WARWICK
THE UNIVERSITY OF WARWICK

**Manuscript version: Author's Accepted Manuscript**
The version presented in WRAP is the author's accepted manuscript and may differ from the published version or Version of Record.

**Persistent WRAP URL:**
http://wrap.warwick.ac.uk/176570

**How to cite:**
Please refer to published version for the most recent bibliographic citation information. If a published version is known of, the repository item page linked to above, will contain details on accessing it.

**Copyright and reuse:**
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

**Publisher's statement:**
Please refer to the repository item page, publisher's statement section, for further information.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk.

**warwick.ac.uk/lib-publications**

# Proof-Carrying Data From Arithmetized Random Oracles

Megan Chen
megchen@bu.edu
Boston University

Alessandro Chiesa
alessandro.chiesa@epfl.ch
EPFL

Tom Gur
Tom.Gur@warwick.ac.uk
University of Warwick

Jack O'Connor
Jack.O-Connor@warwick.ac.uk
University of Warwick

Nicholas Spooner
nicholas.spooner@warwick.ac.uk
University of Warwick

April 24, 2023

**Abstract**

Proof-carrying data (PCD) is a powerful cryptographic primitive that allows mutually distrustful parties to perform distributed computation in an efficiently verifiable manner. Known constructions of PCD are obtained by recursively-composing SNARKs or related primitives. SNARKs with desirable properties such as transparent setup are constructed in the random oracle model. However, using such SNARKs to construct PCD requires heuristically instantiating the oracle and using it in a non-black-box way. [CCS22] constructed SNARKs in the low-degree random oracle model, circumventing this issue, but instantiating their model in the real world appears difficult.

In this paper, we introduce a new model: the *arithmetized random oracle model* (AROM). We provide a plausible standard-model (software-only) instantiation of the AROM, and we construct PCD in the AROM, given only a standard-model collision-resistant hash function. Furthermore, our PCD construction is for arbitrary-depth compliance predicates. We obtain our PCD construction by showing how to construct SNARKs in the AROM for computations that query the oracle, given an accumulation scheme for oracle queries in the AROM. We then construct such an accumulation scheme for the AROM.

We give an efficient "lazy sampling" algorithm (an *emulator*) for the ARO up to some error. Our emulator enables us to prove the security of cryptographic constructs in the AROM and that zkSNARKs in the ROM also satisfy zero-knowledge in the AROM. The algorithm is non-trivial, and relies on results in algebraic query complexity and the combinatorial nullstellensatz.

**Keywords**: proof-carrying data; random oracle model; arithmetization

# Contents

# 1 Introduction

Proof-carrying data (PCD) [CT10] is a powerful cryptographic primitive that allows mutually distrustful parties to perform distributed computation in an efficiently verifiable manner. The notion of PCD generalizes incrementally-verifiable computation (IVC) [Val08] and has recently found exciting applications in enforcing language semantics [CTV13], verifiable MapReduce computations [CTV15], image authentication [NT16], verifiable registries [TFZBT22], blockchains [Mina; BMRS20; CCDW20; KB23], and more.

All known PCD constructions (and practical IVC constructions) are obtained via *recursive proof composition*, a general framework for building PCD from simpler primitives such as SNARKs [BCCT13; BCTV14; COS20] or accumulation schemes [BGH19; BCMS20; BDFG21; BCLMS21; KST22]. While the specific constructions differ, the high-level idea remains the same: to prove the correctness of $t$ steps of computation given proof of correctness for $t-1$ steps, one proves that "the $t$-th step is correct *and* there exists a valid proof for the first $t-1$ steps".

The statement that "there exists a valid proof" refers to the *verifier* of the underlying SNARK or accumulation scheme. As such, the resulting PCD scheme makes non-black-box use of the verifier for the underlying scheme. This leads to a significant theoretical problem when trying to prove security for constructions based on recursive composition: almost all known constructions of SNARKs, and all known constructions of accumulation schemes, are proven secure in the random oracle model (ROM). The random oracle is an inherently black-box object; in particular, it is believed that there is no "nontrivial" proof system for statements about the random oracle.

Most prior work in the area [COS20; BCMS20; BCLMS21] avoids this problem using a *heuristic step*: they assume that there exists some concrete hash function such that replacing the random oracle with the hash function yields a secure SNARK or accumulation scheme in the standard model (without oracles), and then apply recursive composition to this heuristic scheme.

Two prior works [CT10; CCS22] propose a different approach: endow the random oracle with some additional structure. The PCD construction in [CT10] is in a model where the random oracle additionally *signs* its responses using a standard-model signature scheme; the verifier can then check query-answer pairs by verifying the signature rather than querying the oracle. Trading cryptographic structure for algebraic structure, [CCS22] construct PCD in the *low-degree random oracle model* (LDROM), where parties have access to a random low-degree multivariate polynomial.

Both of these oracle models can be instantiated using hardware tokens. Unfortunately, we do not have any standard model (i.e., software-only) instantiation of these oracles, even heuristically. This is in contrast to the (usual) random oracle model, where empirical evidence suggests that "natural" schemes remain secure provided the oracle is replaced with a suitably "random-looking" hash function [BR93]. Our goal in this work is to design a new oracle model that simultaneously achieves both desiderata: (a) there exists a PCD scheme in this model under standard assumptions; and (b) the oracle can be heuristically instantiated.

## 1.1 Our results

In this work we introduce and study a new oracle model, the arithmetized random oracle model (AROM), which provides a random oracle and a corresponding "arithmetization" oracle. As in the standard ROM, the random oracle is an idealized model of some concrete hash function $H$. The arithmetization oracle is an idealized model of a certain *arithmetization* of $H$, which is a low-degree polynomial $P_H$ that can be efficiently computed from the circuit of $H$. As such, the AROM has a plausible heuristic instantiation: replace the random oracle by $H$ and the arithmetization oracle by $P_H$, for a suitable hash function $H$.

Our main result is a construction of PCD in the AROM, based on the [CCS22] construction of PCD in the LDROM. By instantiating the AROM with a suitable hash function, we obtain a candidate "real-world" construction of PCD. Formally, we prove the following theorem.

**Theorem 1** (informal)**.** *There exists transparent*[1] *(zero-knowledge) PCD in the AROM (for computations in the AROM), assuming the existence of collision-resistant hash functions in the standard model.*

Our PCD construction is provably secure (in the AROM) for *all* efficient compliance predicates. This stands in contrast to all other constructions of PCD (with the exception of [CT10], but including [CCS22]), whose security proofs are limited to constant-depth recursion. This is because, like [CT10], our PCD construction preserves the straightline extraction property of the underlying SNARK.[2]

To prove our main theorem, we develop various tools for analyzing cryptographic constructions in the AROM. Our key result here is to show that the additional power provided by the AROM does not help the adversary win any game defined with respect to the random oracle alone.

**Theorem 2** (informal)**.** *Any construction that is secure in the ROM is secure in the AROM.*

An immediate consequence of this theorem is that any construction that is secure in the standard model is secure in the AROM. In contrast, we do not know whether an analogous statement holds in the LDROM. We remark that this result is meaningful even outside the present context: it provides evidence that security in the ROM implies security against a specific type of non-black-box attack, namely, attacks that treat the *arithmetization* of the hash function as a black box.

**Comparison to other oracle models.** As discussed above, both the ROM and the LDROM fall short of our goal. While the ROM has a well-established heuristic instantiation, it is unlikely to support a PCD scheme. PCD exists in the LDROM, but we do not know how to instantiate the oracle. The AROM offers, in some sense, the "best of both worlds": a provable construction of PCD *and* a plausible heuristic instantiation. Moreover, the proposed instantiation of the AROM does not rely on any cryptography beyond "random-oracle-like" hash functions. As such, there are no barriers to implementing our scheme.

**Post-quantum security.** Our scheme does not rely on any pre-quantum assumption; it is plausibly post-quantum secure. Moreover, it is conceivable that the scheme is in fact *provably* post-quantum secure in the "quantum-accessible" AROM; we leave this intriguing question to future work.

## 1.2 Related work

**PCD and IVC in the ROM.** There is theoretical evidence that, unlike for SNARKs, there is no construction of PCD and IVC in the ROM (even allowing for additional "mild" cryptographic assumptions like standard-model CRHs). First, [CL20] shows that the PCP theorem does not hold for various cryptographically relevant oracle models, such as the ROM and the LDROM. This suggests that succinct proofs for computations relative to these oracles may be out of reach. Nevertheless, [CCS22] shows that this is not the whole story by constructing SNARKs for LDROM computations, particularly PCD, from a cryptographic assumption. Second, [HN23] shows various impossibilities for IVC in the ROM. For example, if a particular type of commitment scheme exists, then zero-knowledge IVC (without a CRS) does not exist in the ROM. This result holds even if the IVC construction were to rely on "standard" cryptographic assumptions.[3]

[1] The only setup required is a uniform reference string.

[2] Some other prior PCD constructions are also based on SNARKs with straightline extraction (e.g., [Val08; COS20]). However, this property is lost after the heuristic step is applied.

[3] The paper claims that this result holds for constructions that use *falsifiable* assumptions but does not show this explicitly. Nonetheless, one can check that the proof does work for "benign" cryptographic assumptions.

**Pseudorandom oracles.** [JLLW22] introduce the *pseudorandom oracle model* (PROM) and apply it towards obfuscation. Similarly to the AROM, the PROM aims to capture cryptographic schemes that make a non-black-box use of the random oracle. We outline the PROM and explain how it differs from the AROM.

The PROM is specified relative to a (standard model) pseudorandom function family $F_k$, and has two interfaces. The first accepts a key $k$ and outputs a random handle $h$ (and stores $(h, k)$). The second accepts a handle $h$ and an input $x$ and outputs $F_k(x)$, where $k$ is the key corresponding to $h$. By the security of the PRF, a party holding only $h$ cannot distinguish the latter interface from a random oracle. On the other hand, a party holding the key $k$ can use the circuit for $F_k$ in a non-black-box way. [JLLW22] constructs ideal obfuscation from functional encryption in the PROM.

The key difference between the AROM and the PROM is that the PROM "separates" non-black-box and black-box access to the oracle. Specifically, non-black-box access to the PROM is available only to parties that know $k$, whereas random oracle security holds only against parties that do not know $k$. *In the AROM, there is no such asymmetry: all parties have the same access to the oracle.* This is important in the context of recursive composition (which we study) since completeness requires that both the prover and the verifier have non-black-box access to the oracle. Still, soundness relies on the security of the random oracle against the prover. It is an exciting open question to understand whether, despite this apparent barrier, recursive composition is possible in the PROM.

**Augmented random oracles.** [Zha22] defines the *augmented* random oracle model to analyze the resilience of cryptographic transformations in the ROM against uninstantiability results. While ideas about modeling non-black-box access to the random oracle (and the abbreviation "AROM") are common to both the augmented ROM and the arithmetized ROM, the models are very different both technically and in their applications. We briefly summarize [Zha22] and then explain how our model differs.

Let $\mathsf{ro}$ denote the random oracle, and $\Pi$ denote some protocol. A cryptographic transformation $T$ usually comes with a guarantee like "if $\Pi$ is a secure X, then $T^{\mathsf{ro}}(\Pi)$ is a secure Y". An uninstantiability result for $T$ typically shows that there exists some $\Pi$ such that $T^H(\Pi)$ is insecure for every polynomial-size circuit $H$. Known uninstantiability results use some non-black-box technique to provide a "trapdoor" that can be used with respect to any $H$ but is useless for $\mathsf{ro}$. The augmented ROM captures this paradigm by requiring $T^{\mathsf{ro}}(\Pi)$ to be secure even if $\Pi$ has access to an oracle $M$ that provides some functionality permitted by non-black-box access to $H$, but with respect to $\mathsf{ro}$. [Zha22] shows that key uninstantiability results for transformations (e.g., Fiat–Shamir for arguments [GK03]) lead to insecure protocols in the augmented ROM.

The augmented ROM is a tool for proving a stronger form of security for random oracle transformations. In particular, no "honest" scheme ever accesses the oracle $M$; indeed, the oracle $M$ is chosen adversarially (and may be trivial). On the other hand, in the arithmetized ROM, honest parties use the non-black-box access provided by the arithmetization oracle, whose functionality is (mostly) fixed by the model itself.

# 2 Techniques

Recall that our goal in this work is to construct proof-carrying data (PCD). Our approach follows the widely-used template of *recursive proof composition*. However, our setting imposes several technical and conceptual challenges. We begin by outlining a vital issue in proving security for this type of construction, which our work seeks to address.

Recursive proof composition refers to a set of techniques that enable the construction of PCD (and IVC) from SNARKs or accumulation schemes. With few notable exceptions (e.g., [Gro16]), all constructions of SNARKs and accumulation schemes rely on the Fiat–Shamir heuristic, which converts an interactive public-coin argument system into a non-interactive argument via a cryptographic hash function $H$. For all of these SNARK constructions, it is unknown whether this heuristic can be realized from any concrete (i.e., falsifiable) cryptographic assumption; indeed, there is evidence that this may not be possible [GW11]. However, we can prove these schemes secure in the ROM, treating the hash function $H$ as a truly random function $\mathsf{ro}$ to which the adversary has black-box access.

This leads to a fundamental tension in proving security for the recursive composition of these protocols. On the one hand, to prove security for the protocol itself, we assume that the adversary treats the hash function $H$ as a black box. On the other hand, when recursively composing, the *honest* protocol treats $H$ in a non-black-box way: specifically, as a concrete polynomial-size circuit. The prior work [CL20; HN23] discussed in Section 1.2 suggests that non-black-box use of $H$ may be necessary to achieve PCD (and IVC).

## 2.1 Starting point: the low-degree random oracle model

The work of [CCS22] addresses the aforementioned tension by introducing a new oracle model called the *low-degree random oracle model* (LDROM). They then show how to construct PCD via recursive composition in the LDROM (i.e., using the oracle as a black box).

In the LDROM, all parties have oracle access to a uniformly random low-degree multivariate polynomial $\hat{\rho}\colon \mathbb{F}^m \to \mathbb{F}$. Restricting $\hat{\rho}$ to $\{0,1\}^m \subseteq \mathbb{F}^m$ recovers the usual random oracle, and [CCS22] show that relevant security properties of the random oracle continue to hold in the LDROM; in particular, Micali's SNARK [Mic00] is secure in the LDROM. Unlike the random oracle, the LDROM admits a *query accumulation scheme*: a verifier, with the help of an untrusted accumulation proof, can check the correctness of $n$ queries to $\hat{\rho}$ using only $O(1)$ queries to $\hat{\rho}$. [CCS22] construct such an accumulation scheme and use it to build PCD.

**Instantiating the LDROM.** [CCS22] observe that the LDROM can be instantiated using a hardware token that implements the structured PRF of [BGV11]. Of course, schemes involving hardware tokens have significant drawbacks; finding a plausible "software-only" instantiation would be much preferable. [CCS22] suggest a natural strategy: given a "random-oracle-like" hash function $H$, convert it into an arithmetic circuit gate-by-gate. Such a circuit does define a polynomial with which we could instantiate the LDROM. Unfortunately, as noted in [CCS22], for widely-used hash functions, the degree of this polynomial will be large (at least $2^{25}$). Since the complexity of the verifier in the query accumulation scheme is linear in the degree of the oracle, the resulting PCD scheme would be prohibitively expensive.

## 2.2 The arithmetized random oracle model

Given the above difficulty, a natural next step is to consider techniques for *reducing the degree* of the resulting arithmetic circuit. Since the degree of an arithmetic circuit grows exponentially in its depth, a natural approach is to try to reduce the depth of the circuit for $H$. This can be achieved via the well-known NP

reduction from circuit satisfiability to 3-SAT (a depth-two formula). The output of the reduction is a boolean formula $\Phi_H$ with the following property: there is an efficiently computable *witness function* $W_H$ such that

$$\Phi_H(x, y, z) = \begin{cases} 1 & \text{if } H(x) = y \text{ and } W_H(x) = z \\ 0 & \text{otherwise} \end{cases} .$$

Converting $\Phi_H$ into an arithmetic formula (gate-by-gate) yields a polynomial $P_H$ of total degree $O(|H|)$ that agrees with $\Phi_H$ on boolean inputs.

$P_H$ is *not* a low-degree extension of $H$ (rather of $\Phi_H$) and so this is not a candidate instantiation of the LDROM. As we note later, however, the low-degree structure of $P_H$ will nonetheless allow us to build a query accumulation scheme, inspired by that of [CCS22]. Moreover, the statement "$H(x) = y$" can be verified by querying $P_H$ only, given $z$ as a witness. It is therefore plausible that, following the template developed in the prior work, we can obtain a secure construction of PCD that makes only black-box use of $H$ and $P_H$.

Of course, given the current state of knowledge, we can only hope to prove that this PCD scheme is secure in some idealized model. In particular, we would like to model $H$ as a random oracle. It is then necessary to answer the question: *if $H$ is a random oracle, what should $P_H$ look like?* A central modeling contribution of our work is to propose an answer to this question.

**A new oracle model: the AROM.** We refer to our proposed oracle model as the *arithmetized random oracle model* (AROM). Before presenting the model, we discuss two key modeling challenges that arise. Both relate to the fact that the black-box behavior of $P_H$ depends in a non-black-box way on $H$.

- **Challenge #1: $W_H$ is circuit-dependent.** For a concrete circuit $H$ and input $x$, $W_H(x)$ is a vector representing the assignment to the internal wires of $H$ on input $x$. This of course depends on the size and structure of the circuit for $H$, which is no longer meaningful when $H$ is replaced by a random oracle. We handle this conservatively, by allowing $W_H$ to be adversarial. That is, we require that completeness, soundness, and zero-knowledge hold *regardless* of the choice of $W_H$, which we allow to depend on $x$ and the random oracle, and may even itself be randomized.

  There is, however, an important caveat. While we allow our $W_H$ to depend on the random oracle, we must restrict this dependency; otherwise, the adversary could use $W_H$ to learn information that it cannot otherwise obtain (e.g., $W_H$ could encode a collision in $H$). Similarly, if $W_H$ is computationally unbounded, the adversary could use it to break standard-model cryptography. As such, we restrict $W_H$ to have an efficient implementation (in particular, it can only make polynomially-many queries to $H$).

- **Challenge #2: $P_H$ is not the unique extension.** Even after we have fixed $W_H$ (and hence $\Phi_H$), $P_H$ has a huge number of remaining degrees of freedom. This is because it is of individual degree larger than 1, but its behavior is specified only on boolean inputs. This is a more challenging issue to resolve: letting $P_H$ be chosen adversarially from the set of extensions of $\Phi_H$ would make the adversary unrealistically powerful (see Remark 2.1). Instead, we model $P_H$ as a uniformly random polynomial of the appropriate degree whose restriction to the hypercube is $\Phi_H$. We propose that this captures the inability of the adversary to leverage the structure of $H$ (and hence $P_H$) in breaking security. We leave to future work the question of whether this modeling choice can be weakened (again see Remark 2.1).

We now give an informal definition of the AROM; for details see Section 4. In the AROM, all parties (honest and malicious) have access to three oracles $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}})$:

- a *random oracle* $\mathsf{ro}\colon \{0,1\}^m \to \{0,1\}^\lambda$ drawn uniformly at random;

- a *witness oracle* $\mathsf{wo}\colon \{0,1\}^m \to \{0,1\}^w$ that is an arbitrary PPT-computable function (see below);
- an *extended verification oracle* (arithmetization oracle) $\widehat{\mathsf{vo}}\colon \mathbb{F}^{m+\lambda+w} \to \mathbb{F}$ that is a random extension of individual degree $d \geq 2$ of the verification oracle $\mathsf{vo}\colon \{0,1\}^{m+\lambda+w} \to \{0,1\}$ defined as follows:

$$\mathsf{vo}(x,y,z) := \begin{cases} 1 & \text{if } \mathsf{ro}(x) = y \text{ and } \mathsf{wo}(x) = z \\ 0 & \text{otherwise} \end{cases}.$$

We discuss each oracle in turn.

- The random oracle ro models the hash function $H$, as in the standard ROM.
- The witness oracle wo models the witness function $W_H$. It is defined via a polynomial-size oracle circuit $B$ chosen arbitrarily before the oracle is sampled. On a query $x$, wo outputs $B^{\mathsf{ro}}(x, \mu_x)$ where $\mu_x$ is sampled uniformly at random (and is not resampled if $x$ is queried again). The inclusion of $\mu_x$ allows our definition to subsume, e.g., modeling $W_H$ as a random oracle. The efficiency requirement is necessary to allow for efficient simulation of wo (it prevents wo from being used to break standard-model cryptography).
- The verification function vo models the boolean formula $\Phi_H$. Indeed, the definition of vo is directly obtained from the definition of $\Phi_H$ by replacing $H$ with ro and $W_H$ with wo.
- The extended verification oracle $\widehat{\mathsf{vo}}$ models the polynomial $P_H$. The requirement that $d \geq 2$ arises from a technical concern: as noted in [JKRS09], access to the unique multilinear ($d = 1$) extension of a function can be surprisingly powerful. (E.g., an adversary with access to the multilinear extension of vo can efficiently invert ro, see Remark 2.1.) Requiring $d \geq 2$ avoids this issue and is sufficient for our security proofs. In any case, we want to match the degree of $\widehat{\mathsf{vo}}$ to that of $P_H$ for some concrete hash function $H$, and the degree of $P_H$ will be at least 2 in each variable.[4]

A construction that makes black-box use of $H, W_H, \Phi_H$ can be analyzed in the AROM as suggested by the above discussion: replace $H$ with ro, $W_H$ with wo, and $P_H$ with $\widehat{\mathsf{vo}}$ (with matching degree bound $d$).

In Section 2.3 we describe our construction of PCD in the AROM. This construction relies on a "lazy sampling" procedure for the AROM, a key technical contribution that we describe in Section 2.4.

**AROM vs. LDROM.** Superficially the AROM and LDROM seem quite similar; indeed, they both aim to capture some arithmetization of the random oracle. However, there are notable differences between the two models, even putting aside the differing instantiability considerations. We highlight a few such differences.

