How to Compile Polynomial IOP into Simulation-Extractable SNARKs: A Modular Approach

Markulf Kohlweiss[1,2], Mahak Pancholi[3], and Akira Takahashi[1]

[1] The University of Edinburgh, UK
markulf.kohlweiss@ed.ac.uk,takahashi.akira.58s@gmail.com
[2] Input Output Global
[3] Aarhus University, Denmark
mahakp@cs.au.dk

July 8, 2023

Abstract. Most succinct arguments (SNARKs) are initially only proven knowledge sound (KS). We show that the commonly employed compilation strategy from polynomial interactive oracle proofs (PIOP) via polynomial commitments to knowledge sound SNARKS actually also achieves other desirable properties: weak unique response (WUR) and trapdoorless zero-knowledge (TLZK); and that together they imply simulation extractability (SIM-EXT).
The factoring of SIM-EXT into KS + WUR + TLZK is becoming a cornerstone of the analysis of non-malleable SNARK systems. We show how to prove WUR and TLZK for PIOP compiled SNARKs under mild falsifiable assumptions on the polynomial commitment scheme. This means that the analysis of knowledge soundness from PIOP properties that inherently relies on non-falsifiable or idealized assumption such as the algebraic group model (AGM) or generic group model (GGM) need not be repeated.
While the proof of WUR requires only mild assumptions on the PIOP, TLZK is a different matter. As perfectly hiding polynomial commitments sometimes come at a substantial performance premium, SNARK designers prefer to employ deterministic commitments with some leakage. This results in the need for a stronger zero-knowledge property for the PIOP.
The modularity of our approach implies that any analysis improvements, e.g. in terms of tightness, credibility of the knowledge assumption and model of the KS analysis, or the precision of capturing real-world optimizations for TLZK also benefits the SIM-EXT guarantees.

1 Introduction

Succinct arguments and zero-knowledge proofs are being implemented and deployed. This is both due to their improved practicality and the popularity of use-cases that require efficient and private verification of statements. As it becomes harder to discern real information from automatically generated fakes, we have to increasingly rely on cryptographic chains of evidence [NT16, FFG+16]. Efficient zero-knowledge proofs help institutions become more transparent [BCGW22], help scale blockchains [BBHR19, BMRS20, Sta21], and make applications more private [BCG+14, GGM14, KMS+16, BAZB20].

As SNARKs become more prevalent, we must demand them to have optimal security and not just optimal speed. Recently, a number of new SNARK constructions have been proposed in the literature. They are typically proved to satisfy stand-alone security properties in isolation, namely zero-knowledge and (knowledge) soundness. These basic properties are often unsatisfactory both in theory and practice, due to the fact that NIZK proofs are inherently *transferable*, i.e., whoever observed an existing valid proof can prove a statement by reusing/modifying the proof, even without the knowledge of the corresponding witness. To prevent such *malleability attacks*, the seminal work of Sahai [Sah99] introduced a stronger notion called *simulation-soundness*, which was later extended to *simulation-extractability* (SIM-EXT) [DDO+01]. Essentially, these notions state that no cheating prover can break (knowledge) soundness even after asking a ZK simulator to produce proofs on adaptively chosen statements. There is a long line of research that strengthens NIZK in a generic manner such that the proof system achieves simulation-soundness/extractability [DDO+01,

PR05,CL06,GMY06,JP14,FKMV12,KZM+15,ARS20,BS21,LR22b,LR22a,AGRS23]. These generic lifting techniques often apply additional cryptographic primitives, such as (one-time) signature [DDO+01,GMY06] and pseudo-random function [KZM+15], and then produce an extended proof for OR-statement derived from the target statement to be proven, without looking into inner workings of the base NIZK construction. In contrast, several recent works analyze particular exiting SNARK constructions *without modification* and successfully prove that some of the already deployed schemes satisfy simulation-extractability [GM17,BKSV21, GKK+22, GOP+22, GOP+23, DG23]. The downside of this approach is that analysis must be carried out in an ad-hoc way and tailored to each specific scheme.

Given this state of affairs, our question is whether it's possible to prove simulation-extractability for a large class of exiting SNARKs in an abstract manner. To this end, we turn to the popular paradigms of constructing highly efficient SNARKs from Polynomial Interactive Oracle Proofs (PIOP) [BCS16,CHM+20,BFS20,CFF+21]. This paradigm allows for a modular design of zkSNARK, starting from an information theoretic object, compiling it into an interactive argument system via cryptographic *polynomial commitments*. It is then made non-interactive in the random oracle model via Fiat-Shamir transform [FS87].

1.1 Our Contribution

Framework for proving SIM-EXT for PIOP-based SNARKs In this work, we provide a modular framework to prove SIM-EXT for a class of NIZK arguments constructed from PIOP in the random oracle model. We isolate sufficient and minimal properties required from the PIOP and the polynomial commitment schemes (PCOM) to conclude simulation-extractability (SIM-EXT) for the compiled zkSNARK, while relying on the existing knowledge soundness analysis in a black-box manner. An additional goal here is to minimize modifications to the exiting knowledge sound SNARK constructed via the compiler of [CHM+20, BFS20].

Along the way, we generalize a theorem by [GOP+23] for proving SIM-EXT of Fiat-Shamir arguments in a modular fashion by adapting the notion of canonical simulator. Our canonical simulators supports access to a SRS and an internal PIOP simulator. Finally, we generalize the theorem of [GOP+23] to hold for both straight-line and rewinding-based extractors (assuming knowledge soundness, trapdoor-less zero knowledge (TLZK), and weak unique response (WUR) hold). Our analysis can now focus on proving TLZK and WUR and these properties do not even involve any extractor in the definition; our result holds for any extractor.

Generic strategy for proving weak unique response Since the SNARKs we study are obtained by applying Fiat-Shamir to multi-round interactive arguments, it is often required to show the so-called *(weak) unique response (WUR)* property to conclude SIM-EXT [GOP+23]. As indicated in the very recent works [GOP+23, DG23], proofs of unique response can be quite involved especially if the underlying interactive protocols have many rounds. We show that only a few mild properties of PCOM are sufficient to show that the PIOP-compiler generically outputs a NIZK argument satisfying WUR. Interestingly, this implies that exiting polynomial commitments already have a built-in mechanism to retain SIM-EXT.

Generic strategy for trapdoor-less zero-knowledge simulation Another technical hurdle in proving SIM-EXT in a modular way is that, the zero knowledge simulation must be done in a trapdoor-less manner, i.e., the only additional power available to the zero-knowledge simulation is programming the random oracle. While this was trivial in previous work [GOP+23,GOP+22,DG23] that focus only on NIZK in the random oracle model, PIOP-based zkSNARKs often involve both a RO and structured reference string (SRS) generated by trusted setup. Consequently, the majority of existing ZK simulators for such systems take

advantage of the SRS's trapdoor. In this work, we provide generic strategies for achieving trapdoorless simulation for all PIOP schemes satisfying the standard property of honest verifier zero-knowledge (HVZK). Additionally, we introduce a stronger version of HVZK and refine our simulation strategy for PIOP schemes that satisfy this stronger notion.

Case studies for concrete schemes To show applicability of our framework, we study a few concrete schemes. For PCOM schemes, we study KZG commitments, which is one of the most common commitment scheme used for compiling PIOPs. We also briefly sketch a simple PCOM scheme built using compressed sigma protocols of [AC20]. For PIOP schemes, we study trapdoorless zero-knowledge for Marlin, Lunar, and Plonk. As a consequence of our framework, we conclude that a slight modification of Marlin and Lunar when compiled with the deterministic KZG commitment scheme are SIM-EXT. On the other hand, we show that Plonk needs a hiding version of KZG to be SIM-EXT.

1.2 Technical Overview

We provide a high-level overview of our modular approach towards simulation-extractability.

PIOP and zkSNARK Compiler Let us recap one of the popular paradigms of constructing efficient zkSNARKs [CHM+20, BFS20, CFF+21]. The compiler we study in this paper mostly follows the formalization of Marlin [CHM+20]. Our starting point is a *polynomial interactive oracle proof* (PIOP) system. This is a public-coin interactive protocol between prover $\mathbf{P}(\mathsf{x}, \mathsf{w})$ and verifier $\mathbf{V}(\mathsf{x})$, where x is a statement and w is a witness, respectively. For each round $i = 1, \ldots, \mathsf{r}$, $\mathbf{P}$ sends a *polynomial oracle* $p_i \in \mathbb{F}[X]$ and the verifier $\mathbf{V}$ responds with uniformly sampled challenge ρ_i.[4] The challenge strings $\rho_1, \ldots, \rho_\mathsf{r}$ are then used by $\mathbf{V}$ to derive *evaluation points* $z_1, \ldots, z_\mathsf{r}$, which are queried to the polynomial oracles. Upon receiving $y_i = p_i(z_i)$ for $i = 1, \ldots, \mathsf{r}$ from the oracles, $\mathbf{V}$ outputs a decision bit to accept or reject.

It is well known that the above information-theoretic object can be compiled into a non-interactive argument in the random oracle model, using a cryptographic primitive called *polynomial commitment (PCOM)*. The compilation is two fold. First, one constructs an interactive argument, where for each round i, prover $\mathcal{P}$ internally runs a PIOP prover $\mathbf{P}$ to obtain a polynomial p_i, generates a commitment c_i to p_i, and sends c_i to $\mathcal{V}$, and $\mathcal{V}$ responds with ρ_i generated by the PIOP verifier $\mathbf{V}$. At the end of interaction, $\mathcal{P}$ outputs y_i with *evaluation proofs* π_i guaranteeing that $p_i(z_i) = y_i$ w.r.t. committed polynomials p_i. $\mathcal{V}$ accepts if and only if $\mathbf{V}$ accepts *and* all evaluation proofs are valid. Notice that this protocol follows the typical format of public-coin interactive argument. Therefore, assuming access to random oracle H, one can construct a corresponding non-interactive argument Π by applying a Fiat-Shamir transform.

From Knowledge Soundness to Simulation-Extractability in the PIOP Paradigm Plain knowledge soundness of compiled protocol Π is already analyzed in the literature under various assumptions on PCOM and PIOP [CHM+20, BFS20, CFF+21, MBKM19, GWC19]. To benefit from existing knowledge soundness analyses in a black-box manner, our goal is to lift knowledge soundness to simulation-extractability (SIM-EXT) while being agnostic of concrete behaviors of knowledge extractor. Fortunately, Ganesh et al. [GOP+23] recently proved that a property called *weak unique response (WUR)* is sufficient for Fiat-Shamir non-interactive arguments to have SIM-EXT in the ROM. Essentially, the WUR property of [GOP+23] requires the following: given a transcript $\pi = (a_1, \rho_1, \ldots, a_\mathsf{r}, \rho_\mathsf{r}, a_{\mathsf{r}+1})$ output by simulator

[4] To sketch the core ideas, we provide a simplified version of PIOP where each round involves a single polynomial. Here, we also ignore the role of preprocessing for now. In the detailed proof, we deal with a more general case with multiple polynomials in every round, and the preprocessing phase, and multiple evaluation points for each polynomial.

$\mathcal{S}$ for Fiat-Shamir non-interactive argument (constructed from $(2\mathsf{r}+1)$-move multi-round public-coin interactive argument), for any $i \geq 2$, it is computationally hard to come up with another transcript π' with *shared prefix*, i.e., $\pi' = (a_1, \rho_1, \ldots, a'_i, \rho'_i, \ldots, a'_{\mathsf{r}}, \rho'_{\mathsf{r}}, a'_{\mathsf{r}+1})$ such that $a'_i \neq a_i$.

However, their general theorem only covers a *transparent* case, while many recent PIOP-based zk-SNARKs require trusted generation of SRS in addition to the random oracle (e.g., if PCOM is instantiated with the well-known KZG scheme). It turns out that dependency on both SRS and RO introduces additional challenges in proving SIM-EXT in a modular fashion. In more detail, to invoke the general theorem similar to [GOP+23], one must show the existence of *trapdoor-less* ZK (TLZK) simulator, which only makes use of programmability of RO but without the knowledge of simulation trapdoor for SRS. We formalize this observation in Lemma 2.16. As the existing compiler theorems (such as Marlin and Lunar) do show zero-knowledge with trapdoor, we need an alternative way to prove zero-knowledge. In this work, we show two strategies of TLZK simulation, depending on the power of underlying PIOP simulator. The first path is straightforward: it requires honest verifier zero-knowledge (HVZK) of PIOP and hiding of PCOM similar to the Marlin compiler. As an alternative approach, if PIOP tolerates Ψ additional evaluations on polynomials which are *not* asked by honest PIOP verifier (Ψ-HVZK), then we show only a weak variant of hiding from PCOM is sufficient for Π to be TLZK. This is an observation implicit in several practical constructions, but to the best of our knowledge, no previous compiler theorem explicitly formalized it.

Proving Weak Unique Response for the Compiled Protocol Given a TLZK simulator, our goal is to identify a set of properties allowing us to prove WUR. Recall that a transcript of Π is comprised of

$$(c_1, \rho_1, \ldots, c_{\mathsf{r}}, \rho_{\mathsf{r}}, (y_1, \ldots, y_{\mathsf{r}}), (\pi_1, \ldots, \pi_{\mathsf{r}}))$$

where $\rho_i = \mathsf{H}(\mathsf{srs}, \mathsf{i}, \mathsf{x}, c_1, \rho_1, \ldots, c_i)$. Focusing on the last round response, one can immediately see the need for unique proofs (i.e., for a fixed (c_i, z_i, y_i), there exists a unique proof π_i that verifies) and evaluation binding (i.e., for a fixed (c_i, z_i), there exists unique evaluation outcome y_i that verifies); otherwise, a cheating prover can maul either π_i or y_i of an existing transcript to create a valid transcript without knowing witness for x. We show that these mild properties are already satisfied by KZG which is the most common commitment scheme used for compilation. The hardest part is to prove that an adversary cannot maul a response in the middle of the transcript. Our crucial observation is that the compiler has a built-in mechanism similar to one-time signature, making it difficult for the prover to forge any part of the transcript. In more detail, if any prefix of the transcript is modified, then it inevitably triggers re-sampling of the final Fiat-Shamir challenge, leading to $\rho'_{\mathsf{r}} \neq \rho_{\mathsf{r}}$ with overwhelming probability. Since ρ'_{r} is used as a random coin to derive evaluation points $z'_1, \ldots, z'_{\mathsf{r}}$, without loss of generality, a cheating prover is forced to create a valid evaluation proof for c_1 w.r.t. an evaluation point $z'_1 \neq z_1$.[5] However, if p_1 is randomized enough and the commitment c_1 together with an evaluation proof π_i leaks no more information than $p_1(z_1)$, then it must be hard for an adversary to extrapolate valid evaluation proof for $p_1(z'_1)$ w.r.t. c_1. We formalize this intuition assuming the same evaluation binding and weak hiding assumptions.

1.3 Related Work

Broadly, there are currently three approaches to obtain simulation extractable SNARK:

[5] As we discuss in Section 5, some PIOP protocols do not use the last round challenge ρ_{r} to derive z_1. However, one can cheaply patch them by introducing a random dummy polynomial p' in the first round and having the verifier query p' with a fresh evaluation point derived from ρ_{r}. Note that this can also be seen as a generic method to add weak unique response to *any* Fiat-Shamir NIZKAoK in the ROM.

Generic Lifting Techniques for SIM-EXT NIZK Classically, it is well-known that any sound NIZK proof can be lifted to SIM-EXT NIZK in a general manner. For example, De Santis et al. [DDO+01] combine NIZK for a language L with one-time signature and PRF to realize SIM-EXT by having prover generate a proof for an extended OR-language L' related to the original language L. More recent lifting compilers [KZM+15, ARS20, AGRS23] optimize the approach along these lines and further add *black-box and straight-line knowledge extraction* in order to achieve universally composable [Can01] zkSNARKs, but still introduce performance overhead in the size of SRS, proof size, and proving/verification time. Somewhat related, [GKO+23] introduced a straight-line extraction compiler to lift zkSNARKs which already satisfies SIM-EXT to UC-secure zkSNARKs in the random oracle model, while preserving the asymptotic succinctness of the output proof size.

Scheme Specific Techniques The second is to prove directly that an existing SNARK scheme is simulation extractable [GM17, BKSV21, GKK+22, GOP+23, DG23]. This approach is taken by [GKK+22] for Plonk, Marlin, and Sonic, by [GOP+22, GOP+23] for Bulletproofs, and by [DG23] for Bulletproofs and Spartan, respectively. As these two works share close resemblance with ours, we investigate their strategy for proving specific schemes in some more detail.

All of [GKK+22, GOP+23, DG23] provide very similar frameworks for showing SIM-EXT for zkSNARKs obtained as Fiat-Shamir compiled multi-round protocols (in the updatable SRS setting for [GKK+22] and in the transparent setting for [GOP+23, DG23]). They observe that it suffices to prove that the schemes in question satisfy additional properties that together with existing properties, typically knowledge soundness, imply SIM-EXT. These properties are TLZK, and their specific variants of unique responses (UR). However, each of these properties is directly tied to the compiled zkSNARKs, and as such they have to be analyzed for each scheme separately, e.g., they provide three TLZK simulator and prove that they achieve zero-knowledge for Plonk, Marlin, and Sonic (in [GKK+22]), and prove the unique response property for each scheme (in all works). Moreover, [GKK+22] only considers zkSNARKs whose knowledge soundness proof is based on rewinding in the random oracle.

Our framework simplifies and generalizes the SIM-EXT analysis. We show that the PIOP to zkSNARK compiler [CHM+20] already achieves the additional TLZK and WUR under milder assumptions on the polynomial commitment. TLZK and WUR together with knowledge soundness then imply simulation-extractability.

Another difference from the work of [GKK+22] is in the simulation strategies for TLZK. We observe that the simulation for Plonk as presented in [GKK+22] is flawed as it only works for perfectly hiding commitments, and we explain more in Section 5.

SE-SNARKS from PIOP and polynomial commitments Our result thus falls into a third category which proves SIM-EXT for a PIOP to zkSNARK compiler rather than for individual schemes. As does the following concurrent work.

Concurrent work Faonio et al. [FFK+23] study simulation-extractability (SE) of zkSNARKs constructed from polynomial IOP and polynomial commitment. Their main goal is to identify properties of polynomial commitment such that a compiled zkSNARK satisfies simulation-extractability. Along the way, they define SE tailored to a polynomial commitment parameterized by a *policy predicate* Φ. The policy specifies additional conditions, on top of requiring a valid non-extractable proof, to determine the success of the adversary and is based on the following properties and variables in the SE game: (1) public parameters and honestly sampled commitments, (2) an adversarially created forgery (x, π), (3) the adversary's view, including the set of statement-proof pairs recorded by the simulation oracle, and the set of query-response pairs recorded by the random oracle, (4) auxiliary information which comes along with the forged instance. As a concrete example, they prove the KZG commitment is SE in the AGM w.r.t. a specific class of Φ and this implies SE of several

existing zkSNARKs including Plonk and Marlin. In contrast, our approach to SE analysis of zkSNARKs only requires simpler, easy-to-state properties of polynomial commitment, namely, evaluation binding, unique proof and hiding. We expect that such properties are satisfied by other polynomial commitment than KZG such as inner-product based (IPA) polynomial commitments. Our analysis is also agnostic of the type of extractor for polynomial commitment thanks to the modularity of the generic result proved in [GOP+23].

Intuition from proofs We note that these two works take different routes to the same destination.[6] It is natural, that complex theorems can have multiple, conceptually very different proofs that yield different insights. Our intuition for where the proofs depart—early on, is that their work strengthens the extraction property of the polynomial commitment scheme to also work in the presence of a simulator using the secret trapdoor. In contrast, our work strengthens the zero-knowledge property and requires a simulator that does not have access to the trapdoor. Note that both works resort to random oracle programming for simulation.

More superficially, their polynomial IOP model stems from Lunar's PHP model [CFF+21] from where they also take the inspiration of treating polynomial commitments as commit-and-prove SNARKs, while our PIOP are inspired by Marlin's AHP model [CHM+20].

2 Preliminaries

We assume $[\ell]$ to denote integers $\{1, \ldots, \ell\}$ and $[k, \ell]$ to denote $\{k, \ldots, \ell\}$ for $k < \ell$. The security parameter is denoted by λ. A function $f(\lambda)$ is negligible in λ if for any polynomial $\mathsf{poly}(\lambda)$, $f(\lambda) \leq 1/\mathsf{poly}(\lambda)$ for sufficiently large λ. We denote $f(\lambda) \leq \mathsf{negl}(\lambda)$ to indicate f is negligible. By $y \xleftarrow{\rho} \mathcal{A}(x)$, we mean that a randomized algorithm $\mathcal{A}$ outputs y on input x using a random coin ρ sampled uniformly from a randomness space. For a finite field $\mathbb{F}$, $\mathbb{F}^{\mathsf{d}}[X]$ denotes a set of polynomials over $\mathbb{F}$ of degree at most d.

2.1 Relations

An *indexed relation* $\hat{\mathcal{R}}$ is a set of triples $(\mathsf{i}, \mathsf{x}, \mathsf{w})$ where i is the index, x is the instance, and w is the witness. We assume $\hat{\mathcal{R}}$ can be partitioned using the security parameter $\lambda \in \mathbb{N}$ (e.g., by including the description of field $\mathbb{F}$ such that $|\mathbb{F}|$ is determined by a function of λ). Given a security parameter $\lambda \in \mathbb{N}$, we denote by $\hat{\mathcal{R}}_\lambda$ the restriction of $\hat{\mathcal{R}}$ to triples $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \hat{\mathcal{R}}$ with appropriate length in λ. Given a fixed index i, we denote by $\hat{\mathcal{R}}_\mathsf{i}$ the restriction of $\hat{\mathcal{R}}$ to $\left\{(\mathsf{x}, \mathsf{w}) \ : \ (\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \hat{\mathcal{R}}\right\}$. Given an indexed relation $\hat{\mathcal{R}}$, the corresponding *binary relation* can be defined as $\mathcal{R} = \left\{((\mathsf{i}, \mathsf{x}), \mathsf{w}) \ : \ (\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \hat{\mathcal{R}}\right\}$.

Typically, i describes an arithmetic circuit over a finite field, x denotes public inputs, and w denotes private inputs, respectively. In the rest of this paper, we assume i and x to include the description of finite field $\mathbb{F}$ for the sake of simplicity, but our result holds even if the circuit is over a ring or module.

2.2 Polynomial Interactive Oracle Proofs

We define polynomial interactive oracle proofs (PIOP) with preprocessing. The formulation below is highly inspired by algebraic holographic proofs (AHP) [CHM+20]. We apply the following minor modifications: (1) Interaction starts with prover's message, instead of verifier's public coin. (2) We introduce an additional parameter t to allow multiple queries to a single polynomial. (3) We assume a single maximum degree bound d rather than a distinct bound for each polynomial oracle (following the PIOP formulation of [BFS20]), since the

[6] As fellow travelers we have open lines of communications which even resulted in an author overlap.

degree bound check can be incorporated into PIOP by having prover output an oracle with shifted polynomial.

