Polynomial Commitment Scheme $\mathsf{PC}_{\mathsf{SparseMulti}}$

Proof. Note that the first layer in the tree of transcripts consists of $\sqrt{m}$ distinct verifier's challenges $x_1, \ldots, x_{\sqrt{m}} \in \mathbb{F}^\mu$ that serve as evaluation points; the rest of the tree then corresponds to an instance of $\mathsf{LogDotProdPf}$ described in Figure 15. Our tree extractor $\mathcal{TE}_{\mathsf{PC_{Multi}}}$ consists of two steps. First, we run the tree extractor $\mathcal{TE}_{\mathsf{LogDotProdPf}}$ on each $(4_{\pm}^{\mu/2}, 2)$ subtree to recover the underlying linear combinations

$$\mathbf{l} = \left(\sum_{k \in \{0,1\}^{\mu/2}} p(k + 2^{\mu/2} j) \cdot \widetilde{\mathrm{eq}}(x_{[:\mu/2]}, k) \right)_{j \in \{0,1\}^{\mu/2}}$$

Here, the tree extractor $\mathcal{TE}_{\mathsf{LogDotProdPf}}$ is similar to that of $\mathsf{BP\text{-}IPA}$, and so we refer to the proof of Lemma 6.1. Note that $\mathcal{TE}_{\mathsf{LogDotProdPf}}$ will either succeed, or we can build an adversary $\mathcal{B}$ against $\mathsf{DL\text{-}REL}$. Next, for each $j \in \{0,1\}^{\mu/2}$, we then use the corresponding entry in $\mathbf{l}$ with $\sqrt{m}$ different challenges $x_1, \ldots, x_{\sqrt{m}}$ to solve for $p(k + 2^{\mu/2} j)$ for all $k \in \{0,1\}^{\mu/2}$. This is possible since the Lagrange polynomials $\{\widetilde{\mathrm{eq}}(x_{[:\mu/2]}, k)\}_{k \in \{0,1\}^{\mu/2}}$ are independent for $2^{\mu/2}$ different values of $x_{[:\mu/2]}$. $\square$

Our next lemma summarizes the information-theoretic soundness of the hash functions used in $\mathsf{CircuitSparseEval}$.

Public Input. Dense representation $(\mathsf{row}, \mathsf{col}, \mathsf{val})$ of M, and evaluation point $r = (r_{\mathsf{row}}, r_{\mathsf{col}}) \in \mathbb{F}^{2\mu}$, where $\mu = \log m$.

Witness. $\begin{cases} \mathsf{read\text{-}ts}_{\mathsf{row}}, \mathsf{read\text{-}ts}_{\mathsf{col}}, \mathsf{write\text{-}ts}_{\mathsf{row}}, \mathsf{write\text{-}ts}_{\mathsf{col}} \in \mathbb{F}^n, \\ \mathsf{audit\text{-}ts}_{\mathsf{row}}, \mathsf{audit\text{-}ts}_{\mathsf{col}} \in \mathbb{F}^m, e_{\mathsf{row}}, e_{\mathsf{col}} \in \mathbb{F}^n, \gamma_1, \gamma_2 \in \mathbb{F}. \end{cases}$

1. Compute $\widetilde{M}(r) = \sum_{i=0}^{n-1} \mathsf{val}_i \cdot (e_{\mathsf{row}})_i \cdot (e_{\mathsf{col}})_i$ and return $\widetilde{M}(r)$.
2. For $X \in \{\mathsf{row}, \mathsf{col}\}$, do the following:

(a) Compute

$$\begin{aligned} &\mathsf{mem}_X \leftarrow [\widetilde{\mathrm{eq}}(0, r_X), \ldots, \widetilde{\mathrm{eq}}(m-1, r_X)] \in \mathbb{F}^m, \\ &\mathsf{ISet}_X = \{(i, (\mathsf{mem}_X)_i, 0)\}_{i \in [0, m-1]}, \\ &\mathsf{RSet}_X = \{(X_i, (e_X)_i, (\mathsf{read\text{-}ts}_X)_i)\}_{i \in [0, n-1]}, \\ &\mathsf{WSet}_X = \{(X_i, (e_X)_i, (\mathsf{write\text{-}ts}_X)_i)\}_{i \in [0, n-1]}, \\ &\mathsf{ASet}_X = \{(i, (\mathsf{mem}_X)_i, (\mathsf{audit\text{-}ts}_X)_i)\}_{i \in [0, m-1]}. \end{aligned}$$