- The LDRO is a low-degree extension over a field $\mathbb{F}$ of a random function $\{0,1\}^m \to \mathbb{F}$. Hence the security of the LDRO as a random oracle depends on $|\mathbb{F}|$. The ARO decouples the choice of $\mathbb{F}$ from the random oracle: one may choose the codomain $\{0,1\}^\lambda$ of ro independently from the field $\mathbb{F}$ over which $\widehat{\mathsf{vo}}$ is defined. The security of ro (even in the presence of $\widehat{\mathsf{vo}}$) depends only on $\lambda$. That said, in *both* the LDROM and the AROM, the security of their respective query accumulation schemes depends on $|\mathbb{F}|$.
- The LDRO is a linear code random oracle; i.e., it is sampled at random from a linear space over $\mathbb{F}$. The ARO is also sampled uniformly from some set, but this set does not form a linear space. This means that tools developed in [CCS22] for analyzing linear code random oracles do not directly apply. That said, the ARO does have *some* linear structure: the oracle $\widehat{\mathsf{vo}}$ is sampled uniformly from the (affine) space of low-degree extensions of vo. This fact will be useful for emulating the AROM.

[4]The degree of a variable in $P_H$ is equal to the number of clauses in $\Phi_H$ in which it appears. Every wire appears in at least two clauses in $\Phi_H$: once as an output and once as an input.

- The LDRO has security properties (e.g. collision resistance, unconditional SNARKs) even when $d = 1$ (i.e., it is a random *multilinear* polynomial). The ARO is not even one-way when $d = 1$.

**Remark 2.1** (choice of extension)**.** We set $\widehat{\mathsf{vo}}$ to be a random extension of $\mathsf{vo}$ of individual degree $d \geq 2$. We explain why setting $\widehat{\mathsf{vo}}$ to be an arbitrary extension of $\mathsf{vo}$ would grant the adversary too much power.

First consider the case when $\widehat{\mathsf{vo}}$ is the unique multilinear extension of $\mathsf{vo}$ ($d = 1$). Given oracle access to a multilinear polynomial $P$ over a field $\mathbb{F}$ of characteristic different from 2, a single query to $P$ suffices to efficiently evaluate the sum $\sum_{x\in\{0,1\}^n} P(x)$ [JKRS09]. We can use this capability and the structure of $\mathsf{vo}$ to invert $\mathsf{ro}$: given a target image $y \in \{0,1\}^\lambda$, perform a binary search for a preimage of $y$ by evaluating the sum $\sum_{x_1,z} \widehat{\mathsf{vo}}((x_0, x_1), y, z)$ for different prefixes $x_0$.

Next consider the higher-degree case: $\widehat{\mathsf{vo}}$ is an *adversarially-chosen* extension of $\mathsf{vo}$ of degree $d \geq 2$. Given oracle access to a polynomial $P$ of individual degree $d$, a single query to $P$ suffices to efficiently evaluate the sum $\sum_{x\in H^n} P(x)$ where $H$ is a multiplicative subgroup of $\mathbb{F}$ with $|H| > d$ [CFS17, Lemma A.4]. Assume that $\mathbb{F}$ has such a subgroup $H$ of size $d + 1$, and fix two elements $a, b \in H$. Let $g\colon \mathbb{F} \to \mathbb{F}$ be the unique linear function with $g(a) = 0$ and $g(b) = 1$. Choose $\widehat{\mathsf{vo}}$ to be the polynomial of minimal individual degree satisfying: $\widehat{\mathsf{vo}}(x, y, z) = \mathsf{vo}(x, y, z)$ for $(x, y, z) \in \{0,1\}^n$, and $\widehat{\mathsf{vo}}(w) = 0$ for $w \in g(H)^n \setminus \{0,1\}^n$. Note that $\widehat{\mathsf{vo}}$, and hence also $\widehat{\mathsf{vo}} \circ g^n$, has individual degree at most $|H| - 1 = d$, and $\sum_{x\in H^n} \widehat{\mathsf{vo}}(g(x_1), \ldots, g(x_n)) = \sum_{x\in\{0,1\}^n} \mathsf{vo}(x_1, \ldots, x_n)$. We can then use binary search as in the multilinear case to invert $\mathsf{ro}$.

The above gives some justification for modeling $\widehat{\mathsf{vo}}$ as a *random* low-degree extension of $\mathsf{vo}$. Of course, there are many choices that lie in between adversarial and random. For example, one could set $\widehat{\mathsf{vo}}$ to be drawn from an adversarially-chosen distribution with "enough" entropy. It is not clear, however, whether such a choice would be substantially closer to "reality" than our choice.

## 2.3 Building PCD secure in the AROM

Prior work [CCS22] shows that to obtain PCD in an oracle model $\mathcal{O}$, it suffices to construct: (i) a SNARK for $\mathsf{NP}$ relative to $\mathcal{O}$; and (ii) an accumulation scheme for $\mathcal{O}$-queries relative to $\mathcal{O}$. Further, the resulting PCD scheme is zero-knowledge if the SNARK and accumulation scheme also satisfy zero-knowledge. The PCD construction in the LDROM in [CCS22] follows by establishing these results for the LDROM. Similarly, our construction of PCD will follow by establishing these results for the AROM.

**(i) SNARKs in the AROM.** [CCS22] prove that Micali's SNARK remains (information-theoretically) secure in the LDROM, via a rewinding argument. In the AROM, we show a much more general theorem.

**Theorem 3** (informal)**.** *Let* $\mathsf{p}$ *be a predicate that queries* $\mathsf{ro}$*, and let* $\mathcal{A}$ *be an algorithm querying* $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}})$ *that outputs* $x$ *satisfying* $\mathsf{p}^{\mathsf{ro}}$ *with probability* $\varepsilon$*. Then there is an algorithm* $\mathcal{B}$*, of similar efficiency to* $\mathcal{A}$*, that queries* $\mathsf{ro}$ *only and outputs* $x$ *satisfying* $\mathsf{p}^{\mathsf{ro}}$ *with probability* $\varepsilon - \mathrm{negl}(\lambda)$.

Theorem 3 follows directly from our emulator for $\widehat{\mathsf{vo}}$, which we discuss further in Section 2.4. It is *not* known whether a similar result holds for the LDROM.

As an illustrative example, we can use Theorem 3 to prove that the ARO is collision-resistant. By applying Theorem 3 to the predicate $\mathsf{p}^{\mathsf{ro}}$ that, given $(x, x') \in \{0,1\}^m \times \{0,1\}^m$, checks that $x \neq x'$ and $\mathsf{ro}(x) = \mathsf{ro}(x')$, we deduce that the ARO is collision-resistant from the fact that the RO is collision-resistant.

We use Theorem 3 to prove knowledge soundness and zero knowledge of Micali's SNARK in the AROM.

- **Knowledge soundness.** We use Theorem 3 to prove that Micali's SNARK is secure in the AROM, via a *straightline* extractor. Informally, since we can cast knowledge soundness of Micali's SNARK as an

oracle predicate $\mathsf{p}$, any adversary $\mathcal{A}$ that breaks that security property in the AROM can be transformed via Theorem 3 into an adversary $\mathcal{B}$ that breaks it in the ROM. We can then apply the straightline extractor for Micali's SNARK to $\mathcal{B}$. Since $\mathcal{B}$ invokes $\mathcal{A}$ in a straightline manner, the resulting AROM extractor is also straightline.

- **Zero-knowledge.** We prove that Micali's SNARK is zero knowledge in the AROM. Our zero knowledge simulator that *programs* the oracle; this is a commonality with the zero knowledge simulators for Micali's SNARK in both the ROM and in the LDROM (see [CCS22]). To program the oracle, the simulator relies on a slightly stronger version of our emulator, which emulates oracle queries conditioned on an input list of (real) oracle query-answer pairs. Our hybrid argument invokes Theorem 3 and the Micali SNARK's zero knowledge property in the ROM. Informally, we move between hybrids in the ROM vs. AROM using Theorem 3, setting the predicate $\mathsf{p}$ to be any distinguisher between hybrids.

**(ii) An accumulation scheme for ARO queries.** The accumulation scheme for LDRO queries in [CCS22] is obtained by applying the Fiat–Shamir transformation to the (interactive public-coin) query reduction protocol of [KR08]. We follow the same template in the case of the ARO. The first observation is that it suffices to accumulate queries to $\widehat{\mathsf{vo}}$ only, because a query to $\mathsf{ro}$ or $\mathsf{wo}$ can be verified via a query to $\widehat{\mathsf{vo}}$.[5]

The [KR08] query reduction protocol itself works for any low-degree polynomial: in particular, for $\widehat{\mathsf{vo}}$. As in [CCS22], the central challenge is showing soundness of the Fiat–Shamir transformation in this setting. Note that here we *cannot* appeal to our general theorem above because the verification predicate queries $\widehat{\mathsf{vo}}$.

The soundness of our accumulation scheme is captured by a *zero-finding game* (ZFG). First explicitly described by [BCMS20], the most basic form of a ZFG challenges the adversary to output a commitment $\mathsf{cm}$ (under a standard-model commitment scheme) to a low-degree polynomial $f \not\equiv 0$ such that $f(\mathsf{ro}(\mathsf{cm})) = 0$. Intuitively this is hard because $f$ is fixed by $\mathsf{cm}$ before $\mathsf{ro}(\mathsf{cm})$ is known, and so the probability that $f(\mathsf{ro}(\mathsf{cm})) = 0$ cannot be much larger than the probability that $f(\alpha) = 0$ for a random $\alpha \in \mathbb{F}$, which is negligible for large fields. [CCS22] shows that a more general version of the ZFG holds in the LDROM, where the ZFG polynomial may depend in a restricted way on the LDRO itself. That is, they show that it is hard to find a commitment $\mathsf{cm}$ to polynomials $f, g$ such that $f - \hat{\rho} \circ g \not\equiv 0$ but $(f - \hat{\rho} \circ g)(\hat{\rho}(\mathsf{cm})) = 0$.

The security of our construction depends on the hardness of a similar problem in the AROM, captured by the following lemma.

**Lemma 1** (informal)**.** *It is hard for any polynomial-size adversary with access to the ARO* $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}})$ *to find a commitment* $\mathsf{cm}$ *to a pair of low-degree polynomials* $f, g$ *such that* $f - \widehat{\mathsf{vo}} \circ g \not\equiv 0$ *but* $(f - \widehat{\mathsf{vo}} \circ g)(\mathsf{ro}(\mathsf{cm})) = 0$.

We prove Lemma 1 by adapting the proof of the ZFG in [CCS22]. The proof relies on a *forking lemma* in the LDROM, which in turn relies on the ability to efficiently simulate the oracle in order to sample a forking transcript. For the AROM, we will rely on the emulator described in Section 2.4. The proof proceeds as follows. Looking ahead, we note that the emulator answers queries to $\widehat{\mathsf{vo}}$ using some polynomial $P$. We show that the adversary cannot win the ZFG when $\widehat{\mathsf{vo}}$ is replaced by $P$. This argument uses the forking lemma with respect to the emulator, and follows [CCS22], with one difference: this approach does not require a bespoke forking lemma as in [CCS22], and can be carried out using a general forking lemma [BN06, Lemma 1]. This general forking lemma is designed for random oracle adversaries, however, as we have already replaced $\widehat{\mathsf{vo}}$ with the emulator we can "perfectly emulate" $P$ using the emulator. Further, we can perfectly emulate $\mathsf{wo}$ using the witness circuit $B$. This allows us to reduce the ZFG adversary to a random oracle adversary and thus apply the general forking lemma. Then, since the emulator is statistically indistinguishable from $\widehat{\mathsf{vo}}$, the adversary cannot win the original ZFG.

[5] Recall that $\mathsf{ro}(x) = y$ and $\mathsf{wo}(x) = z$ if and only if $\widehat{\mathsf{vo}}(x, y, z) = 1$.

Before we describe our emulator, we discuss an important feature of our PCD construction.

**Extraction and PCD depth.** Almost all constructions of PCD suffer from the "extractor blowup" problem. To obtain a PCD transcript of depth $d$, we apply the SNARK extractor to itself $d$ times. If the extractor corresponding to a size-$S$ adversary is of size $S^c$, then the final extractor size is $s^{c^d}$, where $s$ is the size of the original PCD adversary. As a result, one obtains meaningful security guarantees when $d$ is a constant.

There is a single construction that does not suffer from this issue: the construction of [CT10]. This is because their SNARK (in their signed random oracle model) is straightline (or "list") extractable. Micali's SNARK is also straightline extractable in the ROM [Val08]. Of course, after heuristically instantiating the oracle there is no longer any notion of "straightline". On the other hand, we can easily show that Micali's SNARK is straightline extractable in the AROM. (We do not know how to show this in the LDROM; [CCS22] instead gives a rewinding extractor for Micali's SNARK.)

As a result, our PCD construction is *secure for arbitrary recursion depth*.

## 2.4 Emulation of the ARO

As discussed in Section 2.3, we aim to construct PCD in the AROM by proving that cryptographic properties in the ROM, specifically knowledge soundness and zero-knowledge of the Micali SNARK, also hold in the AROM. To this end, we design an efficient algorithm $\mathcal{M}$ that answers queries in a way that is statistically indistinguishable from answers of the ARO. We refer to such an algorithm as an *emulator* $\mathcal{M}$ for the AROM.[6]

Recall that the ARO consists of a tuple of oracles $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}})$. Our emulator $\mathcal{M}$ achieves a special (stronger) type of emulation: given oracle access to some $\mathsf{ro}$ and $\mathsf{wo}$, $\mathcal{M}$ can efficiently emulate $\widehat{\mathsf{vo}}$ drawn from the ARO distribution *conditioned* on $(\mathsf{ro}, \mathsf{wo})$.[7] We use this type of emulation to prove Theorem 3.

**Lemma 2** (informal)**.** *There exists a probabilistic algorithm $\mathcal{M}$ such that for every security parameter $\lambda \in \mathbb{N}$, query bound $t \in \mathbb{N}$, and $t$-query adversary $\mathcal{A}$,*

$$\left| \Pr_{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda)} \left[ \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})} = 1 \right] - \Pr_{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda)} \left[ \mathcal{A}^{\mathcal{M}^{(\mathsf{ro},\mathsf{wo})}} = 1 \right] \right| \leq \frac{t}{2^\lambda} \ . \tag{1}$$

*Moreover, $\mathcal{M}$ is* **pass-through** *with respect to $(\mathsf{ro}, \mathsf{wo})$: it answers queries to those oracles by forwarding them to the corresponding "real" oracle (and recording the answers).*

We refer to the absolute difference in Equation 1 as the *emulation error*. An emulator is *perfect* if it has zero emulation error.

**Prior oracle emulators.** Recall that a random oracle is a function $\mathsf{ro}$ chosen uniformly from $(\{0,1\}^m \to \{0,1\}^\lambda)$. It has a well-known perfect (stateless) emulator $\mathcal{M}_{\mathsf{ro}}$ that "lazily" samples answers: given a list of query-answer pairs $\mathsf{tr} \in (\{0,1\}^m \times \{0,1\}^\lambda)^t$ and a new query $x \in \{0,1\}^m$, $\mathcal{M}_{\mathsf{ro}}$ generates a query answer $y \in \{0,1\}^\lambda$, depending on if there exists an entry $(x, y') \in \mathsf{tr}$. If so, $\mathcal{M}_{\mathsf{ro}}$ returns $y := y'$ and $\mathsf{tr}$. Otherwise, $\mathcal{M}_{\mathsf{ro}}$ uniformly samples $y \leftarrow \{0,1\}^\lambda$ and returns the sampled answer $y$ and the updated list $\mathsf{tr}' = \mathsf{tr} \cup \{(x, y)\}$. Note that the lists $\mathsf{tr}$ and $\mathsf{tr}'$ can be omitted from the input and output if $\mathcal{M}_{\mathsf{ro}}$ maintains the list of known query-answer pairs in its state.

The low-degree random oracle [CCS22] also has a perfect emulator, based on *succinct constraint detection* for the Reed–Muller code [BCFGRS17].

[6] Emulators are sometimes known as "lazy samplers" or "simulators". In this paper we reserve the word *simulator* to refer to zero knowledge simulators.

[7] A further strengthening is the ability to emulate oracle queries conditioned on an input list of (real) oracle query-answer pairs. We use this additional property to show that Micali's SNARK maintains zero knowledge in the AROM (see Section 2.3).

**Challenges for the ARO.** The low-degree structure of $\widehat{\mathsf{vo}}$ may suggest that succinct constraint detection directly yields a construction of $\mathcal{M}^{(\mathsf{ro},\mathsf{wo})}$ with perfect emulation. However, the "sparsity" of $\mathsf{vo}$ implies that the set of all possible $\widehat{\mathsf{vo}}$ is *not* a linear space, as we now explain. Recall that for $x \in \{0,1\}^m, y \in \{0,1\}^\lambda, z \in \{0,1\}^w$, $\widehat{\mathsf{vo}}(x,y,z) = \mathsf{vo}(x,y,z) = 1$ if and only if $y = \mathsf{ro}(x)$ and $z = \mathsf{wo}(x)$, and 0 otherwise. Hence, if $\widehat{\mathsf{vo}}_1, \widehat{\mathsf{vo}}_2$ are extended verification oracles, $\widehat{\mathsf{vo}}' = \widehat{\mathsf{vo}}_1 + \widehat{\mathsf{vo}}_2$ may not be an extended verification oracle because there may exist $x, y_1, y_2, z_1, z_2$ such that $\widehat{\mathsf{vo}}'(x, y_1, z_1) = \widehat{\mathsf{vo}}'(x, y_2, z_2) = 1$ and $y_1 \neq y_2$. Hence, unlike for the LDRO, *we cannot directly construct $\mathcal{M}^{(\mathsf{ro},\mathsf{wo})}$ from succinct constraint detection.*

**Our approach.** We adopt a novel approach to simulation. First, we design a query-efficient but time-inefficient perfect emulator for a random low-degree extension $\hat{f}$ of a given arbitrary function $f$.[8] This *almost* suffices for our goal because $\widehat{\mathsf{vo}}$ is a random low-degree extension of the function $\mathsf{vo}$ defined by $(\mathsf{ro}, \mathsf{wo})$, which we can efficiently compute at any point by querying $\mathsf{ro}$ and $\mathsf{wo}$. Second, we additionally achieve time-efficient emulation by leveraging the *sparsity* of $\mathsf{vo}$, at the cost of a small statistical emulation error.

**(1) Time-inefficient emulation of a random low-degree extension.** Let $f\colon \{0,1\}^n \to \mathbb{F}$ be a function and $d \in \mathbb{N}$ a degree bound. We seek an emulator $\mathcal{M}_{\mathsf{LD}}$ such that $\mathcal{M}_{\mathsf{LD}}^f$ answers queries in a way that is identically distributed to a random extension $\hat{f}$ of $f$ with individual degree at most $d$.

We fix some notation. For $w \in \{0,1\}^n$, we denote by $\delta_w$ the unique multilinear polynomial with $\delta_w(w) = 1$ and $\delta_w(x) = 0$ for all $x \in \{0,1\}^n \setminus \{w\}$. For a set $S \subseteq \mathbb{F}^n$, we say that $w$ is $S$-bad if for every $n$-variate polynomial $Q$ of individual degree at most $d$ such that (i) $Q(x) = 0$ for every $x \in \{0,1\}^n \setminus \{w\}$ and (ii) $Q(z) = 0$ for every $z \in S$, it holds that $Q(w) = 0$. For a query-answer list $\mathsf{tr} \in (\mathbb{F}^n \times \mathbb{F})^t$, we denote the query set $\mathrm{supp}(\mathsf{tr}) := \{x : (x,y) \in \mathsf{tr}\}$.

Intuitively, $w$ is $S$-bad if $f(w)$ can be deduced from $\hat{f}|_S$, i.e. given the evaluation table of $f$ everywhere except at $w$ and partial knowledge about the structure of a low-degree extension $\hat{f}$. Note that $S$-badness is monotone with respect to $S$ and that if $w \in S$ then $w$ is $S$-bad.

The query-efficient but time-inefficient emulator $\mathcal{M}_{\mathsf{LD}}$ works as follows.

$\mathcal{M}_{\mathsf{LD}}^f(\mathsf{tr}, x^*)$:

1. Let $S := \mathrm{supp}(\mathsf{tr}) \cup \{x^*\}$, and $W$ be the set of $S$-bad points. Set $P(\vec{X}) := \sum_{w \in W} f(w) \cdot \delta_w(\vec{X})$.
2. Define $g\colon \mathrm{supp}(\mathsf{tr}) \cup \{0,1\}^n \to \mathbb{F}$ by $g(x) := \begin{cases} \mathsf{tr}(x) - P(x) & \text{if } x \in \mathrm{supp}(\mathsf{tr}) \\ 0 & \text{if } x \in \{0,1\}^n \end{cases}$.
3. Sample a random degree-$d$ extension $\hat{g}$ of $g$.
4. Return $y := \hat{g}(x) + P(x)$ and $\mathsf{tr}' := \mathsf{tr} \cup \{(x^*, y)\}$.

Observe that $g$ is well-defined since if $w \in \mathrm{supp}(\mathsf{tr}) \cap \{0,1\}^n$ then $w \in W$ and $\mathsf{tr}(w) = f(w)$ by assumption. Moreover, $\hat{g} + P$ is a low-degree extension of the function $f'\colon \{0,1\}^n \to \mathbb{F}$ given by $f'(w) = 0$ for $S$-good $w$ and $f'(w) = f(w)$ for $S$-bad $w$. That is, $f'$ is consistent with $f$ at every point the adversary "knows", and is zero elsewhere. $\hat{g} + P$ is also consistent with all prior queries as recorded in $\mathsf{tr}$. We show later that this is sufficient to perfectly emulate a random low-degree extension of $f$.

The query complexity of $\mathcal{M}_{\mathsf{LD}}$ is equal to the size of $W$, i.e., the number of $S$-bad points. Aaronson and Wigderson [AW09, Lemma 4.3] proved that, provided $d \geq 2$, the number of $S$-bad points is at most $|S| = |\mathrm{supp}(\mathsf{tr})| + 1$. Moreover since $S$-badness is monotone, querying $\mathcal{M}_{\mathsf{LD}}$ $t$ times results in $t$ queries to $f$ across the whole execution.