Definition 2.1 (Polynomial IOP). *A* polynomial interactive oracle proof (PIOP) *for an indexed relation $\hat{\mathcal{R}}$ is specified by a tuple* PIOP $= (\mathsf{r}, \mathsf{s}, \mathsf{t}, \mathsf{d}, \mathbf{I}, \mathbf{P}, \mathbf{V})$*, where* $\mathsf{r}, \mathsf{s}, \mathsf{t}, \mathsf{d} : \{0,1\}^* \to \mathbb{N}$ *are polynomial-time computable functions and* $\mathbf{I}, \mathbf{P}, \mathbf{V}$ *are three algorithms known as the* indexer*,* prover*, and* verifier*. The parameter* r *specifies the number of interaction rounds,* s *specifies the number of polynomials in each round,* t *specifies the number of queries made to each polynomial, and* d *specifies a maximum degree bound on these polynomials. An execution of* PIOP $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \hat{\mathcal{R}}$ *involves interaction between* $\mathbf{P}$ *and* $\mathbf{V}$*, where* $b \leftarrow \langle \mathbf{P}(\mathsf{i}, \mathsf{x}, \mathsf{w}), \mathbf{V}^{\mathbf{I}(\mathsf{i})}(\mathsf{x}) \rangle$ *denotes the output decision bit, and* $(\mathsf{view}; \mathbf{p}) \leftarrow [\![\mathbf{P}(\mathsf{i}, \mathsf{x}, \mathsf{w}), \mathbf{V}^{\mathbf{I}(\mathsf{i})}(\mathsf{x})]\!]$ *denotes the view (*view*) of* $\mathbf{V}$ *generated during the interaction and the responses of* $\mathbf{I}(\mathsf{i})$*, and the polynomial oracles (*$\mathbf{p}$*) output by* $\mathbf{P}$*. The* view *consists of challenges* $\rho_1, \ldots, \rho_{\mathsf{r}}$ *that* $\mathbf{V}$ *sends to* $\mathbf{P}$ *and vector* $\mathbf{y}$ *of oracle responses defined below. The vector* $\mathbf{p}$ *consists of the polynomial oracles generated by* $\mathbf{P}$ *during the interaction. An execution of* PIOP *proceeds as follows:*

- ***Offline phase*** *The indexer* $\mathbf{I}$ *receives as input index* i *for* $\hat{\mathcal{R}}$*, and outputs* $\mathsf{s}(0)$ *polynomials* $p_{0,1}, \ldots, p_{0,\mathsf{s}(0)} \in \mathbb{F}[X]$ *of degrees at most* $\mathsf{d}(|\mathsf{i}|)$*. Note that the offline phase does not depend on any particular instance or witness, and merely considers the task of encoding the given index* i*.*
- ***Online phase*** *Given an instance* x *and witness* w *such that* $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \hat{\mathcal{R}}$*, the prover* $\mathbf{P}$ *receives* $(\mathsf{i}, \mathsf{x}, \mathsf{w})$ *and the verifier* $\mathbf{V}$ *receives* x *and oracle access to the polynomials output by* $\mathbf{I}(\mathsf{i})$*. The prover* $\mathbf{P}$ *and the verifier* $\mathbf{V}$ *interact over* $(2\mathsf{r}+1)$ *rounds where* $\mathsf{r} = \mathsf{r}(|\mathsf{i}|)$*. For* $i \in [\mathsf{r}]$*, in the* i*-th round of interaction, first the prover* $\mathbf{P}$ *sends* $\mathsf{s}(i)$ *oracle polynomials* $p_{i,1}, \ldots, p_{i,\mathsf{s}(i)} \in \mathbb{F}[X]$ *to the verifier* $\mathbf{V}$*; and* $\mathbf{V}$ *replies with a challenge* $\rho_i \in \mathsf{Ch}$*, where* Ch *is the challenge space determined by* i*. The last round challenge* $\rho_{\mathsf{r}} \in \mathsf{Ch}$ *serves as auxiliary input to* $\mathbf{V}$ *in subsequent phases. We assume the protocol to be* public-coin*, meaning that* ρ_i*'s are public and uniformly sampled from* Ch*. Moreover, observe that* $\mathbf{P}$ *can be interpreted as a series of next message functions such that polynomial oracles for round* i *are obtained by running* $(\mathsf{st}_P, p_{i,1}, \ldots, p_{i,\mathsf{s}(i)}) \leftarrow \mathbf{P}(\mathsf{st}'_P, \rho_{i-1})$*, where* st'_P *is the internal state of* $\mathbf{P}$ *after sending polynomials for round* $i-1$ *and before receiving challenge* ρ_{i-1}*, and* st_P *is the updated state. Here,* ρ_0 *is assumed to be* $\perp$*.*
- ***Query phase*** *Let* $\mathbf{p} = (p_{i,j})_{i \in [0,\mathsf{r}], j \in [\mathsf{s}(i)]}$ *be a vector consisting of all polynomials sent by the prover* $\mathbf{P}$*. The verifier may query any of the polynomials it has received any number of times. Concretely,* $\mathbf{V}$ *executes a subroutine* $\mathbf{Q}_{\mathbf{V}}$ *that receives* $(\mathsf{x}; \rho_1, \ldots, \rho_{\mathsf{r}})$ *and outputs a query vector* $\mathbf{z} = (\mathbf{z}_{i,j})_{i \in [0,\mathsf{r}], j \in [\mathsf{s}(i)]}$*, where each* $\mathbf{z}_{i,j}$ *is to be interpreted as a vector* $(z_{i,j,k})_{k \in [\mathsf{t}(i,j)]} \in \mathbb{D}^{\mathsf{t}(i,j)}$ *and* $\mathbb{D} \subseteq \mathbb{F}$ *is an evaluation domain determined by* i*. We write "*$\mathbf{y}_{i,j} = p_{i,j}(\mathbf{z}_{i,j})$*" to define an evaluation vector* $\mathbf{y}_{i,j} = (y_{i,j,k})_{k \in [\mathsf{t}(i,j)]}$ *where* $y_{i,j,k} = p_{i,j}(z_{i,j,k})$*. Likewise, we write "*$\mathbf{y} = \mathbf{p}(\mathbf{z})$*" to define* $\mathbf{y} = (\mathbf{y}_{i,j})_{i \in [0,\mathsf{r}], j \in [\mathsf{s}(i)]}$ *where* $\mathbf{y}_{i,j} = p_{i,j}(\mathbf{z}_{i,j})$*.*
- ***Decision phase*** *The verifier outputs "accept" or "reject" based on the answers to the queries (and the verifier's randomness). Concretely,* $\mathbf{V}$ *executes a subroutine* $\mathbf{D}_{\mathbf{V}}$ *that receives* $(\mathsf{x}, \mathbf{p}(\mathbf{z}); \rho_1, \ldots, \rho_{\mathsf{r}})$ *as input, and outputs the decision bit* b*.*

  The function d *determines which provers to consider for the completeness and soundness properties of the proof system. In more detail, we say that a (possibly malicious) prover* $\tilde{\mathbf{P}}$ *is* admissible *for* PIOP *if, on every interaction with the verifier* $\mathbf{V}$*, it holds that for every round* $i \in [\mathsf{r}]$ *and oracle index* $j \in [\mathsf{s}(i)]$ *we have* $\deg(p_{i,j}) \leq \mathsf{d}(|\mathsf{i}|)$*. The honest prover* $\mathbf{P}$ *is required to be admissible under this definition.*

Typically PIOP should satisfy completeness, (knowledge) soundness and zero-knowledge as defined below.

Definition 2.2 (Completeness). *A* PIOP *is complete if for any* $(\mathsf{i},\mathsf{x},\mathsf{w}) \in \hat{\mathcal{R}}$,

$$\Pr\left[b \leftarrow \langle \mathbf{P}(\mathsf{i},\mathsf{x},\mathsf{w}) \, , \, \mathbf{V}^{\mathbf{I}(\mathsf{i})}(\mathsf{x})\rangle \; : \; b = 1\right] = 1$$

Definition 2.3 (Knowledge Soundness). *A* PIOP *for* $\hat{\mathcal{R}}$ *is* knowledge sound*, if there exists a* PPT *extractor* $\mathbf{E}$ *such that for all admissible* $\mathbf{A}$*, every index* i*, statement* x*,*

$$\mathbf{Adv}^{\mathsf{KS}}_{\mathbf{A}}(\lambda) := \Pr\left[b \leftarrow \langle \mathbf{A}(\mathsf{i},\mathsf{x}) \, , \, \mathbf{V}^{\mathbf{I}(\mathsf{i})}(\mathsf{x})\rangle; \mathsf{w}^* \leftarrow \mathbf{E}^{\mathbf{A}}(\mathsf{i},\mathsf{x}) \; : \; b = 1 \wedge (\mathsf{i},\mathsf{x},\mathsf{w}^*) \notin \hat{\mathcal{R}}_\lambda\right] \leq \mathsf{negl}(\lambda)$$

where $\mathbf{E}^{\mathbf{A}}$ *denotes that* $\mathbf{E}$ *has black-box access to the next-message function of* $\mathbf{A}$*.*

Remark 2.4. Although in this work we do not discuss how the compiler preserves knowledge soundness, we provide a definition of knowledge soundness for completeness. If r is non-constant, the compiler typically requires a stronger property called *state-restoration* knowledge soundness [BCS16] for the resulting non-interactive argument to satisfy knowledge soundness in the random oracle model.

Now we define a stronger notion of honest verifier zero knowledge (HVZK) for PIOP. First, a straightforward HVZK asks for simulatability of the view of honest verifier $\mathbf{V}$ which comprises of all public coins $\rho_1, \ldots, \rho_{\mathsf{r}}$ and the outcome of evaluations $\mathbf{y}$. It turns out that, if compiled with a non-hiding commitment scheme, the committing function leaks additional evaluations of polynomials which are not queried by an honest PIOP verifier $\mathbf{V}$ (modeled as $\mathbf{p}(\chi)$ below). In order to tolerate such additional leakages, we consider the existence of a more powerful simulator that, along with the proof string, is also able to output some polynomials, such that even after providing additional evaluations w.r.t. these polynomials, the view remains indistinguishable from the real execution.

Note that Lunar [CFF+21] also models a similar notion where zero-knowledge should hold even when the proof might leak some additional evaluation points. However, since their simulation strategy crucially uses the trapdoor information in order to satisfy this notion, our definition is stronger and harder to achieve. One interesting motivation that suggests that we need our stronger formulation is that Plonk [GWC19] (as presented in [GKK+22]) happens to satisfy Lunar's definition but not ours. However, the trapdoorless simulation based on deterministic commitments, as suggested in [GKK+22], is flawed (as we expand more in Section 5). This suggests that Lunar's formulation of leakage is not enough in the context of trapdoorless zero-knowledge; we require something stronger.

We also require the PIOP to satisfy a second condition called *non-extrapolatable first polynomial*. Roughly, it says that there is enough randomness in the first online polynomial p so that, given a certain number of evaluations, the next evaluation remains unpredictable. In this intuition, we implicitly assume that the first online polynomial of the PIOP encodes the witness somehow. Note that this requirement is very easily satisfied by most PIOPs already, since they do encode the witness in the first round polynomials with enough randomness used in the encoding to achieve zero-knowledge.

Definition 2.5 (Ψ-Honest Verifier Zero Knowledge (HVZK)). *Let* PIOP *be a polynomial IOP for relation* $\hat{\mathcal{R}}$*. Let* $\mathbb{D}$ *denote the domain of honest polynomial oracle queries. Let* $\chi = (\chi_{i,j})_{i\in[\mathsf{r}],j\in[\mathsf{s}(i)]}$ *denote an auxiliary query vector which is said to be valid if for each* $i \in [\mathsf{r}], j \in [\mathsf{s}(i)]$*,* $|\chi_{i,j}| \leq \Psi$ *and each query in* $\chi_{i,j}$ *comes from* $\mathbb{D}$*.* PIOP *is* statistical Ψ-honest verifier zero knowledge*, if there exists a* PPT *simulator* $\mathbf{S}$ *such that for any distinguisher* $\mathbf{A}$*, and for all valid auxiliary query vectors* χ*, it holds that*

$$\mathbf{Adv}^{\Psi\text{-}\mathit{HVZK}}_{\mathbf{A}}(\lambda,\chi) := \left|\Pr\left[\mathit{HVZK\text{-}0}_{\mathbf{A}}(1^\lambda,\chi) = 0\right] - \Pr\left[\mathit{HVZK\text{-}1}_{\mathbf{A}}(1^\lambda,\chi) = 0\right]\right| \leq \mathsf{negl}(\lambda)$$

Game 1: HVZK for Polynomial IOP

$\underline{\mathsf{HVZK\text{-}0}_{\mathbf{A}}(1^\lambda, \chi)}$
1: $b \leftarrow \mathbf{A}^{\mathbf{O}_0}(1^\lambda)$
2: **return** b

$\underline{\mathsf{NEXP\text{-}1}_{\mathbf{A}}(1^\lambda, \chi)}$
1: $(\mathsf{i}, \mathsf{x}) \leftarrow \mathbf{A}_1(1^\lambda)$
2: $(\mathsf{view}, \mathbf{p}(\chi), \mathbf{p}(\mathbf{z})) \leftarrow \mathbf{O}'_1(\mathsf{i}, \mathsf{x})$
3: $(z^*, y^*) \leftarrow \mathbf{A}_2(\mathsf{view}, \mathbf{p}(\chi), \mathbf{p}(\mathbf{z}))$
4: $b := (z^* \notin (\mathbf{z}, \chi)) \ \wedge \ (y^* = p(z^*))$
5: **return** b

$\underline{\mathbf{O}_0(\mathsf{i}, \mathsf{x}, \mathsf{w})}$
1: **if** $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \notin \hat{\mathcal{R}}$ **then return** $\perp$
2: $(\mathsf{view}; \mathbf{p}) \leftarrow [\![\mathbf{P}(\mathsf{i}, \mathsf{x}, \mathsf{w}), \mathbf{V}^{\mathbf{I}(\mathsf{i})}(\mathsf{x})]\!]$
3: $(\rho_1, \ldots, \rho_\mathsf{r}, \mathbf{y}) := \mathsf{view}$
4: $\mathbf{z} \leftarrow \mathbf{Q}_\mathbf{V}(\mathsf{x}; \rho_1, \ldots, \rho_\mathsf{r})$
5: **return** $(\mathsf{view}, \mathbf{p}(\chi), \mathbf{p}(\mathbf{z}))$

$\underline{\mathsf{HVZK\text{-}1}_{\mathbf{A}}(1^\lambda, \chi)}$
1: $b \leftarrow \mathbf{A}^{\mathbf{O}_1}(1^\lambda)$
2: **return** b

$\underline{\mathbf{O}_1(\mathsf{i}, \mathsf{x}, \mathsf{w})}$
1: **if** $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \notin \hat{\mathcal{R}}$ **then return** $\perp$
2: $(\mathsf{view}; \mathbf{p}) \leftarrow \mathbf{S}(\mathsf{i}, \mathsf{x})$
3: $(\rho_1, \ldots, \rho_\mathsf{r}, \mathbf{y}) := \mathsf{view}$
4: $\mathbf{z} \leftarrow \mathbf{Q}_\mathbf{V}(\mathsf{x}; \rho_1, \ldots, \rho_\mathsf{r})$
5: **if** $\exists p_{i,j} \in \mathbf{p}, \deg(p_{i,j}) > \mathsf{d}(|\mathsf{i}|)$ **then return** $\perp$
6: **return** $(\mathsf{view}, \mathbf{p}(\chi), \mathbf{p}(\mathbf{z}))$

$\underline{\mathbf{O}'_1(\mathsf{i}, \mathsf{x})}$
1: $(\mathsf{view}; \mathbf{p}) \leftarrow \mathbf{S}(\mathsf{i}, \mathsf{x})$
2: $(\rho_1, \ldots, \rho_\mathsf{r}, \mathbf{y}) := \mathsf{view}$
3: $\mathbf{z} \leftarrow \mathbf{Q}_\mathbf{V}(\mathsf{x}; \rho_1, \ldots, \rho_\mathsf{r})$
4: **if** $\exists p_{i,j} \in \mathbf{p}, \deg(p_{i,j}) > \mathsf{d}(|\mathsf{i}|)$ **then return** $\perp$
5: **return** $(\mathsf{view}, \mathbf{p}(\chi), \mathbf{p}(\mathbf{z}))$

where HVZK-0 and HVZK-1 are defined in Game 1. If the operations highlighted in orange are not executed, then we simply say PIOP *satisfies HVZK.*

Definition 2.6 (Ψ-Non-Extrapolation for the First Polynomial). *Let* PIOP, $\mathbb{D}$, *and* χ *be as defined in Definition 2.5. Let* $\mathbf{S}$ *be the statistical* Ψ*-honest verifier zero knowledge simulator for the* PIOP, *and* t *be the number of distinct evaluations revealed for each polynomial in a* PIOP *proof. Let* $\mathbf{p}$ *be the vector of polynomials output by* $\mathbf{S}$, *and* p *denotes the first polynomial for the online phase in* $\mathbf{p}$. p *is said to be* Ψ-Non-Extrapolatable *if for any adversary* $\mathbf{A} := (\mathbf{A}_1, \mathbf{A}_2)$ *(where* $\mathbf{A}_1, \mathbf{A}_2$ *share internal state), and for all valid auxiliary query vectors* χ *(as defined in Definition 2.5), it holds that,*

$$\mathbf{Adv}^{\Psi\text{-}\mathsf{NEXP}}_{\mathbf{A}}(\lambda, \chi) := \Pr\left[\mathsf{NEXP\text{-}1}_{\mathbf{A}}(1^\lambda, \chi) = 1\right] \leq \mathsf{negl}(\lambda)$$

where NEXP-1 is define in Game 1.

Min-entropy of $\mathbf{Q}_\mathbf{V}$. Recall, the sub-routine $\mathbf{Q}_\mathbf{V}$ takes input $(\mathsf{x}; \rho_1, \ldots, \rho_\mathsf{r})$ and outputs a query vector $\mathbf{z} = (\mathbf{z}_{i,j})_{i \in [0, \mathsf{r}, j \in [\mathsf{s}(i)]}$ where each $\mathbf{z}_{i,j}$ can be parsed as a vector $(z_{i,j,k})_{k \in [\mathsf{t}(i,j)]}$. We isolate one evaluation point $z_{1,1,1}$ for $p_{1,1}$ and require that $z_{1,1,1}$ is equal to some fixed value only with negligible probability. This is captured by assessing min-entropy of the $\mathbf{Q}_\mathbf{V}$ algorithm.

Definition 2.7 (Min-entropy of $\mathbf{Q}_\mathbf{V}$). *Let* PIOP $= (\mathsf{r}, \mathsf{s}, \mathsf{t}, \mathsf{d}, \mathbf{I}, \mathbf{P}, \mathbf{V})$ *be a PIOP for indexed relation* $\hat{\mathcal{R}}$. $(\mathbf{Q}_\mathbf{V}, \mathbf{D}_\mathbf{V})$ *denote the subroutines run by* $\mathbf{V}$. *Let* Ch *be the challenge space from which* $\mathbf{V}$ *samples* ρ_i. *For any fixed* $\lambda \in \mathbb{N}$ *and for any* $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \hat{\mathcal{R}}_\lambda$, *consider the maximum probability that* $z_{1,1,1}$ *is equal to a particular value:*

$$\mu(\lambda, \mathsf{x}) = \max_{\rho_1, \ldots, \rho_{\mathsf{r}-1} \in \mathsf{Ch}, a \in \mathbb{F}} \Pr\left[\rho_\mathsf{r} \xleftarrow{\$} \mathsf{Ch}; \mathbf{z} \leftarrow \mathbf{Q}_\mathbf{V}(\mathsf{x}; \rho_1, \ldots, \rho_\mathsf{r}) \ : \ z_{1,1,1} = a\right].$$

The min-entropy α *of sub-routine* $\mathbf{Q_V}$ *is*

$$\alpha(\lambda) = \min_{(\mathsf{i},\mathsf{x},\mathsf{w})\in\hat{\mathcal{R}}_\lambda} (-\log_2(\mu(\lambda,\mathsf{x}))).$$

We say the min-entropy of $\mathbf{Q_V}$ *is* high *if* $\alpha \in \omega(\log(\lambda))$.

Remark 2.8. As we discuss in Section 5, if the PIOP of interest does not have high min-entropy of $\mathbf{Q_V}$ and/or Ψ-non-extrapolatable first polynomial, one can easily patch in the following manner: a modified prover $\mathbf{P}$ additionally sends a dummy random polynomial $p^* \in \mathbb{F}^{\Psi+1}[X]$ in the first round and the verifier $\mathbf{V}$ queries p^* with a fresh evaluation point $z^* \in \mathbb{D}$ derived from ρ_{r}.

2.3 Non-Interactive Argument and Simulation-Extractability in ROM

Below we write $\mathcal{A}^{\mathsf{H}}$ to denote that an algorithm $\mathcal{A}$ has black-box access to the random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^l$.

Definition 2.9 (Non-Interactive Argument (NARG)). *A Non-Interactive Argument with Universal SRS (henceforth NARG) in the random oracle model for binary relation* $\mathcal{R}$ *is a tuple* $\Pi_{\mathsf{H}} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *of three algorithms:*

- $\mathsf{srs} \leftarrow \mathcal{G}(1^\lambda)$ *is a setup algorithm that samples a* structured reference string srs.
- $\pi \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{srs},\mathsf{x},\mathsf{w})$ *is a prover that outputs a proof* π *asserting* $(\mathsf{x},\mathsf{w}) \in \mathcal{R}$. *If* $(\mathsf{x},\mathsf{w}) \notin \mathcal{R}$, $\mathcal{P}$ *outputs* $\perp$.
- $b \leftarrow \mathcal{V}^{\mathsf{H}}(\mathsf{srs},\mathsf{x},\pi)$ *is a verifier that outputs a decision bit* $b \in \{0,1\}$.

To explicitly model preprocessed SRS in later sections, we also introduce indexed NARG (henceforth iNARG) $\hat{\Pi}$ for indexed relation $\hat{\mathcal{R}}$.

Definition 2.10 (Indexed Non-Interactive Argument (iNARG)). *An Indexed Non-Interactive Argument with Universal SRS in the random oracle model for indexed relation* $\hat{\mathcal{R}}$ *is a tuple* $\hat{\Pi}_{\mathsf{H}} = (\mathcal{G}, \mathcal{I}, \hat{\mathcal{P}}, \hat{\mathcal{V}})$ *of three algorithms:*

- $\mathsf{srs} \leftarrow \mathcal{G}(1^\lambda)$ *works as* $\mathcal{G}$ *of NARG.*
- $(\mathsf{ipk},\mathsf{ivk}) \leftarrow \mathcal{I}(\mathsf{i},\mathsf{srs})$ *is a deterministic* indexer[7] *that takes index* i *and* srs *as input, and produces a proving index key (*ipk*) and a verifier index key* (ivk)*, used respectively by* $\hat{\mathcal{P}}$ *and* $\hat{\mathcal{V}}$.
- $\pi \leftarrow \hat{\mathcal{P}}^{\mathsf{H}}(\mathsf{ipk},\mathsf{x},\mathsf{w})$ *is a prover that outputs a proof* π *asserting* $(\mathsf{i},\mathsf{x},\mathsf{w}) \in \hat{\mathcal{R}}$.
- $b \leftarrow \hat{\mathcal{V}}^{\mathsf{H}}(\mathsf{ivk},\mathsf{x},\pi)$ *is a verifier that outputs a decision bit* b.