(b) Compute $h_{1,X} = \mathcal{H}_{\gamma_1,\gamma_2}(\mathsf{ISet}_X \sqcup \mathsf{WSet}_X)$, $h_{2,X} = \mathcal{H}_{\gamma_1,\gamma_2}(\mathsf{RSet}_X \sqcup \mathsf{ASet}_X)$, and assert that $h_{1,X} = h_{2,X}$.

Fig. 17: $\mathsf{CircuitSparseEval}$ to evaluate a sparse multilinear polynomial $\widetilde{M}$

Lemma B.2. *In* $\mathsf{CircuitSparseEval}$*, we have the following:*

$$\Pr[h_{1,X} = h_{2,X} \mid \mathsf{ISet}_X \sqcup \mathsf{WSet}_X \neq \mathsf{RSet}_X \sqcup \mathsf{ASet}_X] \leq \frac{2(m+n)}{|\mathbb{F}|}.$$

Proof. Note that $\mathcal{H}_{\gamma_1,\gamma_2}(A, V, T)$ for $A, V, T \in \mathbb{F}^{m+n}$ is a polynomial of total degree $2(m+n)$ in γ_1 and γ_2. Applying Schwartz-Zippel gives us the desired bound. □

We now establish the special soundness of $\mathsf{PC}_{\mathsf{SparseMulti}}$, following the discussion about optimizations above.

Lemma B.3. $\mathsf{PC}_{\mathsf{SparseMulti}}$ *satisfies* $((2)^{\mu/2}, \underbrace{4_{\pm}, \ldots, 4_{\pm}}_{(\mu+\nu)/2}, 2)^3$*-computational special soundness. Concretely, there exists a tree-extraction algorithm* $\mathcal{TE}_{\mathsf{PC}_{\mathsf{SparseMulti}}}$ *such that for any PPT adversary* $\mathcal{A}$ *against* SS *of* $\mathsf{PC}_{\mathsf{SparseMulti}}.\mathsf{Open}$*, there exists an EPT adversary* $\mathcal{B}$ *against* $\mathsf{DL\text{-}REL}$ *such that*

$$\mathbf{Adv}^{\mathsf{SS}}_{\Pi,\mathbf{n}}(\mathcal{TE}_{\mathsf{PC}_{\mathsf{Multi}}}, \mathcal{A}) \leq \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},\sqrt{m+n}+2}(\mathcal{B}) + \frac{6(m+n) + O(\mu+\nu)}{|\mathbb{F}|}.$$

Proof. The tree-extractor $\mathcal{TE}_{\mathsf{PC}_{\mathsf{SparseMulti}}}$ simply invokes $\mathcal{TE}_{\mathsf{PC}_{\mathsf{Multi}}}$ three times for the three polynomial openings (listed in the optimizations). Once we get all polynomials in the clear, it remains to argue the soundness of the underlying information-theoretic protocol. This soundness error consists of two kinds: (1) the error in the sumcheck invocations, which are $O(\mu+\nu)/|\mathbb{F}|$, and (2) the error in the hashes for $\mathsf{CircuitSparseEval}$, which when invoked for three matrices A, B, C is at most $6(m+n)/|\mathbb{F}|$. This establishes the desired bound. □

Combining Lemma B.3 with Lemma 4.7, we obtain Lemma 5.13, restated below.

Lemma B.4. $\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}$ *satisfies knowledge soundness. Concretely, there exists an extractor* $\mathcal{E}_{\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}}$ *such that for every PPT prover* $\mathcal{P}^*$ *against* KS *of* $\mathsf{Spartan\text{-}SNARK}$ *making at most* Q *random oracle queries, there exists an EPT adversary* $\mathcal{B}$ *against* $\mathsf{DL\text{-}REL}$ *such that*

$$\mathbf{Adv}^{\mathsf{KS}}_{\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}}(\mathcal{E}_{\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}}, \mathcal{P}^*) \\ \leq \frac{Q(Q-1) + (Q+1)(25\mu + 9\nu + 16) + 6(m+n) + O(\mu+\nu)}{|\mathbb{F}| - 1} + \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G}, \sqrt{m+n}+2}(\mathcal{B}).$$

Both $\mathcal{B}$ and the extractor $\mathcal{E}_{\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}}$ runs in expected time that is at most $O(Q \cdot m^{7.5} \cdot (m+n)^3)$ the running time of $\mathcal{P}^*$.