**(2) Time-efficient emulation from sparsity.** There are two sources of time-inefficiency in the emulator $\mathcal{M}_{\mathsf{LD}}$: (i) sampling the polynomial $\hat{g}$; (ii) computing the set $W$. We consider each of these difficulties in turn.

[8] In contrast, emulating the low-degree random oracle (as in [CCS22]) corresponds to emulating $\hat{f}$ for a *random* function $f$ that the emulator *samples itself*. This considerably simplifies the task, and in particular enables a time-efficient perfect emulation.

(i) *Sampling $\hat{g}$.* Since $\hat{g}$ has an exponentially-large description, we cannot sample it explicitly. Instead, we might hope to make use of the random multivariate polynomial sampling algorithm of [BCFGRS17], which achieves the following guarantee.

**Lemma 3.** *There is an efficient probabilistic algorithm* $\mathsf{LDSample}$ *such that for every degree bound $d \in \mathbb{N}$, set $S \subseteq \mathbb{F}^n$, map $h\colon S \to \mathbb{F}$, $q \in \mathbb{F}^n$, and $\alpha \in \mathbb{F}$,*

$$\Pr[\mathsf{LDSample}(1^d, S, h, x) = \alpha] = \Pr[P(x) = \alpha \mid P|_S = h] \ ,$$

*where $P$ is a uniformly random $n$-variate polynomial of individual degree at most $d$.*[9]

We do not know how to use $\mathsf{LDSample}$ to sample $\hat{g}$ directly, since that would require $S = \mathrm{supp}(\mathsf{tr}) \cup \{0,1\}^n$ and so $\mathsf{LDSample}$ would run in exponential time. Instead, we use a structural result about low-degree extensions of the zero function, the *combinatorial nullstellensatz* [Alo99].

**Lemma 4** (informal)**.** *If a polynomial $P$ is zero on $\{0,1\}^n$, then there exist polynomials $(R_i)_{i=1}^n$ such that*

$$P(\vec{X}) \equiv \sum_{i=1}^{n} X_i(X_i - 1)R_i(\vec{X}) \ . \tag{2}$$

Combining Lemma 4 with a linear-algebraic argument, we show that sampling each $R_i$ in Equation 2 uniformly at random subject to constraints implied by $P$ being a low-degree extension of $g$ yields a uniformly random low-degree extension of $g$. We can then sample each $R_i$ via $\mathsf{LDSample}$.

(ii) *The set $W$.* We do not know of an algorithm that can efficiently compute, given a set $S \subseteq \mathbb{F}^n$, the set of all $S$-bad points. As a result, we do not know how to efficiently realize $\mathcal{M}_{\mathsf{LD}}$. Instead we address this difficulty by leveraging the structure of $\mathsf{vo}$. Specifically, we consider $g = \mathsf{vo}$ (and thus $n = m + \lambda + w$).

We observe that $\mathsf{vo}$ is sparse: it is nonzero only at points $(x, y, z)$ for $\mathsf{ro}(x) = y$ and $\mathsf{wo}(x) = z$. If the adversary has not queried $\mathsf{ro}$ at $x$, then intuitively (since $\mathsf{ro}(x)$ is random) it will not be able to find any $y, z$ such that $\mathsf{vo}(x, y, z) = 1$, even given access to $\widehat{\mathsf{vo}}$. In particular, the probability that the set $W$ contains any $(x, y, z) \in \{0,1\}^{m+\lambda+w}$ such that $\mathsf{vo}(x, y, z) = 1$, but $\mathsf{ro}(x)$ was not yet queried, should be negligible. Indeed, we show that this probability is at most $|S|/2^\lambda$.

Observe that Step 1 of the time-inefficient emulator does nothing if $f(w) = 0$ for all $w \in W$. It follows from the above that to achieve simulation accuracy $O(|S|/2^\lambda)$, it suffices to include in $W$ only points $(x, y, z)$ for which the adversary has already queried $\mathsf{ro}$ at $x$ and $\mathsf{ro}(x) = y, \mathsf{wo}(x) = z$. Since we observe the adversary's queries to $\mathsf{ro}$, this set of points is easy to determine.

To show this formally, we follow an "identical-until-bad-is-set" analysis [BR06].

[9] If the RHS is not well-defined, $\mathsf{LDSample}$ outputs $\perp$.

# 3 Preliminaries

## 3.1 Notations

We define $[n] := \{1, \ldots, n\}$. For a subset $S \subseteq [n]$, we use $\bar{S}$ to denote the complement of $S$. We use $\mathbb{F}^{\leq d}[X_1, \ldots, X_m]$ to denote the set of $m$-variate polynomials of individual degree at most $d$ with coefficients in $\mathbb{F}$; we write $\deg(\cdot)$ to denote the individual degree. For $\vec{d} = (d_1, \ldots, d_m)$, we use $\mathbb{F}^{\leq \vec{d}}[X_1, \ldots, X_m]$ to denote the set of $m$-variate polynomials such that the variable $X_i$ has individual degree at most $d_i$ for each $i \in [m]$.

**Functions.** We use $\mathrm{Dom}(f)$ to denote the domain and $\mathrm{Cod}(f)$ to denote the codomain of a function $f$. We use $(X \to Y)$ to denote the set of all functions $\{f \colon X \to Y\}$, and $(X \rightharpoonup Y)$ to denote the set of all partial functions $\{f \colon X \rightharpoonup Y\}$. For a linear map $\Phi$, we use $\ker(\Phi)$ to denote the kernel of $\Phi$ and $\mathrm{im}(\Phi)$ to denote the image of $\Phi$. We say that a function is total if it is defined for all elements of its domain, and say that it is not total otherwise.

**Low-degree extensions.** For $n, d \in \mathbb{N}$, $S \subset \mathbb{F}^n$ and $f \colon S \to \mathbb{F}$ we denote the set of extensions of degree at most $d$ of $f$ to the field $\mathbb{F}$ by $\mathsf{LDE}_{\mathbb{F},d}[f] := \left\{ \hat{f} \in \mathbb{F}^{\leq d}[X_1, \ldots, X_n] : \hat{f}(x) = f(x)\ \forall x \in S \right\}$.

**Distributions.** For finite set $X$, we write $x \leftarrow X$ to denote that $x$ is drawn uniformly at random from $X$. We use $\mathrm{supp}(\mathcal{D})$ to denote the support of the distribution $\mathcal{D}$. We write $\mathcal{U}(X)$ to denote the uniform distribution over the set $X$.

**Oracle distributions.** An *oracle distribution* $\mathcal{O}$ is a distribution over functions $\theta \colon X \to Y$. We define $\mathrm{Dom}(\mathcal{O}) := X$ and $\mathrm{Cod}(\mathcal{O}) := Y$. A *random oracle* is an oracle distribution $\mathcal{U}(m, n)$ given by $\theta \leftarrow (\{0,1\}^m \to \{0,1\}^n)$ for some $m, n \in \mathbb{N}$.

**Oracle algorithms.** For a function $\theta \colon X \to Y$, we write $A^\theta$ for an algorithm with oracle access to $\theta$. Further, for a tuple of functions $(\theta_1, \ldots, \theta_\nu)$, where $\nu \in \mathbb{N}$, with $\theta_i \colon X_i \to Y_i$, we write $A^{(\theta_1, \ldots, \theta_\nu)}$ for an algorithm with oracle access to each $\theta_i$ for $i \in [\nu]$, and $A^{\theta_S}$ for an algorithm with oracle access to a subset of functions $\{\theta_i |\ i \in S\}$ where $S \subseteq [\nu]$. Often it is useful to view a tuple of oracles $(\theta_1, \ldots, \theta_\nu)$ as a single *combined* oracle $\theta$ such that $\theta(\mathsf{oid}, x) = \theta_{\mathsf{oid}}(x)$ for all $\mathsf{oid} \in [\nu]$ and $x \in \mathrm{Dom}(\theta_{\mathsf{oid}})$.

**Oracle transcripts.** For $\mathscr{F} \subseteq (X \to Y)$, an $\mathscr{F}$-query-answer transcript is a sequence of tuples $\mathsf{tr} := ((x_1, y_1), \ldots, (x_t, y_t)) \in (X \times Y)^t$ for some $t \in \mathbb{N}$, such that there exists $f \in \mathscr{F}$ where for all $i$, $f(x_i) = y_i$. We denote the query set $\mathrm{supp}(\mathsf{tr}) := \{x : (x, y) \in \mathsf{tr}\}$. Note that we can view $\mathsf{tr}$ as a function $\mathrm{supp}(\mathsf{tr}) \to \mathbb{F}$. For an oracle distribution $\mathcal{O}$, we define an $\mathcal{O}$-query-answer transcript to be a $\mathrm{supp}(\mathcal{O})$-query-answer transcript. For $\mathcal{O}$ supported on $\nu$-tuples of oracles, an $\mathcal{O}$-query-answer transcript is defined with respect to the combined oracle; i.e., $\mathsf{tr} = ((\mathsf{oid}_1, x_1, y_1), \ldots, (\mathsf{oid}_t, x_t, y_t)) \in (\bigcup_{\mathsf{oid}=1}^{\nu} (\{\mathsf{oid}\} \times \mathrm{Dom}(\theta_{\mathsf{oid}}) \times \mathrm{Cod}(\theta_{\mathsf{oid}})))^t$. We also define $\mathsf{tr}|_{\mathsf{oid}} := \{(x, y) : (\mathsf{oid}, x, y) \in \mathsf{tr}\}$.

For an oracle algorithm $A$ and oracle $\theta$, the notation $\mathsf{o} \xleftarrow{\mathsf{tr}} A^\theta(z)$ denotes that $A^\theta$ on input $z$ outputs $\mathsf{o}$ and makes the sequence of oracle queries $\mathsf{tr}$.

**Indexed relations.** An *indexed relation* $\mathcal{R}$ is a set of triples $(\mathtt{i}, \mathtt{x}, \mathtt{w})$ where $\mathtt{i}$ is the index, $\mathtt{x}$ is the instance, and $\mathtt{w}$ is the witness; the corresponding *indexed language* $\mathcal{L}(\mathcal{R})$ is the set of index-instance pairs $(\mathtt{i}, \mathtt{x})$ for which there exists a witness $\mathtt{w}$ such that $(\mathtt{i}, \mathtt{x}, \mathtt{w}) \in \mathcal{R}$. For example, the indexed relation of satisfiable boolean circuits consists of triples where $\mathtt{i}$ is the description of a boolean circuit, $\mathtt{x}$ is a partial assignment to its input wires, and $\mathtt{w}$ is an assignment to the remaining wires that makes the circuit output 0.

**Oracle relations.** For a set of oracle distributions $\mathcal{X}$, we write $\mathcal{R}^{\mathcal{X}}$ to denote the set of indexed relations $\{\mathcal{R}^\theta : \theta \in \bigcup_{\mathcal{O} \in \mathcal{X}} \mathrm{supp}(\mathcal{O})\}$. When considering sets of oracle distributions $\mathcal{X}$ for which each $\mathcal{O} \in \mathcal{X}$ is such that $\mathrm{supp}(\mathcal{O})$ contains tuples of oracles $(\theta_1, \ldots, \theta_\nu)$, for $\nu \in \mathbb{N}$, with oracle identifiers $(\mathsf{oid}_1, \ldots \mathsf{oid}_\nu)$, we

write $\mathcal{R}^{(\mathcal{X},\mathsf{oid})}$ to denote the set of indexed relations $\{\mathcal{R}^{\theta_{\mathsf{oid}}} : (\theta_1,\ldots,\theta_\nu) \in \bigcup_{\mathcal{O}\in\mathcal{X}} \mathrm{supp}(\mathcal{O})\}$. We define $\mathcal{R}^{(\mathcal{X},\mathsf{oid})} \in \mathsf{NP}^{(\mathcal{X},\mathsf{oid})}$ if and only if there exists a polynomial-time oracle Turing machine $M$ such that, for every $(\theta_1,\ldots,\theta_\nu) \in \bigcup_{\mathcal{O}\in\mathcal{X}} \mathrm{supp}(\mathcal{O})$, $\mathcal{R}^{\theta_{\mathsf{oid}}} = \{(\mathbb{i},\mathbb{x},\mathbb{w}) : M^{\theta_{\mathsf{oid}}}(\mathbb{i},\mathbb{x},\mathbb{w}) = 1\}$.

**Security parameters.** We assume for simplicity that all public parameters have a length of at least $\lambda$ so that efficient algorithms that receive such parameters can run in time (at least) polynomial in $\lambda$.

**Adversaries.** An adversary (or extractor) is *polynomial-size* if it can be expressed as a circuit of polynomial size. We also consider a relaxed definition: an adversary (or extractor) running in *(non-uniform) expected polynomial-time* is a Turing machine provided with a *polynomial-size* non-uniform advice string and access to an infinite random tape, whose expected running time for all choices of advice is polynomial.

An adversary $\mathcal{A}$ with expected running time $t$ and success probability $p$ can be converted into a circuit of size $O(t/\epsilon)$ with success probability $p-\epsilon$ as follows: first truncate the execution of $\mathcal{A}$ at running time $t/\epsilon$; then choose as advice the randomness that maximizes the success probability of the truncated $\mathcal{A}$.

For $\nu \in \mathbb{N}$ and a distribution $\mathcal{O}$, whose support contains tuples of oracles $(\theta_1,\ldots,\theta_\nu)$, we refer to an adversary with access to $(\theta_1,\ldots,\theta_\nu) \leftarrow \mathcal{O}$ as an $\mathcal{O}$-adversary.

### 3.2 Linear algebra and the combinatorial nullstellensatz

**Definition 3.1.** *An affine subspace $S$ of a vector space $V$ is a set $a + S_0 = \{a + s_0 : s_0 \in S_0\}$ where $a \in V$ and $S_0$ is a subspace of $V$.*

**Claim 3.2.** *Let $\Phi\colon V \to W$ be a linear map, and let $S$ be a finite affine subspace of $V$. If $s \sim \mathcal{U}(S)$ then $\Phi(s) \sim \mathcal{U}(\Phi(S))$.*

*Proof.* Since $S$ is a finite affine subspace of $V$, there is a vector $a \in V$ and a finite subspace $S_0 \subseteq V$ such that $S_0 := \{s - a : s \in S\}$. Fix $w \in \Phi(S)$ and $s_0' \in S_0$ such that $\Phi(s_0') = w - \Phi(a)$; then

$$\begin{aligned}\Pr_{s\leftarrow S}[\Phi(s) = w] &= \Pr_{s_0\leftarrow S_0}[\Phi(s_0) = w - \Phi(a)] = \Pr_{s_0\leftarrow S_0}[s_0 \in \ker(\Phi) + s_0'] \\ &= |\ker(\Phi)|/|S_0| = 1/|\Phi(S_0)| = 1/|\Phi(S)|\,.\end{aligned}$$

□

**Lemma 3.3** (Combinatorial nullstellensatz [Alo99])**.** *Let $\mathbb{F}$ be an arbitrary field, and let $f$ be a polynomial in $\mathbb{F}[X_1,\ldots,X_n]$. Let $S_1,\ldots,S_n$ be nonempty subsets of $\mathbb{F}$ and define $g_i(x_i) = \prod_{s\in S_i}(x_i - s)$. If $f$ vanishes over all the common zeros of $g_1,\ldots,g_n$ (that is, if $f(s_1,\ldots,s_n) = 0$ for all $s_i \in S_i$), then there are polynomials $h_1,\ldots,h_n \in \mathbb{F}[X_1,\ldots,X_n]$ so that $\deg(h_i) \le \deg(f) - \deg(g_i)$ so that*

$$f = \sum_{i=1}^{n} h_i g_i \ .$$

### 3.3 Non-interactive arguments in oracle models

Given a set of oracle distributions $\mathcal{X}$, a (preprocessing) *non-interactive argument* relative for an indexed oracle relation $\mathcal{R}^{\mathcal{X}}$ is a tuple of algorithms $\mathsf{ARG} = (\mathcal{G},\mathcal{I},\mathcal{P},\mathcal{V})$ that works as follows. Below we denote by $\theta$ an oracle (or tuple of oracles) in the set $\bigcup_{\mathcal{O}\in\mathcal{X}} \mathrm{supp}(\mathcal{O})$.

- $\mathcal{G}(1^\lambda) \to \mathsf{pp}$. On input a security parameter $\lambda$ (in unary), the generator $\mathcal{G}$ samples public parameters $\mathsf{pp}$.
- $\mathcal{I}^\theta(\mathsf{pp},\mathbb{i}) \to (\mathsf{ipk},\mathsf{ivk})$. On input public parameters $\mathsf{pp}$ and an index $\mathbb{i}$ for the relation $\mathcal{R}$, the indexer $\mathcal{I}$ deterministically computes index-specific proving and verification keys $(\mathsf{ipk},\mathsf{ivk})$.

- $\mathcal{P}^\theta(\mathsf{ipk}, \mathbb{x}, \mathbb{w}) \to \pi$. On input an index-specific proving key $\mathsf{ipk}$, an instance $\mathbb{x}$, and a corresponding witness $\mathbb{w}$, the prover $\mathcal{P}$ computes a proof $\pi$ that attests to the claim that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}^\theta$.
- $\mathcal{V}^\theta(\mathsf{ivk}, \mathbb{x}, \pi) \to b$. On input an index-specific verification key $\mathsf{ivk}$, an instance $\mathbb{x}$, and a corresponding proof $\pi$, the verifier $\mathcal{V}$ computes a bit indicating whether $\pi$ is a valid proof.

We require ARG to satisfy the following completeness and soundness properties.

- *Completeness.* For every oracle distribution $\mathcal{O} \in \mathcal{X}$ and adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|r} & \theta \leftarrow \mathcal{O}(\lambda) \\ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}^\theta & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ \Downarrow & (\mathbb{i}, \mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}^\theta(\mathsf{pp}) \\ \mathcal{V}^\theta(\mathsf{ivk}, \mathbb{x}, \pi) = 1 & (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}^\theta(\mathsf{pp}, \mathbb{i}) \\ & \pi \leftarrow \mathcal{P}^\theta(\mathsf{ipk}, \mathbb{x}, \mathbb{w}) \end{array}\right] = 1 \ .$$

  The above formulation of completeness allows $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ to depend on the oracle $\theta$ and public parameters $\mathsf{pp}$.

- *Soundness.* For every oracle distribution $\mathcal{O} \in \mathcal{X}$ and polynomial-size adversary $\tilde{\mathcal{P}}$,

$$\Pr\left[\begin{array}{c|r} \mathcal{V}^\theta(\mathsf{ivk}, \mathbb{x}, \pi) = 1 & \theta \leftarrow \mathcal{O}(\lambda) \\ \wedge & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathbb{i}, \mathbb{x}) \notin \mathcal{L}(\mathcal{R}^\theta) & (\mathbb{i}, \mathbb{x}, \pi) \leftarrow \tilde{\mathcal{P}}^\theta(\mathsf{pp}) \\ & (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}^\theta(\mathsf{pp}, \mathbb{i}) \end{array}\right] \leq \mathrm{negl}(\lambda) \ .$$

The above formulation of soundness allows $(\mathbb{i}, \mathbb{x})$ to depend on the oracle $\theta$ and public parameters $\mathsf{pp}$.

We also consider straightline knowledge soundness properties and zero knowledge for ARG.

**Straightline knowledge soundness.** ARG has *straightline knowledge soundness* (with respect to auxiliary input distribution $\mathcal{D}$) if there exists a deterministic polynomial-time extractor $\mathcal{E}$ such that for every oracle distribution $\mathcal{O} \in \mathcal{X}$ and (non-uniform) polynomial-time adversary $\tilde{\mathcal{P}}$,

$$\Pr\left[\begin{array}{c|r} & \theta \leftarrow \mathcal{O}(\lambda) \\ \mathcal{V}^\theta(\mathsf{ivk}, \mathbb{x}, \pi) = 1 & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ \wedge & \mathsf{ai} \leftarrow \mathcal{D}(\mathsf{pp}) \\ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}^\theta & (\mathbb{i}, \mathbb{x}, \pi) \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}^\theta(\mathsf{pp}, \mathsf{ai}) \\ & (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}^\theta(\mathsf{pp}, \mathbb{i}) \\ & \mathbb{w} \leftarrow \mathcal{E}(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \pi, \mathsf{tr}) \end{array}\right] \leq \mathrm{negl}(\lambda) \ .$$

**Zero knowledge.** ARG has statistical zero knowledge if there exists a probabilistic polynomial-time stateful simulator $\mathcal{S}$ such that for every oracle distribution $\mathcal{O} \in \mathcal{X}$ and polynomial-size honest stateful adversary $\mathcal{A}$, the following distributions are $\mathrm{negl}(\lambda)$-close in statistical distance:

$$\left\{\mathcal{A}^\theta(\pi) \;\middle|\; \begin{array}{r} \theta \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}^\theta(\mathsf{pp}) \\ (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}^\theta(\mathsf{pp}, \mathbb{i}) \\ \pi \leftarrow \mathcal{P}^\theta(\mathsf{ipk}, \mathbb{x}, \mathbb{w}) \end{array}\right\} \quad \text{and} \quad \left\{\mathcal{A}^{\mathcal{S}^\theta}(\pi) \;\middle|\; \begin{array}{r} \theta \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathcal{S}(1^\lambda) \\ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \xleftarrow{\mathsf{tr}} \mathcal{A}^\theta(\mathsf{pp}) \\ \pi \leftarrow \mathcal{S}^\theta(\mathbb{i}, \mathbb{x}, \mathsf{tr}) \end{array}\right\} \ . \tag{3}$$

An adversary $\mathcal{A}$ is *honest* if it outputs $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}^\theta$ with probability $\geq 1 - \mathrm{negl}(\lambda)$. Above, the notation $\mathcal{A}^{\mathcal{S}^\theta}$ indicates that the simulator $\mathcal{S}$ (with oracle access to $\theta$) answers the oracle queries of $\mathcal{A}$.