It is easy to convert $\hat{\Pi}$ for $\hat{\mathcal{R}}$ into the corresponding Π for binary relation $\mathcal{R} = \left\{((\mathsf{i},\mathsf{x}),\mathsf{w}) \ : \ (\mathsf{i},\mathsf{x},\mathsf{w}) \in \hat{\mathcal{R}}\right\}$ by defining $\mathcal{P}^{\mathsf{H}}(\mathsf{srs},(\mathsf{i},\mathsf{x}),\mathsf{w})$ to be an algorithm outputting π after running $(\mathsf{ipk},\mathsf{ivk}) \leftarrow \mathcal{I}(\mathsf{srs},\mathsf{i})$ and $\pi \leftarrow \hat{\mathcal{P}}^{\mathsf{H}}(\mathsf{ipk},\mathsf{x},\mathsf{w})$, and $\mathcal{V}^{\mathsf{H}}(\mathsf{srs},(\mathsf{i},\mathsf{x}),\pi)$ to be an algorithm outputting b after running $(\mathsf{ipk},\mathsf{ivk}) \leftarrow \mathcal{I}(\mathsf{srs},\mathsf{i})$ and $b \leftarrow \hat{\mathcal{V}}^{\mathsf{H}}(\mathsf{ivk},\mathsf{x},\pi)$, respectively. Therefore, we only state security properties for NARG without loss of generality.

Definition 2.11 (Completeness). *A NARG* $\Pi_{\mathsf{H}} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *for relation* $\mathcal{R}$ *satisfies* perfect completeness *if for all* λ*, all* $N \in \mathbb{N}$*, and for all PPT adversaries* $\mathcal{A}$ *it holds that*

$$\Pr\left[\mathsf{srs} \leftarrow \mathcal{G}(1^\lambda); (\mathsf{x},\mathsf{w}) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathsf{srs}); \pi \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{srs},\mathsf{x},\mathsf{w}) \ : \ \mathcal{V}^{\mathsf{H}}(\mathsf{srs},\mathsf{x},\pi) = 1 \wedge (\mathsf{x},\mathsf{w}) \in \mathcal{R}\right] = 1.$$

[7] In the literature, indexer is also referred to as Derive algorithm.

Game 2: NIZK

$\underline{\mathsf{NIZK\text{-}0}_{\mathcal{A}}(1^\lambda)}$
1: $\mathsf{srs} \leftarrow \mathcal{G}(1^\lambda)$
2: $b \leftarrow \mathcal{A}^{\mathsf{H},\mathcal{P}^{\mathsf{H}}(\mathsf{srs},\cdot,\cdot)}(\mathsf{srs})$
3: **return** b

$\underline{\mathsf{NIZK\text{-}1}_{\mathcal{A}}(1^\lambda)}$
1: $(\mathsf{srs},\mathsf{td}) \leftarrow \mathcal{S}_0(1^\lambda)$
2: $\mathsf{st} := \mathsf{td}$
3: $b \leftarrow \mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}(\mathsf{srs})$
4: **return** b

$\underline{\mathsf{H}(t)}$
1: **if** $\mathcal{Q}_{\mathsf{H}}(t) = \bot$ **then**
2: $\quad \mathcal{Q}_{\mathsf{H}}(t) \xleftarrow{\$} \{0,1\}^l$
3: **return** $\mathcal{Q}_{\mathsf{H}}(t)$

$\underline{\mathcal{S}_1(t)}$
1: $(h,\mathsf{st}) \leftarrow \mathcal{S}(1,\mathsf{st},t)$
2: **return** h

$\underline{\mathcal{S}_2(\mathsf{x},\mathsf{w})}$
1: **if** $(\mathsf{x},\mathsf{w}) \notin \mathcal{R}$ **then**
2: $\quad$ **return** $\bot$
3: $(\pi,\mathsf{st}) \leftarrow \mathcal{S}(2,\mathsf{st},(\mathsf{srs},\mathsf{x}))$
4: **return** π

We define zero-knowledge for NARGs relying on both SRS and programmable RO. Note that in the general definition below, a simulator may take advantage of both the trapdoor of srs and programmability of the random oracle. Concretely, a simulated SRS generator $\mathcal{S}_0$ may potentially output a simulation trapdoor td. The zero-knowledge simulator $\mathcal{S}$ is defined as a stateful algorithm that operates in two modes. In the first mode, $(h,\mathsf{st}') \leftarrow \mathcal{S}(1,\mathsf{st},t)$ responds to a random oracle query on input t. In the second mode, $(\pi,\mathsf{st}') \leftarrow \mathcal{S}(2,\mathsf{st},(\mathsf{srs},\mathsf{x}))$ simulates a proof string generated by an honest prover $\mathcal{P}$.

Definition 2.12 (Non-Interactive Zero Knowledge in the SRS and Programmable Random Oracle Model). *Let $\Pi_{\mathsf{H}} = (\mathcal{G},\mathcal{P},\mathcal{V})$ be a NARG for relation $\mathcal{R}$. Π_{H} is unbounded* non-interactive zero knowledge *(NIZK) in the programmable random oracle model, if there exist a tuple of* PPT *algorithms $(\mathcal{S}_0,\mathcal{S})$ such that for all PPT distinguisher $\mathcal{A}$, it holds that*

$$\mathbf{Adv}^{\mathit{NIZK}}_{\mathcal{A}}(\lambda) := \left|\Pr\left[\mathit{NIZK\text{-}0}_{\mathcal{A}}(1^\lambda) = 0\right] - \Pr\left[\mathit{NIZK\text{-}1}_{\mathcal{A}}(1^\lambda) = 0\right]\right| \leq \mathsf{negl}(\lambda)$$

where NIZK-0 and NIZK-1 are defined in Game 2. As a special case, if Π_{H} is NIZK w.r.t. $\mathcal{S}_0 = \mathcal{G}$ (and therefore it outputs $\mathsf{td} = \bot$), then it is said to be trapdoor-less *NIZK (TLZK).*

Now we define our final goal: an adaptive version of simulation-extractability for NARG in the ROM. On a high-level, the simulation-extractability (SIM-EXT) property ensures that extractability holds even if the cheating adversary is able to observe simulated proofs. Without the texts highlighted in orange (i.e., without access to the simulation oracle $\mathcal{S}_2'$), the property degrades to the standard extraction property (EXT). This is also known as knowledge soundness.

Definition 2.13 (Simulation-Extractability (SIM-EXT)). *Consider a NARG $\Pi_{\mathsf{H}} = (\mathcal{G},\mathcal{P},\mathcal{V})$ for relation $\mathcal{R}$ with a NIZK simulator $(\mathcal{S}_0,\mathcal{S})$. Π_{H} is* simulation-extractable *(SIM-EXT) with respect to $(\mathcal{S}_0,\mathcal{S})$, if for any* PPT *adversary $\mathcal{A}$, there exists a* PPT *extractor $\mathcal{E}_{\mathcal{A}}$ such that, it holds that*

$$\mathbf{Adv}^{\mathit{SIM\text{-}EXT}}_{\mathcal{A}}(\lambda) := \Pr\left[\mathit{SIM\text{-}EXT}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where SIM-EXT is defined in Game 3.

Game 3: SIM-EXT and WUR

$\underline{\mathsf{WUR}_{\mathcal{A}}(1^\lambda)}$
1: $\mathcal{Q}_1 := \varnothing;\ \mathcal{Q}_2 := \varnothing; \mathcal{Q}_{\mathsf{H}} := \varnothing$
2: $(\mathsf{srs}, \mathsf{td}) \leftarrow \mathcal{S}_0(1^\lambda)$
3: $\mathsf{st} := (\mathcal{Q}_{\mathsf{H}}, \mathsf{td})$
4: $(\mathsf{x}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_1^{\mathcal{S}_1}(\mathsf{srs})$
5: $\widetilde{\pi} \leftarrow \mathcal{S}_2'(\mathsf{x})$
6: $\pi \leftarrow \mathcal{A}_2^{\mathcal{S}_1}(\mathsf{st}_{\mathcal{A}}, \widetilde{\pi})$
7: $b \leftarrow \mathcal{V}^{\mathcal{S}_1}(\mathsf{srs}, \mathsf{x}, \pi)$
8: $b' := (\exists i \in [1, \mathsf{r}] : \pi|_i = \widetilde{\pi}|_i \ \wedge\ a_{i+1} \neq \tilde{a}_{i+1})$
9: **return** $b \wedge b'$

$\underline{\mathcal{S}_1(t)}$
1: $(h, \mathsf{st}) \leftarrow \mathcal{S}(1, \mathsf{st}, t)$
2: $\mathcal{Q}_1(t) := h$
3: **return** h

$\underline{\mathcal{S}(1, \mathsf{st}, t)}$
1: Retrieve $\mathcal{Q}_{\mathsf{H}}$ from st
2: **if** $\mathcal{Q}_{\mathsf{H}}(t) = \perp$ **then**
3: $\quad \mathcal{Q}_{\mathsf{H}}(t) \xleftarrow{\$} \{0,1\}^l$
4: $\mathsf{st} := \mathcal{Q}_{\mathsf{H}}$
5: **return** $(\mathcal{Q}_{\mathsf{H}}(t), \mathsf{st})$

$\underline{\mathsf{SIM\text{-}EXT}_{\mathcal{A}}(1^\lambda)}$
1: $\mathcal{Q}_1 := \varnothing;\ \mathcal{Q}_2 := \varnothing; \mathcal{Q}_{\mathsf{H}} := \varnothing$
2: $(\mathsf{srs}, \mathsf{td}) \leftarrow \mathcal{S}_0(1^\lambda)$
3: $\mathsf{st} := (\mathcal{Q}_{\mathsf{H}}, \mathsf{td})$
4: $(\mathsf{x}^*, \pi^*) \xleftarrow{\rho} \mathcal{A}^{\mathcal{S}_1, \mathcal{S}_2'}(\mathsf{srs})$
5: $b \leftarrow \mathcal{V}^{\mathcal{S}_1}(\mathsf{srs}, \mathsf{x}^*, \pi^*)$
6: $\mathsf{w}^* \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{srs}, \rho, \mathcal{Q}_1, \mathcal{Q}_2)$
7: **return** $b = 1 \ \wedge\ (\mathsf{x}^*, \mathsf{w}^*) \notin \mathcal{R} \wedge (\mathsf{x}^*, \pi^*) \notin \mathcal{Q}_2$

$\underline{\mathcal{S}_2'(\mathsf{x})}$
1: $(\pi, \mathsf{st}) \leftarrow \mathcal{S}(2, \mathsf{st}, (\mathsf{srs}, \mathsf{x}))$
2: $\mathcal{Q}_2 := \mathcal{Q}_2 \cup \{(\mathsf{x}, \pi)\}$
3: **return** π

$\underline{\mathcal{S}(2, \mathsf{st}, (\mathsf{srs}, \mathsf{x}))}$
1: Retrieve $\mathcal{Q}_{\mathsf{H}}$ from st
2: $\pi = (a_1, \rho_1, \ldots, a_{\mathsf{r}+1}) \leftarrow \bar{\mathcal{S}}(\mathsf{srs}, \mathsf{x})$
3: **for** $i \in [1, \mathsf{r}]$ **do**
4: $\quad$ **if** $\mathcal{Q}_{\mathsf{H}}(\mathsf{srs}, \mathsf{x}, \pi|_i) \neq \perp$ **then**
5: $\quad\quad$ **return** `abort`
6: $\quad$ **else**
7: $\quad\quad \mathcal{Q}_{\mathsf{H}}(\mathsf{srs}, \mathsf{x}, \pi|_i) := \rho_i$
8: **return** $(\pi, \mathcal{Q}_{\mathsf{H}})$

Depending on whether $\mathcal{E}_{\mathcal{A}}(\mathsf{srs}, \rho, \mathcal{Q}_1, \mathcal{Q}_2)$ depends on $\mathcal{A}$ and uses ρ, or there exists an $\mathcal{E}$ that is independent of $\mathcal{A}$ and uses only $\mathcal{Q}_1$ we get either white-box extraction, or "Fischlin's" straightline extraction, respectively. [8]

2.4 Simulation-Extractability of Fiat-Shamir Non-Interactive Arguments

In this paper, we consider a special class of NARG characterized as *Fiat-Shamir NARG (FS-NARG)*. $\Pi_{\mathsf{H}} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ is said to be FS-NARG, if $\mathcal{P}$ and $\mathcal{V}$ satisfy the following conditions:

- $\mathcal{P}^{\mathsf{H}}(\mathsf{srs}, \mathsf{x}, \mathsf{w})$ outputs a proof string that can be parsed as $\pi = (a_1, \rho_1, \ldots, a_{\mathsf{r}}, \rho_{\mathsf{r}}, a_{\mathsf{r}+1})$. We denote by $\pi|_i$ the i-th prefix $(a_1, \rho_1, \ldots, a_i)$ of π for $i \in [\mathsf{r}]$.
- There exists a PPT verdict algorithm Ver such that $\mathcal{V}^{\mathsf{H}}(\mathsf{srs}, \mathsf{x}, \pi)$ outputs 1 if and only if (1) $\mathsf{Ver}(\mathsf{srs}, \mathsf{x}, \pi) = 1$, and (2) $\rho_i = \mathsf{H}(\mathsf{srs}, \mathsf{x}, \pi|_i)$ for $i \in [\mathsf{r}]$.

In [GOP+22, GOP+23], the authors define the *weak unique response* property tailored to FS-NARGs but without SRS. In particular, this notion is useful for showing SIM-EXT of FS-NARGs constructed from multi-round public-coin protocols.

Definition 2.14 (WUR). *Consider a FS-NARG $\Pi_{\mathsf{H}} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ for $\mathcal{R}$ with a NIZK simulator $(\mathcal{S}_0, \mathcal{S})$. Π_{H} is said to have* weak unique responses *(WUR) with respect to $(\mathcal{S}_0, \mathcal{S})$, if given a proof string $\widetilde{\pi} = (\tilde{a}_1, \tilde{\rho}_1, \ldots, \tilde{a}_{\mathsf{r}}, \tilde{\rho}_{\mathsf{r}}, \tilde{a}_{\mathsf{r}+1})$ simulated by $\mathcal{S}$, it is hard to find another*

[8] Dependence or independence from $\mathcal{A}$ can be formalized by requiring that there exists a function f such that for any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}_{\mathcal{A}} = f(\mathcal{A})$. If f is a constant function we have independence otherwise dependence.

accepting transcript $\pi = (a_1, \rho_1, \ldots, a_{\mathsf{r}}, \rho_{\mathsf{r}}, a_{\mathsf{r}+1})$ *that both have a common prefix up to the* i*th challenge for an instance* x*. That is, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *(where* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *share the internal states), it holds that*

$$\mathbf{Adv}^{\mathit{WUR}}_{\mathcal{A}}(\lambda) = \Pr\left[\mathit{WUR}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where WUR is defined in Game 3.

To capture typical behaviors of TLZK simulator in an abstract manner, we define the notion of canonical simulation.

Definition 2.15 (Canonical Simulator). *Let* Π_{H} *be a FS-NARG with TLZK simulator* $\mathcal{S}$*.* $\mathcal{S}$ *is said to be canonical, if* $\mathcal{S}$ *in mode 1 answers random oracle queries as defined in Game 3, and* $\mathcal{S}$ *in mode 2 follows the procedures defined in Game 3 by invoking some stateless algorithm* $\bar{\mathcal{S}}$*.*

To enhance modularity of our security proof, we provide the following lemma updating extractability to simulation-extractability assuming weak unique response and a trapdoor-less canonical simulator. We note it can be seen as adaptation of [GOP+23, Lemma 3.2] which only covers NARGs without srs. In [DG23, Theorem 3.4], the authors prove a similar result in the transparent setting, although it relies on different assumptions, i.e., k-unique response (k-UR) and k-zero knowledge. In [GKK+22, Theorem 1], the authors deal with FS-NARG with (updatable) srs also assuming k-UR, but their analysis only covers the case where knowledge soundness is rewinding-based.

Lemma 2.16. *Consider FS-NARG* $\Pi_{\mathsf{H}} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *for relation* $\mathcal{R}$ *in the ROM. If* Π_{H} *satisfies EXT and additionally satisfies WUR w.r.t. a canonical TLZK simulator* $\mathcal{S}$*, then* Π_{H} *satisfies SIM-EXT w.r.t.* $\mathcal{S}$*.*

Proof. The proof is almost identical to that of [GOP+23, Lemma 3.2], thereby we often refer to their proof to avoid redundancy. Let $\hat{\mathcal{A}}$ be a SIM-EXT adversary. Consider the following hybrids:

- $\mathsf{Hyb}_0(1^\lambda)$: Identical to SIM-EXT except that there is no extractor in the experiment and it outputs 1 as long as $b = 1 \wedge (\mathsf{x}^*, \pi^*) \notin \mathcal{Q}_2$.
- $\mathsf{Hyb}_1(1^\lambda)$: Identical to the previous, except that it aborts if there exists $(\mathsf{x}^*, \pi) \in \mathcal{Q}_2$ such that $\exists i \in [1, \mathsf{r}] : \pi^*|_i = \pi|_i \ \wedge \ \pi^* \neq \pi$. Assuming that $\hat{\mathcal{A}}$ makes at most $q_2(\lambda) \in \mathsf{poly}(\lambda)$ queries to $\mathcal{S}'_2$, the probability that Hyb_1 aborts can be bounded by $q_2 \cdot \mathbf{Adv}^{\mathsf{WUR}}_{\mathcal{B}}(\lambda)$ for some reduction $\mathcal{B}$ [GOP+23, Lemma 3.2]. Therefore, the experiment only aborts with negligible probability.

Given $\hat{\mathcal{A}}$ causing Hyb_1 to output 1, we construct a EXT adversary $\mathcal{A}$ that only has access to the random oracle H. $\mathcal{A}$ proceeds as in [GOP+23, Alg.4]: (1) Upon receiving srs (corresponding to pp in [GOP+23]), $\mathcal{A}$ internally runs $\hat{\mathcal{A}}$. (2) Upon receiving a simulation query from $\hat{\mathcal{A}}$, $\mathcal{A}$ runs the algorithm of $\mathcal{S}$ in mode 2. (3) Upon receiving a random oracle query from $\hat{\mathcal{A}}$, $\mathcal{A}$ relays a query to H unless the input is already programmed by $\mathcal{S}$. (4) Upon receiving (x^*, π^*) from $\hat{\mathcal{A}}$, $\mathcal{A}$ outputs it unless the aforementioned abort conditions are met. Thanks to the hash prefix $(\mathsf{srs}, \mathsf{x})$, if $\mathsf{x}^* \neq \mathsf{x}$ for every previously queried x, then the random oracle entry prefixed by $(\mathsf{srs}, \mathsf{x}^*)$ has never been programmed by the canonical simulator $\mathcal{S}$. Moreover, if $\mathsf{x}^* = \mathsf{x}$ for some previously queried $(\mathsf{x}, \pi) \in \mathcal{Q}_2$, then the random oracle entry prefixed by $(\mathsf{srs}, \mathsf{x}, \pi^*|_i)$ has never been programmed by the canonical simulator $\mathcal{S}$ due to the abort condition of Hyb_1. Therefore, (x^*, π^*) output by $\mathcal{A}$ always gets accepted by $\mathcal{V}^{\mathsf{H}}$.

Overall, if $\hat{\mathcal{A}}$ outputs a valid proof in Hyb_0, then $\mathcal{A}$ also succeeds in outputting a valid proof w.r.t. H except with negligible probability. Since Π_{H} satisfies EXT, there exists a PPT

extractor $\mathcal{E}_{\mathcal{A}}$ such that it can extract a valid witness from successful $\mathcal{A}$ except with negligible probability. Therefore, one can construct a SIM-EXT extractor $\hat{\mathcal{E}}_{\hat{\mathcal{A}}}$ that internally runs the procedures of $\mathcal{A}$ and extracts a witness by invoking $\mathcal{E}_{\mathcal{A}}$. Overall such $\hat{\mathcal{E}}$ succeeds in extracting a valid witness from successful $\hat{\mathcal{A}}$ except with negligible probability. □

Remark 2.17. Note that trapdoor-less simulation is crucial for replicating the modular argument of [GOP+23]. In the above proof, an outer prover $\mathcal{A}$ only receives srs from an honest setup algorithm $\mathcal{G}$ while having to simulate the view of SIM-EXT adversary $\hat{\mathcal{A}}$. Therefore, $\mathcal{A}$ cannot use the trapdoor and must perform simulation by programming the random oracle responses only.

2.5 Polynomial Commitment Scheme

We define a polynomial commitment scheme [KZG10].

Definition 2.18 (Polynomial Commitment Scheme). *A polynomial commitment scheme denoted by PCOM is a tuple of algorithms* $(\mathsf{KGen}, \mathsf{Com}, \mathsf{Eval}, \mathsf{Check})$*:*

1. $\mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda, D)$*: Takes as input the security parameter* λ *and the maximum degree bound* D *and generates commitment key* ck *as output. We assume* ck *to include description of the finite field* $\mathbb{F}$*.*
2. $c \xleftarrow{\rho} \mathsf{Com}(\mathsf{ck}, f)$*: Takes as input* ck*, the polynomial* $f \in \mathbb{F}^D[X]$*, and outputs a commitment* c*. In case the commitment scheme is deterministic,* $\rho = \perp$*. We also denote* $c := \mathsf{Com}(\mathsf{ck}, f; \rho)$ *if the committing function deterministically generates* c *from fixed randomness* ρ*. If the input is a vector of polynomials* $\mathbf{f}$ *with dimension* n*, we assume* Com *to output a vector of commitments* $\mathbf{c}$ *with dimension* n *by invoking* Com n *times.*
3. $\pi \leftarrow \mathsf{Eval}(\mathsf{ck}, c, z, f, \rho)$*: Takes as input* ck*, the commitment* c*, evaluation point* $z \in \mathbb{F}$*, the polynomial* f*, and outputs a non-interactive proof of evaluation* π*. The randomness* ρ *must equal the one previously used in* Com*. If the input is vectors* $(\mathbf{c}, \mathbf{z}, \mathbf{f}, \boldsymbol{\rho})$ *with dimension* n*, we assume* Eval *to output a vector of proofs* $\boldsymbol{\pi}$ *with dimension* n *by invoking* Eval n *times.*
4. $b \leftarrow \mathsf{Check}(\mathsf{ck}, c, z, y, \pi)$*: Takes as input statement* (ck, c, z, y)*, where* $y \in \mathbb{F}$ *is a claimed polynomial evaluation, and the proof of evaluation* π *and outputs a bit* b*. If the input is vectors* $(\mathbf{c}, \mathbf{z}, \mathbf{y}, \boldsymbol{\pi})$ *with dimension* n*, we assume* Check *to invoke* Check n *times and output* 1 *if and only if all of them output* 1*.*

We define security properties for PCOM. All of the experiments are described in Game 4.

Definition 2.19 (Completeness). *A PCOM is said to be* complete*, if for any* $\lambda \in \mathbb{N}$*,* $D \in \mathbb{N}$*, polynomial* $f \in \mathbb{F}^D[X]$*, evaluation point* $z \in \mathbb{F}$

$$\Pr\left[\mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda, D); c \leftarrow \mathsf{Com}(\mathsf{ck}, f); \pi \leftarrow \mathsf{Eval}(\mathsf{ck}, c, z, f, \rho) \ : \ \mathsf{Check}(\mathsf{ck}, c, z, f(z), \pi) = 1\right] = 1.$$

The evaluation binding property essentially guarantees that, it is infeasible to open the same commitment c to two distinct outcomes of evaluation y and y' for the fixed evaluation point z.