Proof. We first show that $\mathsf{Spartan\text{-}SNARK}_{\mathsf{FS}}$ is $\mathbf{n}'$-computational special sound, where

$$\mathbf{n}' = (1, (1, 2_{\mathsf{li}}, 2)^{\mu}, 2, 2, 2, 1, (2, 2_{\mathsf{li}}, 2)^{\mu}, (\underbrace{4_{\pm}, \ldots, 4_{\pm}}_{\mu/2}, 2), ((2)^{\mu/2}, \underbrace{4_{\pm}, \ldots, 4_{\pm}}_{(\mu+\nu)/2}, 2)^3, 2).$$

This follows from combining the tree-extractor for $\mathcal{TE}_{\mathsf{Spartan\text{-}NIZK}_{\mathsf{FS}}}$ constructed in Lemma 5.6 with the tree-extractor $\mathcal{TE}_{\mathsf{PC}_{\mathsf{SparseMulti}}}$ constructed in Lemma B.3. We then conclude the bound above using Lemma 4.7. □

C Omitted details for Bulletproofs

We describe the protocol $\mathsf{BP\text{-}ACSPf}$ in Figure 18. Note that n is the number of multiplication gates, m is the number of committed inputs, and q is the number of equations involving $\mathbf{W}$s. Our proof of Theorem 6.7 follows from Theorem 3.4 combined with the results proved in this section.

Lemma C.1. $\mathsf{BP\text{-}ACSPf}_{\mathsf{FS}}$ *satisfies knowledge soundness. Let n be the number of multiplication gates and q the number of equations involving committed inputs. Concretely, there exists an extractor $\mathcal{E}_{\mathsf{BP\text{-}ACSPf}_{\mathsf{FS}}}$ such that for every PPT adversary $\mathcal{P}^*$ against* KS *making at most Q random oracle queries, there exists an adversary $\mathcal{A}$ against* $\mathsf{DL\text{-}REL}$ *with*

$$\mathbf{Adv}^{\mathsf{KS}}_{\mathsf{BP\text{-}ACSPf}_{\mathsf{FS}}}(\mathcal{E}_{\mathsf{BP\text{-}ACSPf}_{\mathsf{FS}}}, \mathcal{P}^*) \leq \mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G}, 2n+1}(\mathcal{A}) + \frac{Q(Q-1) + 2(Q+1)(n+q+3\log n + 6)}{|\mathbb{F}| - 1}\ .$$

Both $\mathcal{A}$ and the extractor $\mathcal{E}_{\mathsf{BP\text{-}ACSPf}_{\mathsf{FS}}}$ run in expected time that is at most $O(Q \cdot q \cdot n^3)$ that of $\mathcal{P}^*$'s runtime.

Proof. By the result of [16], given an $\mathbf{n}$-tree of accepting transcripts where $\mathbf{n} = (n, q+1, 7, 2, 4_{\pm}, \overset{\log n}{\ldots}, 4_{\pm})$, we can extract. Applying our special soundness result yields the bound.

The expected runtime of the extractor $\mathcal{E}_{\mathsf{BP\text{-}ACSPf}_{\mathsf{FS}}}$, as well as the adversary $\mathcal{A}$, is at most $O(Q \cdot n \cdot (q + 1) \cdot 12 \cdot n^2) = O(Q \cdot q \cdot n^3)$. □

Lemma C.2. $\mathsf{BP\text{-}ACSPf}_{\mathsf{FS}}$ *satisfies perfect 2-*ZK*.*

Proof. We present the 2-ZK simulator $\mathcal{S}_{\mathsf{BP\text{-}ACSPf}_{\mathsf{FS}}, x}$, and argue that its output is identically distributed to that of honestly generated proofs.