**Succinctness.** In this work, we say that a non-interactive argument system $\mathsf{ARG}$ for $\mathcal{R}^{\mathcal{X}}$ is *succinct* if there is a fixed polynomial $p$ such that *both* the length of the proof and the running time of the argument verifier are bounded by $p(\lambda, |\mathbb{x}|)$. In this case, we refer to $\mathsf{ARG}$ as a SNARG; if $\mathsf{ARG}$ also has knowledge soundness, it is a SNARK.

### 3.4 Proof-carrying data

A triple of algorithms $\mathsf{PCD} = (\mathbb{G}, \mathbb{I}, \mathbb{P}, \mathbb{V})$ is a (preprocessing) *proof-carrying data scheme* (PCD scheme) for a class of compliance predicates $\mathsf{F}$ relative to a set of oracle distributions $\mathcal{X}$ if the properties below hold.

**Definition 3.4.** *A* **transcript** $\mathsf{T}$ *is a directed acyclic graph where each vertex $u \in V(\mathsf{T})$ is labeled by local data $z^{(u)}_{\mathsf{loc}}$ and each edge $e \in E(\mathsf{T})$ is labeled by a message $z^{(e)} \neq \bot$. The* **output** *of a transcript $\mathsf{T}$, denoted $\mathsf{o}(\mathsf{T})$, is $z^{(e)}$ where $e = (u, v)$ is the lexicographically-first edge such that $v$ is a sink.*

**Definition 3.5.** *A vertex $u \in V(\mathsf{T})$ is* **$\Phi$-compliant** *for $\Phi \in \mathsf{F}$ if for all outgoing edges $e = (u, v) \in E(\mathsf{T})$ and for all $\theta \in \bigcup_{\mathcal{O} \in \mathcal{X}} \mathrm{supp}(\mathcal{O})$:*

- *(base case) if $u$ has no incoming edges, $\Phi^\theta(z^{(e)}, z^{(u)}_{\mathsf{loc}}, \bot, \ldots, \bot) = 1$;*
- *(recursive case) if $u$ has incoming edges $e_1, \ldots, e_m$, $\Phi^\theta(z^{(e)}, z^{(u)}_{\mathsf{loc}}, z^{(e_1)}, \ldots, z^{(e_m)}) = 1$.*

*We say that $\mathsf{T}$ is* **$\Phi$-compliant** *if $E(\mathsf{T})$ is non-empty and all vertices incident to an edge are $\Phi$-compliant.*

**Completeness.** For every oracle distribution $\mathcal{O} \in \mathcal{X}$ and adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c} \left(\begin{array}{c} \Phi \in \mathsf{F} \\ \wedge \Big((\wedge_{i=1}^m z_i = \bot) \vee (\wedge_{i=1}^m \mathbb{V}^\theta(\mathbb{ivk}, z_i, \pi_i) = 1)\Big) \\ \wedge\ \Phi^\theta(z, z_{\mathsf{loc}}, z_1, \ldots, z_m) = 1 \\ \Downarrow \\ \mathbb{V}^\theta(\mathbb{ivk}, z, \pi) = 1 \end{array}\right) \end{array} \middle| \begin{array}{r} \theta \leftarrow \mathcal{O}(\lambda) \\ \mathbb{pp} \leftarrow \mathbb{G}(1^\lambda) \\ (\Phi, z, z_{\mathsf{loc}}, [z_i, \pi_i]_{i=1}^m) \leftarrow \mathcal{A}^\theta(\mathbb{pp}) \\ (\mathbb{ipk}, \mathbb{ivk}) \leftarrow \mathbb{I}^\theta(\mathbb{pp}, \Phi) \\ \pi \leftarrow \mathbb{P}^\theta(\mathbb{ipk}, z, z_{\mathsf{loc}}, [z_i, \pi_i]_{i=1}^m) \end{array} \right] = 1 \ .$$

**Straightline knowledge soundness.** $\mathsf{PCD} = (\mathbb{G}, \mathbb{I}, \mathbb{P}, \mathbb{V})$ has straightline knowledge soundness (with respect to auxiliary input distribution $\mathcal{D}$) if there exists a deterministic polynomial-time extractor $\mathbb{E}$ such that for every oracle distribution $\mathcal{O} \in \mathcal{X}$ and (non-uniform) polynomial-time adversary $\tilde{\mathbb{P}}$,

$$\Pr\left[\begin{array}{l} \\ \Phi \in \mathsf{F} \\ \wedge\, \mathbb{V}(\mathbb{ivk}, \mathsf{o}, \pi) = 1 \\ \wedge \Big(\mathsf{T} \text{ is not } \Phi\text{-compliant} \vee\ \mathsf{o}(\mathsf{T}) \neq \mathsf{o}\Big) \\ \\ \end{array} \middle| \begin{array}{r} \theta \leftarrow \mathcal{O}(\lambda) \\ \mathbb{pp} \leftarrow \mathbb{G}(1^\lambda) \\ \mathsf{ai} \leftarrow \mathcal{D}(\mathbb{pp}) \\ (\Phi, \mathsf{o}, \pi) \xleftarrow{\mathsf{tr}} \tilde{\mathbb{P}}^\theta(\mathbb{pp}, \mathsf{ai}) \\ (\mathbb{ipk}, \mathbb{ivk}) \leftarrow \mathbb{I}^\theta(\mathbb{pp}, \Phi) \\ \mathsf{T} \leftarrow \mathbb{E}(\mathbb{pp}, \Phi, \mathsf{o}, \pi, \mathsf{tr}) \end{array} \right] \leq \mathrm{negl}(\lambda) \ .$$

**Zero knowledge.** $\mathsf{PCD}$ has statistical zero knowledge if there exists a probabilistic polynomial-time stateful simulator $\mathbb{S}$ such that for every oracle distribution $\mathcal{O} \in \mathcal{X}$ and polynomial-size honest (stateful) adversary $\mathcal{A}$,

the following distributions are $\mathrm{negl}(\lambda)$-close in statistical distance:

$$\left\{ \mathcal{A}^{\theta}(\pi) \;\middle|\; \begin{array}{r} \theta \leftarrow \mathcal{O}(\lambda) \\ \mathbb{pp} \leftarrow \mathbb{G}(1^\lambda) \\ (\Phi, z, z_{\mathsf{loc}}, [z_i, \pi_i]_{i=1}^m) \leftarrow \mathcal{A}^{\theta}(\mathbb{pp}) \\ (\mathbb{ipk}, \mathbb{ivk}) \leftarrow \mathbb{I}^{\theta}(\mathbb{pp}, \Phi) \\ \pi \leftarrow \mathbb{P}^{\theta}(\mathbb{ipk}, \Phi, z, z_{\mathsf{loc}}, [z_i, \pi_i]_{i=1}^m) \end{array} \right\} \text{ and } \left\{ \mathcal{A}^{\mathbb{S}^{\theta}}(\pi) \;\middle|\; \begin{array}{r} \theta \leftarrow \mathcal{O}(\lambda) \\ \mathbb{pp} \leftarrow \mathbb{S}(1^\lambda) \\ (\Phi, z, z_{\mathsf{loc}}, [z_i, \pi_i]_{i=1}^m) \xleftarrow{\mathsf{tr}} \mathcal{A}^{\theta}(\mathbb{pp}) \\ \pi \leftarrow \mathbb{S}^{\theta}(\Phi, z, \mathsf{tr}) \end{array} \right\} .$$

An adversary $\mathcal{A}$ is *honest* if its output satisfies the implicant of the completeness condition with probability $\geq 1 - \mathrm{negl}(\lambda)$ (i.e., $\Phi \in \mathsf{F}$, $\Phi^{\theta}(z, z_{\mathsf{loc}}, z_1, \ldots, z_m) = 1$, and either for all $i$, $z_i = \bot$, or for all $i$, $\mathbb{V}^{\theta}(\mathbb{ivk}, z_i, \pi_i) = 1$). Above, the notation $\mathcal{A}^{\mathbb{S}}$ indicates that the simulator $\mathbb{S}$ answers oracle queries of $\mathcal{A}$.

**Efficiency.** The generator $\mathbb{G}$, prover $\mathbb{P}$, indexer $\mathbb{I}$ and verifier $\mathbb{V}$ run in polynomial time. A proof $\pi$ has size $\mathrm{poly}(\lambda, |\Phi|)$; in particular, it does not grow with each application of $\mathbb{P}$.

### 3.5 Accumulation schemes

We recall the definition of an accumulation scheme from [BCMS20], extended to any set of oracle distributions; then, in Definition 3.6 below, we describe how to specialize that notion to the case of accumulating oracle queries.

Let $\Phi \colon \bigcup_{\mathcal{O} \in \mathcal{X}} \mathrm{supp}(\mathcal{O}(*)) \times (\{0,1\}^*)^3 \to \{0,1\}$ be a predicate (for clarity we write $\Phi^{\theta}(\mathsf{pp}_\Phi, \mathsf{i}_\Phi, \mathsf{q})$ for $\Phi(\theta, \mathsf{pp}_\Phi, \mathsf{i}_\Phi, \mathsf{q})$). Let $\mathcal{H}$ be a probabilistic algorithm with access to $\theta$, which outputs predicate parameters $\mathsf{pp}_\Phi$.

An **accumulation scheme for** $(\Phi, \mathcal{H})$ is a tuple of algorithms $\mathsf{AS} = (\mathrm{G}, \mathrm{I}, \mathrm{P}, \mathrm{V}, \mathrm{D})$ that have access to the same oracle $\theta$ (except for G). These algorithms satisfy *completeness* and *soundness*, and optionally also *zero knowledge*, as specified below.

**Completeness.** For every oracle distribution $\mathcal{O} \in \mathcal{X}$ and (unbounded) adversary $\mathcal{A}$,

$$\Pr\left[ \begin{array}{c} \forall\, j \in [\ell],\ \mathrm{D}^{\theta}(\mathsf{dk}, \mathsf{acc}_j) = 1 \\ \forall\, i \in [n],\ \Phi^{\theta}(\mathsf{pp}_\Phi, \mathsf{i}_\Phi, \mathsf{q}_i) = 1 \\ \Downarrow \\ \mathrm{V}^{\theta}(\mathsf{avk}, [\mathsf{q}_i]_{i=1}^n, [\mathsf{acc}_j]_{j=1}^\ell, \mathsf{acc}, \pi_{\mathrm{V}}) = 1 \\ \mathrm{D}^{\theta}(\mathsf{dk}, \mathsf{acc}) = 1 \end{array} \;\middle|\; \begin{array}{r} \theta \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathrm{G}(1^\lambda) \\ \mathsf{pp}_\Phi \leftarrow \mathcal{H}^{\theta}(1^\lambda) \\ (\mathsf{i}_\Phi, [\mathsf{q}_i]_{i=1}^n, [\mathsf{acc}_j]_{j=1}^\ell) \leftarrow \mathcal{A}^{\theta}(\mathsf{pp}, \mathsf{pp}_\Phi) \\ (\mathsf{apk}, \mathsf{avk}, \mathsf{dk}) \leftarrow \mathrm{I}^{\theta}(\mathsf{pp}, \mathsf{pp}_\Phi, \mathsf{i}_\Phi) \\ (\mathsf{acc}, \pi_{\mathrm{V}}) \leftarrow \mathrm{P}^{\theta}(\mathsf{apk}, [\mathsf{q}_i]_{i=1}^n, [\mathsf{acc}_j]_{j=1}^\ell) \end{array} \right] = 1 \ .$$

Note that for $\ell = n = 0$, the precondition on the left-hand side holds vacuously; this is required for the completeness condition to be non-trivial.

**Soundness.** For every oracle distribution $\mathcal{O} \in \mathcal{X}$ and polynomial-size adversary $\mathcal{A}$,

$$\Pr\left[ \begin{array}{c} \mathrm{V}^{\theta}(\mathsf{avk}, [\mathsf{q}_i]_{i=1}^n, [\mathsf{acc}_j]_{j=1}^\ell, \mathsf{acc}, \pi_{\mathrm{V}}) = 1 \\ \mathrm{D}^{\theta}(\mathsf{dk}, \mathsf{acc}) = 1 \\ \Downarrow \\ \forall\, j \in [\ell],\ \mathrm{D}^{\theta}(\mathsf{dk}, \mathsf{acc}_j) = 1 \\ \forall\, i \in [n],\ \Phi^{\theta}(\mathsf{pp}_\Phi, \mathsf{i}_\Phi, \mathsf{q}_i) = 1 \end{array} \;\middle|\; \begin{array}{r} \theta \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathrm{G}(1^\lambda) \\ \mathsf{pp}_\Phi \leftarrow \mathcal{H}^{\theta}(1^\lambda) \\ \begin{pmatrix} \mathsf{i}_\Phi & [\mathsf{q}_i]_{i=1}^n & [\mathsf{acc}_j]_{j=1}^\ell \\ \mathsf{acc} & \pi_{\mathrm{V}} & \end{pmatrix} \leftarrow \mathcal{A}^{\theta}(\mathsf{pp}, \mathsf{pp}_\Phi) \\ (\mathsf{apk}, \mathsf{avk}, \mathsf{dk}) \leftarrow \mathrm{I}^{\theta}(\mathsf{pp}, \mathsf{pp}_\Phi, \mathsf{i}_\Phi) \end{array} \right] \geq 1 - \mathrm{negl}(\lambda) \ .$$

**Zero knowledge.** There exists a polynomial-time stateful simulator S such that for every oracle distribution $\mathcal{O} \in \mathcal{X}$ and polynomial-size stateful "honest" adversary $\mathcal{A}$ (see below), the following distributions are

(statistically/computationally) indistinguishable:

$$\left\{ \mathcal{A}^{\theta}(\mathsf{acc}) \;\middle|\; \begin{array}{r} \theta \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathrm{G}(1^\lambda) \\ \mathsf{pp}_\Phi \leftarrow \mathcal{H}^{\theta}(1^\lambda) \\ (\mathsf{i}_\Phi, [\mathsf{q}_i]_{i=1}^n, [\mathsf{acc}_j]_{j=1}^\ell) \leftarrow \mathcal{A}^{\theta}(\mathsf{pp}, \mathsf{pp}_\Phi) \\ (\mathsf{apk}, \mathsf{avk}, \mathsf{dk}) \leftarrow \mathrm{I}^{\theta}(\mathsf{pp}, \mathsf{pp}_\Phi, \mathsf{i}_\Phi) \\ (\mathsf{acc}, \pi_{\mathrm{V}}) \leftarrow \mathrm{P}^{\theta}(\mathsf{apk}, [\mathsf{q}_i]_{i=1}^n, [\mathsf{acc}_j]_{j=1}^\ell) \end{array} \right\}$$

and

$$\left\{ \mathcal{A}^{\theta}(\mathsf{acc}) \;\middle|\; \begin{array}{r} \theta \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathrm{S}(1^\lambda) \\ \mathsf{pp}_\Phi \leftarrow \mathcal{H}^{\theta}(1^\lambda) \\ (\mathsf{i}_\Phi, [\mathsf{q}_i]_{i=1}^n, [\mathsf{acc}_j]_{j=1}^\ell) \xleftarrow{\mathsf{tr}} \mathcal{A}^{\theta}(\mathsf{pp}, \mathsf{pp}_\Phi) \\ \mathsf{acc} \leftarrow \mathrm{S}^{\theta}(\mathsf{pp}_\Phi, \mathsf{i}_\Phi, \mathsf{tr}) \end{array} \right\} .$$

Here $\mathcal{A}$ is *honest* if it outputs, with probability 1, a tuple $(\mathsf{i}_\Phi, [\mathsf{q}_i]_{i=1}^n, [\mathsf{acc}_j]_{j=1}^\ell)$ such that $\Phi^{\theta}(\mathsf{pp}_\Phi, \mathsf{i}_\Phi, \mathsf{q}_i) = 1$ and $\mathrm{D}^{\theta}(\mathsf{dk}, \mathsf{acc}_j) = 1$ for every $i \in [n]$ and $j \in [\ell]$. Note that the simulator S is *not* required to simulate the accumulation verifier proof $\pi_{\mathrm{V}}$.

**Accumulation scheme for oracle queries.** We explain how to specialize the general notion of an accumulation scheme above to the particular case of accumulating queries to a tuple of oracles.

**Definition 3.6.** *Let $\mathcal{X}$ be a set of oracle distributions. An* **accumulation scheme for $\mathcal{X}$-queries** *is an accumulation scheme where: (i) the accumulation verifier* V *does not access the oracle; (ii) $\mathcal{H} = \bot$ (and so $\mathsf{pp}_\Phi = \bot$); (iii) predicate inputs* $\mathsf{q}$ *are of the form $(x, y)$;*[10] *(iv) the predicate $\Phi$ is defined such that $\Phi^{\theta}(\mathsf{pp}_\Phi, \mathsf{i}_\Phi, x, y) = 1$ if and only if $\theta(x) = y$ (in particular, $\mathsf{pp}_\Phi$ and $\mathsf{i}_\Phi$ are ignored).*

## 3.6 Commitment schemes

Let $\nu \in \mathbb{N}$ and let $\mathcal{X}$ be a set of oracle distributions, such that each $\mathcal{O} \in \mathcal{X}$ is a distribution over tuples of oracles $(\theta_1, \ldots, \theta_\nu)$. A *commitment scheme in $\mathcal{X}$* is a tuple $\mathsf{CM} = (\mathsf{CM.Setup}, \mathsf{CM.Commit})$ with the following syntax.

- $\mathsf{CM.Setup}$, on input a security parameter $1^\lambda$, outputs a commitment key $\mathsf{ck}$.
- $\mathsf{CM.Commit}$, on input a commitment key $\mathsf{ck}$, a message $m \in \{0,1\}^*$, and randomness $\omega$, outputs a commitment $\mathsf{cm}$.

The tuple $\mathsf{CM}$ satisfies a binding property and, optionally, a hiding property.

- **Binding.** For every $\mathcal{O} \in \mathcal{X}$ and efficient adversary $\mathcal{A}$,

$$\Pr\left[ \begin{array}{c} m_0 \neq m_1 \\ \wedge \\ \mathsf{CM.Commit}(\mathsf{ck}, m_0; \omega_0) = \mathsf{CM.Commit}(\mathsf{ck}, m_1; \omega_1) \end{array} \;\middle|\; \begin{array}{r} (\theta_1, \ldots, \theta_\nu) \leftarrow \mathcal{O}(\lambda) \\ \mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda) \\ ((m_0, \omega_0), (m_1, \omega_1)) \leftarrow \mathcal{A}^{(\theta_1, \ldots, \theta_\nu)}(\mathsf{ck}) \end{array} \right] \leq \mathrm{negl}(\lambda) \ .$$

[10] If $\mathcal{X}$ is a set of oracle distributions whose support contains tuples of oracles, then $x$ is assumed to start with the oracle identifier corresponding to the oracle being queried.

- **Hiding.** For every $\mathcal{O} \in \mathcal{X}$ and efficient stateful adversary $\mathcal{A}$ that outputs two messages of the same length, the following distributions are (statistically or computationally) indistinguishable:

$$\mathcal{D}_0(\lambda) := \left\{ (\mathsf{pp}, \mathsf{cm}, \mathsf{aux}) \;\middle|\; \begin{array}{r} (\theta_1, \ldots, \theta_\nu) \leftarrow \mathcal{O}(\lambda) \\ \mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda) \\ (m_0, m_1, \mathsf{aux}) \leftarrow \mathcal{A}^{(\theta_1,\ldots,\theta_\nu)}(\mathsf{ck}) \\ \omega \leftarrow \{0,1\}^{\mathrm{poly}(\lambda)} \\ \mathsf{cm} := \mathsf{CM.Commit}(\mathsf{ck}, m_0; \omega) \end{array} \right\}$$

$$\text{and } \mathcal{D}_1(\lambda) := \left\{ (\mathsf{pp}, \mathsf{cm}, \mathsf{aux}) \;\middle|\; \begin{array}{r} (\theta_1, \ldots, \theta_\nu) \leftarrow \mathcal{O}(\lambda) \\ \mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda) \\ (m_0, m_1, \mathsf{aux}) \leftarrow \mathcal{A}^{(\theta_1,\ldots,\theta_\nu)}(\mathsf{ck}) \\ \omega \leftarrow \{0,1\}^{\mathrm{poly}(\lambda)} \\ \mathsf{cm} := \mathsf{CM.Commit}(\mathsf{ck}, m_1; \omega) \end{array} \right\} .$$

Note that in this definition $\mathsf{CM}$ does not have access to $(\theta_1, \ldots, \theta_\nu)$. The above generalizes the notion of a commitment scheme, which is recovered from the above by setting the oracles to be empty.

Moreover, we say that $\mathsf{CM}$ is **$s$-succinct** if for every commitment key $\mathsf{ck} \in \mathsf{CM.Setup}(1^\lambda)$, message $m \in \{0,1\}^*$, and randomness $\omega$, it holds that $\mathsf{CM.Commit}(\mathsf{ck}, m; \omega) \in \{0,1\}^{s(\lambda)}$.

We have the following simple claim about any binding and hiding commitment scheme.

**Claim 3.7.** *Let* $\mathsf{CM}$ *be a binding and hiding commitment scheme. Then for every message* $m$*,*

$$\Pr_{\omega,\omega'}\left[\mathsf{CM.Commit}(\mathsf{ck}, m, \omega) = \mathsf{CM.Commit}(\mathsf{ck}, m, \omega')\right] = \mathrm{negl}(\lambda) \ .$$

A proof of the above claim appears in Claim 3.4 of [CCS22].