Definition 2.20 ((Weak) Evaluation Binding). *PCOM is said to be evaluation binding if, for any* $\lambda \in \mathbb{N}$*,* $D \in \mathbb{N}$*, for all* PPT *adversaries* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\textit{PC-EBIND}}_{\mathcal{A}}(\lambda) := \Pr\left[\textit{PC-EBIND}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

If instead

$$\mathbf{Adv}^{\mathsf{PC\text{-}wEBIND}}_{\mathcal{A}}(\lambda) := \Pr\left[\mathsf{PC\text{-}wEBIND}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

then PCOM *is* weak *evaluating binding.*

The unique proof states that, it is infeasible to create two distinct valid proofs π and π' for fixed c, z, y.

Definition 2.21 ((Weak) Unique Proof). *PCOM is said to be unique proof if, for any $\lambda \in \mathbb{N}$, $D \in \mathbb{N}$, for all* PPT *adversaries $\mathcal{A}$,*

$$\mathbf{Adv}^{\mathsf{PC\text{-}UNIQ}}_{\mathcal{A}}(\lambda) := \Pr\left[\mathsf{PC\text{-}UNIQ}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

If instead

$$\mathbf{Adv}^{\mathsf{PC\text{-}wUNIQ}}_{\mathcal{A}}(\lambda) := \Pr\left[\mathsf{PC\text{-}wUNIQ}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

then PCOM *is* weak *unique proof.*

Unlike the usual hiding definition for a commitment scheme, Com inevitably leaks evaluations of the committed polynomials. As we shall see later, some schemes such as KZG further leak evaluation at an additional point $\chi \in \mathbb{F}$. To capture this, we consider a weak variant of hiding.

Definition 2.22 ((Weak) Hiding). *PCOM is said to be weak hiding if, for any $\lambda \in \mathbb{N}$, $D \in \mathbb{N}$, there exists a* PPT *simulator* $(\mathsf{SKGen}, \mathsf{SCom})$ *such that for all* PPT *adversaries $\mathcal{A}$,*

$$\mathbf{Adv}^{\mathsf{PC\text{-}wHIDE}}_{\mathcal{A}}(\lambda) := |\Pr\left[\mathsf{PC\text{-}wHIDE}_{\mathcal{A}}(1^\lambda) = 1\right] - 1/2| \leq \mathsf{negl}(\lambda).$$

As a special case, if SKGen *outputs $\chi = \perp$ (and thus $f(\chi) = \perp$), then PCOM is said to be hiding.*

For our results, we require the probability that a commitment is equal to a fixed value is low. This requirement is captured by assessing min-entropy of the PCOM commitment scheme.

Definition 2.23 (Min-entropy of commitments). *Let PCOM be a polynomial commitment scheme over $\mathbb{F}$. For any fixed $\lambda \in \mathbb{N}$,$D \in \mathbb{N}$, $\mathsf{ck} \in \mathsf{KGen}(1^\lambda, D)$, and $f \in \mathbb{F}^D[X]$, consider the maximum probability that a commitment to f is equal to a particular value:*

$$\mu(\lambda, \mathsf{ck}, f) = \max_c \Pr\left[\mathsf{Com}(\mathsf{ck}, f) = c\right].$$

The min-entropy *α of scheme PCOM is*

$$\alpha(\lambda) = \min_{\mathsf{ck} \in \mathsf{KGen}(1^\lambda, D) \ \wedge \ f \in \mathbb{F}^D[X]} (-\log_2(\mu(\lambda, \mathsf{ck}, f))).$$

We say that PCOM has high min-entropy if $\alpha \in \omega(\log(\lambda))$.

Game 4: PCOM Security Games

$\underline{\mathsf{PC\text{-}EBIND}_{\mathcal{A}}(1^\lambda), \mathsf{PC\text{-}wEBIND}_{\mathcal{A}}(1^\lambda)}$

$ctr := 0; \mathcal{Q}_{\mathsf{Com}} = \varepsilon; \mathcal{Q}_{\mathsf{Eval}} := \varepsilon$
$\mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda, D)$
$(i, c, z, y, y', \pi, \pi') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Com}}, \mathcal{O}_{\mathsf{Eval}}}(\mathsf{ck})$
if $\mathcal{Q}_{\mathsf{Eval}}(i) \neq (*, *, c, z, y, \pi)$ **then**
    return 0
$b \leftarrow \mathsf{Check}(\mathsf{ck}, c, z, y, \pi)$
$b' \leftarrow \mathsf{Check}(\mathsf{ck}, c, z, y', \pi')$
return $(y \neq y') \wedge b \wedge b'$

$\underline{\mathsf{PC\text{-}UNIQ}_{\mathcal{A}}(1^\lambda), \mathsf{PC\text{-}wUNIQ}_{\mathcal{A}}(1^\lambda)}$

$ctr := 0; \mathcal{Q}_{\mathsf{Com}} = \varepsilon; \mathcal{Q}_{\mathsf{Eval}} := \varepsilon$
$\mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda, D)$
$(i, c, z, y, \pi, \pi') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Com}}, \mathcal{O}_{\mathsf{Eval}}}(\mathsf{ck})$
if $\mathcal{Q}_{\mathsf{Eval}}(i) \neq (*, *, c, z, y, \pi)$ **then**
    return 0
$b \leftarrow \mathsf{Check}(\mathsf{ck}, c, z, y, \pi)$
$b' \leftarrow \mathsf{Check}(\mathsf{ck}, c, z, y, \pi')$
return $(\pi \neq \pi') \wedge b \wedge b'$

$\underline{\mathsf{PC\text{-}wHIDE}_{\mathcal{A}}(1^\lambda)}$

$b \xleftarrow{\$} \{0, 1\}$
if $b = 0$ **then**
    $\mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda, D)$
else
    $(\mathsf{ck}, \chi) \leftarrow \mathsf{SKGen}(1^\lambda, D)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{ComEval\text{-}b}}}(\mathsf{ck})$
return $(b = b')$

$\underline{\mathcal{O}_{\mathsf{Com}}(f)}$

if $\deg(f) > D$ **then return** $\perp$
$c \xleftarrow{\rho} \mathsf{Com}(\mathsf{ck}, f)$
$ctr := ctr + 1$
$\mathcal{Q}_{\mathsf{Com}}(ctr) := (f, \rho, c)$
return c

$\underline{\mathcal{O}_{\mathsf{Eval}}(i, z)}$

if $\mathcal{Q}_{\mathsf{Com}}(i) = \varepsilon$ **then return** $\perp$
$(f, \rho, c) := \mathcal{Q}_{\mathsf{Com}}(i)$
$\pi \leftarrow \mathsf{Eval}(\mathsf{ck}, c, z, f, \rho)$
$\mathcal{Q}_{\mathsf{Eval}}(i) := (f, \rho, c, z, f(z), \pi)$
return (y, π)

$\underline{\mathcal{O}_{\mathsf{ComEval\text{-}0}}(f, (z_1, \ldots, z_n))}$

if $\deg(f) > D$ **then return** $\perp$
$c \xleftarrow{\rho} \mathsf{Com}(\mathsf{ck}, f)$
for $i = 1, \ldots, n$ **do**
    $\pi_i \leftarrow \mathsf{Eval}(\mathsf{ck}, c, z_i, f, \rho)$
return $(c, (\pi_1, \ldots, \pi_n))$

$\underline{\mathcal{O}_{\mathsf{ComEval\text{-}1}}(f, (z_1, \ldots, z_n))}$

if $\deg(f) > D$ **then return** $\perp$
$(c, \boldsymbol{\pi}) \leftarrow \mathsf{SCom}(\mathsf{ck}, \chi, f(\chi), \mathbf{z}, f(z_1), \ldots, f(z_n))$
return $(c, \boldsymbol{\pi})$

3 Analysis of PIOP Compiled into Non-interactive Argument

In this section, we analyze a standard compiler that outputs iNARG (Definition 2.10) in the random oracle model. The compiler takes following building blocks as input:

- Polynomial IOP $\mathsf{PIOP} = (\mathsf{r}, \mathsf{s}, \mathsf{t}, \mathsf{d}, \mathbf{I}, \mathbf{P}, \mathbf{V})$ (Definition 2.1) for an indexed relation $\hat{\mathcal{R}}$.
- Polynomial commitment $\mathsf{PCOM} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Eval}, \mathsf{Check})$ (Definition 2.18)

It then outputs iNARG $\hat{\Pi}_{\mathsf{H}} = (\mathcal{G}, \mathcal{I}, \hat{\mathcal{P}}, \hat{\mathcal{V}})$ described in Protocol 1. On a high-level, the outer prover $\hat{\mathcal{P}}$ internally runs a PIOP prover $\mathbf{P}$ in order to obtain polynomials and then commit to them using the polynomial commitment scheme. Then by hashing the transcript obtained until i-th round, $\hat{\mathcal{P}}$ obtains PIOP challenge ρ_i, which is fed to $\mathbf{P}$ to advance to the next round. When the PIOP prover terminates, $\hat{\mathcal{P}}$ runs a PIOP query algorithm to sample query points $\mathbf{z}$ and evaluates polynomial oracles on $\mathbf{z}$. Finally, $\hat{\mathcal{P}}$ produces evaluation proofs to guarantee that polynomial evaluations are done correctly with respect to commitments produced in earlier rounds.

Remark 3.1. The iNARG $\hat{\Pi}_{\mathsf{H}}$ detailed in Protocol 1 is almost identical to the compiled protocol in [CHM+20], except that we are explicit about strings hashed to derive Fiat-Shamir challenge (the Marlin compiler does not specify what needs to be hashed when applying Fiat-Shamir). We stress it is crucial to hash index i (i.e., the circuit description) on

top of statement x and transcript; otherwise, the proof system is susceptible to the following malleability attack. Suppose the adversary receives an honestly generated proof π for (i, x). Then the adversary constructs a modified i^* such that for any w, it holds that $(\mathsf{i}^*, \mathsf{x}, \mathsf{w}) \in \hat{\mathcal{R}}$ iff $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \hat{\mathcal{R}}$, e.g. by introducing redundancy in the circuit. In this way, π is a valid proof for $(\mathsf{i}^*, \mathsf{x})$ which allows the adversary to trivially win the SIM-EXT game. Although syntactically ivk now contains $(\mathsf{srs}, \mathsf{i})$, this does not penalize verification performance in practice because hashing of the prefix srs, i can be preprocessed.

As mentioned in Section 2.3, iNARG can be converted into NARG $\Pi_{\mathsf{H}} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ (Definition 2.10) for the corresponding binary relation $\mathcal{R}$, which is amenable to analysis of simulation-extractability. In the rest of this section we will show Π_{H} satisfies the following security properties under certain assumptions on PIOP and PCOM.

- Section 3.1 Trapdoor-less zero knowledge (TLZK) with canonical simulation (Definition 2.15),
- Section 3.2 Weak unique response (WUR) with respect to simulators provided in Section 3.1.

Knowledge soundness of Π_{H} is already proved in the literature from knowledge soundness of PIOP and extractability of PCOM under various assumptions [CHM+20, BFS20, CFF+21, MBKM19, GWC19]. Put together with Lemma 2.16 we conclude SIM-EXT for the compiled Π_{H}.

Corollary 3.2. *Let Π_{H} be the FS-NARG protocol derived from $\hat{\Pi}_{\mathsf{H}}$.*

1. *Suppose the PIOP satisfies Ψ-HVZK and Ψ-NEXP with $\Psi = 1$, and $\mathbf{Q}_{\mathbf{V}}$ has high min-entropy. PCOM satisfies PC-wHIDE, PC-wEBIND, PC-wUNIQ. If Π_{H} is non-trivial and knowledge sound, then it is SIM-EXT.*
2. *Suppose the PIOP satisfies HVZK, high min-entropy of $\mathbf{Q}_{\mathbf{V}}$, and $\mathsf{t}(1,1) \leq \mathsf{d}(|\mathsf{i}|)$. PCOM has high min-entropy (Definition 2.23), and satisfies PC-HIDE, PC-wEBIND, PC-wUNIQ. If Π_{H} is knowledge sound, then it is SIM-EXT.*

3.1 Trapdoor-Less Non-Interactive Zero Knowledge of Compiled NARG

Our analysis for showing TLZK will be split in two directions, which will exploit the type of properties satisfied by the two core building blocks, PIOP and PCOM schemes. First, we consider a class of PIOPs satisfying the *stronger* property of Ψ-HVZK, which in turn, requires only a *weaker* hiding property from the PCOM scheme. Namely, it suffices to use a deterministic PCOM scheme for the compilation. This characterization allows us to reuse randomness already introduced by the PIOP while committing to the polynomials. Note that the previous generic PIOP-to-iNARG compilers do not give us a clear picture of scenarios when using a deterministic commitment suffices for trapdoorless NIZK: Marlin [CHM+20, Thorem 8.4] and Dark [BFS20, Theorem 4] require a hiding commitment scheme as well as trapdoor to perform simulation; and Lunar [CFF+21, Theorem 5] requires a weaker "somewhat hiding" commitment scheme, which, however, crucially relies on the knowledge of commitment trapdoor.

In the second direction, we consider all other PIOPs that satisfy the *weaker* property of HVZK, and require the PCOM scheme to be hiding. This is similar to Marlin, however, unlike Marlin, we cannot use the trapdoor information in order to simulate. Hence, we present a more direct trapdoorless simulation strategy similar to Dark.[9]

[9] They assume PIOP is HVZK, a committing function Com is hiding, and Eval satisfies HVZK. The latter two roughly correspond to our combined notion of hiding for PCOM.

Protocol 1: iNARG $\hat{\Pi}_{\mathsf{H}}$ for $\hat{\mathcal{R}}$

$\underline{\mathcal{G}(1^\lambda)}$

1. Let $D = \max_{(\mathsf{i},\cdot,\cdot)\in\hat{\mathcal{R}}_\lambda} \mathsf{d}(|\mathsf{i}|)$.
2. Run $\mathsf{ck} \leftarrow \mathsf{PCOM.KGen}(1^\lambda, D)$ and output $\mathsf{srs} := \mathsf{ck}$.

$\underline{\mathcal{I}(\mathsf{i}, \mathsf{srs})}$

1. Compute $(p_{0,1}, \ldots, p_{0,\mathsf{s}(0)}) \leftarrow \mathsf{PIOP.}\mathbf{I}(\mathsf{i})$ where each polynomial $p_{0,j}$, for $j \in [\mathsf{s}(0)]$ is of degree at most $\mathsf{d}(|\mathsf{i}|)$.
2. For $j \in [\mathsf{s}(i)]$: Compute $c_{0,j} \leftarrow \mathsf{PCOM.Com}(\mathsf{ck}, p_{0,j}; \perp)$ with empty randomness.
3. Set $\mathsf{ipk} := (\mathsf{i}, \mathsf{srs}, (c_{0,1}, \ldots, c_{0,\mathsf{s}(0)}), (p_{0,1}, \ldots, p_{0,\mathsf{s}(0)}))$, and $\mathsf{ivk} := (\mathsf{i}, \mathsf{srs}, (c_{0,1}, \ldots, c_{0,\mathsf{s}(0)}))$.

$\underline{\hat{\mathcal{P}}^{\mathsf{H}}(\mathsf{ipk}, \mathsf{x}, \mathsf{w})}$

1. Initialize round $i := 1$, transcript $\pi := \varnothing$, polynomials $\mathbf{p} := \varnothing$, evaluation proofs $\boldsymbol{\pi}_{\mathsf{PCOM}} := \varnothing$, initial state $\mathsf{st}_P := (\mathsf{i}, \mathsf{x}, \mathsf{w})$, $\rho_0 := \perp$.
2. **Online phase:** While $i \le \mathsf{r}$,
   (a) Compute $(p_{i,1}, \ldots, p_{i,\mathsf{s}(i)}, \mathsf{st}_P) \leftarrow \mathsf{PIOP.}\mathbf{P}(\mathsf{st}_P, \rho_{i-1})$.
   (b) For $j \in [\mathsf{s}(i)]$: Compute $c_{i,j} \xleftarrow{r_{i,j}} \mathsf{PCOM.Com}(\mathsf{ck}, p_{i,j})$.
   (c) Update $\pi := \pi||(c_{i,1}, \ldots, c_{i,\mathsf{s}(i)})$ and $\mathbf{p} := \mathbf{p}||(p_{i,1}, \ldots, p_{i,\mathsf{s}(i)})$
   (d) Derive challenge $\rho_i \leftarrow \mathsf{H}(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi)$.
   (e) Update $\pi := \pi||\rho_i$, and $i := i + 1$.
3. **Query phase:** If $i > \mathsf{r}$,
   (a) Compute $\mathbf{z} \leftarrow \mathsf{PIOP.}\mathbf{Q}_{\mathbf{V}}(\mathsf{x}; \rho_1, \ldots, \rho_{\mathsf{r}})$.
   (b) For each $i \in [0, \mathsf{r}], j \in [\mathsf{s}(i)], k \in [\mathsf{t}(i,j)]$: Compute $\pi_{i,j,k} \leftarrow \mathsf{PCOM.Eval}(\mathsf{ck}, c_{i,j}, z_{i,j,k}, p_{i,j}, r_{i,j})$ and $\boldsymbol{\pi}_{\mathsf{PCOM}} := \boldsymbol{\pi}_{\mathsf{PCOM}}||\pi_{i,j,k}$
   (c) $\mathbf{y} := \mathbf{p}(\mathbf{z})$.
   (d) Update $\pi := \pi||(\mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCOM}})$.
4. Output π.

$\underline{\hat{\mathcal{V}}^{\mathsf{H}}(\mathsf{ivk}, \mathsf{x}, \pi)}$

1. Initialize round $i := 1$.
2. Parse $\pi := (c_{1,1}, \ldots, \rho_{\mathsf{r}}, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCOM}})$.
3. **Online phase:** While $i \le \mathsf{r}$,
   (a) Define $\pi|_i := (c_{1,1}, \ldots, c_{i,\mathsf{s}(i)})$.
   (b) Derive challenge $\rho'_i \leftarrow \mathsf{H}(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi|_i)$. If $\rho'_i \neq \rho_i$, abort by outputting 0.
   (c) Update round $i := i + 1$.
4. **Query phase:** Compute $\mathbf{z} \leftarrow \mathbf{Q}_{\mathbf{V}}(\mathsf{x}; \rho_1, \ldots, \rho_{\mathsf{r}})$.
5. **Decision phase:** Output 1 if $\mathsf{PCOM.Check}(\mathsf{ck}, \mathbf{c}, \mathbf{z}, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCOM}}) = 1$ and $\mathbf{D}_{\mathbf{V}}(\mathsf{x}, \mathbf{y}; \rho_1, \ldots, \rho_{\mathsf{r}}) = 1$.

Simulator 1: $\mathcal{S}$

The simulator internally runs an 1-HVZK simulator $\mathbf{S}$ for PIOP.

SRS generation. On input 1^λ with mode 0, $\mathcal{S}$ behaves exactly like $\mathcal{G}$ and sets $\mathsf{st} = \bot$.

Random oracle queries. $\mathcal{S}$ maintains an initially empty look-up table $\mathcal{Q}_\mathsf{H}$. On input query t with mode 1, do: check whether $\mathcal{Q}_\mathsf{H}(t)$ is already defined. If this is the case, return the previously assigned value; otherwise, sample ρ_i from Ch, set $\mathcal{Q}_\mathsf{H}(t) := \rho_i$, and return ρ_i.

Simulation queries. On input $(\mathsf{srs}, (\mathsf{i}, \mathsf{x}))$ with mode 2, do:

1. Run $\mathcal{I}(\mathsf{i}, \mathsf{srs})$ to generate $(\mathsf{ipk}, \mathsf{ivk})$.
2. Run $(\mathsf{view}; \tilde{\mathbf{p}}) \leftarrow \mathbf{S}(\mathsf{i}, \mathsf{x})$.
3. Parse view as $(\rho_1, \ldots, \rho_\mathsf{r}, \mathbf{y})$, and $\tilde{\mathbf{p}}$ as $(\tilde{p}_{i,j})_{i \in [\mathsf{r}], j \in [\mathsf{s}(i)]}$.
4. Run $\mathbf{z} \leftarrow \mathbf{Q}_\mathbf{V}(\mathsf{x}; \rho_1, \ldots, \rho_\mathsf{r})$.
5. Parse ipk to obtain $(p_{0,1}, \ldots, p_{0,\mathsf{s}(0)})$.
6. For each $i \in [\mathsf{r}], j \in [\mathsf{s}(i)]$: compute $c_{i,j} \xleftarrow{r_{i,j}} \mathsf{PCOM.Com}(\mathsf{ck}, \tilde{p}_{i,j})$, and for each $i \in [0, \mathsf{r}], j \in [\mathsf{s}(i)], k \in [\mathsf{t}(i,j)]$ $\pi_{i,j,k} \leftarrow \mathsf{PCOM.Eval}(\mathsf{ck}, c_{i,j}, z_{i,j,k}, \tilde{p}_{i,j}, r_{i,j})$, where $\tilde{p}_{0,j} := p_{0,j}$. Let $\boldsymbol{\pi}_\mathsf{PCOM} := (\pi_{i,j,k})_{i,j,k}$.
7. Set $\pi := (c_{1,1}, \ldots, c_{\mathsf{r},\mathsf{s}(\mathsf{r})}, \rho_\mathsf{r}, \mathbf{y}, \boldsymbol{\pi}_\mathsf{PCOM})$.
8. Program the table as follows: for $i \in [\mathsf{r}]$, set $\pi|_i := (c_{1,1}, \ldots, c_{i,\mathsf{s}(i)})$, and $\mathcal{Q}_\mathsf{H}(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi|_i) := \rho_i$. If $\mathcal{Q}_\mathsf{H}(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi|_i)$ is already set, then output $\bot_1$.
9. Output π.

Compilation with Weak Hiding Polynomial Commitments We first handle the case where PCOM only satisfies a weak variant of hiding, which means that commitment and evaluation are potentially deterministic. In this case, the committing function itself does not have high min-entropy as in Definition 2.23. Combined with a "sufficiently randomized first polynomial" of PIOP, we can still retain high min-entropy of the compiled protocol, which we formalize below. Non-triviality is often required for Fiat-Shamir to retain zero knowledge (cf. [AABN02]), and existing PIOP-based zkSNARKs are already non-trivial.