Arithmetic Circuit Satisfiability Relation.

$$\mathcal{R}_{\textsf{BP-ACSPf}} = \left\{ \begin{array}{c} \begin{pmatrix} (m, n, q, \mathbf{g}, \mathbf{h}, g, h, u), \\ (\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{W}_V, \mathbf{V}, \mathbf{c}), \\ (\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O, \mathbf{v}, \gamma) \end{pmatrix} : \\ V_i = g^{v_i} h^{\gamma_i} \ \forall j \in [1, m] \ \wedge \ \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O \ \wedge \\ \mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_O = \mathbf{W}_V \cdot \mathbf{v} + \mathbf{c} \end{array} \right\}$$

Interaction Phase.

1. $\mathcal{P}$ samples $\alpha, \beta, \rho \xleftarrow{\$} \mathbb{F}$, $\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{F}^n$ and computes

$$A_I = h^{\alpha} \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R}, \qquad A_O = h^{\beta} \mathbf{g}^{\mathbf{a}_O}, \qquad S = h^{\rho} \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}.$$

$\mathcal{P}$ sends A_I, A_O, S to $\mathcal{V}$.

2. $\mathcal{V}$ sends challenges $y, z \xleftarrow{\$} \mathbb{F}^*$.
3. Denote $\mathbf{y}^n = (1, y, \ldots, y^{n-1}) \in \mathbb{F}^n$ and $\mathbf{z}_{[1:]}^{q+1} = (z, z^2, \ldots, z^q) \in \mathbb{F}^q$.

$\mathcal{P}$ samples $\beta_i \xleftarrow{\$} \mathbb{F}$ for all $i \in \{1, 3, 4, 5, 6\}$ and computes

$$\begin{aligned} \ell(X) &= \mathbf{a}_L \cdot X + \mathbf{a}_O \cdot X^2 + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{q+1} \cdot \mathbf{W}_R) \cdot X + \mathbf{s}_L \cdot X^3, \\ r(X) &= (\mathbf{y}^n \circ \mathbf{a}_R) \cdot X - \mathbf{y}^n + \mathbf{z}_{[1:]}^{q+1} \cdot (\mathbf{W}_L \cdot X + \mathbf{W}_O) + (\mathbf{y}^n \circ \mathbf{s}_R) \cdot X^3, \\ t(X) &= \langle \ell(X), r(X) \rangle = \sum_{i=1}^{6} t_i X^i, \\ T_i &= g^{t_i} h^{\beta_i} \quad \forall\, i \in \{1, 3, 4, 5, 6\}. \end{aligned}$$

$\mathcal{P}$ sends T_1, T_3, T_4, T_5, T_6 to $\mathcal{V}$.

4. $\mathcal{V}$ sends challenge $x \xleftarrow{\$} \mathbb{F}^*$.
5. $\mathcal{P}$ computes

$$\begin{aligned} &\mathbf{l} = \ell(x), \quad \mathbf{r} = r(x), \quad \hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle, \quad \mu = \alpha \cdot x + \beta \cdot x^2 + \rho \cdot x^3, \\ &\beta_x = \beta_1 \cdot x + \langle \mathbf{z}_{[1:]}^{q+1}, \mathbf{W}_V \cdot \gamma \rangle \cdot x^2 + \sum_{i=3}^{6} \beta_i \cdot x^i. \end{aligned}$$

$\mathcal{P}$ sends $\hat{t}, \beta_x, \mu$ to $\mathcal{V}$.

6. $\mathcal{V}$ sends challenge $w \xleftarrow{\$} \mathbb{F}^*$.
7. $\mathcal{P}, \mathcal{V}$ both compute

$$\begin{aligned} &\mathbf{h}' = \mathbf{h}^{\mathbf{y}^{-n}}, \quad u' = u^w, \\ &W_L = (\mathbf{h}')^{\mathbf{z}_{[1:]}^{q+1} \cdot \mathbf{W}_L}, \quad W_R = \mathbf{g}^{\mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{q+1} \cdot \mathbf{W}_R)}, \quad W_O = (\mathbf{h}')^{\mathbf{z}_{[1:]}^{q+1} \cdot \mathbf{W}_O}, \\ &P' = h^{-\mu} \cdot A_I^x \cdot A_O^{x^2} \cdot S^{x^3} \cdot (\mathbf{h}')^{-\mathbf{y}^n} \cdot W_L^x \cdot W_R^x \cdot W_O \cdot (u')^{\hat{t}}. \end{aligned}$$

8. $\mathcal{P}, \mathcal{V}$ engage in $\textsf{BP-IPA}$ for the triple $((n, \mathbf{g}, \mathbf{h}', u'), P', (\mathbf{l}, \mathbf{r}))$.