## 3.7 Constraint detection for low-degree polynomials

**Definition 3.8.** *Let* $\vec{d} = (d_1, \ldots, d_m) \in \mathbb{N}^m$*. The low-degree polynomial evaluation code is defined as follows:*

$$\mathrm{LD}[\mathbb{F}, m, \vec{d}] := \left\{ \, c \in (\mathbb{F}^m \to \mathbb{F}) : \exists\, p \in \mathbb{F}^{\leq \vec{d}}[X_1, \ldots, X_m] \text{ s.t. } \forall\, x \in \mathbb{F}^m, c(x) = p(x) \, \right\} \ .$$

*Further, let* $\mathscr{F} = \{\mathbb{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *be a family of fields,* $m \colon \mathbb{N} \to \mathbb{N}$ *an arity function, and* $\vec{d} \colon \mathbb{N} \to \mathbb{N}^m$ *a degree function. We define*

$$\mathrm{LD}[\mathscr{F}, m, \vec{d}] := \left\{ \, \mathrm{LD}[\mathbb{F}_\lambda, m(\lambda), \vec{d}(\lambda)] \, \right\}_{\lambda\in\mathbb{N}} \ .$$

We recall the notion of constraints for linear codes.

**Definition 3.9.** *Let* $\mathcal{C} \subseteq (D \to \mathbb{F})$ *be a linear code. A subset* $Q \subseteq D$ *is* **constrained** *if there exists a nonzero* $z \colon Q \to \mathbb{F}$ *such that, for every* $c \in \mathcal{C}$*,* $\sum_{x\in Q} z(x)c(x) = 0$ *(equivalently, if there exists* $z \neq 0 \in \mathcal{C}^\perp$ *with* $\mathrm{supp}(z) \subseteq Q$*); we refer to* $z$ *as a* **constraint** *on* $Q$*. We say that* $Q$ *is* **unconstrained** *if it is not constrained. We say that* $Q \subseteq D$ **determines** $x \in D$ *if* $x \in Q$ *or there exists a constraint* $z$ *on* $Q \cup \{x\}$ *such that* $z(x) \neq 0$*.*

We recall the definition of a constraint detector [BCFGRS17], which is an algorithm that determines whether a set of queries $Q$ is constrained and, if so, outputs a constraint.

**Definition 3.10.** *Let $\mathcal{C} \subseteq (D \to \mathbb{F})$ be a linear code. An algorithm* $\mathsf{CD}$ *is a* **constraint detector** *for $\mathcal{C}$ if, given as input a set $Q \subseteq D$, outputs: (i) a basis for the space of constraints $\{z\colon Q \to \mathbb{F} : \forall\, c \in \mathcal{C}, \sum_{x\in Q} z(x)c(x) = 0\}$ on $Q$ if $Q$ is constrained; (ii) $\perp$ if $Q$ is unconstrained; A code family $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ has* **efficient constraint detection** *if there exists a polynomial-time algorithm* $\mathsf{CD}$ *such that, for every $\lambda \in \mathbb{N}$, $\mathsf{CD}(1^\lambda, \cdot)$ is a constraint detector for $\mathcal{C}_\lambda$.*

The following theorem is proved in [BCFGRS17]:

**Theorem 3.11.** *The code family $\mathrm{LD}[\mathscr{F}, m, \vec{d}]$ has a constraint detector $\mathsf{CD}(1^\lambda, \cdot)$ that runs in time $\mathrm{poly}(m(\lambda), d(\lambda), \log|\mathbb{F}_\lambda|)$, where $d(\lambda) := \max_{i\in[m]} d(\lambda)_i$. In particular, it has efficient constraint detection.*

## 3.8 Forking lemmas

We state a general forking lemma proved in [BN06].

**Lemma 3.12.** *Fix $t, \lambda \in \mathbb{N}$. Let $\mathcal{A}$ be a probabilistic algorithm that on input $x, y_1, \ldots, y_t$ returns a pair $(I, \sigma)$, where $I \in [t]$ and $\sigma$ is referred to as a side output. Let* $\mathsf{IG}$ *be a probabilistic algorithm that we call the input generator. The accepting probability of $\mathcal{A}$, denoted* acc*, is defined as follows:*

$$\mathrm{acc} := \Pr\left[\; I \geq 1 \;\middle|\; \begin{array}{r} x \leftarrow \mathsf{IG} \\ y_1, \ldots, y_t \leftarrow \mathcal{U}(\{0,1\}^\lambda) \\ (I, \sigma) \leftarrow \mathcal{A}(x, y_1, \ldots, y_t) \end{array} \right] .$$

*The forking algorithm $\mathsf{Fork}_\mathcal{A}$ associated to $\mathcal{A}$ is the probabilistic algorithm that takes input $x$ and proceeds as follows:*

*(i) Pick coins $\rho$ for $\mathcal{A}$ at random.*
*(ii) Sample $y_1, \ldots, y_t \leftarrow \mathcal{U}(\{0,1\}^\lambda)$, and run $\mathcal{A}(x, y_1, \ldots, y_t; \rho)$ to obtain $(I, \sigma)$.*
*(iii) If $I = 0$ then return $(0, \varepsilon, \varepsilon)$.*
*(iv) Otherwise, sample $y'_I, \ldots, y'_t \leftarrow \mathcal{U}(\{0,1\}^\lambda)$ and run $\mathcal{A}(x, y_1, \ldots, y_{I-1}, y'_I, \ldots, y'_t; \rho)$ to obtain $(I', \sigma')$.*
*(v) If $(I = I'$ and $y_I \neq y'_I)$ then return $(1, \sigma, \sigma')$.*
*(vi) Otherwise return $(0, \varepsilon, \varepsilon)$.*

*Let*

$$\mathrm{frk} := \Pr\left[\; b = 1 \;\middle|\; \begin{array}{r} x \leftarrow \mathsf{IG} \\ (b, \sigma, \sigma') \leftarrow \mathsf{Fork}_\mathcal{A}(x) \end{array} \right] .$$

*Then*

$$\mathrm{frk} \geq \mathrm{acc} \cdot \left( \frac{\mathrm{acc}}{t} - \frac{1}{2^\lambda} \right) ,$$

*alternatively,*

$$\mathrm{acc} \leq \frac{t}{2^\lambda} + \sqrt{t \cdot \mathrm{frk}} .$$

## 3.9 Identical-until-bad

We consider two programs, $G$ and $H$, which are written in some pseudocode. We say that $G$ and $H$ are *identical-until-bad* if they are syntactically identical except for statements that follow the setting of a bad flag to true. Somewhat more formally, let $G$ and $H$ be programs written in some pseudocode and let bad be a flag that occurs in both of them. We say that $G$ and $H$ are *identical-until-bad* if their code is the same except

possibly places where $G$ has a statement "set the bad flag" followed by some statements $S_G$ while $H$ has a corresponding statement "set the bad flag" followed by some statements $S_H$, different from $S_G$.

We refer the reader to [BR06] for further details and a full formal treatment of the notion of identical-until-bad, which requires specification of the programming language in question to fully formalize. We stress that that identical-until-bad is a purely syntactic requirement.

We state the fundamental lemma of game-playing, which is proved in [BR06].

**Lemma 3.13.** *Let $G$ and $H$ be identical-until-bad programs and let $\mathcal{A}$ be an adversary. Then*

$$\left|\Pr[\mathcal{A}^G = 1] - \Pr[\mathcal{A}^H = 1]\right| \leq \Pr[\mathcal{A}^G \textit{ sets } \mathsf{bad}] \ .$$

# 4 Arithmetized random oracle model

We define the arithmetized random oracle model. As a first step, we define the arithmetized random oracle *distribution*, which is defined over tuples $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}})$, and explain how the oracles $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}})$ are sampled.

**Definition 4.1.** *Let $m \in \mathbb{N}$ be an arity parameter, $\lambda \in \mathbb{N}$ be a security parameter, $r \in \mathbb{N}$ be a randomness-size parameter, $w \in \mathbb{N}$ be a witness-size parameter, and $d \in \mathbb{N}$ be a degree parameter. For all oracle circuits $B\colon \{0,1\}^{m+r} \to \{0,1\}^w$, we define an* **arithmetized random oracle distribution** $\mathrm{ARO}[\mathbb{F}, m, \lambda, d, B]$,[11] *where $\mathbb{F}$ is a finite field and the support of $\mathrm{ARO}[\mathbb{F}, m, \lambda, d, B]$ contains triples $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}})$ that are sampled as follows:*

1. *Sample the* **random oracle** $\mathsf{ro}$ *uniformly at random from $(\{0,1\}^m \to \{0,1\}^\lambda)$.*
2. *For every $x \in \{0,1\}^m$, sample a random string $\mu_x \in \{0,1\}^r$. Then define the* **witness oracle** $\mathsf{wo}\colon \{0,1\}^m \to \{0,1\}^w$ *as $\mathsf{wo}(x) := B^{\mathsf{ro}}(x, \mu_x)$.*
3. *Define the* **verification function** $\mathsf{vo}\colon \{0,1\}^{m+\lambda+w} \to \{0,1\}$ *as*

$$\mathsf{vo}(x,y,z) := \begin{cases} 1 & \text{if } \mathsf{ro}(x) = y \wedge \mathsf{wo}(x) = z \\ 0 & \text{o.w.} \end{cases} .$$

4. *Sample the* **(extended) verification oracle** $\widehat{\mathsf{vo}}\colon \mathbb{F}^{m+\lambda+w} \to \mathbb{F}$ *uniformly at random from the set*

$$\left\{ p \in \mathbb{F}^{\leq d}[X_1, \ldots, X_{m+\lambda+w}] : p \text{ equals } \mathsf{vo} \text{ on } \{0,1\}^{m+\lambda+w} \right\} .$$

5. *Output $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}})$.*

Next, we define a *family* of ARO distributions, which is parameterized by a family of finite fields $\mathscr{F} = \{\mathbb{F}_\lambda\}_{\lambda\in\mathbb{N}}$ and a family of oracle circuits $\mathcal{B} = \{B_\lambda^{(\cdot)}\colon \{0,1\}^{m(\lambda)} \to \{0,1\}^{w(\lambda)}\}_{\lambda\in\mathbb{N}}$. Here, $\mathcal{B}$ can be interpreted as the set of all possible adversarial strategies for learning information about the random oracle, and $\lambda$ is the security parameter.

**Definition 4.2.** *Let $\mathscr{F} = \{\mathbb{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of fields, $m\colon \mathbb{N} \to \mathbb{N}$ be an arity function, $w\colon \mathbb{N} \to \mathbb{N}$ be a witness-size function, $\mathcal{B} = \{B_\lambda^{(\cdot)}\colon \{0,1\}^{m(\lambda)} \to \{0,1\}^{w(\lambda)}\}_{\lambda\in\mathbb{N}}$ be a family of oracle circuits, and $d\colon \mathbb{N} \to \mathbb{N}$ be a degree function. We define the* **arithmetized random oracle family** *as*

$$\mathrm{ARO}[\mathscr{F}, m, d, \mathcal{B}] := \{\ \mathrm{ARO}[\mathbb{F}_\lambda, m(\lambda), \lambda, d(\lambda), B_\lambda^{(\cdot)}]\ \}_{\lambda\in\mathbb{N}} .$$

The "arithmetized random oracle" is the set of all ARO distributions for polynomial-sized circuit families $\mathcal{B}$.

**Definition 4.3.** *Let $\mathscr{F} = \{\mathbb{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of fields, $m\colon \mathbb{N} \to \mathbb{N}$ be an arity function, $w\colon \mathbb{N} \to \mathbb{N}$ be a witness-size function, and $d\colon \mathbb{N} \to \mathbb{N}$ be a degree function. Then, we define a* **set of arithmetized random oracle families** *as*

$$\mathbf{ARO}[\mathscr{F}, m, d] := \{\mathrm{ARO}[\mathscr{F}, m, d, \mathcal{B}]\colon \mathcal{B} \text{ is a family of } \mathrm{poly}(\lambda)\text{-size oracle circuits}\} ,$$

*where above $\mathcal{B} = \{B_\lambda^{(\cdot)}\colon \{0,1\}^{m(\lambda)} \to \{0,1\}^{w(\lambda)}\}_{\lambda\in\mathbb{N}}$.*

[11] Given $m \in \mathbb{N}$ and the oracle circuit $B$, the randomness length $r$ and witness size $w$ parameters are determined. Thus $r, w$ do not appear in the parameterization of ARO.

# 5 Stateful emulation of the ARO

We define the notion of a stateful emulator for the ARO, present our stateful emulator construction, and then prove its correctness. We start by giving a general definition of stateful emulators for distributions over tuples of oracles, and stating the main result of this section.

**Definition 5.1** (Stateful oracle algorithms)**.** *For a randomized algorithm $\mathcal{M}\colon ((X \rightharpoonup Y) \times X) \to Y$ and oracle algorithm $A$, we denote by $A^{\mathcal{M}}(z)$ the following procedure:*

1. *Initialize $\mathsf{tr}\colon X \rightharpoonup Y$ to be undefined everywhere.*
2. *(Continue to) run $A$ (on input $z$) until it makes an oracle query $x$ or terminates. If $A$ terminates, then stop and return $A$'s output.*
3. *If $A$ has not terminated, first check if $x \in \operatorname{supp}(\mathsf{tr})$; if so, then set $y := \mathsf{tr}(x)$. Otherwise, compute $y \leftarrow \mathcal{M}(\mathsf{tr}, x)$ and add the mapping $x \mapsto y$ to $\mathsf{tr}$.*
4. *Answer $A$'s query with $y$ and go to Step 2.*

*We refer to $\mathcal{M}$ as a* **stateful oracle algorithm**.

Note that $\mathcal{M}$ is not *itself* stateful, but oracle answers are sampled in a stateful way (i.e., by storing prior queries and answers in $\mathsf{tr}$).

**Definition 5.2.** *Let $\mathcal{O}$ be an oracle distribution supported on tuples of oracles $(\theta_1, \ldots, \theta_\nu)$ for some $\nu \in \mathbb{N}$, let $S \subseteq [\nu]$ and let $\varepsilon\colon \mathbb{N} \to [0,1]$. Then, a* **stateful $(\mathcal{O}, S)$-emulator with error** *$\varepsilon$ is a stateful oracle algorithm $\mathcal{M}$ such that for every $t \in \mathbb{N}$ and probabilistic $t$-query adversary $\mathcal{A}$:*

$$\left|\Pr\left[\mathcal{A}^{\theta} = 1 \;\middle|\; \theta \leftarrow \mathcal{O}\right] - \Pr\left[\mathcal{A}^{\mathcal{M}^{\theta_{\bar{S}}}} = 1 \;\middle|\; \theta \leftarrow \mathcal{O}\right]\right| \leq \varepsilon(t) \ .$$

*Moreover, we say that $\mathcal{M}$ is* **pass-through** *if it answers queries to $\theta_i$ for any $i \in \bar{S}$ by querying $\theta_i$ and returning the answer. A* **stateful $\mathcal{O}$-emulator** *is a stateful $(\mathcal{O}, [\nu])$-emulator.*

Prior to stating our main theorem, we first introduce the definition of $\mathsf{vo}_{\mathsf{tr}}$, which is a function encoding the view of $\widehat{\mathsf{vo}}$ that is known, given only the $\mathsf{ro}$ queries in an ARO query-answer transcript.

**Definition 5.3.** *Let $\mathcal{X} := \mathbf{ARO}[\mathscr{F}, m, d]$, let $\mathcal{O} \in \mathcal{X}$, let $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda)$ and let $\mathsf{tr}$ be an $\mathcal{O}(\lambda)$-query-answer transcript. We define $\mathsf{vo}_{\mathsf{tr}}\colon \{0,1\}^{m+\lambda+w} \to \{0,1\}$ as follows:*

$$\mathsf{vo}_{\mathsf{tr}}(x, y, z) := \begin{cases} 1 & \text{if } \mathsf{tr}|_{\mathsf{ro}}(x) = y \text{ and } \mathsf{wo}(x) = z \\ 0 & \text{otherwise} \end{cases} \ .$$

Now, we state the main theorem of this section.

**Theorem 5.4.** *Let $\mathbb{F}$ be a finite field, $m \in \mathbb{N}$ be an arity parameter, $\lambda \in \mathbb{N}$ be a security parameter, $d \in \mathbb{N}$ be a degree parameter with $d \geq 2$, and $B\colon \{0,1\}^{m+r} \to \{0,1\}^w$ be an oracle circuit. Let $\mathcal{O}(\lambda) := \mathrm{ARO}[\mathbb{F}, m, \lambda, d, B]$. Then $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ in Construction 5.11 is a pass-through stateful $(\mathcal{O}(\lambda), \{\widehat{\mathsf{vo}}\})$-emulator with error $\frac{t}{2^\lambda}$. In fact, for every $t_{\widehat{\mathsf{vo}}} \in \mathbb{N}$ and probabilistic adversary $\mathcal{A}$ that makes at most $t_{\widehat{\mathsf{vo}}}$ queries to $\widehat{\mathsf{vo}}$,*

$$\left|\Pr\left[\mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})} = 1 \;\middle|\; (\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda)\right] - \Pr\left[\mathcal{A}^{\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}} = 1 \;\middle|\; (\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda)\right]\right| \leq \frac{t_{\widehat{\mathsf{vo}}}}{2^\lambda} \ .$$

*$\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ additionally satisfies the following properties:*

- ***Output distribution.*** *For every $\mathcal{O}$-query-answer transcript $\mathsf{tr}$, $x \in \{0,1\}^{m+\lambda+w}$ and $y \in \mathbb{F}$ we have that*

$$\Pr_{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})\leftarrow\mathcal{O}}\left[\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}(\mathsf{tr},(\widehat{\mathsf{vo}},x))=(\mathsf{tr}',y)\right]=\Pr\left[P(x)=y \mid P\leftarrow\mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}}\cup\mathsf{tr}|_{\widehat{\mathsf{vo}}}]\right)\right] \ ,$$

  *where $\mathsf{tr}' := \mathsf{tr} \cup \{(\widehat{\mathsf{vo}}, x, y)\}$ and $\mathsf{vo}_{\mathsf{tr}}$ is defined as in Definition 5.3.*

- ***Efficiency.*** *$\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ runs in time $\mathrm{poly}(m, \lambda, w, d, t, \log|\mathbb{F}|)$. For each oracle query $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ emulates, $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ makes at most 1 oracle query.*

We often write $\varepsilon_{\mathrm{ARO}}(t,\lambda) := \frac{t}{2^\lambda}$ to refer to the above error. We will abuse notation and refer to such an emulator as a pass-through stateful (ARO, $\widehat{\mathsf{vo}}$)-emulator. Throughout this section, we assume without loss of generality that any adversary that queries $\mathsf{ro}$ on input $x$ also queries $\mathsf{wo}$ with $x$. The proof that $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ has *emulation error* at most $\frac{t}{2^\lambda}$ proceeds via a sequence of hybrids.

The remainder of this section is broken up as follows.

- In Section 5.1, we construct an inefficient perfect emulator $\mathcal{M}_{\mathsf{LD}}$ for low-degree extensions of any function defined over the boolean hypercube.
- In Section 5.2, we construct an *efficient* pass-through stateful $(\mathrm{ARO}, \widehat{\mathsf{vo}})$-emulator $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$.
- In Section 5.3 we prove Theorem 5.4. In order to do so, we use $\mathcal{M}_{\mathsf{LD}}$ to obtain an inefficient perfect emulator for the ARO, $\mathcal{M}_{\mathrm{ARO}^*}^{(\mathsf{ro},\mathsf{wo})}$, and prove it is statistically close to $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$.
- In Section 5.4, we give efficient implementations of certain subroutines used by $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$.

## 5.1 Inefficient stateful emulator for low-degree extensions

We describe a stateful emulator algorithm for low-degree extensions and prove that the emulation is perfect. Our analysis uses tools from algebraic query complexity; specifically the proof of [AW09, Lemma 4.5] inspired the "shift" polynomial used in our analysis.

**Lemma 5.5.** *Let $n, d \in \mathbb{N}$ with $d \geq 2$. Let $\mathcal{M}_{\mathsf{LD}}$ be the stateful oracle algorithm in Construction 5.9. Then for every $f\colon \{0,1\}^n \to \mathbb{F}$ and adversary $\mathcal{A}$:*

$$\Pr\left[\mathcal{A}^{\hat{f}}=1 \;\middle|\; \hat{f}\leftarrow\mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[f]\right)\right]=\Pr\left[\mathcal{A}^{\mathcal{M}_{\mathsf{LD}}^{f}}=1\right] \ ;$$

*i.e., $\mathcal{M}_{\mathsf{LD}}^{f}$ is a stateful $\mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[f]\right)$-emulator.*

Before giving our construction of a stateful emulator for low-degree extensions, we define some notions required for its description.

**Definition 5.6.** *Let $f\colon \{0,1\}^n \to \mathbb{F}$ and let $\mathsf{tr} \in (\mathbb{F}^n \times \mathbb{F})^t$ be an $\mathsf{LDE}_{\mathbb{F},d}[f]$-query-answer transcript. Then we define $(f \cup \mathsf{tr})\colon \{0,1\}^n \cup \mathrm{supp}(\mathsf{tr}) \to \mathbb{F}$ by*

$$(f\cup\mathsf{tr})(x) := \begin{cases} f(x) & \text{if } x\in\{0,1\}^n \\ \mathsf{tr}(x) & \text{otherwise} \end{cases} \ .$$

**Definition 5.7.** *Given a point $w \in \{0,1\}^n$ and a query set $\mathsf{X} \in (\mathbb{F}^n)^t$, let $\mathbf{Q}_{w,\mathsf{X}}$ be the set containing polynomials $Q_{w,\mathsf{X}} \in \mathbb{F}^{\leq 2}[X_1,\ldots,X_n]$ such that: (i) $Q_{w,\mathsf{X}}(w) = 1$; (ii) $Q_{w,\mathsf{X}}(x) = 0$ for every $x \in \{0,1\}^n$ such that $x \neq w$; and (iii) $Q_{w,\mathsf{X}}(x) = 0$ if $x \in \mathsf{X}$.*

**Definition 5.8.** *Given a query set* $\mathsf{X} \in (\mathbb{F}^n)^t$*, define the set*

$$\mathsf{BAD}_{\mathsf{X}} := \{w \in \{0,1\}^n \mid \mathbf{Q}_{w,\mathsf{X}} \textit{ is empty}\} \quad .$$

**Construction 5.9.** For a given $f\colon \{0,1\}^n \to \{0,1\}$, the emulator $\mathcal{M}^f_{\mathsf{LD}}$ takes input $(n, d, \mathsf{tr}, x)$ and works as follows.