Definition 3.3 (Min-entropy of the first commitment). *Let $\mathsf{Coin}_P(\lambda)$ be the set of random coins used by the PIOP prover $\mathbf{P}$ on any input $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \hat{\mathcal{R}}_\lambda$. For any fixed $\lambda \in \mathbb{N}$, $\mathsf{ck} \in \mathsf{KGen}(1^\lambda)$, and $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \hat{\mathcal{R}}_\lambda$, consider the maximum probability that a commitment to the first polynomial hits a particular value:*

$$\mu(\mathsf{ck}, \mathsf{i}, \mathsf{x}, \mathsf{w}) = \max_c \Pr\left[r \xleftarrow{\$} \mathsf{Coin}_P(\lambda); (p_{1,1}, \ldots) \leftarrow \mathbf{P}((\mathsf{i}, \mathsf{x}, \mathsf{w}); r) \ : \ \mathsf{Com}(\mathsf{ck}, p_{1,1}) = c\right]$$

The min-entropy $\alpha_{\hat{\Pi}}$ of protocol $\hat{\Pi}$ is

$$\alpha_{\hat{\Pi}}(\lambda) := \min_{\mathsf{ck} \in \mathsf{KGen}(1^\lambda) \wedge (\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \hat{\mathcal{R}}_\lambda} (-\log_2 \mu(\mathsf{ck}, \mathsf{i}, \mathsf{x}, \mathsf{w}))$$

We say that $\hat{\Pi}$ is non-trivial *if $\alpha_{\hat{\Pi}} \in \omega(\log(\lambda))$.*

Lemma 3.4. *If PIOP is Ψ-HVZK with $\psi = 1$, PCOM is weak hiding (PC-wHIDE), and the corresponding iNARG $\hat{\Pi}_\mathsf{H}$ (Protocol 1) is non-trivial. Then FS-NARG protocol Π_H derived from $\hat{\Pi}_\mathsf{H}$ is TLZK with canonical simulator.*

Proof. We show the existence of canonical simulator $\mathcal{S}$ for Π_{H}, detailed in Simulator 1. By inspection $\mathcal{S}$ is trapdoor-less. Moreover, it is indeed canonical, because all of the operation except Step 8 do not depend on the RO query table and can be turned into a stateless algorithm $\bar{\mathcal{S}}$ that takes $(\mathsf{srs}, (\mathsf{i}, \mathsf{x}))$ as input. Then programming of RO entries at Step 8 follows the format of canonical simulation.

With respect to $\mathcal{S}$, our goal is to prove

$$\mathbf{Adv}^{\mathsf{NIZK}}_{\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{NIZK\text{-}0}_{\mathcal{A}}(1^\lambda) = 0\right] - \Pr\left[\mathsf{NIZK\text{-}1}_{\mathcal{A}}(1^\lambda) = 0\right]\right| \leq \mathsf{negl}(\lambda)$$

where the two experiments are described in Game 2. Let $\mathsf{Hyb}_0(1^\lambda)$ be an experiment identical to $\mathsf{NIZK\text{-}0}_{\mathcal{A}}(1^\lambda)$. Consider the following hybrids.

- $\mathsf{Hyb}_1(1^\lambda)$ is identical to the previous hybrid, except at derivation of challenge ρ_i for each i (Step 2d of the online phase): once the input $(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi|_i)$ is obtained, sample ρ_i uniformly from the challenge domain Ch, program the $\mathcal{Q}_{\mathsf{H}}(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi|_i) := \rho_i$ if $\mathcal{Q}_{\mathsf{H}}(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi|_i)$ is undefined, and abort otherwise. Assuming that $\mathcal{A}$ making at most $\mathsf{poly}(\lambda)$ queries, the probability that $\mathsf{Hyb}_1(1^\lambda)$ aborts is negligible due to non-triviality. Thus, we obtain

$$\left|\Pr\left[\mathsf{Hyb}_0(1^\lambda) = 0\right] - \Pr\left[\mathsf{Hyb}_1(1^\lambda) = 0\right]\right| \leq \mathsf{negl}(\lambda).$$

- $\mathsf{Hyb}_2(1^\lambda)$ is identical to the previous hybrid, except at the SRS generation and the committing phase: the $\mathcal{G}$ is modified such that it invokes $(\mathsf{ck}, \chi) \leftarrow \mathsf{SKGen}(1^\lambda)$ and outputs ck, and during the online phase $\hat{\mathcal{P}}$ runs simulated algorithm $(c_{i,j}, \boldsymbol{\pi}_{i,j}) \leftarrow \mathsf{SCom}(\mathsf{ck}, \chi, p_{i,j}(\chi), \mathbf{z}_{i,j}, p_{i,j}(\mathbf{z}_{i,j}))$, instead of invoking Com and Eval. Note that $\rho_1, \ldots, \rho_{\mathsf{r}}$ are picked in advance thanks to the previous hybrid, and therefore a simulated prover can indeed derive the whole $\mathbf{p}$ via $\mathbf{P}$ and $\mathbf{z}$ via $\mathbf{Q}_{\mathbf{V}}$, respectively, before generating any commitments.

  To argue Hyb_2 is indistinguishable from Hyb_1, we construct reduction $\mathcal{B}$ against weak hiding of PCOM. Upon receiving ck, $\mathcal{B}$ runs the procedures of Hyb_1 until the PIOP prover $\mathbf{P}$ outputs polynomials. As $\mathcal{B}$ obtains $\mathbf{p}$ and $\mathbf{z}$, it forwards $(p_{i,j}, \mathbf{z}_{i,j})$ to the oracle $\mathcal{O}_{\mathsf{ComEval\text{-}b}}$ of PC-wHIDE for all i, j. Upon receiving $(c_{i,j}, \boldsymbol{\pi}_{i,j})$ from the oracle, $\mathcal{B}$ completes the remaining operations of the simulated prover in Hyb_1, to obtain a simulated proof π, and hands over π to $\mathcal{A}$. $\mathcal{B}$ repeats the above procedures whenever $\mathcal{A}$ requests a proof on $((\mathsf{i}, \mathsf{x}), \mathsf{w})$.

  Upon receiving a bit b' from $\mathcal{A}$, $\mathcal{B}$ forwards b' to PC-wHIDE. If $b = 0$ is chosen in PC-wHIDE, the view of $\mathcal{A}$ is identically distributed to that of Hyb_1; if $b = 1$, the view of $\mathcal{A}$ is identically distributed to that of Hyb_2. Thus, we have

$$\left|\Pr\left[\mathsf{Hyb}_1(1^\lambda) = 0\right] - \Pr\left[\mathsf{Hyb}_2(1^\lambda) = 0\right]\right| \leq \mathbf{Adv}^{\mathsf{PC\text{-}wHIDE}}_{\mathcal{B}}(\lambda).$$

- $\mathsf{Hyb}_3(1^\lambda)$ is identical to the previous hybrid, except at generation of polynomials $\mathbf{p}$: run the Ψ-HVZK simulator $\mathbf{S}(\mathsf{i}, \mathsf{x})$ to obtain $\mathsf{view} = (\rho_1, \ldots, \rho_{\mathsf{r}}, \mathbf{y})$ and $\tilde{\mathbf{p}} = (\tilde{p}_{i,j})_{i \in [0,\mathsf{r}], j \in [\mathsf{s}(i)]}$, instead of invoking the PIOP prover $\mathbf{P}$. Then derive $\mathbf{z} \leftarrow \mathbf{Q}_{\mathbf{V}}(\mathsf{x}; \rho_1, \ldots, \rho_{\mathsf{r}})$, generate simulated commitments and proofs $(c_{i,j}, \boldsymbol{\pi}_{i,j}) \leftarrow \mathsf{SCom}(\mathsf{ck}, \chi, \tilde{p}_{i,j}(\chi), \mathbf{z}_{i,j}, \tilde{p}_{i,j}(\mathbf{z}_{i,j}))$, and proceeds as in Hyb_2.

  Since PIOP is Ψ-HVZK with $\psi = 1$ and both Hyb_2 and Hyb_3 only require evaluations of $\mathbf{p}$ at $\mathbf{z}$ and χ, we have

$$\left|\Pr\left[\mathsf{Hyb}_2(1^\lambda) = 0\right] - \Pr\left[\mathsf{Hyb}_3(1^\lambda) = 0\right]\right| \leq \mathbf{Adv}^{\Psi\text{-}\mathsf{HVZK}}_{\mathcal{C}}(\lambda).$$

- $\mathsf{Hyb}_4(1^\lambda)$ is identical to the previous hybrid, except at the SRS generation and the committing phase: the $\mathcal{G}$ is modified back to the original algorithm invoking KGen, and during

Simulator 2: $\mathcal{S}'$

The simulator internally runs an HVZK simulator $\mathbf{S}$ for PIOP.

SRS generation. Same as Simulator 1.

Random oracle queries. Same as Simulator 1.

Simulation queries. On input $(\mathsf{srs}, \mathcal{Q}_{\mathsf{H}}, (\mathsf{i}, \mathsf{x}))$ with mode 2, do:

1. Run $\mathcal{I}(\mathsf{i}, \mathsf{srs})$ to generate $(\mathsf{ipk}, \mathsf{ivk})$.
2. Run $\mathsf{view} \leftarrow \mathbf{S}(\mathsf{i}, \mathsf{x})$.
3. Parse view as $(\rho_1, \ldots, \rho_{\mathsf{r}}, \mathbf{y})$.
4. Parse ipk to obtain $(c_{0,1}, \ldots, c_{0,\mathsf{s}(0)}, p_{0,1}, \ldots, p_{0,\mathsf{s}(0)})$.
5. Run $\mathbf{z} \leftarrow \mathbf{Q}_{\mathbf{V}}(\mathsf{x}; \rho_1, \ldots, \rho_{\mathsf{r}})$. Let $\mathbf{z}_0$ be the vector of evaluation points w.r.t. $(c_{0,1}, \ldots, c_{0,\mathsf{s}(0)})$, and $\mathbf{z}_1$ be for $(c_{1,1}, \ldots, c_{\mathsf{r},\mathsf{s}(\mathsf{r})})$. Similarly, $\mathbf{y}_0$ ($\mathbf{y}_1$) be evaluations w.r.t $\mathbf{z}_0$ (resp, $\mathbf{z}_1$).
6. Compute $(\mathbf{c}, \boldsymbol{\pi}_{\mathsf{PCOM}}) \leftarrow \mathsf{SCom}(\mathsf{ck}, \perp, \perp, \mathbf{z}_1, \mathbf{y}_1)$.
7. For each $j \in [\mathsf{s}(0)]$, $k \in [\mathsf{t}(0,j)]$, compute $\pi_{0,j,k} \leftarrow \mathsf{PCOM.Eval}(\mathsf{ck}, c_{0,j}, z_{0,j,k}, p_{0,j}, \perp)$. Let $\boldsymbol{\pi}_{\mathsf{PCOM}} := (\pi_{0,j,k})_{j,k} || \boldsymbol{\pi}_{\mathsf{PCOM}}$.
8. Set $\pi := (c_{1,1}, \ldots, c_{\mathsf{r},\mathsf{s}(\mathsf{r})}, \rho_{\mathsf{r}}, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCOM}})$.
9. Program random oracle as follows: for $i \in [\mathsf{r}]$, set $\pi|_i := (c_{1,1}, \ldots, c_{i,\mathsf{s}(i)})$, and $\mathcal{Q}_{\mathsf{H}}(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi|_i) := \rho_i$. If $\mathcal{Q}_{\mathsf{H}}(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi|_i)$ is already set, then output $\perp_1$.
10. Output π.

the online phase (resp. query phase) $\hat{\mathcal{P}}$ runs Com (resp. Eval) instead of SCom. Analogous to Hyb_2, one can construct reduction $\mathcal{D}$ against weak hiding of PCOM, hence:

$$\left|\Pr\left[\mathsf{Hyb}_3(1^\lambda) = 0\right] - \Pr\left[\mathsf{Hyb}_4(1^\lambda) = 0\right]\right| \leq \mathbf{Adv}^{\mathsf{PC\text{-}wHIDE}}_{\mathcal{D}}(\lambda).$$

Notice that in Hyb_4 the prover's behavior is identical to $\mathcal{S}$ described in the box. This concludes the proof. □

Compilation with Hiding Polynomial Commitments For completeness, we provide an alternative simulation strategy.

Lemma 3.5. *If PIOP is HVZK and PCOM is hiding (PC-HIDE) and has high min-entropy (Definition 2.23), then the FS-NARG protocol Π_{H} derived from $\hat{\Pi}_{\mathsf{H}}$ (Protocol 1) is TLZK with canonical simulator.*

Proof. We show the existence of canonical simulator $\mathcal{S}'$ for Π_{H}, detailed in Simulator 2. By inspection $\mathcal{S}'$ is canonical and trapdoor-less. The proof follows outline of the previous theorem. Let $\mathsf{Hyb}_0(1^\lambda)$ be an experiment identical to $\mathsf{NIZK\text{-}0}_{\mathcal{A}}(1^\lambda)$. Consider the following hybrids.

– $\mathsf{Hyb}_1(1^\lambda)$ is identical to the previous hybrid, except at derivation of challenge ρ_i for each i: once the input $(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi|_i)$ is obtained, sample ρ_i uniformly from the challenge domain, program the $\mathcal{Q}_{\mathsf{H}}(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi|_i) := \rho_i$ if $\mathcal{Q}_{\mathsf{H}}(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi|_i)$ is undefined, and abort otherwise. By high min-entropy of the commitment, the probability that $\mathsf{Hyb}_1(1^\lambda)$ aborts is negligible. Thus, we obtain

$$\left|\Pr\left[\mathsf{Hyb}_0(1^\lambda) = 0\right] - \Pr\left[\mathsf{Hyb}_1(1^\lambda) = 0\right]\right| \leq \mathsf{negl}(\lambda).$$

– $\mathsf{Hyb}_2(1^\lambda)$ is identical to the previous hybrid, except at the SRS generation and the committing phase: the $\mathcal{G}$ is modified such that it invokes $(\mathsf{ck}, \bot) \leftarrow \mathsf{SKGen}(1^\lambda)$ and outputs ck, and during the online phase $\hat{\mathcal{P}}$ runs $\mathsf{SCom}(\mathsf{ck}, \bot, \bot, \mathbf{z}, \mathbf{p}(\mathbf{z}))$ to obtain simulated $(\mathbf{c}, \boldsymbol{\pi}_{\mathsf{PCOM}})$, instead of invoking Com and Eval. Note that $\rho_1, \ldots, \rho_{\mathsf{r}}$ are picked in advance thanks to the previous hybrid, and therefore a simulated prover can indeed derive the whole $\mathbf{p}$ and $\mathbf{z}$ before generating any commitments.

  To argue Hyb_2 is indistinguishable from Hyb_1, we construct reduction $\mathcal{B}$ against hiding of PCOM analogously to the previous theorem. Thus, we have

$$\left|\Pr\left[\mathsf{Hyb}_1(1^\lambda) = 0\right] - \Pr\left[\mathsf{Hyb}_2(1^\lambda) = 0\right]\right| \leq \mathbf{Adv}^{\text{PC-HIDE}}_{\mathcal{B}}(\lambda).$$

– $\mathsf{Hyb}_3(1^\lambda)$ is identical to the previous hybrid, except at generation of polynomials $\mathbf{p}$: run the HVZK simulator for PIOP $\mathbf{S}(\mathsf{i}, \mathsf{x})$ to obtain $\mathsf{view} = (\rho_1, \ldots, \rho_{\mathsf{r}}, \mathbf{y})$, instead of invoking the PIOP prover $\mathbf{P}$. Note that SCom now takes as input $(\mathsf{ck}, \bot, \bot, \mathbf{z}, \mathbf{y})$, where $\mathbf{z} \leftarrow \mathbf{Q}_{\mathbf{V}}(\mathsf{x}; \rho_1, \ldots, \rho_{\mathsf{r}})$. Since PIOP is HVZK, we have

$$\left|\Pr\left[\mathsf{Hyb}_2(1^\lambda) = 0\right] - \Pr\left[\mathsf{Hyb}_3(1^\lambda) = 0\right]\right| \leq \mathbf{Adv}^{\text{HVZK}}_{\mathcal{C}}(\lambda).$$

Notice that in Hyb_3 the prover's behavior is identical to $\mathcal{S}'$ described in the box. This concludes the proof. □

3.2 Weak Unique Response of Compiled NARG

For a PIOP satisfying stronger Ψ-HVZK along with Ψ-NEXP and high min-entropy for the $\mathbf{Q}_{\mathbf{V}}$ algorithm, we only need weaker hiding property for PCOM. For a PIOP satisfying just HVZK along with high min-entropy for the $\mathbf{Q}_{\mathbf{V}}$ algorithm, and the constraint that $\mathsf{t}(1,1) \leq \mathsf{d}(|\mathsf{i}|)$, we require PCOM to be hiding in a stronger sense. In both cases, we require the assumption that $\mathbf{Q}_{\mathbf{V}}$ has high min-entropy. This condition is met by PIOPs such as Plonk, but is not met by other PIOPs such as Marlin and Lunar. The latter can be modified slightly to meet the condition: add a dummy polynomial in the first round and evaluate it on a random point chosen in the last round (see Section 5 for details). In the first case, we require the PIOP to also satisfy Ψ-NEXP, which as remarked in Section 2.2, just captures the intuition that many PIOPs encode the witness in the first polynomial and thus generate it with enough randomness in order to achieve zero-knowledge. Finally, in the second case, we require that the PIOP does not reveal the entire first polynomial as a part of the proof. When the first polynomial encodes the witness, this constraint again is easily satisfied by most PIOPs in order to achieve zero-knowledge.

We state our main theorem now.

Theorem 3.6. *Let $\mathcal{S}$ (Simulator 1) and $\mathcal{S}'$ (Simulator 2) be canonical TLZK simulators for FS-NARG Π_{H} derived from $\hat{\Pi}_{\mathsf{H}}$ (Protocol 1).*

1. *If PIOP satisfies Ψ-NEXP, high min-entropy of $\mathbf{Q}_{\mathbf{V}}$, and if PCOM is weak evaluation binding (PC-wEBIND), weak unique proof (PC-wUNIQ), and weak hiding (PC-wHIDE), then Π_{H} satisfies weak unique responses (WUR) with respect to $\mathcal{S}$. Concretely, for every PPT adversary $\mathcal{A}$ against WUR of Π_{H} that makes q queries to $\mathcal{S}_1$, there exist adversaries $\mathcal{B}, \mathcal{C}, \mathcal{D}$ such that,*

$$\mathbf{Adv}^{\text{WUR}}_{\mathcal{A},\mathcal{S}}(\lambda) \leq \mathbf{Adv}^{\text{PC-wUNIQ}}_{\mathcal{B}}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{PC-wEBIND}}_{\mathcal{C}}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{PC-wHIDE}}_{\mathcal{D}}(\lambda) + \frac{q\ell}{2^\alpha} + \mathsf{negl}(\lambda)$$

where $\ell := \sum_{i=0}^{i=\mathsf{r}} \left(\sum_{j=1}^{\mathsf{s}(i)} \mathsf{t}(i,j)\right)$, and α is the min-entropy of $\mathbf{Q}_{\mathbf{V}}$ (Definition 2.7).

2. *If* PIOP *satisfies high min-entropy of* $\mathbf{Q_V}$, $\mathsf{t}(1,1) \leq \mathsf{d}(|\mathsf{i}|)$, *and if* PCOM *is hiding (PC-HIDE) and satisfies all the other properties above, then* Π_{H} *satisfies weak unique responses (WUR) with respect to* $\mathcal{S}'$. *Concretely,*

$$\mathbf{Adv}^{\mathsf{WUR}}_{\mathcal{A},\mathcal{S}'}(\lambda) \leq \mathbf{Adv}^{\mathsf{PC\text{-}wUNIQ}}_{\mathcal{B}}(\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{PC\text{-}wEBIND}}_{\mathcal{C}}(\lambda) + 5 \cdot \mathbf{Adv}^{\mathsf{PC\text{-}HIDE}}_{\mathcal{D}}(\lambda) + \frac{q\ell}{2^{\alpha}} + \mathsf{negl}(\lambda)$$

Proof. Given an adversary $\mathcal{A} := \{\mathcal{A}_1, \mathcal{A}_2\}$ against WUR with respect to the simulator $\mathcal{S} := \{\mathcal{S}_1, \mathcal{S}'_2\}$, we will construct another adversary who violates some PCOM property. In the following let $\tilde{\pi}$ denote the simulated transcript and π denote the transcript output by the adversary. We define a sequence of hybrids.

- Hyb_0: Let this be the same as WUR game.
- Hyb_1: This is the same as the Hyb_0 except that the challenger aborts if π and $\tilde{\pi}$ differ for the first time for some evaluation proof $\pi_{i,j,k}$ and otherwise have identical prefix.
- Hyb_2: This is the same as the Hyb_1 game except an additional abort condition. The challenger additionally aborts if π and $\tilde{\pi}$ differ for the first time for some evaluation $y_{i,j,k}$ and otherwise have the same prefix.
- Hyb_3: This is the same as the Hyb_2 game except for additional abort conditions: The challenger additionally aborts in the following scenario. On receiving an RO query from $\mathcal{A}_2$ for a new input of the form $(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi|_{\mathsf{r}})$, i.e., a query with respect to round index r, it samples uniform $\rho_{\mathsf{r}} \in \mathsf{Ch}$, and $z_{1,1,1} \leftarrow \mathbf{Q_V}(\mathsf{x}, \rho_1, \ldots, \rho_{\mathsf{r}})$, where $\mathsf{x}, \rho_1, \ldots, \rho_{\mathsf{r}-1}$ are derived from the RO query. Abort if $z_{1,1,1} \in \tilde{\mathbf{z}}$, where $\tilde{\mathbf{z}}$ is from the simulated transcript.

Claim 1. *Let* $\mathcal{A}$ *be an adversary succeeding in the* WUR *game. Then, if the* PCOM *satisfies* PC-wHIDE, *then there exists another adversary* $\mathcal{B}$ *such that*

$$|\Pr\left[\mathsf{Hyb}_0(1^\lambda) = 1\right] - \Pr\left[\mathsf{Hyb}_1(1^\lambda) = 1\right]| \leq \mathbf{Adv}^{\mathsf{PC\text{-}wUNIQ}}_{\mathcal{B}}(\lambda).$$

And if PCOM *satisfies strong* PC-HIDE, *then there exists adversaries* $\mathcal{B}, \mathcal{B}_1$ *such that*

$$|\Pr\left[\mathsf{Hyb}_0(1^\lambda) = 1\right] - \Pr\left[\mathsf{Hyb}_1(1^\lambda) = 1\right]| \leq \mathbf{Adv}^{\mathsf{PC\text{-}wUNIQ}}_{\mathcal{B}}(\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{PC\text{-}HIDE}}_{\mathcal{B}_1}(\lambda).$$

We will bound the probability that $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$ succeeds in the WUR game, but fails in Hyb_1. First observe that the only difference between the two hybrids is the additional abort probability in Hyb_1. This happens only if π and $\tilde{\pi}$ share the same prefix and differ for the first time for some evaluation proof, i.e., for some (i,j,k), $\pi_{i,j,k} \neq \tilde{\pi}_{i,j,k}$, where $\pi_{i,j,k} \in \boldsymbol{\pi}_{\mathsf{PCOM}}$ and $\tilde{\pi}_{i,j,k} \in \tilde{\boldsymbol{\pi}}_{\mathsf{PCOM}}$. If this holds, then we construct an adversary $\mathcal{B}$ who breaks the weak unique evaluation proof property for the PCOM scheme. We separate the analysis according to the type of PCOM scheme used in compilation. The **Non Hiding case** covers the case when PCOM satisfies only weak PC-wHIDE, and the **Hiding case** covers the strong version.