Fig. 18: Bulletproofs' Arithmetic Circuit Satisfiability $\textsf{BP-ACSPf}$

Verification.

1. $\mathcal{V}$ rejects if BP-IPA fails.
2. $\mathcal{V}$ computes

$$\delta(y,z) = \langle \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{q+1} \cdot \mathbf{W}_R), \mathbf{z}_{[1:]}^{q+1} \cdot \mathbf{W}_L \rangle,$$

$$R = g^{x^2 \cdot \left(\delta(y,z) + \langle \mathbf{z}_{[1:]}^{q+1}, \mathbf{c} \rangle\right)} \cdot \mathbf{V}^{x^2 \cdot \left(\mathbf{z}_{[1:]}^{q+1} \cdot \mathbf{W}_V\right)} \cdot T_1^x \cdot \prod_{i=3}^{6} T_i^{x^i}.$$

3. $\mathcal{V}$ checks whether $g^{\hat{t}} h^{\beta_x} \stackrel{?}{=} R$.

Fig. 18: Bulletproofs' Arithmetic Circuit Satisfiability BP-ACSPf (cont.)

Simulator $\mathcal{S}_{\mathsf{BP\text{-}ACSPf_{FS}},x}(\mathsf{pp}, \mathbb{x})$:

1. Sample $\alpha, \beta, \rho \xleftarrow{\$} \mathbb{F}$ and $\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O, \mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{F}^n$ and compute

$$A_I = h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R}, \qquad A_O = h^\beta \mathbf{g}^{\mathbf{a}_O}, \qquad S = h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}.$$

2. Query random oracle for challenges $y, z \leftarrow \mathsf{H}(\mathsf{pp}, \mathsf{x}, A_I, A_O, S)$.
3. Sample $x \xleftarrow{\$} \mathbb{F}^*$.
4. Compute
$$\begin{cases} \mathbf{l} = \mathbf{a}_L \cdot x + \mathbf{a}_O \cdot x^2 + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R) \cdot x + \mathbf{s}_L \cdot x^3, \\ \mathbf{r} = \mathbf{y}^n \circ \mathbf{a}_R \cdot x - \mathbf{y}^n + \mathbf{z}_{[1:]}^{Q+1} \cdot (\mathbf{W}_L \cdot x + \mathbf{W}_O) + \mathbf{y}^n \circ \mathbf{s}_R \cdot x^3, \\ \hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle. \end{cases}$$
5. Sample $\beta_x \xleftarrow{\$} \mathbb{F}$, $T_3, T_4, T_5, T_6 \xleftarrow{\$} \mathbb{G}$ and compute
$$T_1 = \left(g^{x^2 \cdot (\delta(y,z) + \langle \mathbf{z}_{[1:]}^{q+1}, \mathbf{c} \rangle) - \hat{t}} \cdot h^{-\beta_x} \cdot \mathbf{V}^{-x^2 \cdot \mathbf{z}_{[1:]}^{q+1} \cdot \mathbf{W}_V} \cdot \prod_{i=3}^{6} T_i^{x^i} \right)^{-x^{-1}}.$$
6. Program $\mathsf{H}(\mathsf{pp}, V, A, S, T_1, T_2) := x$.
7. Query random oracle for challenge $w \leftarrow \mathsf{H}(\mathsf{pp}, V, A, S, T_1, T_2, \hat{t}, \beta_x, \mu)$.
8. Compute
$$\begin{cases} \mathbf{h}' = \mathbf{h}^{\mathbf{y}^{-n}}, \\ u' = u^w, \\ P' = h^{-\mu} \cdot A_I^x \cdot A_O^{x^2} \cdot S^{x^3} \cdot (\mathbf{h}')^{-\mathbf{y}^n} \cdot W_L^x \cdot W_R^x \cdot W_O \cdot (u')^{\hat{t}}. \end{cases}$$
9. Generate honest proof $\pi_{\mathsf{BP\text{-}IPA_{FS}}}$ for the triple $((n, \mathbf{g}, \mathbf{h}', u'), P', (\mathbf{l}, \mathbf{r}))$.
10. Output $\pi_{\mathsf{BP\text{-}ACSPf_{FS}}} = (A_I, A_O, S, T_1, T_3, T_4, T_5, T_6, \hat{t}, \beta_x, \mu, \pi_{\mathsf{BP\text{-}IPA_{FS}}})$.