1. Set $P(\vec{X}) := \sum_{b \in \mathsf{BAD}_{\mathrm{supp}(\mathsf{tr}) \cup \{x\}}} f(b) \cdot \delta_b(\vec{X})$.
2. Define $\mathsf{tr}_s := \{(x, y - P(x)) : (x, y) \in \mathsf{tr}\}$.
3. Sample $\hat{Z} \leftarrow \mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_s]$, where $Z\colon \{0,1\}^n \to \{0,1\}$ is the zero function.
4. Output $y := \hat{Z}(x) + P(x)$.

Before proving Lemma 5.5, we first prove a useful claim.

**Claim 5.10.** *Let* $Z\colon \{0,1\}^n \to \{0,1\}$ *be the zero function and let* $h\colon \{0,1\}^n \to \mathbb{F}$ *be arbitrary. Let* $\mathsf{tr}_{\hat{Z}}, \mathsf{tr}_{\hat{h}} \in (\mathbb{F}^n \times \mathbb{F})^t$ *be a* $\mathsf{LDE}_{\mathbb{F},d}[Z]$*-query-answer transcript and a* $\mathsf{LDE}_{\mathbb{F},d}[h]$*-query-answer transcript respectively, such that* $\mathrm{supp}(\mathsf{tr}_{\hat{Z}}) = \mathrm{supp}(\mathsf{tr}_{\hat{h}})$*. Let* $\hat{Z} \leftarrow \mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_{\hat{Z}}]\right)$*, and let* $\hat{h} \in \mathsf{LDE}_{\mathbb{F},d}[h \cup \mathsf{tr}_{\hat{h}}]$*. Then* $\hat{Z} + \hat{h}$ *is distributed as* $\mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[h \cup \mathsf{tr}_{\hat{Z}+\hat{h}}]\right)$*, where* $\mathsf{tr}_{\hat{Z}+\hat{h}} := \{(x, y_{\hat{Z}} + y_{\hat{h}}) : (x, y_{\hat{Z}}) \in \mathsf{tr}_{\hat{Z}}, (x, y_{\hat{h}}) \in \mathsf{tr}_{\hat{h}}\}$.

*Proof.* Consider the bijective affine map $T_{\hat{h}}$ on $\mathbb{F}^{\leq d}[X_1, \ldots, X_n]$ defined by $T_{\hat{h}}(\hat{Z}) := \hat{Z} + \hat{h}$. We show that $T_{\hat{h}}(\mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_{\hat{Z}}]) = \mathsf{LDE}_{\mathbb{F},d}[h \cup \mathsf{tr}_{\hat{Z}+\hat{h}}]$. For any $x \in \mathrm{supp}(\mathsf{tr}_{\hat{Z}})$ we have that $(\hat{Z}+\hat{h})(x) = y_{\hat{Z}} + y_{\hat{h}}$, where $(x, y_{\hat{Z}}) \in \mathsf{tr}_{\hat{Z}}$ and $(x, y_{\hat{h}}) \in \mathsf{tr}_{\hat{h}}$. Thus, for $(x, y_{\hat{Z}+\hat{h}}) \in \mathsf{tr}_{\hat{Z}+\hat{h}}$, we have $(\hat{Z} + \hat{h})(x) = y$. Further, for any $x \in \{0,1\}^n$, $(\hat{Z}+\hat{h})(x) = \hat{h}(x) = h(x)$. Hence, for any $\hat{Z} \in \mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_{\hat{Z}}]$, $T_{\hat{h}}(\hat{Z}) \in \mathsf{LDE}_{\mathbb{F},d}[h \cup \mathsf{tr}_{\hat{Z}+\hat{h}}]$.

As $Z$ is uniformly random in $\mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_{\hat{Z}}]$ and $T_{\hat{h}}$ is a bijection between $\mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_{\hat{Z}}]$ and $\mathsf{LDE}_{\mathbb{F},d}[h \cup \mathsf{tr}_{\hat{Z}+\hat{h}}]$, we have that $T_{\hat{h}}(\hat{Z}) = \hat{Z} + \hat{h}$ is uniformly random in $\mathsf{LDE}_{\mathbb{F},d}[h \cup \mathsf{tr}_{\hat{Z}+\hat{h}}]$. □

*Proof of Lemma 5.5.* We show that for all $\mathsf{LDE}_{\mathbb{F},d}[f]$-query-answer transcripts, $\mathsf{tr} = [(x_i, y_i)]_{i=1}^t \in (\mathbb{F}^n \times \mathbb{F})^t$, $x \in \mathbb{F}^n$ and $y \in \mathbb{F}$, we have that

$$\Pr\left[\hat{f}(x) = y \;\middle|\; \hat{f} \leftarrow \mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[f \cup \mathsf{tr}]\right)\right] = \Pr\left[\mathcal{M}^f_{\mathsf{LD}}(\mathsf{tr}, x) = y\right] \quad . \tag{4}$$

We will obtain Eq. 4 in three steps.

1. We show that the set $\mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_s]$ is non-empty provided that $\mathsf{LDE}_{\mathbb{F},d}[f \cup \mathsf{tr}]$ is non-empty, which ensures that Step 3 of Construction 5.9 does not fail.
2. Note that the output of $\mathcal{M}_{\mathsf{LD}}$ is determined by the polynomial $M(\vec{X}) := \hat{Z}(\vec{X}) + P(\vec{X})$, which is computed in Step 4 of Construction 5.9. We show that $M(\vec{X})$ is distributed like $\mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[g \cup \mathsf{tr}]\right)$, for a function $g\colon \{0,1\}^n \to \mathbb{F}$ defined below.
3. We demonstrate that evaluations of polynomials distributed like $\mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[g \cup \mathsf{tr}]\right)$ are perfectly indistinguishable from evaluations of polynomials distributed like $\mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[f \cup \mathsf{tr}]\right)$.

**Step 1.** We show that $\mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_s]$ is non-empty if $\mathsf{LDE}_{\mathbb{F},d}[f \cup \mathsf{tr}]$ is non-empty. As in Step 1 of Construction 5.9, let $P(\vec{X}) := \sum_{b \in \mathsf{BAD}_{\mathrm{supp}(\mathsf{tr}) \cup \{x\}}} f(b) \cdot \delta_b(\vec{X})$. Then we have that $\hat{f}(\vec{X}) - P(\vec{X}) \in \mathsf{LDE}_{\mathbb{F},d}[h \cup \mathsf{tr}_s]$, where $h\colon \{0,1\}^n \to \mathbb{F}$ is defined by

$$h(x) = \begin{cases} 0 & \text{if } x \in \mathsf{BAD}_{\mathrm{supp}(\mathsf{tr}) \cup \{x\}} \\ f(x) & \text{otherwise} \end{cases} \quad ,$$

and $\mathsf{tr}_s$ is as defined in Step 2 of Construction 5.9. By definition of $\mathsf{BAD}_{\mathrm{supp}(\mathsf{tr})\cup\{x\}}$, for each $w \in \{0,1\}^n \setminus \mathsf{BAD}_{\mathrm{supp}(\mathsf{tr})\cup\{x\}}$, the set $\mathbf{Q}_{w,\mathrm{supp}(\mathsf{tr})\cup\{x\}}$ is non-empty; for each $w$, choose an arbitrary element $Q_w \in \mathsf{BAD}_{\mathrm{supp}(\mathsf{tr})\cup\{x\}}$. Let $S_1(\vec{X}) := \sum_{w\in\{0,1\}^n\setminus\mathsf{BAD}_{\mathrm{supp}(\mathsf{tr})\cup\{x\}}} f(w)\cdot Q_w(\vec{X})$. It is easily verified that $S_1$ has the following properties: (i) $S_1(x) = 0$ for all $x \in \mathrm{supp}(\mathsf{tr})$; (ii) $S_1(w) = f(w)$ for all $w \in \{0,1\}^n \setminus \mathsf{BAD}_{\mathrm{supp}(\mathsf{tr})\cup\{x\}}$, and (iii) $S_1(b) = 0$ for all $b \in \mathsf{BAD}_{\mathrm{supp}(\mathsf{tr})\cup\{x\}}$. Thus, the polynomial $\hat{f}(\vec{X}) - P(\vec{X}) - S_1(\vec{X}) \in \mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_s]$, as desired.

**Step 2.** Define

$$g(x) = \begin{cases} f(x) & \text{if } x \in \mathsf{BAD}_{\mathrm{supp}(\mathsf{tr})\cup\{x\}} \\ 0 & \text{otherwise.} \end{cases}$$

We show that the distribution of $M(\vec{X})$ is $\mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[g \cup \mathsf{tr}]\right)$.

By definition, $P(\vec{X}) \in \mathsf{LDE}_{\mathbb{F},1}[g \cup \mathsf{tr}_P]$, where $\mathsf{tr}_P$ is defined by $\mathsf{tr}_P = \{(x, P(x)) : x \in \mathrm{supp}(\mathsf{tr})\}$. Thus, by Claim 5.10, as $M(\vec{X}) = \hat{Z}(\vec{X}) + P(\vec{X})$ and $\hat{Z}(\vec{X}) \leftarrow \mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_s]$, we have that $M(\vec{X}) \leftarrow \mathsf{LDE}_{\mathbb{F},d}[g \cup \mathsf{tr}_{s+P}]$. However, $\mathsf{tr}_{s+P} = \mathsf{tr}$, so $M(\vec{X})$ is distributed like $\mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[g \cup \mathsf{tr}]\right)$.

In particular,

$$\Pr\left[\mathcal{M}^f_{\mathsf{LD}}(\mathsf{tr}, x) = y\right] = \Pr\left[\hat{g}(x) = y \mid \hat{g} \leftarrow \mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[g \cup \mathsf{tr}]\right)\right] \quad .$$

**Step 3.** Consider another "shift" polynomial $S_2(\vec{X}) = \sum_{w\in\{0,1\}^n\setminus\mathsf{BAD}_{\mathrm{supp}(\mathsf{tr})\cup\{x\}}} (f(w) - g(w))Q_w(\vec{X})$. $S_2$ has the property that if $\hat{g} \in \mathsf{LDE}_{\mathbb{F},d}[g \cup \mathsf{tr}]$, then $\hat{g} + S_2 \in \mathsf{LDE}_{\mathbb{F},d}[f \cup \mathsf{tr}]$. Further, $S_2(x) = 0$. Hence the map $\hat{g} \to \hat{g} + S_2$ is a bijection between the sets

$$\left\{\hat{f} \in \mathsf{LDE}_{\mathbb{F},d}[f \cup \mathsf{tr}] : \hat{f}(x) = y\right\}$$

and

$$\{\hat{g} \in \mathsf{LDE}_{\mathbb{F},d}[g \cup \mathsf{tr}] : \hat{g}(x) = y\} \quad .$$

Thus, we have that

$$\Pr\left[\hat{f}(x) = y \;\middle|\; \hat{f} \leftarrow \mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[f \cup \mathsf{tr}]\right)\right] = \Pr\left[\hat{g}(x) = y \mid \hat{g} \leftarrow \mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[g \cup \mathsf{tr}]\right)\right] \quad ,$$

which yields the result. □

## 5.2 Stateful emulator for the ARO

We present the stateful emulator algorithm for the ARO.

For the purpose of making an identical-until-bad argument (see Section 3.9) in our analysis, we include the setting of a bad flag in this construction. This is not required for the emulator to function.

**Construction 5.11.** We define a pass-through stateful $(\mathrm{ARO}[\mathbb{F}, m, \lambda, d, B], \{\widehat{\mathsf{vo}}\})$-emulator as follows:

- *Query* $\mathcal{M}^{(\mathsf{ro},\mathsf{wo})}_{\mathrm{ARO}}$ *with parameters* $(\mathbb{F}, m, \lambda, w, d)$ *and with input* $(\mathsf{tr}, (\mathsf{oid}, x))$.
  - If $\mathsf{oid} = \mathsf{ro}$, output $y := \mathsf{ro}(x)$.
  - If $\mathsf{oid} = \mathsf{wo}$, output $y := \mathsf{wo}(x)$.
  - If $\mathsf{oid} = \widehat{\mathsf{vo}}$:

1. Compute the set $V := \{(x, y, z) : (x, y) \in \mathsf{tr}|_{\mathsf{ro}} \wedge (x, z) \in \mathsf{tr}|_{\mathsf{wo}}\}$.
2. Set $P(\vec{X}) := \sum_{b \in V} \delta_b(\vec{X})$.
3. If the set $B := \{b \in \mathsf{BAD}_{\mathrm{supp}(\mathsf{tr}|_{\widehat{\mathsf{vo}}}) \cup \{x\}} \setminus V : \mathsf{vo}(b) \neq 0\}$ is non-empty, set the bad flag.
4. For each $x \in \mathrm{supp}(\mathsf{tr}|_{\widehat{\mathsf{vo}}})$, compute $P(x)$ and set $\mathsf{tr}_s := \{(x, y - P(x)) : (x, y) \in \mathsf{tr}|_{\widehat{\mathsf{vo}}}\}$.
5. Sample $\hat{Z} \leftarrow \mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_s]$, where $Z \colon \{0,1\}^{m+\lambda+w} \to \{0,1\}$ is the zero function.
6. Output $y := \hat{Z}(x) + P(x)$.

**Remark 5.12.** We obtain an efficient implementation of Construction 5.11 by using the $\mathsf{ZSample}_{\mathbb{F},m+\lambda+w,d}$ algorithm (Construction 5.20) in Step 5 and Step 6.

## 5.3 Proof of Theorem 5.4

It is clear that the answers to ro and wo queries are indistinguishable. Hence, we focus on proving the indistinguishability of answers to $\widehat{\mathsf{vo}}$ queries.

We use a hybrid argument with the following hybrids.

- $\mathbf{H}_0$: $\mathcal{A}$ queries $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}})$.
- $\mathbf{H}_1$: $\mathcal{A}$ queries $(\mathsf{ro}, \mathsf{wo}, \mathcal{M}^{\mathsf{vo}}_{\mathsf{LD}})$, the emulator described in Construction 5.9.
- $\mathbf{H}_2$: $\mathcal{A}$ queries $\mathcal{M}^{(\mathsf{ro},\mathsf{wo})}_{\mathrm{ARO}^*}$, the emulator described in Construction 5.13, below.
- $\mathbf{H}_3$: $\mathcal{A}$ queries $\mathcal{M}^{(\mathsf{ro},\mathsf{wo})}_{\mathrm{ARO}}$, the emulator described in Construction 5.11.

For every $\mathcal{O}$-query-answer transcript $\mathsf{tr}$, let $\mathsf{tr}_{<\ell}$ denote the queries in $\mathsf{tr}$ up to and *excluding* the $\ell$-th query, i.e. $\mathsf{tr}_{<\ell} := [(\mathsf{oid}_i, x_i, y_i)]_{i=1}^{\ell-1}$.

**Construction 5.13.** We specify an inefficient hybrid stateful emulator for the ARO, which runs based on the stateful emulator for random low-degree extension (Construction 5.9).

- *Query* $\mathcal{M}^{(\mathsf{ro},\mathsf{wo})}_{\mathrm{ARO}^*}$ *with parameters* $(\mathbb{F}, m, \lambda, w, d)$ *and with input* $(\mathsf{tr}, (\mathsf{oid}, x))$.
  - If $\mathsf{oid} = \mathsf{ro}$, output $y := \mathsf{ro}(x)$.
  - If $\mathsf{oid} = \mathsf{wo}$, output $y := \mathsf{wo}(x)$.
  - If $\mathsf{oid} = \widehat{\mathsf{vo}}$:
    1. Compute the set $V := \{(x, y, z) : (x, y) \in \mathsf{tr}|_{\mathsf{ro}} \wedge (x, z) \in \mathsf{tr}|_{\mathsf{wo}}\}$.
    2. Set $P(\vec{X}) := \sum_{b \in V} \delta_b(\vec{X})$.
    3. If the set $B := \{b \in \mathsf{BAD}_{\mathrm{supp}(\mathsf{tr}|_{\widehat{\mathsf{vo}}}) \cup \{x\}} \setminus V : \mathsf{vo}(b) \neq 0\}$ is non-empty, set the bad flag and update:
    $$P(\vec{X}) := P(\vec{X}) + \sum_{b \in B} \mathsf{vo}(b) \cdot \delta_b(\vec{X}).$$
    4. For each $x \in \mathrm{supp}(\mathsf{tr}|_{\widehat{\mathsf{vo}}})$, compute $P(x)$ and set $\mathsf{tr}_s := \{(x, y - P(x)) : (x, y) \in \mathsf{tr}|_{\widehat{\mathsf{vo}}}\}$.
    5. Sample $\hat{Z} \leftarrow \mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_s]$, where $Z \colon \{0,1\}^{m+\lambda+w} \to \{0,1\}$ is the zero function.
    6. Output $y := \hat{Z}(x) + P(x)$.

**$\mathbf{H_0}$ vs. $\mathbf{H_1}$.** $\mathbf{H}_0$ and $\mathbf{H}_1$ are perfectly indistinguishable by Lemma 5.5.

**$\mathbf{H_1}$ vs. $\mathbf{H_2}$.** We show that $\mathbf{H}_1$ and $\mathbf{H}_2$ are perfectly indistinguishable. We do this by showing that for all $\mathcal{O}$-query-answer transcripts $\mathsf{tr}$, $x \in \mathbb{F}^m$ and $y \in \mathbb{F}$ we have

$$\Pr\left[\mathcal{M}_{\mathsf{LD}}^{\mathsf{vo}}(\mathsf{tr}|_{\widehat{\mathsf{vo}}}, x) = y\right] = \Pr\left[\mathcal{M}_{\mathrm{ARO}^*}^{(\mathsf{ro},\mathsf{wo})}(\mathsf{tr}, x) = y\right] \ .$$

In both experiments $y$ is sampled as $\hat{Z}(x)+P(x)$ where $\hat{Z} \leftarrow \mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_s]$, but where $P$ is constructed differently in each experiment. We show that $P$ is in fact the same polynomial in both experiments, from which the claim follows. In Construction 5.13, we have $P(\vec{X}) := \sum_{b\in V} \delta_b(\vec{X}) + \sum_{b\in B} \mathsf{vo}(b)\delta_b(\vec{X})$. But $\mathsf{vo}(b) = 1$ for all $b \in V$. Thus $P(\vec{X}) = \sum_{b\in B\cup V} \mathsf{vo}(b)\delta_b = \sum_{b\in \mathsf{BAD}_{\mathrm{supp}(\mathsf{tr}|_{\widehat{\mathsf{vo}}})\cup\{x\}}} \mathsf{vo}(b)\delta_b(\vec{X})$, which is precisely how $P(\vec{X})$ is constructed by $\mathcal{M}_{\mathsf{LD}}^{\mathsf{vo}}$. (Note that $\mathsf{tr}$ in $\mathcal{M}_{\mathsf{LD}}^{\mathsf{vo}}$ is an $\mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}]$-query-answer transcript.)

**$\mathbf{H_2}$ vs. $\mathbf{H_3}$.** We show that $t$-query adversary $\mathcal{A}$ distinguishes between $\mathbf{H}_2$ and $\mathbf{H}_3$ with probability $\varepsilon \le \frac{t}{2^\lambda}$. We do this by showing that $\mathcal{M}_{\mathrm{ARO}^*}^{(\mathsf{ro},\mathsf{wo})}$ and $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ are identical-until-bad (Section 3.9). Then we argue that the probability that the bad flag is set in either construction is at most $\frac{t_{\widehat{\mathsf{vo}}}}{2^\lambda}$, where $t_{\widehat{\mathsf{vo}}} \le t$ is the total number of queries made to the verification oracle.

**Claim 5.14.** *$\mathcal{M}_{\mathrm{ARO}^*}^{(\mathsf{ro},\mathsf{wo})}$ and $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ are identical-until-bad.*

*Proof.* $\mathcal{M}_{\mathrm{ARO}^*}^{(\mathsf{ro},\mathsf{wo})}$ and $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ are syntactically identical until the bad flag is set. □

Since $\mathcal{M}_{\mathrm{ARO}^*}^{(\mathsf{ro},\mathsf{wo})}$ and $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ are identical-until-bad, the fundamental lemma of game playing (Lemma 3.13) states that

$$\begin{aligned}&\left|\Pr\left[\mathcal{A}^{\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}} = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{M}_{\mathrm{ARO}^*}^{(\mathsf{ro},\mathsf{wo})}} = 1\right]\right| \\ &\le \Pr\left[\text{the bad flag is raised by } \mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}\right] \ .\end{aligned}$$

Next we bound the probability that the bad flag is raised within $t$ queries. First observe that the bad flag is only ever raised when $\mathsf{oid} = \widehat{\mathsf{vo}}$.