Non Hiding Case. $\mathcal{B}$ starts a weak unique proof game with respect to PCOM. It receives ck from the challenger. It then starts the WUR experiment with $\mathcal{A}$ by invoking $\mathcal{A}_1$ on $\mathsf{crs} := \mathsf{ck}$. All random oracle queries by $\mathcal{A}_1$ are answered by $\mathcal{B}$ by running $\mathcal{S}_1$ locally. $\mathcal{A}_1$ finally outputs x and its internal state $\mathsf{st}_{\mathcal{A}}$. $\mathcal{B}$ runs steps in $\mathcal{S}'_2$ explicitly on input (i, x), i.e., it runs step 2 in Simulator 1 to receive $(\mathsf{view}, \tilde{\mathbf{p}})$. $\mathcal{B}$ queries $\tilde{\mathbf{p}}$ to the oracle $\mathcal{O}_{\mathsf{Com}}$ to receive commitments, and then queries $\mathcal{O}_{\mathsf{Eval}}$ on $\tilde{\mathbf{z}}$ to receive evaluation proofs. Here, $\tilde{\mathbf{z}}$ is derived from view. $\mathcal{B}$ computes $\tilde{\pi}$ given view, commitments and evaluation proofs (received from $\mathcal{O}_{\mathsf{Com}}$ and $\mathcal{O}_{\mathsf{Eval}}$), and sends $\tilde{\pi}$ to $\mathcal{A}_2$. On running $\mathcal{A}_2$ on $\tilde{\pi}$ and $\mathsf{st}_{\mathcal{A}}$, it receives π as output. Let (i,j,k) be the first tuple for which $\pi_{i,j,k} \neq \tilde{\pi}_{i,j,k}$. $\mathcal{B}$ submits $(c_{i,j}, z_{i,j,k}, y_{i,j,k}, \pi_{i,j,k}, \tilde{\pi}_{i,j,k})$ to the challenger. Note that the view generated by $\mathcal{B}$ is exactly as the one that would be generated by running

$\mathcal{S}_2'$. Thus, difference in the winning probability of $\mathcal{A}$ when participating in Hyb_0 vs Hyb_1 is bounded by the advantage of $\mathcal{B}$ in weak unique evaluation proofs game. Thus,

$$|\Pr\left[\mathsf{Hyb}_0(1^\lambda)=1\right]-\Pr\left[\mathsf{Hyb}_1(1^\lambda)=1\right]| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{PC\text{-}wUNIQ}}(\lambda).$$

Hiding Case. Consider the following sub-hybrid,

- $\mathsf{Hyb}_{0,1}$: Same as the Hyb_0 except the following changes. On receiving x from $\mathcal{A}_1$, run steps for $\mathcal{S}_2'$ in Simulator 2 explicitly, i.e., run step 2 to receive view. Now, for $i \in [\mathsf{r}], j \in \mathsf{s}(i)$, randomly sample $\hat{p}_{i,j} \leftarrow \mathbb{F}^{\mathsf{d}(|\mathsf{i}|)}[X]$ such that $\hat{p}_{i,j}(\tilde{\mathbf{z}}_{i,j}) := \tilde{\mathbf{y}}_{i,j}$. Instead of executing SCom, compute Com honestly on polynomials $\hat{p}_{i,j}$. Continue with the rest of the steps as in Simulator 2. Notice that this is indistinguishable from the view in Hyb_0 because of hiding of the PCOM scheme, i.e., there exists an adversary $\mathcal{B}_1$ such that,

$$|\Pr\left[\mathsf{Hyb}_0(\lambda)=1\right]-\Pr\left[\mathsf{Hyb}_{0,1}(\lambda)=1\right]| \leq \mathbf{Adv}_{\mathcal{B}_1}^{\mathsf{PC\text{-}HIDE}}(\lambda).$$

- $\mathsf{Hyb}_{0,2}$: Same as the previous hybrid, except that it aborts when adversary submits π containing an evaluation proof $\pi_{i,j,k}$ that differs from $\tilde{\pi}_{i,j,k}$ of the simulated $\tilde{\pi}$. Using such an adversary $\mathcal{A}$, one can break the weak unique proof property. The reduction $\mathcal{B}$ playing the weak unique proof game receives ck as input and emulates the procedures of $\mathsf{Hyb}_{1,2}$, but instead of computing Com locally, on polynomials $\hat{p}_{i,j}$, it queries these to $\mathcal{O}_{\mathsf{Com}}$ to receive the commitments, and queries $\mathcal{O}_{\mathsf{Eval}}$ to receive evaluation proofs. It constructs $\tilde{\pi}$ given these values. Similar to the non-hiding case, here too the winning probability of $\mathcal{A}$ is bounded by the advantage of $\mathcal{B}$ in weak unique evaluation proofs game.

$$|\Pr\left[\mathsf{Hyb}_{0,1}(1^\lambda)=1\right]-\Pr\left[\mathsf{Hyb}_{0,2}(1^\lambda)=1\right]| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{PC\text{-}wUNIQ}}(\lambda).$$

- Hyb_1: Same as the $\mathsf{Hyb}_{0,2}$ except that commitments and proofs are generated by SCom as in $\mathcal{S}'$. Again, due to the hiding property we get

$$|\Pr\left[\mathsf{Hyb}_{0,2}(1^\lambda)=1\right]-\Pr\left[\mathsf{Hyb}_1(1^\lambda)=1\right]| \leq \mathbf{Adv}_{\mathcal{B}_1}^{\mathsf{PC\text{-}HIDE}}(\lambda).$$

This gives,

$$|\Pr\left[\mathsf{Hyb}_0(1^\lambda)=1\right]-\Pr\left[\mathsf{Hyb}_1(1^\lambda)=1\right]| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{PC\text{-}wUNIQ}}(\lambda)+2\cdot\mathbf{Adv}_{\mathcal{B}_1}^{\mathsf{PC\text{-}HIDE}}(\lambda).$$

Claim 2. *Let $\mathcal{A}$ be an adversary succeeding in the Hyb_1 game. Then, if PCOM satisfies PC-wHIDE, then there exists another adversary $\mathcal{C}$ such that*

$$|\Pr\left[\mathsf{Hyb}_1(1^\lambda)=1\right]-\Pr\left[\mathsf{Hyb}_2(1^\lambda)=1\right]| \leq \mathbf{Adv}_{\mathcal{C}}^{\mathsf{PC\text{-}wEBIND}}(\lambda).$$

And if PCOM satisfies strong PC-HIDE, then there exist adversaries $\mathcal{C}, \mathcal{C}_1$ such that

$$|\Pr\left[\mathsf{Hyb}_1(1^\lambda)=1\right]-\Pr\left[\mathsf{Hyb}_2(1^\lambda)=1\right]| \leq \mathbf{Adv}_{\mathcal{C}}^{\mathsf{PC\text{-}wEBIND}}(\lambda)+2\cdot\mathbf{Adv}_{\mathcal{C}_1}^{\mathsf{PC\text{-}HIDE}}(\lambda).$$

We will bound the probability that $\mathcal{A}=\{\mathcal{A}_1,\mathcal{A}_2\}$ succeeds in the Hyb_1 game, but fails in Hyb_2. This happens only if π and $\tilde{\pi}$ share the same prefix and differ for the first time for some evaluation $y_{i,j,k}$, i.e., for some (i,j,k), $y_{i,j,k} \neq \tilde{y}_{i,j,k}$. Here, we construct an adversary $\mathcal{C}$ who breaks weak evaluation binding for PCOM. As in the previous case, this analysis is also separated into the **Non Hiding** and the **Hiding** case.

Non Hiding Case. $\mathcal{C}$ starts the weak evaluation binding game with respect to PCOM scheme. $\mathcal{C}$'s execution is very similar to the previous case. It receives ck from the challenger,

and starts the Hyb_1 experiment with $\mathcal{A}$ by invoking $\mathcal{A}_1$ on $\mathsf{crs} := \mathsf{ck}$. All random oracle queries are answered by running $\mathcal{S}_1$. When $\mathcal{A}_1$ outputs x and state $\mathsf{st}_{\mathcal{A}}$, $\mathcal{C}$ runs step 2 in $\mathcal{S}_2'$ explicitly, queries $\mathcal{O}_{\mathsf{Com}}$ and $\mathcal{O}_{\mathsf{Eval}}$ as before to construct $\widetilde{\pi}$. It then invokes $\mathcal{A}_2$ with inputs $(\widetilde{\pi}, \mathsf{st}_{\mathcal{A}})$ to receive π. Since the prefix is the same for both transcripts, $\rho_1, \ldots, \rho_{\mathsf{r}}$ is the same as well, which means that they have the same query vector $\mathbf{z}_{i,j}$. Since $\mathcal{A}$ fails in Hyb_2 game but not in Hyb_1, there exists an (i, j, k) and valid $\pi_{i,j,k}, \tilde{\pi}_{i,j,k}$ such that $y_{i,j,k} \neq \tilde{y}_{i,j,k}$ for the same commitment $\tilde{c}_{i,j}$ and evaluation point $\tilde{z}_{i,j,k}$. $\mathcal{C}$ simply outputs $(\tilde{c}_{i,j}, \tilde{z}_{i,j,k}, \tilde{y}_{i,j,k}, y_{i,j,k}, \tilde{\pi}_{i,j,k}, \pi_{i,j,k})$ to the weak evaluation binding challenger and wins the game. Thus, the difference in the winning probability of $\mathcal{A}$ when participating in Hyb_1 vs Hyb_2 is bounded by the advantage of $\mathcal{C}$ in the weak evaluation binding game, i.e.,

$$|\Pr\left[\mathsf{Hyb}_1(1^\lambda) = 1\right] - \Pr\left[\mathsf{Hyb}_2(1^\lambda) = 1\right]| \leq \mathbf{Adv}_{\mathcal{C}}^{\text{PC-wEBIND}}(\lambda).$$

Hiding Case. Here too, as in the hiding case of Claim 1, we consider two sub hybrids with the same behaviour except that, instead of invoking PC-wUNIQ challenger in $\mathsf{Hyb}_{1,2}$, we will invoke the challenger of PC-wEBIND game. As before, we can conclude that,

$$|\Pr\left[\mathsf{Hyb}_1(1^\lambda) = 1\right] - \Pr\left[\mathsf{Hyb}_2(1^\lambda) = 1\right]| \leq \mathbf{Adv}_{\mathcal{C}}^{\text{PC-wEBIND}}(\lambda) + 2 \cdot \mathbf{Adv}_{\mathcal{C}_1}^{\text{PC-HIDE}}(\lambda).$$

Claim 3. *Let $\mathcal{A}$ be an adversary who makes up to q random oracle queries and succeeds in the Hyb_2 game. Then,*

$$|\Pr\left[\mathsf{Hyb}_2(1^\lambda) = 1\right] - \Pr\left[\mathsf{Hyb}_3(1^\lambda) = 1\right]| \leq (q\ell)/2^\alpha.$$

where $\ell := \sum_{i=0}^{\mathsf{r}} \left(\sum_{j=1}^{\mathsf{s}(i)} \mathsf{t}(i,j)\right)$, and α is the min-entropy of $\mathbf{Q}_{\mathbf{V}}$.

An adversary wins in Hyb_2 but fails to win in Hyb_3 when it makes an unrecorded RO query $(\mathsf{srs}, \mathsf{i}, \mathsf{x}, \pi|_{\mathsf{r}})$ such that $z_{1,1,1} \in \tilde{\mathbf{z}}$, where $\tilde{\mathbf{z}} \leftarrow \mathbf{Q}_{\mathbf{V}}(\mathsf{x}; \rho_1, \ldots, \rho_{\mathsf{r}})$, and $\rho_1, \ldots, \rho_{\mathsf{r}-1}$ are derived from the query. For a given random oracle query, and any given value of $z \in \tilde{\mathbf{z}}$, $z_{1,1,1} = z$ only with probability $2^{-\alpha}$, which is negligible assuming high min-entropy of $\mathbf{Q}_{\mathbf{V}}$. Let q be the total number of random oracle queries made, and let $\ell := |\tilde{\mathbf{z}}|$, the additional abort probability introduced by this hybrid is $q\ell/2^\alpha$.

Claim 4. *Let $\mathcal{A}$ be an adversary succeeding in the Hyb_3 game. Then, if PCOM satisfies PC-wHIDE, underlying PIOP satisfies Ψ-NEXP, and high min-entropy of $\mathbf{Q}_{\mathbf{V}}$, then there exist adversaries $\mathcal{D}_1, \mathcal{D}_2$ such that*

$$\Pr\left[\mathsf{Hyb}_3(1^\lambda) = 1\right] \leq 2\mathbf{Adv}_{\mathcal{D}_1}^{\text{PC-wHIDE}}(\lambda) + \mathbf{Adv}_{\mathcal{D}_2}^{\text{PC-wEBIND}}(\lambda) + \mathsf{negl}(\lambda)$$

And if PCOM satisfies PC-HIDE, the underlying PIOP satisfies high min-entropy of $\mathbf{Q}_{\mathbf{V}}$, and $\mathsf{t}(1,1) \leq \mathsf{d}(|\mathsf{i}|)$, then there exist adversaries $\mathcal{D}_1, \mathcal{D}_2$ such that

$$\Pr\left[\mathsf{Hyb}_3(1^\lambda) = 1\right] \leq \mathbf{Adv}_{\mathcal{D}_1}^{\text{PC-HIDE}}(\lambda) + \mathbf{Adv}_{\mathcal{D}_2}^{\text{PC-wEBIND}}(\lambda) + \mathsf{negl}(\lambda)$$

We will bound the probability that $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$ succeeds in the Hyb_3 game. This happens only if π and $\widetilde{\pi}$ differ for the first time for some polynomial commitment $c_{i,j}$, i.e., for some $1 < i \leq \mathsf{r}$ and $j \in [\mathsf{s}(i)]$, $c_{i,j} \neq \tilde{c}_{i,j}$. As before, we consider two cases: (1) when the PIOP satisfies Ψ-HVZK, Ψ-NEXP and PCOM scheme satisfies the weak hiding property, and (2) when the PCOM scheme satisfies the strong hiding property while the PIOP satisfies plain HVZK. For both cases we will construct an adversary who breaks hiding (weak or strong depending on the case).

Non Hiding Case. Consider the following sub-hybrids.

– $\mathsf{Hyb}_{3,1}$: This is the same as Hyb_3 except that ck is generated as $(\mathsf{ck}, \chi) \leftarrow \mathsf{SKGen}(1^\lambda)$ and $\tilde{\mathbf{p}}$ is committed to using SCom. As was argued in Hyb_2 of Lemma 3.4, this switch is indistinguishable from Hyb_3 because of weak hiding of PCOM, i.e.,

$$|\Pr\left[\mathsf{Hyb}_3(1^\lambda) = 1\right] - \Pr\left[\mathsf{Hyb}_{3,1}(1^\lambda) = 1\right]| \leq \mathbf{Adv}^{\mathsf{PC\text{-}wHIDE}}_{\mathcal{D}_1}(\lambda).$$

– $\mathsf{Hyb}_{3,2}$: This is the same as $\mathsf{Hyb}_{3,1}$ except the following abort conditions. As before, let π be the transcript submitted by $\mathcal{A}_2$, and $z_{1,1,1}, y_{1,1,1}, \pi_{1,1,1}$ be the first evaluation and proof for commitment $c_{1,1}$ in π. Let $\tilde{p}_{1,1}$ be the corresponding polynomial in $\tilde{\mathbf{p}}$ output by $\mathbf{S}$. The experiment aborts by outputting $\perp$ if $z_{1,1,1} = \chi$ or $\tilde{p}_{1,1}(z_{1,1,1}) = y_{1,1,1}$.

  First note that, assuming that the prefix $\pi|_{i'}$ was queried to the RO, $\rho_\mathsf{r} \neq \tilde{\rho}_\mathsf{r}$ and $z_{1,1,1} \notin \tilde{\mathbf{z}}$ (because of the argument in Claim 3). Moreover, $z_{1,1,1} \neq \chi$ because of high min-entropy of $\mathbf{Q_V}$. Finally, since $\tilde{p}_{1,1}$ satisfies the non-extrapolatable property for the first polynomial (in Definition 2.6), $\tilde{p}_{1,1}(z_{1,1,1}) = y_{1,1,1}$ only with negligible probability, for a new $z_{1,1,1} \notin (\tilde{\mathbf{z}}, \chi)$. Thus, $p_{1,1}(z_{1,1,1}) \neq y_{1,1,1}$, except with $\mathsf{negl}(\lambda)$ probability. Hence, we obtain

$$|\Pr\left[\mathsf{Hyb}_{3,1}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Hyb}_{3,2}(1^\lambda) = 1\right]| \leq \mathsf{negl}(\lambda).$$

– $\mathsf{Hyb}_{3,3}$: This is the same as Hyb_3 again, i.e., we switch back to using KGen and Com. Once again, because of weak hiding, this switch should be indistinguishable to any adversary, i.e.,

$$|\Pr\left[\mathsf{Hyb}_{3,2}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Hyb}_{3,3}(1^\lambda) = 1\right]| \leq \mathbf{Adv}^{\mathsf{PC\text{-}wHIDE}}_{\mathcal{D}_1}(\lambda).$$

This implies that, here too, $\tilde{p}_{1,1}(z_{1,1,1}) \neq y_{1,1,1}$ except with $\mathsf{negl}(\lambda)$ probability. However, when this happens, we can build an adversary $\mathcal{D}_2$ against PC-wEBIND game. $\mathcal{D}_2$ receives ck, invokes $\mathcal{A}_1$ internally on ck, and receives x in response. $\mathcal{D}_2$ generates a view for $\mathcal{A}$ that is identical to $\mathsf{Hyb}_{3,3}$ by generating commitments and evaluation proofs by querying $\mathcal{O}_{\mathsf{Com}}$ and $\mathcal{O}_{\mathsf{Eval}}$ oracles on $\tilde{\mathbf{p}}$. Now, if $\mathcal{A}$ returns a transcript such that $y^* := \tilde{p}_{1,1}(z_{1,1,1}), y^* \neq y_{1,1,1}$, then $\mathcal{D}_2$ can query $\mathcal{O}_{\mathsf{Eval}}$ to generate an honest evaluation proof π^* for $(y^*, z_{1,1,1})$, and output tuple $(\tilde{c}_{1,1}, z_{1,1,1}, y_{1,1,1}, y^*, \pi_{1,1,1}, \pi^*)$ to the weak evaluation binding challenger, and win the game. Thus, we get,

$$\Pr\left[\mathsf{Hyb}_{3,3}(1^\lambda) = 1\right] \leq \mathbf{Adv}^{\mathsf{PC\text{-}wEBIND}}_{\mathcal{D}_2}(\lambda)$$

Overall, in the non hiding case we obtain,

$$\Pr\left[\mathsf{Hyb}_3(1^\lambda) = 1\right] \leq 2\mathbf{Adv}^{\mathsf{PC\text{-}wHIDE}}_{\mathcal{D}_1}(\lambda) + \mathbf{Adv}^{\mathsf{PC\text{-}wEBIND}}_{\mathcal{D}_2}(\lambda) + \mathsf{negl}(\lambda)$$

Hiding Case. Consider the following sub-hybrid.

– $\mathsf{Hyb}_{3,1}$: This is the same as Hyb_3 except the following changes. For $i \in [\mathsf{r}], j \in [\mathsf{s}(i)]$, random polynomials $\hat{p}_{i,j} \in \mathbb{F}^{\mathsf{d}(|\mathsf{i}|)}[X]$ are sampled and stored locally by the simulator, with the constraint that $\hat{p}_{i,j}(\tilde{\mathbf{z}}_{i,j}) := \tilde{\mathbf{y}}_{i,j}$. Once again commitments $\tilde{\mathbf{c}}$ are obtained by calling SCom as in Hyb_3. When $\mathcal{A}$ submits π, let $z_{1,1,1}, y_{1,1,1}, \pi_{1,1,1}$ be the first evaluation and proof for commitment $c_{1,1}$ in π. If $\hat{p}_{1,1}(z_{1,1,1}) = y_{1,1,1}$ then $\mathsf{Hyb}_{3,1}$ aborts by outputting $\perp$.

  The adversary's view is identical in $\mathsf{Hyb}_{3,1}$ and Hyb_3 since the inputs to SCom remain identical, and the all of the steps executed in Hyb_3 are executed with identical inputs here as well. The only difference is that, here, the simulator stores local polynomials $\hat{p}_{i,j}$ and the experiment aborts if an evaluation of the secret polynomial $\hat{p}_{1,1}$ is guessed by the adversary. We argue that the probability of abort is negligible.

  Assume that the number of evaluations revealed for the first polynomial in the PIOP is at most equal to the highest allowed degree, i.e., $\mathsf{t}(1,1) \leq \mathsf{d}(|\mathsf{i}|)$. Given this assumption, we

can conclude that the inputs to SCom, which consists of evaluation points and evaluations with respect to the first polynomial, are not sufficient to define a degree $\mathsf{d}(|\mathsf{i}|)$ polynomial. This also means that the commitment $\tilde{c}_{1,1}$ output by SCom leaks nothing to $\mathcal{A}$ about the additional evaluation needed to interpolate the polynomial. Since, we pick a degree $\mathsf{d}(|\mathsf{i}|)$ polynomial $\hat{p}_{1,1}$ at random, it has at least one extra degree of freedom. Finally, because of Claim 3, we know that $z_{1,1,1} \notin \tilde{\mathbf{z}}_1$. Thus, for a new point $z_{1,1,1}$, the probability that $\hat{p}_{1,1}(z_{1,1,1}) = y_{1,1,1}$ is $1/|\mathbb{F}|$.

$$|\Pr\left[\mathsf{Hyb}_3(1^\lambda) = 1\right] - \Pr\left[\mathsf{Hyb}_{3,1}(1^\lambda) = 1\right]| \leq 1/|\mathbb{F}|.$$

– $\mathsf{Hyb}_{3,2}$: Same as $\mathsf{Hyb}_{3,1}$ except that all the polynomials are committed to using Com. This is indistinguishable from $\mathsf{Hyb}_{3,1}$ because of hiding of PCOM:

$$|\Pr\left[\mathsf{Hyb}_{3,1}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Hyb}_{3,2}(1^\lambda) = 1\right]| \leq \mathbf{Adv}^{\mathsf{PC\text{-}HIDE}}_{\mathcal{D}_1}(\lambda).$$

Finally, we bound the probability of an adversary winning in $\mathsf{Hyb}_{3,2}$. Here, the transcript $\widetilde{\pi}$ is constructed according to $\mathsf{Hyb}_{3,2}$, and thus, the commitment $\tilde{c}_{1,1}$ is for polynomial $\hat{p}_{1,1}$. $\mathcal{A}$ outputs π such that $c_{i,j}$, $1 < i \leq \mathsf{r}, j \in [\mathsf{s}(\mathsf{r})]$, $c_{i,j} \neq \tilde{c}_{i,j}$. Because of the abort condition in $\mathsf{Hyb}_{3,1}$, here, $\hat{p}_{1,1}(z_{1,1,1}) \neq y_{1,1,1}$ which violates PC-wEBIND. We can construct a reduction towards breaking PC-wEBIND in a similar manner to the hybrids in the non-hiding case.

$$\Pr\left[\mathsf{Hyb}_{3,2}(1^\lambda) = 1\right] \leq \mathbf{Adv}^{\mathsf{PC\text{-}wEBIND}}_{\mathcal{D}_2}(\lambda)$$

And overall, we get,

$$\Pr\left[\mathsf{Hyb}_3(1^\lambda) = 1\right] \leq \mathbf{Adv}^{\mathsf{PC\text{-}HIDE}}_{\mathcal{D}_1}(\lambda) + \mathbf{Adv}^{\mathsf{PC\text{-}wEBIND}}_{\mathcal{D}_2}(\lambda) + 1/|\mathbb{F}|$$

Putting the three claims together we get the probability bound stated in the theorem. □

4 Case Studies: Polynomial Commitment Schemes

4.1 KZG commitment scheme

We show that the polynomial commitment scheme $\mathsf{KZG} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Eval}, \mathsf{Check})$ satisfies all the necessary properties. If the operations highlighted in orange are executed, then we obtain the randomized variant, rKZG.