The indistinguishability argument between the outputs of $\mathcal{S}_{\mathsf{BP\text{-}ACSPf_{FS}}}$, and honestly generated proofs goes as follows. In both cases, the proof elements $A, T_3, T_4, T_5, T_6, \beta_x, \mu$ and the underlying vectors $\mathbf{l}, \mathbf{r}$ are distributed uniformly among their respective domains. The remaining proof elements $S, T_1, \hat{t}, \pi_{\mathsf{BP\text{-}IPA_{FS}}}$ are then uniquely determined from the previous ones, and thus are identically distributed as well. This establishes the 2-ZK property of $\mathsf{BP\text{-}ACSPf_{FS}}$. □

Finally, we show the unique response property of BP-ACSPf, starting from after the x challenge.

Lemma C.3. $\mathsf{BP\text{-}ACSPf_{FS}}$ *satisfies* 2-UR. *In particular, for any adversary* $\mathcal{A}$ *against* 2-UR *of* $\mathsf{BP\text{-}ACSPf_{FS}}$, *there exists an adversary* $\mathcal{B}$ *against* DL-REL *such that*

$$\mathbf{Adv}_{\mathsf{BP\text{-}ACSPf_{FS}}}^{\mathsf{2\text{-}UR}}(\mathcal{A}) \leq 3 \cdot \mathbf{Adv}_{\mathbb{G},2n+1}^{\mathsf{DL\text{-}REL}}(\mathcal{B}) + \frac{2Q(Q-1) + 12(Q+1)\log n}{|\mathbb{F}| - 1}.$$

$\mathcal{B}$ *runs in expected time at most* $O(Q \cdot n^2)$ *that of* $\mathcal{A}$*'s runtime.*

Proof. We proceed through a sequence of hybrids. The high-level idea is to analyze different cases for where the two proofs π, π' first differ after the x challenge, and reduce each case to breaking $\mathsf{DL\text{-}REL}$ or the knowledge soundness KS of $\mathsf{BP\text{-}IPA}$ (which in turn reduces to breaking $\mathsf{DL\text{-}REL}$).

– Hyb_0 is the game $2\text{-}\mathsf{UR}^{\mathcal{A}}_{\mathsf{BP\text{-}ACSPf_{FS}}}$. Recall that in this game, an adversary $\mathcal{A}$ outputs an input $\mathbf{V}$, a challenge $x \in \mathbb{F}^*$, and two proofs π, π' that agrees up to the x challenge, i.e. we have

$$\pi = (A_I, A_O, S, T_1, T_3, T_4, T_5, T_6, \hat{t}, \beta_x, \mu, \pi_{\mathsf{BP\text{-}IPA_{FS}}}),$$
$$\pi' = (A_I, A_O, S, T_1, T_3, T_4, T_5, T_6, \hat{t}', \beta'_x, \mu', \pi'_{\mathsf{BP\text{-}IPA_{FS}}}).$$

$\mathcal{A}$ wins if $\pi \neq \pi'$ and both proofs are accepting with respect to the x challenge that it chose.

– Hyb_1 is the same as Hyb_0, except that we also run $\mathcal{E}_{\mathsf{BP\text{-}IPA_{FS}}}$ on the proofs $\pi_{\mathsf{BP\text{-}IPA_{FS}}}, \pi'_{\mathsf{BP\text{-}IPA_{FS}}}$ to extract witnesses $(\mathbf{l}, \mathbf{r})$ and $(\mathbf{l}', \mathbf{r}')$. Hyb_1 returns 0 if the extractor aborts on either proofs, or the witnesses are not satisfying, i.e. that $\hat{t} \neq \langle \mathbf{l}, \mathbf{r} \rangle$ or $\hat{t}' \neq \langle \mathbf{l}', \mathbf{r}' \rangle$.