Let $E_i$ be the event that the bad flag is raised on the $i$-th query to the verification oracle, and let $B_i$ be the set defined in Step 3 of Construction 5.13, on the $i$-th query. Further, define $\mathsf{BAD}_1 := B_1$ and $\mathsf{BAD}_i := B_i \setminus B_{i-1}$ for $i \in \{2, \dots t_{\widehat{\mathsf{vo}}}\}$. Then by a union bound over the points in $\mathsf{BAD}_i$ we have

$$\begin{aligned}\Pr[E_i] &\le \sum_{(x_b,y_b,z_b)\in\mathsf{BAD}_i} \Pr\left[\mathsf{ro}(x_b) = y_b \mid (\mathsf{ro}, x_b, \mathsf{ro}(x_b)) \notin \mathsf{tr}_{<i}\right] \\ &\le \frac{|\mathsf{BAD}_i|}{2^\lambda} \ ,\end{aligned}$$

where we use the fact that for every $b := (x_b, y_b, z_b) \in \mathsf{BAD}_i$, we have that

$$\Pr\left[\mathsf{ro}(x_b) = y_b \mid (\mathsf{ro}, x_b) \notin \mathrm{supp}(\mathsf{tr}_{<i})\right] = \frac{1}{2^\lambda} \ .$$

To conclude, we upper bound $\mathcal{A}$'s overall advantage over $t_{\widehat{\mathsf{vo}}}$ queries to $\widehat{\mathsf{vo}}$.

$$\Pr\left[\bigcup_{i\in[t_{\widehat{\mathsf{vo}}}]} E_i\right] = \sum_{s_1,\dots,s_{t_{\widehat{\mathsf{vo}}}}\in\mathbb{N}} \Pr\left[\bigcup_{i\in[t_{\widehat{\mathsf{vo}}}]} E_i \;\middle|\; |\mathsf{BAD}_j| = s_j \ \forall j \in [t_{\widehat{\mathsf{vo}}}]\right] \Pr\left[|\mathsf{BAD}_j| = s_j \ \forall j \in [t_{\widehat{\mathsf{vo}}}]\right]$$

$$= \sum_{\substack{s_1,\ldots,s_{t_{\widehat{\mathsf{vo}}}} \in \mathbb{N} \\ \sum_{k\in[t_{\widehat{\mathsf{vo}}}]} s_k \le t_{\widehat{\mathsf{vo}}}}} \Pr\left[\bigcup_{i\in[t_{\widehat{\mathsf{vo}}}]} E_i \;\middle|\; |\mathsf{BAD}_j| = s_j \;\forall j \in [t_{\widehat{\mathsf{vo}}}]\right] \Pr\left[|\mathsf{BAD}_j| = s_j \;\forall j \in [t_{\widehat{\mathsf{vo}}}]\right]$$

$$\le \sum_{\substack{s_1,\ldots,s_{t_{\widehat{\mathsf{vo}}}} \in \mathbb{N} \\ \sum_{k\in[t_{\widehat{\mathsf{vo}}}]} s_k \le t_{\widehat{\mathsf{vo}}}}} \left(\sum_{i\in[t_{\widehat{\mathsf{vo}}}]} \Pr\left[E_i \mid |\mathsf{BAD}_j| = s_j \;\forall j \in [t_{\widehat{\mathsf{vo}}}]\right]\right) \Pr\left[|\mathsf{BAD}_j| = s_j \;\forall j \in [t_{\widehat{\mathsf{vo}}}]\right]$$

$$\le \sum_{\substack{s_1,\ldots,s_{t_{\widehat{\mathsf{vo}}}} \in \mathbb{N} \\ \sum_{k\in[t_{\widehat{\mathsf{vo}}}]} s_k \le t_{\widehat{\mathsf{vo}}}}} \left(\sum_{i\in[t_{\widehat{\mathsf{vo}}}]} \frac{s_i}{2^\lambda}\right) \Pr\left[|\mathsf{BAD}_j| = s_j \;\forall j \in [t_{\widehat{\mathsf{vo}}}]\right] \le \frac{t_{\widehat{\mathsf{vo}}}}{2^\lambda} \ .$$

The equality in the second line follows from [AW09, Lemma 4.3], which states that $\sum_{i\in[t_{\widehat{\mathsf{vo}}}]} |\mathsf{BAD}_i| = |\mathsf{BAD}_{\mathrm{supp}(\mathsf{tr}|_{\widehat{\mathsf{vo}}})}| \le t_{\widehat{\mathsf{vo}}}$.

**Output distribution.** Finally, we show that for every $\mathcal{O}$-query-answer transcript $\mathsf{tr}$, $x \in \{0,1\}^{m+\lambda+w}$ and $y \in \mathbb{F}$ we have that

$$\Pr\left[P(x) = y \mid P \leftarrow \mathcal{U}\left(\mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}|_{\widehat{\mathsf{vo}}}]\right)\right] = \Pr_{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})\leftarrow\mathcal{O}}\left[\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}(\mathsf{tr},(\widehat{\mathsf{vo}},x)) = (\mathsf{tr}',y)\right] \ .$$

The output of Construction 5.11 is determined by the polynomial $M(\vec{X}) := \hat{Z}(\vec{X}) + P(\vec{X})$. We show that the distribution of $M(\vec{X})$ is $\mathcal{U}(\mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}|_{\widehat{\mathsf{vo}}}])$.

By definition, $P(\vec{X}) \in \mathsf{LDE}_{\mathbb{F},1}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}_P]$, where $\mathsf{tr}_P := \{(x, P(x) : x \in \mathrm{supp}(\mathsf{tr})\}$. Also, $\hat{Z}(\vec{X}) \leftarrow \mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_s]$, where $\mathsf{tr}_s := \{(x, y - P(x)) : (x,y) \in \mathsf{tr}\}$. Thus by Claim 5.10, $M(\vec{X}) \leftarrow \mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}_{P+S}]$. But $\mathsf{tr}_{P+S} = \mathsf{tr}$, which yields the result.

**Efficiency.** Running $\mathcal{M}_{\mathrm{ARO}^*}^{(\mathsf{ro},\mathsf{wo})}$ requires computing $P(\vec{X})$ at all of the points in the set $\mathrm{supp}(\mathsf{tr}|_{\widehat{\mathsf{vo}}})$ which takes time $O(t)$ and running $\mathsf{ZSample}_{\mathbb{F},d}$ which takes time $\mathrm{poly}(m,\lambda,w,d,\log(|\mathbb{F}|),t)$ by Lemma 5.21. Thus $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ runs in time $\mathrm{poly}(m,\lambda,w,d,\log(|\mathbb{F}|),t)$. Finally, $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ makes at most one query to $(\mathsf{ro},\mathsf{wo})$ for every query it receives, so its query complexity is $O(t)$.

## 5.4 Efficiently implementing Construction 5.11

We describe the subroutines used to efficiently implement the ARO stateful emulator in Construction 5.11.

### 5.4.1 Efficiently sampling a random low-degree polynomial

We describe $\mathsf{LDSample}_{\mathbb{F},m,\vec{d}}$, a stateful oracle that samples evaluations of a random low-degree polynomial in $\mathbb{F}^{\le\vec{d}}[X_1,\ldots,X_m]$.

**Lemma 5.15.** *Let $\mathbb{F}$ be a field, let $m,t \in \mathbb{N}$, $\vec{d} \in \mathbb{N}^m$ and let $\mathsf{LDSample}_{\mathbb{F},m,\vec{d}}$ be the algorithm described in Construction 5.16. Then for all $\mathsf{tr} \in (\mathbb{F}^m \times \mathbb{F})^t$, for which $\mathsf{LDE}_{\mathbb{F},\vec{d}}[\mathsf{tr}] \neq \emptyset$, and for all $x \in \mathbb{F}^m, y \in \mathbb{F}$:*

$$\Pr\left[\mathsf{LDSample}_{\mathbb{F},m,\vec{d}}(\mathsf{tr},x) = (\mathsf{tr}',y)\right] = \Pr\left[P(x) = y \;\middle|\; P(\vec{X}) \leftarrow \mathsf{LDE}_{\mathbb{F},\vec{d}}[\mathsf{tr}]\right] \ ,$$

*where $\mathsf{tr}' := \mathsf{tr} \cup \{(x,y)\}$. Moreover, $\mathsf{LDSample}_{\mathbb{F},m,\vec{d}}$ runs in time $\mathrm{poly}(m,d,\log|\mathbb{F}|,t)$, where $t = |\mathsf{tr}|$.*

**Construction 5.16.** Given a constraint detector $\mathsf{CD}_{\mathbb{F},m,\vec{d}}$ for $\mathrm{LD}[\mathbb{F}, m, \vec{d}]$ (Definition 3.10) the algorithm $\mathsf{LDSample}_{\mathbb{F},m,\vec{d}}$ operates as follows.

- *Evaluate polynomial:* $\mathsf{LDSample}_{\mathbb{F},m,\vec{d}}(\mathsf{tr}, x) \to y$.
  1. Run $\mathsf{CD}_{\mathbb{F},m,\vec{d}}(\mathrm{supp}(\mathsf{tr}) \cup \{x\})$:
     (a) If $\mathsf{CD}_{\mathbb{F},m,\vec{d}}$ outputs a constraint $z$ for which $z(x) \neq 0$, compute $y$ such that $(x, y)$ that is consistent with $z$; i.e., $y := -\frac{1}{z(x)} \sum_{(x',y') \in \mathsf{tr}} z(x')y'$.
     (b) If $\mathsf{CD}_{\mathbb{F},m,\vec{d}}$ outputs $\perp$ or a basis $z_1, \ldots, z_k$ where $z(x) = 0$ for all $i \in [k]$, sample $y \leftarrow \mathbb{F}$.
  2. Output $y$.

*Proof of Lemma 5.15.* Follows from Lemma 4.3 of [BCFGRS17]. □

*Proof.* This algorithm appears in appendix B of [BCFGRS17], and its correctness and efficiency are argued in Lemma 4.3 of [BCGRS17]. □

### 5.4.2 Efficiently sampling a RLDE of the boolean zero function

We first give an inefficient subroutine (Construction 5.18) that samples evaluations of a uniformly random low-degree extension of the boolean zero function conditioned on a set of preprogrammed points in Lemma 5.17 and prove its correctness. Subsequently, in Construction 5.20 we give an efficient stateful algorithm that realizes Lemma 5.17 in polynomial time, based on succinct constraint detection.

Throughout the following, given an arity $m \in \mathbb{N}$ and a degree parameter $d \geq 2 \in \mathbb{N}$ we denote $\vec{d_i} := (d, \ldots, d-2, \ldots, d) \in \mathbb{N}^m$ to be the vector which takes the value $d-2$ at its $i$-th entry, and the value $d$ everywhere else.

**Lemma 5.17.** *Let $\mathbb{F}$ be a field, let $m, d, t \in \mathbb{N}$, let $\mathsf{ZSample}_{\mathbb{F},m,d}$ be the algorithm described in Construction 5.18 and let $Z\colon \{0,1\}^m \to \{0,1\}$ denote the zero function. Then for all $\mathsf{tr} \in (\mathbb{F}^m \times \mathbb{F})^t$ which agree with $Z$ and are such that $\mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}] \neq \emptyset$, and for all $x \in \mathbb{F}^m, y \in \mathbb{F}$:*

$$\Pr\left[\mathsf{ZSample}_{\mathbb{F},m,\vec{d}}(\mathsf{tr}, x) = (\mathsf{tr}', y)\right] = \Pr\left[P(x) = y \;\middle|\; P(\vec{X}) \leftarrow \mathsf{LDE}_{\mathbb{F},\vec{d}}[Z \cup \mathsf{tr}]\right] \quad ,$$

*where $\mathsf{tr}' := \mathsf{tr} \cup \{(x, y)\}$.*

**Construction 5.18.** $\mathsf{ZSample}_{\mathbb{F},m,d}(\mathsf{tr}, x^*)$.

1. Define the set

$$S_{\mathsf{tr}} := \Big\{(\mathsf{tr}_1, \ldots, \mathsf{tr}_m) \in (\mathrm{supp}(\mathsf{tr}) \to \mathbb{F})^m : \forall i \in [m],\ \mathsf{LDE}_{\mathbb{F},\vec{d_i}}[\mathsf{tr}_i] \neq \emptyset \\ \wedge\ \forall (x, y) \in \mathsf{tr},\ \sum_{i=1}^{m} x_i(x_i - 1)\mathsf{tr}_i(x) = y\Big\} \quad .$$

2. Sample $(\mathsf{tr}_1, \ldots, \mathsf{tr}_m) \leftarrow \mathcal{U}(S_{\mathsf{tr}})$.
3. For each $i \in [m]$ sample $R_i(\vec{X}) \leftarrow \mathsf{LDE}_{\mathbb{F},\vec{d_i}}[\mathsf{tr}_i]$.

4. Output $y := \sum_{i=1}^{m} x_i^*(x_i^* - 1)R_i(x^*)$.

We will require the following simple, but useful claim.

**Claim 5.19.** *Let $\mathbb{F}$ be a field, $m \in \mathbb{N}$, $\vec{d} := (d_1, \ldots, d_m) \in \mathbb{N}^m$ and $S \subseteq \mathbb{F}^m$. Let $f\colon S \to \mathbb{F}$ and $g\colon S \to \mathbb{F}$ be such that $\mathsf{LDE}_{\mathbb{F},\vec{d}}[f]$ and $\mathsf{LDE}_{\mathbb{F},\vec{d}}[g]$ are non-empty. Then $|\mathsf{LDE}_{\mathbb{F},\vec{d}}[f]| = |\mathsf{LDE}_{\mathbb{F},\vec{d}}[g]|$.*

*Proof.* Fix some $\hat{f} \in \mathsf{LDE}_{\mathbb{F},\vec{d}}[f]$, $\hat{g} \in \mathsf{LDE}_{\mathbb{F},\vec{d}}[g]$. Let $T(P) := P - \hat{f} + \hat{g}$, and observe that $T$ is a bijection between $\mathsf{LDE}_{\mathbb{F},\vec{d}}[f]$ and $\mathsf{LDE}_{\mathbb{F},\vec{d}}[g]$. □

*Proof of Lemma 5.17.* Denote $\mathcal{D}_0 := \mathcal{U}(\mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}])$. Define

$$\mathcal{D}_Z := \left\{ (R_1(\vec{X}), \ldots, R_m(\vec{X})) \;\middle|\; \begin{array}{c} (\mathsf{tr}_1, \ldots, \mathsf{tr}_m) \leftarrow \mathcal{U}(S_{\mathsf{tr}}) \\ R_i(\vec{X}) \leftarrow \mathsf{LDE}_{\mathbb{F},\vec{d_i}}[\mathsf{tr}_i] \; \forall i \in [m] \end{array} \right\} .$$

Observe that the distribution of polynomials sampled by Step 3 of $\mathsf{ZSample}_{\mathbb{F},m,d}(\mathsf{tr})$ is $\mathcal{D}_Z$.

Let $\Phi(R_1(\vec{X}), \ldots, R_m(\vec{X})) := \sum_{i=1}^{m} X_i(X_i - 1)R_i$. We introduce a hybrid distribution, as follows

$$R_{\mathsf{tr}} := \left\{ \vec{R} \in \prod_{i=1}^{m} \mathbb{F}^{\leq \vec{d}_i}[X_1, \ldots, X_m] : (\Phi(\vec{R}))(x) = y \; \forall \; (x, y) \in \mathsf{tr} \right\} ,$$

and define the hybrid $\mathcal{D}_{CN}$ to be the uniform distribution over the set $R_{\mathsf{tr}}$.

The proof proceeds in two steps: first we show that $\mathcal{D}_Z$ is identical to $\mathcal{D}_{CN}$. Second, we use the combinatorial nullstellensatz [Alo99] to show that if $\vec{R} \sim \mathcal{U}(R_{\mathsf{tr}})$ then $\Phi(\vec{R}) \sim \mathcal{U}(\mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}])$.

**$\mathcal{D}_Z$ vs. $\mathcal{D}_{CN}$:** The distribution $\mathcal{D}_{CN}$ is defined as the uniform distribution over $R_{\mathsf{tr}}$ so $\Pr[\vec{R} \leftarrow \mathcal{D}_{CN}] = \frac{1}{|R_{\mathsf{tr}}|}$. We show that $\Pr[\vec{R} \leftarrow \mathcal{D}_Z] = \frac{1}{|R_{\mathsf{tr}}|}$.

The set $S_{\mathsf{tr}}$ partitions $R_{\mathsf{tr}}$ into a disjoint union of sets, each of the same cardinality. In particular, for $\vec{\mathsf{tr}} \in S_{\mathsf{tr}}$ denote

$$T_{\vec{\mathsf{tr}}} := \left\{ (R_1(\vec{X}), \ldots, R_m(\vec{X})) : R_i(\vec{X}) \in \mathsf{LDE}_{\mathbb{F},\vec{d_i}}[\mathsf{tr}_i] \; \forall i \in [m] \right\},$$

then we have that

$$\bigcup_{\vec{\mathsf{tr}} \in S_{\mathsf{tr}}} T_{\vec{\mathsf{tr}}} = \mathrm{supp}(\mathcal{D}_Z) = R_{\mathsf{tr}} \ .$$

Observe that for $\vec{\mathsf{tr}} \neq \vec{\mathsf{tr}}'$, $T_{\vec{\mathsf{tr}}} \cap T_{\vec{\mathsf{tr}}'} = \varnothing$. Moreover, by definition of $S_{\mathsf{tr}}$, $\mathsf{LDE}_{\mathbb{F},\vec{d_i}}[\mathsf{tr}_i] \neq \emptyset$ for each $i \in [m]$, so by Claim 5.19, every $T_{\vec{\mathsf{tr}}}$ is of the same cardinality for any $\vec{\mathsf{tr}} \in S_{\mathsf{tr}}$. Thus $|T_{\vec{\mathsf{tr}}}| = \frac{|R_{\mathsf{tr}}|}{|S_{\mathsf{tr}}|}$ for all $\vec{\mathsf{tr}} \in S_{\mathsf{tr}}$.

Let $\vec{r} \in R_{\mathsf{tr}}$, let $\vec{R} \leftarrow \mathcal{D}_Z$, and let $\vec{\mathsf{tr}}^*_{\vec{r}} \in S_{\mathsf{tr}}$ be uniquely defined to be such that $\vec{r} \in T_{\vec{\mathsf{tr}}^*_{\vec{r}}}$. For each $\vec{\mathsf{tr}} \in S_{\mathsf{tr}}$, let $E_{\vec{\mathsf{tr}}}$ be the event that $\vec{R} \in T_{\vec{\mathsf{tr}}}$. Note that by definition of $\mathcal{D}_Z$, $\Pr[E_{\vec{\mathsf{tr}}}] = \frac{1}{|S_{\mathsf{tr}}|}$ for all $\vec{\mathsf{tr}} \in S_{\mathsf{tr}}$. Conditioning on $E_{\vec{\mathsf{tr}}}$, we have

$$\begin{aligned} \Pr\left[\vec{R} = \vec{r}\right] &= \sum_{\vec{\mathsf{tr}} \in S_{\mathsf{tr}}} \Pr\left[\vec{R} = \vec{r} \;\middle|\; E_{\vec{\mathsf{tr}}}\right] \Pr\left[E_{\vec{\mathsf{tr}}}\right] \\ &= \Pr\left[\vec{R} = \vec{r} \;\middle|\; E_{\vec{\mathsf{tr}}^*_{\vec{r}}}\right] \Pr\left[E_{\vec{\mathsf{tr}}^*_{\vec{r}}}\right] \end{aligned}$$

$$= \frac{1}{|T_{\vec{\mathsf{tr}}^*_{\vec{r}}}||S_{\mathsf{tr}}|}$$
$$= \frac{1}{|R_{\mathsf{tr}}|} \ ,$$

where the second and third equalities follow from the fact that $\Pr\left[\vec{R} = \vec{r} \;\middle|\; E_{\vec{\mathsf{tr}}}\right] = \frac{1}{|T_{\vec{\mathsf{tr}}}|}$ if $\vec{\mathsf{tr}} = \vec{\mathsf{tr}}^*_{\vec{r}}$ and is 0 otherwise.

**$\mathcal{D}_{CN}$ vs. $\mathcal{D}_0$.** We show that $\Phi(\vec{R})$ for $R \leftarrow \mathcal{D}_{CN}$ is distributed as $\mathcal{D}_0 = \mathcal{U}(\mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}])$.

Note that $\Phi$ is a linear map, and $R_{\mathsf{tr}}$ is an affine space, so by Claim 3.2, if $\vec{R} \sim \mathcal{D}_{CN}$ then $\Phi(\vec{R}) \sim \mathcal{U}(\Phi(R_{\mathsf{tr}}))$. It remains to show that $\Phi(R_{\mathsf{tr}}) = \mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}]$. First, we show that $\Phi(R_{\mathsf{tr}}) \subseteq \mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}]$: fix $\vec{R} \in R_{\mathsf{tr}}$; then since $\mathrm{im}(\Phi) \subseteq \mathsf{LDE}_{\mathbb{F},d}[Z]$ and $(\Phi(\vec{R}))(x) = y \;\forall\; (x,y) \in \mathsf{tr}$, $\Phi(\vec{R}) \in \mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}]$.

Finally we show that $\mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}] \subseteq \Phi(R_{\mathsf{tr}})$, which completes the proof. Fix some $\hat{Z} \in \mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}]$. Since $\mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}] \subseteq \mathsf{LDE}_{\mathbb{F},d}[Z]$, by Lemma 3.3, there exist polynomials $R_1, \ldots, R_m$ where $R_i \in \mathbb{F}^{\leq \vec{d}_i}[X_1, \ldots, X_m]$ such that $\hat{Z} = \Phi(R_1, \ldots, R_m)$. Then for all $(x,y) \in \mathsf{tr}$, $\Phi(R_1, \ldots, R_m)(x) = \hat{Z}(x) = y$. Hence $\vec{R} \in R_{\mathsf{tr}}$, and so $\hat{Z} \in \Phi(R_{\mathsf{tr}})$. □

**Construction 5.20.** Given a constraint detector $\mathsf{CD}_{\mathbb{F},m,\vec{d}}$ for $\mathrm{LD}[\mathbb{F}, m, \vec{d}]$ (Definition 3.10), we efficiently implement $\mathsf{ZSample}_{\mathbb{F},m,d}$ as a stateful oracle for $d \geq 2$ as follows:

- *Evaluate polynomial:* $\mathsf{ZSample}_{\mathbb{F},m,d}(\mathsf{tr}, x^*) \to y$.
  1. If $\mathsf{tr} = \emptyset$:
     (a) For each $i \in [m]$, set $\mathsf{tr}_i := \emptyset$.
  2. If $\mathsf{tr} \neq \emptyset$:
     (a) For each $i \in [m]$, run $\mathsf{CD}_{\mathbb{F},m,\vec{d}_i}(\mathrm{supp}(\mathsf{tr}))$ to obtain constraints $z_{i,j}$ for $j \in [k]$.
     (b) Using Gaussian elimination, solve the following linear system of constraints

$$\sum_{x \in \mathrm{supp}(\mathsf{tr})} z_{i,j}(x)\mathsf{tr}_i(x) = 0 \quad \forall j \in [k],\ \forall i \in [m] \ ,$$

$$\sum_{i=1}^{m} x_i(1-x_i)\mathsf{tr}_i(x) = y \quad \forall (x,y) \in \mathsf{tr} \ ,$$

     for the variables $\mathsf{tr}_i(x)$, to obtain a description of the solution space $S_{\mathsf{tr}} \subseteq \mathbb{F}^{m \times |\mathrm{supp}(\mathsf{tr})|}$ of vectors satisfying the constraints.
     (c) Sample $(\mathsf{tr}_1, \ldots, \mathsf{tr}_m) \leftarrow S_{\mathsf{tr}}$ uniformly.
  3. For each $i \in [m]$, sample $y_i \leftarrow \mathsf{LDSample}_{\mathbb{F},m,\vec{d}_i}(\mathsf{tr}_i, x^*)$.
  4. Output $y := \sum_{i=1}^{m} x_i^*(x_i^* - 1)y_i$.