– $\mathsf{KGen}(1^\lambda, D)$: Generate the parameters of a bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, g, h, e)$ where $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = q$ is prime, $\langle g \rangle = \mathbb{G}_1$, $\langle h \rangle = \mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable, non-degenerate bilinear map. The group order p also determines $\mathbb{F} := \mathbb{F}_p$ and a set of supported polynomials $\mathbb{F}^D[X]$. The algorithm samples $\alpha, \beta \in \mathbb{F}$ uniformly, and compute $\sigma = (g, g^\alpha, \ldots, g^{\alpha^D}, g^{\beta\alpha}, \ldots, g^{\beta\alpha^D}, h, h^\alpha)$. Output $\mathsf{ck} = (\mathcal{G}, \sigma)$.

– $\mathsf{Com}(\mathsf{ck}, f)$: On input ck, a polynomial $f \in \mathbb{F}^D[X]$, sample $\hat{f} \xleftarrow{\$} \mathbb{F}^D[X]$, and generate a commitment as $c = g^{f(\alpha)+\beta\hat{f}(\alpha)}$ and output c.

– $\mathsf{Eval}(\mathsf{ck}, c, z, f(z), f, \hat{f})$: Compute $\omega(X) = (f(X) - f(z))/(X - z)$ and $\hat{\omega}(X) = (\hat{f}(X) - \hat{f}(z))/(X - z)$. Output $\pi = (g^{\omega(\alpha)+\beta\hat{\omega}(\alpha)}, \hat{f}(z))$.

– $\mathsf{Check}(\mathsf{ck}, c, z, y, \pi)$: Parse $\pi = (\pi, \hat{y})$. Accept if and only if $e(c/(g^y g^{\beta\hat{y}}), h) = e(\pi, h^\alpha/h^z)$.

We recall the SDH assumption [BB04].

Definition 4.1 (SDH Assumption). *The strong Diffie-Hellman assumption (SDH) holds with respect to a bilinear group generator* BGen *if for all* PPT *adversaries* $\mathcal{A}$ *and degree bound* $D > 0$,

$$\Pr\left[t = g^{\frac{1}{\alpha+c}} \;:\; \mathcal{G} \leftarrow \mathsf{BGen}(1^\lambda); \alpha \xleftarrow{\$} \mathbb{F}; \sigma := (\{g^{\alpha^i}\}_{i=0}^D, h^\alpha); (t, c) \leftarrow \mathcal{A}(\mathcal{G}, \sigma)\right] \leq \mathsf{negl}(\lambda)$$

Evaluation binding of both schemes is known under the SDH assumption [KZG10].

Lemma 4.2. *KZG is perfectly unique and perfectly weak hiding. rKZG is computationally unique under the SDH assumption and perfectly hiding.*

Proof. **Unique Proof.** Unique proof for KZG is proven in [GKO+23]. Unique proof for rKZG requires the SDH assumption. Suppose the adversary outputs $(c, z, y, (\pi, \hat{y}), (\pi', \hat{y}'))$ such that $(\pi, \hat{y}) \neq (\pi', \hat{y}')$ and $e(c/(g^y g^{\beta\hat{y}}), h) = e(\pi, h^\alpha/h^z)$ and $e(c/(g^y g^{\beta\hat{y}'}), h) = e(\pi', h^\alpha/h^z)$. If $\hat{y} = \hat{y}'$ or $\pi = \pi'$, then the verification condition uniquely determines the other part of valid proof string. Therefore, we only need to rule out the case $\hat{y} \neq \hat{y}'$ and $\pi \neq \pi'$. To this end, we construct reduction $\mathcal{B}$, given $\mathcal{G}$ and $\sigma = (\{g^{\alpha^i}\}_{i=0}^D, h^\alpha)$ as input, breaks the SDH assumption. $\mathcal{B}$ first samples uniform $\beta \in \mathbb{F}$ and hands over a commitment key $\mathsf{ck} = (\mathcal{G}, (\{g^{\alpha^i}\}_{i=0}^D, \{g^{\beta\alpha^i}\}_{i=0}^D, h^\alpha))$ to a unique response adversary $\mathcal{A}$. Upon receiving $(c, z, y, (\pi, \hat{y}), (\pi', \hat{y}'))$ from $\mathcal{A}$, $\mathcal{B}$ outputs $t = (\pi/\pi')^{1/(\beta(\hat{y}'-\hat{y}))}$ and $-z$ as a solution to the SDH problem. Note that we indeed have that $(\pi/\pi')^{1/(\beta(\hat{y}'-\hat{y}))} = g^{1/(\alpha-z)}$ thanks to the verification condition and therefore $\mathcal{B}$ outputs a valid solution as long as $\mathcal{A}$ outputs a non-unique proof such that $\hat{y} \neq \hat{y}'$ and $\pi \neq \pi'$.

(Weak) Hiding. For KZG, SKGen is identical to KGen except that it additionally outputs the trapdoor $\chi = \alpha$. $\mathsf{SCom}(\mathsf{ck}, \chi, y_\chi, \mathbf{z}, \mathbf{y})$ outputs $(c, \pi) = (g^{y_\chi}, g^{(y_\chi - y_i)/(\chi - z_i)})$. Note that SCom can simulate these without knowing f, thanks to the knowledge of χ and $y_\chi = f(\chi)$.

For rKZG, SKGen is identical to KGen and $\mathsf{SCom}(\mathsf{ck}, \perp, \perp, \mathbf{z}, \mathbf{y})$ proceeds as follows: (1) uniformly sample $f' \in \mathbb{F}^D[X]$ such that $f'(\mathbf{z}) = \mathbf{y}$, (2) $c \xleftarrow{\hat{f}} \mathsf{Com}(\mathsf{ck}, f')$, (3) $\pi \leftarrow \mathsf{Eval}(\mathsf{ck}, c, \mathbf{z}, \mathbf{y}, f', \hat{f})$, and (4) output (c, π).

Remark 4.3. The extractability of (r)KZG holds under the algebraic group model. All of the above properties also hold for a version of (r)KZG where the commitment c consists of two group elements. Since this variant is known to be extractable under the (d)PKE assumption [CHM+20, B.2], one can obtain simulation-extractable iNARG both in the standard model and in the AGM.

4.2 Commitment schemes based on the hardness of Discrete Log assumption

To show application towards commitment schemes in a different setting, we consider polynomial commitment schemes with transparent setup build on hardness of discrete log assumption. We consider a simple PCOM scheme built from compressed sigma protocols in [AC20]. In the following we will only provide proof sketches for showing weak unique proofs, weak evaluation binding, and strong hiding properties for it. The intuition for proving these properties are generic enough to be applicable to other similar schemes such as [BBB+18, BG18].

Definition 4.4 (DL-REL). *Given an adversary $\mathcal{A}$ and $n \geq 2$, the advantage of finding a non-trivial discrete logarithm relation between random n generators is*

$$\mathbf{Adv}_{\mathcal{G}}^{\mathsf{DL\text{-}REL}}(\mathcal{A}) = \Pr\left[\mathbf{g}^{\mathbf{x}} = 1_{\mathbb{G}} \wedge \mathbf{x} \neq \mathbf{0} \;:\; \begin{array}{r} \mathbb{G} \leftarrow \mathcal{G}(1^\lambda); \mathbf{g} := (g_1, \ldots, g_n) \xleftarrow{\$} \mathbb{G}^n; \\ \mathbf{x} := (x_1, \ldots, x_n) \leftarrow \mathcal{A}(\mathbb{G}, \mathbf{g}) \end{array}\right].$$

PCOM scheme from Compressed Sigma Protocols [AC20] The work of [AC20] gives an interactive protocol for proving evaluations of a public linear function to a committed vector with communication that is logarithmic in the length of the vector. They give a protocol $\Pi := (\mathcal{P}, \mathcal{V})$ (Protocol 5 in [AC20], and denoted below by Π) for linear form evaluations, which proves statements from the following relation:

$$\mathcal{R}_L = \{(P \in \mathbb{G}, L \in \mathcal{L}(\mathbb{Z}_p^n), y \in \mathbb{Z}_p); (\mathbf{x} \in \mathbb{Z}_p^n, \rho_c \in \mathbb{Z}_p) : P = \mathbf{g}^{\mathbf{x}} h^{\rho_c}, y = L(\mathbf{x})\}$$

Π is proven to be complete, special-HVZK, and a special sound protocol in [AC20]. These properties are summarized in Theorem 4.5.

Theorem 4.5 (Theorem 3 in [AC20]). *Π is a $(2\ell+3)$-round protocol for relation $\mathcal{R}_L$, where $\ell = \lceil \log_2(n+1) \rceil$. It is perfectly complete, special honest-verifier zero-knowledge and computationally $(2,2,3,\ldots,3)$-special sound, under the discrete logarithm assumption.*

Π is a public coin interactive protocol that can be compiled into a non-interactive one using Fiat-Shamir transform in the random oracle model. Since Π satisfies special soundness, from [AFK21], we know that the Fiat-Shamir compiled non-interactive Π_{H} satisfies EXT. Moreover, an HVZK simulation of interactive argument (without trapdoor) implies canonical NIZK simulator for the Fiat-Shamir compiled protocol [GOP+23, FKMV12].

Theorem 4.6 (Informal). *Π_{H} derived from Fiat-Shamir transformation of Π satisfies black-box EXT, and NIZK.*

Given Π_{H}, one can construct a PCOM scheme (CSP) to commit to polynomials of degree at most n, as:

1. $(\mathbb{G}, \mathbf{g}, h) \leftarrow \mathsf{KGen}(1^\lambda)$: Set pp as group elements $\mathbf{g}, h \in \mathbb{G}$ where discrete-log is hard. Output pp.
2. $c \xleftarrow{\rho_c} \mathsf{Com}(\mathsf{pp}, f)$: Compute Pedersen commitment of the coefficients as $c = \mathbf{g}^{\mathbf{x}} h^{\rho_c}$, where $\mathbf{x}$ denotes the coefficients of polynomial f.
3. $\pi \leftarrow \mathsf{Eval}(\mathsf{pp}, c, z, f, \rho_c)$: Compute a linear form L with coefficients $(1, z, \ldots, z^d)$, and evaluation proof $\pi = \Pi_{\mathsf{H}}.\mathcal{P}(\mathsf{pp}, c, L, y; \mathbf{x}, \rho_c)$, where $\mathbf{x}$ denotes the coefficients of polynomial f. Output π.
4. $b \leftarrow \mathsf{Check}(\mathsf{pp}, \pi, c, z, y)$: Compute L as the coefficients $(1, z, \ldots, z^d)$ and output $b \leftarrow \Pi_{\mathsf{H}}.\mathcal{V}(\mathsf{pp}, c, L, y, \pi)$.

Now we provide intuition for why Π_{H} satisfies properties required by our framework.

Lemma 4.7. *CSP satisfies PC-HIDE, and assuming solving DL-REL is hard, it satisfies PC-wEBIND and PC-wUNIQ.*

Proof sketch.
Strong Hiding. Existence of SCom algorithm is implied by the NIZK property. Given (ck, $\perp, \perp, \mathbf{z}, f(z_1), \ldots, f(z_m)$), SCom computes a random $P \leftarrow \mathbb{G}$ and, for each $z_i, f(z_i)$, executes the NIZK simulator for Π_{H} on $(P, L, f(z_i))$, where $L := (1, z_i, \ldots, z_i^d)$, and d is the degree of f. SCom outputs P and the proofs generated by the underlying simulator.

Weak Unique Proofs. One can split the analysis here into two cases: first, when the proof submitted by the adversary π, shares no common prefix with the simulated one $\widetilde{\pi}$; and second, when they share a common prefix. In the first, we can invoke the extractor of Π_{H} (which is guaranteed by Theorem 4.6) and obtain a witness which corresponds to the polynomial coefficients and randomness in the polynomial commitment P. This extraction is guaranteed as otherwise one can show a reduction contradicting EXT similar to Lemma 2.16. This would imply that given such an adversary, there is a way to extract the randomness used for the commitment. This also means that this strategy can be used to break the hiding property of this scheme:

- Hyb_0: Let this be the same as PC-wHIDE game.
- Hyb_1: The change here is that instead of using Com in the challenge responses, run SCom instead. The two hybrids remain indistinguishable because of strong hiding property. Note that this is true only if the oracles in PC-wUNIQ game are accessed once in the beginning in a non-adaptive way, i.e., similar to oracle accesses in PC-HIDE game. This

requirement can be met by weakening the PC-wHIDE definition even more and adjusting the proofs where PC-wHIDE is used in a minor way.

– Hyb_2: Here, on receiving a polynomial f and evaluation points $\mathbf{z}$ from the adversary, sample a random polynomial $\hat{f}$ that agrees with f on $\mathbf{z}$ and is random otherwise. Commit to this using Com and generate proofs using Eval. This hybrid should remain indistinguishable since the amount of information seen by SCom remains the same in both hybrids.

The above argument tells us that hybrids Hyb_0 and Hyb_2 are indistinguishable. However, if there exists an adversary $\mathcal{A}$ against PC-wHIDE (satisfying the first case above), then we can construct a distinguisher $\mathcal{D}$ or Hyb_0 and Hyb_2. $\mathcal{D}$ receives $f, \mathbf{z}$ from $\mathcal{A}$ and relays all the messages between the adversary and the hybrids (either Hyb_0 or Hyb_2). Observe that if $\mathcal{D}$ sees a view with respect to Hyb_0 then the commitment P and proofs $\tilde{\pi}$ received by it correspond to f. Otherwise, it corresponds to a random $\hat{f}$. On receiving π from the adversary, extract the randomness, and the polynomial from the commitment. Using these, $\mathcal{D}$ checks if the extracted polynomial matches f. If it does then this is the case with Hyb_0, otherwise it is Hyb_2.

For proving the other case, the ideas here are very similar to the ones used in [GOP+23] to prove weak unique proofs for Bulletproofs. Here, we are restricted by the common prefix, and cannot rewind the adversary all the way in order to obtain a witness. Instead, we only rewind up until the point where the two transcripts deviate. Using just this partial rewinding, we can now derive the coefficients in the exponent for the last matching message, and derive a deterministic relation between this message, the next round challenge, and the next round message. This would mean that, unless the adversary breaks DL-REL, since the matching message and the next round challenge are the same (because of the shared prefix), the next round message has to be the same. This argument can be repeated for each round to conclude that the entire proof string has to be the same.

Weak Evaluation Binding. Changing the evaluation changes the statement and we can directly extract from the adversary's proof transcript. The extracted witness includes the polynomial and commitment randomness. Again, this gives a strategy for breaking the hiding property of this scheme.

5 Case Studies: Polynomial Interactive Oracle Proofs

Marlin. We present a slightly modified PIOP for Marlin in Protocol 2. Notations follow [CHM+20]. The b denotes the query bound which determines the degree of masking polynomials for $\hat{w}, \hat{z}_A, \hat{z}_B, \hat{z}_C$. Here, the bound is incremented by Ψ in order to tolerate one additional query at χ.

Note that the original Marlin protocol does not have high min-entropy of $\mathbf{Q_V}$ in terms of 2.7, because an evaluation query to all the first-round polynomials is fixed by the challenge β_1.[10] This is why we introduce a dummy polynomial $r(X)$ of degree $\Psi+1$ and have the verifier query r with the last-round challenge β_3. Clearly, this variant of $\mathsf{PIOP}_{\mathsf{Marlin}}$ has $\log(|\mathbb{F}|)$-bit min-entropy for $\mathbf{Q_V}$. We then show that $\mathsf{PIOP}_{\mathsf{Marlin}}$ retains Ψ-HVZK and Ψ-NEXP. We also highlight an evaluation proof for a dummy polynomial as a generic method that can generically add the weak unique response property to any Fiat-Shamir NIZKAoK.

Theorem 5.1. $\mathsf{PIOP}_{\mathsf{Marlin}}$ *with* $\mathsf{b} = \Psi + 2$ *is perfectly* Ψ*-HVZK and* Ψ*-NEXP.*

Proof. We give a Ψ-HVZK simulator $\mathcal{S}_{\mathsf{Marlin}}$ for Marlin in Simulator 3. First consider the public polynomials $g_2, \ldots, h_3$. These are computed honestly both in the real and the simulated world, and so their evaluations are identically distributed. In the real world, up to b

[10] This also implies that the unique response analysis for Marlin presented in [GKK+22] does not hold. We thank the authors of [FFK+23] for bringing the issue to our attention.

Protocol 2: Protocol $\mathsf{PIOP}_{\mathsf{Marlin}}$

Modifications with respect to the original protocol of [CHM+20] are written in orange.

Offline phase. The indexer $\mathbf{I}$ is given as input subsets H, K of $\mathbb{F}$, and matrices $A, B, C \in \mathbb{F}^{n \times n}$ representing the R1CS instance, and outputs three univariate polynomial oracles $\{\hat{\mathsf{row}}_M, \hat{\mathsf{col}}_M, \hat{\mathsf{val}}_M\}$ of degree less than $|K|$ for each matrix $M \in A, B, C$, such that the following polynomial is a low-degree extension of M.

$$\hat{M}(X, Y) := \sum_{k \in K} u_H(X, \hat{\mathsf{row}}_M(k)) u_H(Y, \hat{\mathsf{col}}_M(k)) \hat{\mathsf{val}}_M(k)$$

Input. $\mathbf{P}$ receives $(H, K, A, B, C, \mathsf{i}, x, w)$, and $\mathbf{V}$ receives (H, K, x) and oracle access to the nine polynomials output by $\mathbf{I}(\mathsf{i})$.

Online phase: first round. $\mathbf{P}$ sends the masked oracle polynomials $\hat{w}(X) \in \mathbb{F}^{<|w|+\mathsf{b}}[X]$, $\hat{z}_A(X)$, $\hat{z}_B(X), \hat{z}_C(X) \in \mathbb{F}^{<|H|+\mathsf{b}}[X]$, $h_0(X) \in \mathbb{F}^{<|H|+2\mathsf{b}-1}[X]$ as defined in [CHM+20, 5.3.2] and uniformly random $r(X) \in \mathbb{F}^{\Psi+1}[X]$. It samples a random $s(X) \in \mathbb{F}^{<2|H|+\mathsf{b}-1}[X]$ and sends polynomial oracle $s(X)$ together with $\sigma_1 \in \mathbb{F}$ where $\sigma_1 := \sum_{a \in H} s(a)$, and $\hat{z}_A(X)\hat{z}_B(X) - \hat{z}_C(X) = h_0(X)v_H(X)$.

Online phase: second round. Upon receiving challenges $\alpha, \eta_A, \eta_B, \eta_C \in \mathbb{F}$ from $\mathbf{V}$, $\mathbf{P}$ sends oracle polynomials $g_1(X) \in \mathbb{F}^{<|H|-1}[X]$, $h_1(X) \in \mathbb{F}^{<|H|+\mathsf{b}-1}[X]$ to $\mathbf{V}$, where

$$\begin{aligned} &s(X) + u_H(\alpha, X)\Big(\sum_{M \in \{A,B,C\}} \eta_M \hat{z}_M(X)\Big) - \Big(\sum_{M \in \{A,B,C\}} \eta_M r_M(\alpha, X)\Big)\hat{z}(X) \\ =&h_1(X)v_H(X) + Xg_1(X) + \sigma_1/|H| \end{aligned}$$

Online phase: third round. Upon receiving challenge $\beta_1 \in \mathbb{F} \setminus H$ from the $\mathbf{V}$, $\mathbf{P}$ sends oracle polynomials $g_2(X), h_2(X) \in \mathbb{F}^{<|H|-1}[X]$ and $\sigma_2 \in \mathbb{F}$ to $\mathbf{V}$, where

$$\begin{aligned} \sigma_2 &= \sum_{k \in H} u_H(\alpha, k) \sum_{M \in \{A,B,C\}} \eta_M \hat{M}(k, \beta_1) \\ u_H(\alpha, X) &\sum_{M \in \{A,B,C\}} \eta_M \hat{M}(X, \beta_1) = h_2(X)v_H(X) + Xg_2(X) + \sigma_2/|H| \end{aligned}$$

Online phase: fourth round. Upon receiving challenge $\beta_2 \in \mathbb{F} \setminus H$ from the $\mathbf{V}$, $\mathbf{P}$ sends oracle polynomials $g_3(X) \in \mathbb{F}^{<|K|-1}[X]$, $h_3(X) \in \mathbb{F}^{<6|K|-6}[X]$ and $\sigma_3 \in \mathbb{F}$ to $\mathbf{V}$, where,

$$\begin{aligned} \sigma_3 &= \sum_{k \in K} \sum_{M \in \{A,B,C\}} \eta_M \frac{v_H(\beta_2)v_H(\beta_1)\hat{\mathsf{val}}_M(k)}{(\beta_2 - \hat{\mathsf{row}}_M(k))(\beta_1 - \hat{\mathsf{col}}_M(k))} \\ h_3(X)v_K(X) &= a(X) - b(X)(Xg_3(X) + \sigma_3/|K|) \\ a(X) &= \sum_{M \in \{A,B,C\}} \eta_M v_H(\beta_2)v_H(\beta_1)\hat{\mathsf{val}}_M(X) \prod_{L \in \{A,B,C\} \setminus \{M\}} (\beta_2 - \hat{\mathsf{row}}_L(X))(\beta_1 - \hat{\mathsf{col}}_L(X)) \\ b(X) &= \prod_{M \in \{A,B,C\}} (\beta_2 - \hat{\mathsf{row}}_M(X))(\beta_1 - \hat{\mathsf{col}}_M(X)) \end{aligned}$$

Query phase. $\mathbf{V}$ queries the oracles $\hat{w}(X), \hat{z}_A(X), \hat{z}_B(X), \hat{z}_C(X), h_0(X), s(X), h_1(X), g_1(X)$ at β_1; $h_2(X), g_2(X)$ at β_2; $h_3(X), g_3(X), r(X)$ and all offline oracles $\{\hat{\mathsf{row}}_M, \hat{\mathsf{col}}_M, \hat{\mathsf{val}}_M\}$ for each $M \in A, B, C$ at a random query point $\beta_3 \in \mathbb{F}$.

Decision phase. $\mathbf{V}$ accepts if the following tests pass:

- $h_3(\beta_3)v_K(\beta_3) = a(\beta_3) - b(\beta_3)(\beta_3 g_3(\beta_3) + \sigma_3/|K|)$
- $h_2(\beta_2)v_H(\beta_2) + \beta_2 g_2(\beta_2) + \sigma_2/|H| = u_H(\alpha, \beta_2)\sigma_3$
- $s(\beta_1) + u_H(\alpha, \beta_1)(\sum_M \eta_M \hat{z}_M(\beta_1)) - \sigma_2 \hat{z}(\beta_1) = h_1(\beta_1)v_H(\beta_1) + \beta_1 g_1(\beta_1) + \sigma_1/|H|$
- $\hat{z}_A(\beta_1)\hat{z}_B(\beta_1) - \hat{z}_C(\beta_1) = h_0(\beta_1)v_H(\beta_1)$

Simulator 3: $\mathcal{S}_{\mathsf{Marlin}}$

On input (i, x) do:

Offline Phase: Compute polynomials $(\hat{\mathsf{row}}_A, \ldots, \hat{\mathsf{val}}_C)$ honestly. Set $\mathbf{p} := \{\hat{\mathsf{row}}_A, \ldots, \hat{\mathsf{val}}_C\}$.