We can see that Hyb_1 is identical to Hyb_0, except when the extractor $\mathcal{E}_{\mathsf{BP\text{-}IPA_{FS}}}$ fails in extracting from either proofs $\pi_{\mathsf{BP\text{-}IPA_{FS}}}, \pi'_{\mathsf{BP\text{-}IPA_{FS}}}$. The probability that this happens is precisely bounded by (twice) the KS advantage of $\mathsf{BP\text{-}IPA_{FS}}$. Concretely, by Theorem 6.1 there exists an adversary $\mathcal{B}$ against $\mathsf{DL\text{-}REL}$, running in expected time at most $O(Q \cdot m^2 \cdot n^2)$ that of $\mathcal{A}$'s runtime, such that

$$|\Pr[\mathsf{Hyb}_0] - \Pr[\mathsf{Hyb}_1]| \leq 2\frac{Q(Q-1) + 6(Q+1)\log n}{|\mathbb{F}| - 1} + 2\mathbf{Adv}^{\mathsf{DL\text{-}REL}}_{\mathbb{G},2n+1}(\mathcal{B}) .$$

It remains to show that if Hyb_1 returns 1, then there exists an adversary $\mathcal{B}'$ that returns a non-trivial discrete log relation. The adversary $\mathcal{B}'$ works as follows:

- **If $\hat{t} \neq \hat{t}'$ or $\beta_x \neq \beta'_x$:** since both proofs are accepting and are the same up to the x challenge, we have

$$\begin{aligned} g^{\hat{t}} \cdot h^{\beta_x} &= g^{x^2 \cdot \left(\delta(y,z) + \langle \mathbf{z}^{q+1}_{[1:]}, \mathbf{c} \rangle\right)} \cdot \mathbf{V}^{x^2 \cdot \left(\mathbf{z}^{q+1}_{[1:]} \cdot \mathbf{W}_V\right)} \cdot T_1^x \cdot \prod_{i=3}^{6} T_i^{x^i} \\ &= g^{\hat{t}'} \cdot h^{\beta'_x}. \end{aligned}$$

  This gives a non-trivial discrete-log relation for $\mathcal{B}'$ to output.
- **If $(\hat{t}, \beta_x) = (\hat{t}', \beta'_x)$ but $\mu \neq \mu'$:** since both proofs $\pi_{\mathsf{BP\text{-}IPA_{FS}}}, \pi'_{\mathsf{BP\text{-}IPA_{FS}}}$ are accepting, we have

$$\begin{aligned} \mathbf{g}^{\mathbf{l}} \cdot \mathbf{h}^{(\mathbf{y}^{-m \cdot n} \circ \mathbf{r})} \cdot h^{\mu} &= A_I^x \cdot A_O^{x^2} \cdot S^{x^3} \cdot (\mathbf{h}')^{-\mathbf{y}^n} \cdot W_L^x \cdot W_R^x \cdot W_O \cdot u^{w \cdot \hat{t}} \\ &= \mathbf{g}^{\mathbf{l}'} \cdot \mathbf{h}^{(\mathbf{y}^{-m \cdot n} \circ \mathbf{r}')} \cdot h^{\mu'}. \end{aligned}$$

  $\mathcal{B}'$ then outputs this non-trivial discrete-log relation.
- **If $(\hat{t}, \beta_x, \mu) = (\hat{t}', \beta'_x, \mu')$ but $\pi_{\mathsf{BP\text{-}IPA_{FS}}} \neq \pi'_{\mathsf{BP\text{-}IPA_{FS}}}$:** we consider the same equation as in the previous case. Here, by the same reasoning as in Theorem 6.2, it must be that $(\mathbf{l}, \mathbf{r}) \neq (\mathbf{l}', \mathbf{r}')$, or otherwise the two inner product arguments are the same. $\mathcal{B}'$ thus outputs the same non-trivial discrete-log relation.

Note that if Hyb_1 returns 1, then $\pi \neq \pi'$, hence at least one of the above cases must happen. Putting everything together (and in particular unifying adversaries $\mathcal{B}, \mathcal{B}'$), we get the desired bound. $\square$