Online Phase:

1. Sample polynomials $\hat{z}_A, \hat{z}_B, \hat{w}, h_0, r$ uniformly at random from the appropriate domain.
2. Set $\hat{z}_C(X) := \hat{z}_A(X)\hat{z}_B(X) - h_0(X)v_H(X)$.
3. Sample challenges $\alpha, \eta_A, \eta_B, \eta_C \leftarrow \mathbb{F}$.
4. Sample polynomials h_1, g_1 uniformly at random from the appropriate domain, and $\sigma_1 \leftarrow \mathbb{F}$.
5. Derive $\hat{z}$ from $\hat{w}$ and x, and set $s(X) := h_1(X)v_H(X) + Xg_1(X) + \sigma_1/|H| - u_H(\alpha, X)(\sum_{M\in\{A,B,C\}} \eta_M \hat{z}_M(X)) + (\sum_{M\in\{A,B,C\}} \eta_M r_M(\alpha, X))\hat{z}(X)$.
6. Sample challenges $\beta_1, \beta_2 \leftarrow \mathbb{F} \setminus H$.
7. Compute third and fourth round messages $(g_2, h_2, g_3, h_3, \sigma_2, \sigma_3)$ honestly.
8. Set $\mathsf{view} := \{\sigma_1, \alpha, \eta_A, \eta_B, \eta_C, \beta_1, \sigma_2, \beta_2, \sigma_3\}$.
9. Update $\mathbf{p} := \mathbf{p}||\{r, \hat{w}, \hat{z}_A, \hat{z}_B, \hat{z}_C, h_0, s, g_1, h_1, g_2, h_2, g_3, h_3\}$.

Query Phase: Honestly evaluate polynomials $(\hat{z}_A, \hat{z}_B, \hat{z}_C, \hat{w}, h_0, s, h_1, g_1)$ at β_1, (h_2, g_2) at β_2, and (h_3, g_3, r), and all offline polynomials at $\beta_3 \leftarrow \mathbb{F}$. Update view with these evaluations.

Output: $(\mathsf{view}; \mathbf{p})$.

evaluations of polynomials $\hat{z}_A$, $\hat{z}_B$,$\hat{z}_C$ at points in $\mathbb{F} \setminus H$, are distributed uniformly random in $\mathbb{F}$. h_0 is set such that $\hat{z}_A(X)\hat{z}_B(X) - \hat{z}_C(X) = h_0(X)v_H(X)$, for some evaluation point $\gamma \in \mathbb{F} \setminus H$, $h_0(\gamma) := (\hat{z}_A(\gamma)\hat{z}_B(\gamma) - \hat{z}_C(\gamma))/v_H(\gamma)$. Since, $\hat{z}_A(\gamma), \hat{z}_B(\gamma), \hat{z}_C(\gamma)$ are uniformly distributed in $\mathbb{F}$, and $v_H(\gamma)$ is a deterministic value given γ, $h_0(\gamma)$ is also uniformly random in $\mathbb{F}$. Since up to b evaluations of $\hat{z}_A$, $\hat{z}_B$,$\hat{z}_C$ are uniformly distributed in $\mathbb{F}$, h_0 evaluated on the same points is also uniformly distributed in the real world. In the ideal world, $\hat{z}_A, \hat{z}_B, h_0$ are sampled uniformly at random, and thus their evaluation (up to b) are uniformly random here too. Moreover, since $\hat{z}_C(X)$ is set to satisfy the same constraint, its evaluations are uniform in the ideal world.

Now, we argue that the evaluations of s, g_1, h_1 and the value σ_1 are also distributed identically. First notice that σ_1 is a random field element in the real world since $s(X)$ is a random polynomial. In the ideal world, it is picked at random (before picking $s(X)$). This however, still remains identical to the real world.

In the real world, $s(X)$ is sampled at random and then g_1, h_1 are set such that $s(X) + p(X) = h_1(X)v_H(X) + Xg_1(X) + \sigma_1/|H|$, where $p(X) := u_H(\alpha, X)\Big(\sum_{M\in\{A,B,C\}} \eta_M \hat{z}_M(X)\Big) - \Big(\sum_{M\in\{A,B,C\}} \eta_M r_M(\alpha, X)\Big)\hat{z}(X)$. Polynomial $s(X)$ can be rewritten as $h_s(X)v_H(X) + Xg_s(X) + \sigma_1/|H|$. We can conclude that the constant terms is $\sigma/|H|$ because of the guarantees from sub-check protocol (i.e., the sum over H is σ_1 iff the constant term is $\sigma_1/|H|$. Because $h_s \in \mathbb{F}^{<|H|+\mathsf{b}-1}$ and $g_s \in \mathbb{F}^{<|H|-1}$, both h_s and g_s have to be uniform polynomials as well.

Moreover, σ_1 will be a random field element. Similarly, $p(X)$ can be written as $h_p(X)v_H(X)+Xg_p(X)+c_p$, for some h_p, g_p and a constant c_p. This implies, $h_s(X)+h_p(X)=h_1(X)$, $g_s(X)+g_p(X)=g_1(X)$, and $c_p=0$. This means that, in the real world, up to b evaluations of $h_1(X)$ and $g_1(X)$ look random because of h_s and g_s. Since in the ideal world, these polynomials are picked completely at random, their evaluations are going to be identically distributed.

Finally, we argue for evaluations of $s(X)$. Observe that, both in the real and ideal world, $s(X)$ is determined once all other polynomials are fixed. Hence, its evaluation is fixed after fixing all other polynomials. As long as $s(X)$ is evaluated at the same points that g_1, h_1 have already been evaluated at, its evaluations are going to be identically distributed.

Proving Ψ-NEXP is straightforward: since the dummy polynomial r is of degree $\Psi+1$, a Ψ-NEXP adversary cannot guess $r(z^*)$ for $z^* \notin (\chi, \beta_3)$ except with probability $1/|\mathbb{F}|$.

Remark 5.2. Lunar PIOP for RICS-lite also follows a very similar structure as Marlin. We observe that the same simulation strategy as for Marlin will also work for this PIOP. Since the simulation is almost the same, we skip the details here.

PLONK. We recall PIOP for PLONK [GWC19] in Protocol 3. Unlike Marlin, $\mathsf{PIOP}_{\mathsf{PLONK}}$ already has high min-entropy of $\mathbf{Q_V}$ since the evaluation point z is only sampled at the very end. It also satisfies the condition that $\mathsf{t}(1,1) \leq \mathsf{d}(|\mathsf{i}|)$ since the witness carrying polynomial $\hat{f}_L$ is only queried at one point. [GKK+22, Lemma 7] claims that $\mathsf{PIOP}_{\mathsf{PLONK}}$ compiled with deterministic KZG satisfies perfect trapdoor-less ZK. We first explain why the simulator provided there is not perfect. Essentially, the simulator of [GKK+22, Lemma 7] works as follows:

1. Sample uniform challenge $\beta, \gamma, \alpha \in \mathbb{F}$ and $z \in \mathbb{D}$.
2. Let $\mathsf{w} = \mathbf{0}$ be a dummy witness. Compute and commit to all polynomial oracles until the second round as an honest prover would do.
3. Compute $T(X)$ of the third round as an honest prover would do, except that $T(X)$ is now a rational function instead of polynomial, because a dummy witness may not satisfy the constraint. Define a polynomial $\tilde{T}(X)$ such that $\tilde{T}(z) = T(z)$ and create a KZG commitment $c_T = g^{\tilde{T}(\chi)}$.
4. Using the simulated polynomial $\tilde{T}(X)$ and other polynomials, compute KZG evaluation proofs with an evaluation point z as input.

Although this is trapdoor-less simulation, we observe that perfect simulation is achieved due to the third step. In the real protocol, $T(X)$ is computed and committed honestly and the equation (5) is satisfied even if evaluated at the KZG trapdoor χ. However, the above simulator commits to $\tilde{T}(X)$ via deterministic KZG, which does not equal the righthand side of (5) when evaluated at χ. Therefore, an unbounded adversary that can compute exponents of KZG commitments can trivially distinguish simulated from real transcripts by checking whether these exponents satisfy the equation or not. Note that one might be able to justify computational ZK of the above strategy by assuming a variant of Uber assumption [Boy08], which is an interesting future direction.

Instead, we provide a more straightforward simulation strategy for $\mathsf{PIOP}_{\mathsf{PLONK}}$ leading to perfect TLZK if compiled via rKZG.

Theorem 5.3. *PLONK PIOP is perfectly HVZK.*

Proof. We give an HVZK simulator $\mathcal{S}_{\mathsf{PLONK}}$ in Simulator 4. The only difference with an honest prover is that, $\mathcal{S}_{\mathsf{PLONK}}$ encodes a dummy witness in polynomials f_L, f_R, f_O. In a real protocol, since $\hat{f}_L = f_L(X) + r_L(X) \cdot Z_H(X)$ where $r_L(X)$ is a degree-1 uniformly random

Protocol 3: $\mathsf{PIOP}_{\mathsf{PLONK}}$

Offline phase. The indexer $\mathbf{I}$ receives as input $\mathsf{i} = (\mathbb{F}, n, m, l, \mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O, \mathbf{q}_M, \mathbf{q}_C, \sigma, \mathcal{T}_\mathcal{C})$, and computes the following polynomial oracles as described in the text: selector polynomials $(q_L, q_R, q_O, q_M, q_C)$; preprocessed polynomials for permutation argument $(S_{L,\mathsf{ID}}, S_{R,\mathsf{ID}}, S_{O,\mathsf{ID}}, S_{L,\sigma}, S_{R,\sigma}, S_{O,\sigma})$; vanishing polynomial of H, $Z_H(X) = X^n - 1$.

Input. $\mathbf{P}$ receives $(\mathsf{i}, (\mathsf{w}_i)_{i\in[l]}, (\mathsf{w}_i)_{i\in[l+1,3n]})$ and $\mathbf{V}$ receives $\mathsf{x} = (\mathsf{w}_i)_{i\in[l]}$ and oracle access to the polynomials output by $\mathbf{I}(\mathsf{i})$.

Online phase: first round. $\mathbf{P}$ computes $f_{\mathsf{pub}}(X) = \sum_{i\in[l]} \mathsf{w}_i \cdot L_i(X)$, and witness-carrying polynomials $f_L(X) = \sum_{i\in[n]} \mathsf{w}_i \cdot L_i(X)$, $f_R(X) = \sum_{i\in[n]} \mathsf{w}_{i+n} \cdot L_i(X)$, $f_O(X) = \sum_{i\in[n]} \mathsf{w}_{i+2n} \cdot L_i(X)$. Define $\hat{f}_L = f_L + (a_0 + a_1 X) Z_H(X)$, $\hat{f}_R = f_R + (b_0 + b_1 X) Z_H(X)$, $\hat{f}_O = f_O + (c_0 + c_1 X) Z_H(X)$, where $a_i, b_i, c_i \xleftarrow{\$} \mathbb{F}$. Send $(\hat{f}_L(X), \hat{f}_R(X), \hat{f}_O(X))$ to $\mathbf{V}$.

Online phase: second round. Upon receiving challenges $\beta, \gamma \in \mathbb{F}$ from the $\mathbf{V}$, $\mathbf{P}$ computes $h_{\mathsf{ID}}(X)$, $h_\sigma(X)$ and a permutation polynomial $s(X)$ as described in [GWC19]. Then $\mathbf{P}$ sends a masked oracle polynomial $\hat{s}(X) = s + (d_0 + d_1 X + d_2 X^2) Z_H(X)$ to $\mathbf{V}$, where $d_i \xleftarrow{\$} \mathbb{F}$.

Online phase: third round. Upon receiving challenge $\alpha \in \mathbb{F}$ from the $\mathbf{V}$, $\mathbf{P}$ computes

$$F_\mathcal{C}(X) = q_L(X)\hat{f}_L(X) + q_R(X)\hat{f}_R(X) + q_O(X)\hat{f}_O(X) \tag{1}$$

$$+ q_M(X)\hat{f}_L(X)f_R(X) + q_C(X) + f_{\mathsf{pub}}(X) \tag{2}$$

$$F_1(X) = h_{\mathsf{ID}}(X)\hat{s}(X) - h_\sigma(X)\hat{s}(\zeta X) \tag{3}$$

$$F_2(X) = L_1(X)(\hat{s}(X) - 1) \tag{4}$$

$$T(X) = \frac{F_\mathcal{C}(X) + F_1(X)\cdot\alpha + F_2(X)\cdot\alpha^2}{Z_H(X)} \tag{5}$$

and sends an oracle polynomial $T(X)$ to $\mathbf{V}$.

Query phase. $\mathbf{V}$ queries online oracles $(\hat{f}_L(X), \hat{f}_R(X), \hat{f}_O(X), f_{\mathsf{pub}}(X), T(X))$ and all offline oracles with a random query point $z \in \mathbb{D}$. Moreover, it makes an additional query to the permutation polynomial $\hat{s}(X)$ with ζz.

Decision phase. $\mathbf{V}$ first computes $f_{\mathsf{pub}}(X)$ from the statement. Then $\mathbf{V}$ constructs $F_\mathcal{C}(z)$, $F_1(z)$ and $F_2(z)$ based on the outputs of polynomial oracles. It then checks that $(F_\mathcal{C}(z) + F_1(z)\cdot\alpha + F_2(z)\cdot\alpha^2) = T(z)\cdot Z_H(z)$.

Simulator 4: $\mathcal{S}_{\mathsf{PLONK}}$

On input (i, x) do:

Offline Phase: Compute preprocessed polynomials as in the real $\mathbf{I}$.

Online & Query Phase: Let $\mathsf{w} = \mathbf{0}$ be a dummy witness.

1. Construct witness-carrying polynomials f_L, f_R, f_O encoding dummy witness w. Construct masked polynomials $\hat{f}_L, \hat{f}_R, \hat{f}_O$ as an honest prover would.
2. Uniformly sample permutation argument challenges $\beta, \gamma \in \mathbb{F}$, and construct polynomials $h_{\mathsf{ID}}, h_\sigma, \hat{s}$ as an honest prover would.
3. Uniformly sample amortization challenge $\alpha \in \mathbb{F}$ and define

$$\begin{aligned} F_{\mathcal{C}} &= q_L \cdot \hat{f}_L + q_R \cdot \hat{f}_R + q_O \cdot \hat{f}_O + q_M \cdot f_L \cdot \hat{f}_R + q_C + f_{\mathsf{pub}} \\ F_1 &= h_{\mathsf{ID}} \cdot \hat{s} - h_\sigma \cdot \hat{s}(\zeta X) \\ F_2 &= L_1 \cdot (\hat{s} - 1) \\ T(X) &= \frac{F_{\mathcal{C}}(X) + \alpha \cdot F_1(X) + \alpha^2 \cdot F_2(X)}{Z_H(X)} \end{aligned}$$

   where $T(X)$ is a rational function since the dummy witness may not satisfy the relation.
4. Uniformly sample $z \in \mathbb{D}$ and define $y_L = \hat{f}_L(z), y_R = \hat{f}_R(z), y_O = \hat{f}_O(z), y_s = \hat{s}(z)$, $y'_s = \hat{s}(\zeta z), y_T = T(z)$.

Output: $\mathsf{view} = ((\beta, \gamma), \alpha, z, (y_L, y_R, y_O, y_s, y'_s, y_T))$.

polynomial, a single evaluation of $\hat{f}_L$ at $z \in \mathbb{D}$ is independent of f_L. The same argument applies to f_R and f_O, and thus we have that the joint distribution of simulated $(z, (y_L, y_R, y_O))$ is identical to that of a real protocol. Analogously, since the permutation polynomial is $\hat{s} = s(X) + r_s(X) \cdot Z_H(X)$ where $r_s(X)$ is a degree-2 uniformly random polynomial, two evaluations of $\hat{s}$ at $z, z\zeta \in \mathbb{D}$ are independent of s. Finally, once $((\beta, \gamma), \alpha, (y_L, y_R, y_O, y_s, y'_s))$ are fixed, a valid y_T is uniquely determined. This concludes that the distribution of simulated $((\beta, \gamma), \alpha, (y_L, y_R, y_O, y_s, y'_s, y_T))$ is identical to that of real transcript.

Acknowledgment

We thank Matteo Campanelli, Antonio Faonio, Dario Fiore, Luigi Russo and Michal Zajac for helpful comments and discussions.

References

AABN02. M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In *EUROCRYPT 2002*, 2002.

AC20. T. Attema and R. Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In *CRYPTO 2020, Part III*, 2020.

AFK21. T. Attema, S. Fehr, and M. Klooß. Fiat-shamir transformation of multi-round interactive proofs. Cryptology ePrint Archive, Report 2021/1377, 2021. https://eprint.iacr.org/2021/1377.

AGRS23. B. Abdolmaleki, N. Glaeser, S. Ramacher, and D. Slamanig. Universally composable nizks: Circuit-succinct, non-malleable and crs-updatable. Cryptology ePrint Archive, Paper 2023/097, 2023. https://eprint.iacr.org/2023/097.

ARS20. B. Abdolmaleki, S. Ramacher, and D. Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In *ACM CCS 2020*, 2020.

BAZB20. B. Bünz, S. Agrawal, M. Zamani, and D. Boneh. Zether: Towards privacy in a smart contract world. In *FC 2020*, 2020.

BB04. D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *EUROCRYPT 2004*, 2004.

BBB+18. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, 2018.

BBHR19. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable zero knowledge with no trusted setup. In *CRYPTO 2019, Part III*, 2019.

BCG+14. E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, 2014.

BCGW22. D. Bitan, R. Canetti, S. Goldwasser, and R. Wexler. Using zero-knowledge to reconcile law enforcement secrecy and fair trial rights in criminal cases. In *Proceedings of the 2022 Symposium on Computer Science and Law, CSLAW 2022, Washington DC, USA, November 1-2, 2022*, pp. 9–22. ACM, 2022.

BCS16. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *TCC 2016-B, Part II*, 2016.

BFS20. B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. In *EUROCRYPT 2020, Part I*, 2020.

BG18. J. Bootle and J. Groth. Efficient batch zero-knowledge arguments for low degree polynomials. In *PKC 2018, Part II*, 2018.

BKSV21. K. Baghery, M. Kohlweiss, J. Siim, and M. Volkhov. Another look at extraction and randomization of groth's zk-snark. In *FC 2021*, vol. 12674 of *Lecture Notes in Computer Science*, pp. 457–475. Springer, 2021.

BMRS20. J. Bonneau, I. Meckler, V. Rao, and E. Shapiro. Coda: Decentralized cryptocurrency at scale. Cryptology ePrint Archive, Report 2020/352, 2020. https://eprint.iacr.org/2020/352.

Boy08. X. Boyen. The uber-assumption family. In *Pairing-Based Cryptography - Pairing 2008, Second International Conference, Egham, UK, September 1-3, 2008. Proceedings*, vol. 5209 of *Lecture Notes in Computer Science*, pp. 39–56. Springer, 2008.

BS21. K. Baghery and M. Sedaghat. Tiramisu: Black-box simulation extractable nizks in the updatable CRS model. In *CANS 2021*, vol. 13099 of *Lecture Notes in Computer Science*, pp. 531–551. Springer, 2021.

Can01. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, 2001.

CFF+21. M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In *ASIACRYPT 2021, Part III*, 2021.

CHM+20. A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *EUROCRYPT 2020, Part I*, 2020.

CL06. M. Chase and A. Lysyanskaya. On signatures of knowledge. In *CRYPTO 2006*, 2006.

DDO+01. A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *CRYPTO 2001*, 2001.

DG23. Q. Dao and P. Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). In *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II*, vol. 14005 of *Lecture Notes in Computer Science*, pp. 531–562. Springer, 2023.

FFG+16. D. Fiore, C. Fournet, E. Ghosh, M. Kohlweiss, O. Ohrimenko, and B. Parno. Hash first, argue later: Adaptive verifiable computations on outsourced data. In *ACM CCS 2016*, 2016.

FFK+23. A. Faonio, D. Fiore, M. Kohlweiss, L. Russo, and M. Zajac. From polynomial iop and commitments to non-malleable zksnarks. Cryptology ePrint Archive, Paper 2023/569, 2023. https://eprint.iacr.org/2023/569.

FKMV12. S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi. On the non-malleability of the Fiat-Shamir transform. In *INDOCRYPT 2012*, 2012.

FS87. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, 1987.

GGM14. C. Garman, M. Green, and I. Miers. Decentralized anonymous credentials. In *NDSS 2014*, 2014.

GKK+22. C. Ganesh, H. Khoshakhlagh, M. Kohlweiss, A. Nitulescu, and M. Zajac. What makes fiat-shamir zksnarks (updatable SRS) simulation extractable? In *SCN 2022*, vol. 13409 of *Lecture Notes in Computer Science*, pp. 735–760. Springer, 2022.

GKO+23. C. Ganesh, Y. Kondi, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Witness-succinct universally-composable snarks. In *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II*, vol. 14005 of *Lecture Notes in Computer Science*, pp. 315–346. Springer, 2023.

GM17. J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In *CRYPTO 2017, Part II*, 2017.

GMY06. J. A. Garay, P. D. MacKenzie, and K. Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, (2), 2006.

GOP+22. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In *EUROCRYPT 2022, Part II*, 2022.

GOP+23. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat-shamir bulletproofs are non-malleable (in the random oracle model). Cryptology ePrint Archive, Paper 2023/147, 2023. https://eprint.iacr.org/2023/147.

GWC19. A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

JP14. A. Jain and O. Pandey. Non-malleable zero knowledge: Black-box constructions and definitional relationships. In *SCN 14*, 2014.

KMS+16. A. E. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, 2016.

KZG10. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT 2010*, 2010.

KZM+15. A. Kosba, Z. Zhao, A. Miller, Y. Qian, H. Chan, C. Papamanthou, R. Pass, a. shelat, and E. Shi. C∅c∅: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093, 2015. https://eprint.iacr.org/2015/1093.

LR22a. A. Lysyanskaya and L. N. Rosenbloom. Efficient and universally composable non-interactive zero-knowledge proofs of knowledge with security against adaptive corruptions. Cryptology ePrint Archive, Paper 2022/1484, 2022. https://eprint.iacr.org/2022/1484.

LR22b. A. Lysyanskaya and L. N. Rosenbloom. Universally composable sigma-protocols in the global random-oracle model. Cryptology ePrint Archive, Report 2022/290, 2022. https://eprint.iacr.org/2022/290.

MBKM19. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In *ACM CCS 2019*, 2019.

NT16. A. Naveh and E. Tromer. PhotoProof: Cryptographic image authentication for any set of permissible transformations. In *2016 IEEE Symposium on Security and Privacy*, 2016.

PR05. R. Pass and A. Rosen. New and improved constructions of non-malleable cryptographic protocols. In *37th ACM STOC*, 2005.

Sah99. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, 1999.

Sta21. StarkWare. ethSTARK documentation. Cryptology ePrint Archive, Report 2021/582, 2021. https://eprint.iacr.org/2021/582.