**Lemma 5.21.** *The outputs of Construction 5.18 and Construction 5.20 are identical. Moreover, Construction 5.20 runs in time* $\mathrm{poly}(m, d, \log|\mathbb{F}|, t)$*, where* $t = |\mathsf{tr}|$.

*Proof.* **Correctness.** By definition, for each $i \in [m]$, $\mathsf{CD}_{\mathbb{F},m,\vec{d}_i}(\mathrm{supp}(\mathsf{tr}))$ will output a basis $z_{i,j}$ for the space of constraints $\{z \colon \mathbb{F}^m \to \mathbb{F} : \forall\ p \in \mathbb{F}^{\leq \vec{d}_i}[X_1, \ldots, X_m],\ \sum_{x \in \mathrm{supp}(\mathsf{tr})} z(x)p(x) = 0\}$ on $\mathrm{supp}(\mathsf{tr})$.

Therefore for any given $i \in [m]$, we have that $\mathsf{tr}_i$ satisfies

$$\sum_{x \in \mathrm{supp}(\mathsf{tr})} z_{i,j}(x)\mathsf{tr}_i(x) = 0 \quad \forall j \in [k] \ ,$$

if and only if $\mathsf{LDE}_{\mathbb{F},\vec{d}_i}[\mathsf{tr}_i] \neq \emptyset$. Therefore, the query-answer transcripts which are sampled in Step 2c of Construction 5.20 are distributed identically to those sampled in Step 2 of Construction 5.18. By Lemma 5.15 for each $i \in [m]$ we have that

$$\Pr\left[\mathsf{LDSample}_{\mathbb{F},m,\vec{d}}(\mathsf{tr}_i, x) = (\mathsf{tr}'_i, y)\right] = \Pr\left[R_i(x) = y \;\middle|\; R_i(\vec{X}) \leftarrow \mathsf{LDE}_{\mathbb{F},\vec{d}}[\mathsf{tr}_i]\right] \ ,$$

meaning that the output distribution of Construction 5.20 is identical to Construction 5.18.

**Efficiency.** Running $\mathsf{ZSample}_{\mathbb{F},m,d}$ requires running $\mathsf{CD}_{\mathbb{F},m,\vec{d}_i}$ $m$ times, which requires $\mathrm{poly}(m, d, \log(|\mathbb{F}|))$ time. It further requires using Gaussian elimination to solve a system of at most $mt$ equations for $mt$ unknowns, which requires time $O(m^3t^3)$. Finally, it requires running $\mathsf{LDSample}_{\mathbb{F},m,\vec{d}_i}$ which also requires time $\mathrm{poly}(m, d, \log(|\mathbb{F}|))$, by Lemma 5.15. Thus $\mathsf{ZSample}_{\mathbb{F},m,d}$ runs in time $\mathrm{poly}(m, d, \log(|\mathbb{F}|), t)$. □

# 6 From ROM to AROM security

We prove that security properties in the ROM also hold in the AROM.

- In Section 6.1 we prove that the witness oracle in the AROM can be emulated.
- In Section 6.2 we show that any pass-through stateful (ARO, $\widehat{\mathsf{vo}}$)-emulator can be transformed into a pass-through stateful $(\mathrm{ARO}, \{\mathsf{wo}, \widehat{\mathsf{vo}}\})$-emulator that only accesses $\mathsf{ro}$, while preserving the emulation error. This is done using the witness oracle emulator from Section 6.1.
- In Section 6.3 we build on the above result to prove that security in the ROM implies security in the AROM.
- In Section 6.4 we prove that commitment schemes in the ROM remain secure in the AROM.

## 6.1 Emulating access to the witness oracle

We transform any adversary that queries the witness oracle $\mathsf{wo}$ of the ARO to an adversary that does not query $\mathsf{wo}$. In Section 6.3 we use this result to prove Theorem 6.5.

**Lemma 6.1.** *Let $\mathcal{O} := \mathrm{ARO}[\mathbb{F}, m, \lambda, d, B]$, where $\mathbb{F}$ is a finite field, the parameters $m, \lambda, d \in \mathbb{N}$, and $B\colon \{0,1\}^{m+r} \to \{0,1\}^w$ is an oracle circuit. Define the distribution $\mathcal{O}' := \{(\mathsf{ro}, \mathsf{wo}) : (\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}\}$. The algorithm $\mathcal{W}[B]$ (Construction 6.2) is a pass-through stateful $(\mathcal{O}', \{\mathsf{wo}\})$-emulator with zero error. $\mathcal{W}$ answers each query in time $O(|B|)$.*

**Construction 6.2.** Define $\mathcal{W}[B]$ as follows:

- $\mathcal{W}^{\mathsf{ro}}[B](\mathsf{tr}, \mathsf{oid}, x) \to y$.
  1. If $\mathsf{oid} = \mathsf{ro}$, return $\mathsf{oid}(x)$.
  2. If $\mathsf{tr}(x) \neq \bot$, return $\mathsf{tr}(x)$.
  3. Otherwise, sample uniform $\mu \leftarrow \{0,1\}^r$ and set $y := B^{\mathsf{ro}}(x, \mu)$. Output $y$.

## 6.2 Stateful emulator for the ARO w.r.t. the random oracle

As a consequence of Lemma 6.1, there exists an emulator for the ARO that only accesses the random oracle.

**Lemma 6.3.** *Let $\mathcal{O} := \mathrm{ARO}[\mathbb{F}, m, \lambda, d, B]$, where $\mathbb{F}$ is a finite field, $m, \lambda, d \in \mathbb{N}$ respectively are arity, security, and degree parameters, and $B\colon \{0,1\}^{m+r} \to \{0,1\}^w$ is a $t_B$-query oracle circuit. Let $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ be a pass-through stateful $(\mathcal{O}, \{\widehat{\mathsf{vo}}\})$-emulator with error $\varepsilon_{\mathrm{ARO}}(t, \lambda)$. Then, there exists a pass-through stateful $(\mathcal{O}, \{\mathsf{wo}, \widehat{\mathsf{vo}}\})$-emulator $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}$ with error $\varepsilon_{\mathrm{ARO}}(t, \lambda)$.*

*Further, if $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ makes $t_{\mathcal{M}}$ oracle queries and runs in time $T_{\mathcal{M}}$ for emulating a single query, then for emulating $t$ queries, $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}$ has a runtime of $t \cdot (T_{\mathcal{M}} + t_{\mathcal{M}} \cdot O(|B|))$ and a query complexity of $t \cdot t_{\mathcal{M}} \cdot t_B$.*

*Proof.* Define the emulator $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}} := \mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathcal{W}[B])}$, i.e. $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}$ works exactly like $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ except that $\mathsf{wo}$-queries are answered using $\mathcal{W}[B]$ (Construction 6.2). We show that for any $t$-query adversary $\mathcal{A}$,

$$\left|\Pr\left[\mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})} = 1 \;\middle|\; (\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda)\right] - \Pr\left[\mathcal{A}^{\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}} = 1 \;\middle|\; (\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda)\right]\right| \leq \varepsilon_{\mathrm{ARO}}(t, \lambda) \ .$$

By Lemma 6.1, in which the adversary $\mathcal{A}$ is the composed adversary $\mathcal{A}^{\mathcal{M}_{\mathrm{ARO}}^{(\cdot)}}$, we have

$$\Pr\left[\mathcal{A}^{\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}} = 1 \;\middle|\; (\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda)\right] = \Pr\left[\mathcal{A}^{\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}} = 1 \;\middle|\; (\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda)\right] \ . \qquad (5)$$

Since $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ is a pass-through stateful $(\mathcal{O}, \{\widehat{\mathsf{vo}}\})$-emulator, by Definition 5.2, we have

$$\left|\Pr\left[\mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}=1 \;\middle|\; (\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}}) \leftarrow \mathcal{O} \;\right] - \Pr\left[\mathcal{A}^{\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}}=1 \;\middle|\; (\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda) \;\right]\right| \leq \varepsilon_{\mathrm{ARO}}(t,\lambda) \;. \tag{6}$$

Substituting the probability on the right side of Equation 6 with Equation 5 yields the claim.

Finally, we consider the efficiency of $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}$. Let $t_{\mathcal{M}}, T_{\mathcal{M}}$ respectively denote the number of queries and runtime of $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ for emulating a single query. Then, $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}$ has a runtime of $T_{\mathcal{M}} + t_{\mathcal{M}} \cdot O(|B|)$ per emulated query, since $\mathcal{W}[B]$ answers each query in time $O(|B|)$ (Lemma 6.1). Hence the runtime for emulating $t$ queries is $t \cdot (T_{\mathcal{M}} + t_{\mathcal{M}} \cdot O(|B|))$. Moreover, if a single execution of $B$ makes $t_B$ queries to $\mathsf{ro}$, then $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}$ makes $t \cdot t_{\mathcal{M}} \cdot t_B$ oracle queries when emulating $t$ queries. □

**Corollary 6.4.** *There exists an efficient pass-through stateful $(\mathcal{O}, \{\mathsf{wo}, \widehat{\mathsf{vo}}\})$-emulator with error $\frac{t}{2^\lambda}$ and a query complexity of $t \cdot t_B$.*

*Proof.* This is a consequence using the $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ of Theorem 5.4 in Lemma 6.3. □

## 6.3 Security in the ROM is preserved in the AROM

We prove that security properties in the ROM are preserved in the AROM; this is a straightforward application of Corollary 6.4.

**Theorem 6.5.** *Let $\mathcal{O} := \mathrm{ARO}[\mathbb{F}, m, \lambda, d, B]$, where $\mathbb{F}$ is a finite field, $m, \lambda, d \in \mathbb{N}$ respectively are arity, security, and degree parameters, and $B\colon \{0,1\}^{m+r} \to \{0,1\}^w$ is a $t_B$-query polynomial-size oracle circuit. There exists a polynomial-size circuit $\mathcal{C}$ such that for all $t_{\mathcal{A}}$-query adversaries $\mathcal{A}$ and all $t_{\mathsf{p}}$-query $\mathsf{ro}$-oracle predicates $\mathsf{p}$ where*

$$\Pr\left[\mathsf{p}^{\mathsf{ro}}(x)=1 \;\middle|\; \begin{array}{c}(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}}) \leftarrow \mathcal{O} \\ x \leftarrow \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}\end{array}\right] \geq \delta \;, \tag{7}$$

*we have*

$$\Pr\left[\mathsf{p}^{\mathsf{ro}}(x)=1 \;\middle|\; \begin{array}{c}\mathsf{ro} \leftarrow \mathcal{U}(m,\lambda) \\ x \leftarrow \mathcal{C}^{(\mathsf{ro},\mathcal{A})}\end{array}\right] \geq \delta - \frac{t_{\mathcal{A}} + t_{\mathsf{p}}}{2^\lambda} \;.$$

*$\mathcal{C}$ makes at most $t_{\mathcal{A}} \cdot t_B$ queries to $\mathsf{ro}$ and accesses $\mathcal{A}$ in a straightline fashion.*

*Proof.* Let $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}$ be the pass-through stateful $(\mathcal{O}, \{\mathsf{wo}, \widehat{\mathsf{vo}}\})$-emulator guranteed by Corollary 6.4. Define the adversary $\mathcal{C}^{(\mathsf{ro},\mathcal{A})} := \mathcal{A}^{\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}}$. Note that $\mathcal{C}^{(\mathsf{ro},\mathcal{A})}$ runs $\mathcal{A}$ in a straightline fashion and answers $\mathcal{A}$'s oracle queries using $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}$; the efficiency of $\mathcal{C}$ is straightforward. Define the algorithm $\bar{\mathcal{A}}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$, which runs $x \leftarrow \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$ and outputs $\mathsf{p}^{\mathsf{ro}}(x)$; the query complexity of $\bar{\mathcal{A}}$ is $t_{\mathcal{A}} + t_{\mathsf{p}}$. The theorem follows immediately from the definition of pass-through stateful emulator (Definition 5.2) applied to $\bar{\mathcal{A}}$. □

## 6.4 Commitment schemes in the AROM

A corollary of Theorem 6.5 is that any commitment scheme secure in the ROM is also secure in the AROM. The weaker statement that any commitment scheme in the standard model is secure in the AROM also holds, since any standard-model commitment scheme is secure in the ROM. (Because any adversary in the ROM that breaks the commitment scheme can also break it in the standard model by simulating the random oracle.)

**Lemma 6.6.** *Denote $\mathcal{X} := \mathbf{ARO}[\mathscr{F}, m, d]$, in which $\mathscr{F} = \{\mathbb{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is a family of fields, $m\colon \mathbb{N} \to \mathbb{N}$ be an arity function and $d\colon \mathbb{N} \to \mathbb{N}$ be a degree function. If* CM *is a binding (resp. hiding) commitment scheme in the ROM with binding error $\varepsilon_{\mathsf{bind}}(\lambda)$ (resp. hiding error $\varepsilon_{\mathsf{hide}}(\lambda)$), then* CM *is a commitment scheme in $\mathcal{X}$ with binding error $\varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathrm{ARO}}(t, \lambda)$ (resp. hiding error $\varepsilon_{\mathsf{hide}}(\lambda) + \varepsilon_{\mathrm{ARO}}(t, \lambda)$), in which $t$ is the adversary's query bound.*

*Proof.* Let $\mathcal{O} \in \mathcal{X}$ and $\lambda \in \mathbb{N}$. Let $\delta$ be the binding error of a commitment scheme CM in the AROM. Suppose $\mathcal{A}$ is a $t$-query efficient adversary such that

$$\Pr\left[\begin{array}{c|r} m_0 \neq m_1 & (\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda) \\ \wedge & \mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda) \\ \mathsf{CM.Commit}(\mathsf{ck}, m_0; \omega_0) = \mathsf{CM.Commit}(\mathsf{ck}, m_1; \omega_1) & ((m_0, \omega_0), (m_1, \omega_1)) \leftarrow \mathcal{A}^{(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}})}(\mathsf{ck}) \end{array}\right] > \delta \ .$$

By Theorem 6.5, there exists an adversary $\mathcal{C}$ so that

$$\Pr\left[\begin{array}{c|r} m_0 \neq m_1 & \mathsf{ro} \leftarrow \mathcal{U}(m(\lambda), \lambda) \\ \wedge & \mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda) \\ \mathsf{CM.Commit}(\mathsf{ck}, m_0; \omega_0) = \mathsf{CM.Commit}(\mathsf{ck}, m_1; \omega_1) & ((m_0, \omega_0), (m_1, \omega_1)) \leftarrow \mathcal{C}^{(\mathsf{ro}, \mathcal{A})}(\mathsf{ck}) \end{array}\right]$$
$$\geq \delta - \varepsilon_{\mathrm{ARO}}(t, \lambda) \ .$$

By CM's binding property in the ROM, we have that $\delta - \varepsilon_{\mathrm{ARO}}(t, \lambda) \leq \varepsilon_{\mathsf{bind}}(\lambda)$, so $\delta \leq \varepsilon_{\mathsf{bind}}(\lambda) + \varepsilon_{\mathrm{ARO}}(t, \lambda)$. Note that when $\varepsilon_{\mathrm{ARO}}(t, \lambda)$ is the error from Theorem 5.4, we get $\delta \leq \mathrm{negl}(\lambda)$ since CM is binding in the ROM; hence CM is binding in the AROM. A similar argument shows that CM is hiding in the AROM. □

# 7 Zero-finding game in the AROM

We state and prove our lemma for zero-finding games in the AROM.

**Lemma 7.1.** *Let $\mathcal{X} := \mathbf{ARO}[\mathscr{F}, m, d]$, let $\ell \in \mathbb{N}$ be a degree bound, and let $\mathsf{CM}$ be a commitment scheme that has binding error $\varepsilon_{\mathsf{CM}}(\lambda)$ (and is not necessarily hiding). For every efficient probabilistic $t$-query oracle algorithm $\mathcal{A}$ that outputs tuples of the form $(f \in \mathbb{F}^{\leq d\ell(m+\lambda+w)}[X], g \in (\mathbb{F}^{\leq \ell}[X])^{m+\lambda+w}, \omega)$ and every $\mathcal{O} \in \mathcal{X}$, the following holds:*

$$\Pr\left[\begin{array}{l|r} f(X) \not\equiv \widehat{\mathsf{vo}}(g(X)) & (\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda) \\ \wedge\ f(\beta) = \widehat{\mathsf{vo}}(g(\beta)) & \mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda) \\ & (f, g, \omega) \leftarrow \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{ck}) \\ & \mathsf{cm} := \mathsf{CM.Commit}(\mathsf{ck}, (g, f); \omega) \\ & \beta \in \{0,1\}^\lambda := \mathsf{ro}(\mathsf{cm}) \end{array}\right]$$
$$\leq O\left(\sqrt{t \cdot \left(\frac{d\ell(m+\lambda+w)}{|\mathbb{F}|} + \varepsilon_{\mathsf{CM}}(\lambda)\right)}\right) + \varepsilon_{\mathrm{ARO}}(t + d\ell(m+\lambda+w), \lambda) \ .$$

*Above, $\varepsilon_{\mathrm{ARO}}(t + d\ell(m+\lambda+w), \lambda)$ is the emulation error of a stateful emulator for the ARO.*

## 7.1 Partial oracles

We introduce partial oracles for the stateful emulator for the ARO, which extend an ARO-query-answer transcript to include evaluations which are determined by the structure of the emulator. First, we define a partial oracle for $\widehat{\mathsf{vo}}$ queries as follows.

**Definition 7.2.** *Let $\mathcal{X} := \mathbf{ARO}[\mathscr{F}, m, d]$, let $\mathcal{O} \in \mathcal{X}$, and let $\mathsf{tr}$ be an $\mathcal{O}$-query-answer transcript. We define the partial function $\overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}} \colon \{0,1\}^{m+\lambda+w} \rightharpoonup \mathbb{F}$ to be:*

$$\overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(x) := \begin{cases} y & \text{if } P(x) = y \text{ for all } P \in \mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}|_{\widehat{\mathsf{vo}}}] \\ \perp & \text{otherwise} \end{cases} \ .$$

Second, we relate $\overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}$ to the output distribution of $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ for $\widehat{\mathsf{vo}}$ queries.

**Claim 7.3.** *Let $\mathcal{X} := \mathbf{ARO}[\mathscr{F}, m, d]$. For every $\mathcal{O} \in \mathcal{X}$, every $\mathcal{O}$-query-answer transcript $\mathsf{tr}$, every $x \in \mathbb{F}^{m+\lambda+w}$ and every $y \in \mathbb{F}$, we have that*

$$\Pr_{P \leftarrow \mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}|_{\widehat{\mathsf{vo}}}]}[P(x) = y] = \begin{cases} \frac{1}{|\mathbb{F}|} & \text{if } \overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(x) = \perp \\ 1 & \text{if } \overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(x) = y \\ 0 & \text{otherwise} \end{cases} \ .$$

*Proof.* Let $x \in \mathbb{F}^{m+\lambda+w}$. The cases when $\overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(x) \neq \perp$ are clear, and so we consider the case when $\overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(x) = \perp$. Let $\mathsf{FIXED} = \left\{x \in \mathbb{F}^{m+\lambda+w} \;\middle|\; \overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(x) \neq \perp\right\}$. In this case, there exist $P_1, P_2 \in \mathrm{supp}(\mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}|_{\widehat{\mathsf{vo}}}])$ such that $P_1(x') = P_2(x')$ for every $x' \in \mathsf{FIXED}$, but $P_1(x) \neq P_2(x)$. Since $\mathrm{supp}(\mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}|_{\widehat{\mathsf{vo}}}])$ is an affine space, there exists $P^* = P_1 - P_2 \in \mathsf{LDE}_{\mathbb{F},d}[Z \cup \mathsf{tr}_0]$, where $Z \colon \{0,1\}^m \to \{0,1\}$ is the zero function and $\mathsf{tr}_0 := \{(x, 0) : x \in \mathrm{supp}(\mathsf{tr})\}$, such that $P^*(x') = 0$ for every $x \in \mathsf{FIXED}$ and $P^*(x) \neq 0$. Thus, we can sample from the conditional distribution in the claim by uniformly sampling $P'' \in \mathrm{supp}(\mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}|_{\widehat{\mathsf{vo}}}])$ such that $P''(x') = \overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(x')$ for every $x' \in \mathsf{FIXED}$, uniformly sampling $\alpha \in \mathbb{F}$, and returning $P'' + \alpha P^* \in \mathrm{supp}(\mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}|_{\widehat{\mathsf{vo}}}])$. The claim follows since $P''(x) + \alpha P^*(x)$ is uniformly random in $\mathbb{F}$. □

## 7.2 Proof of Lemma 7.1

We will prove Lemma 7.1 by first introducing an emulated hybrid world, in which the adversary plays the zero-finding game against the emulator, which we show is statistically close to the zero-finding game in the lemma statement. We then invoke a forking lemma (Lemma 3.12) to obtain a lower bound on the probability that a forking adversary wins both forks in this hybrid world. Finally, we invoke Schwartz–Zippel and the binding property of CM to obtain an upper bound on the same quantity, and combine the two bounds to obtain the result.

**Emulated hybrid world.** Let $\mathcal{O} \in \mathcal{X}$ and let $\mathcal{A}$ be an adversary that wins the zero-finding game above with probability $\delta$. Let $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ be a pass-through stateful $(\mathcal{O}, \{\widehat{\mathsf{vo}}\})$-emulator with error $\varepsilon_{\mathrm{ARO}}(t, \lambda)$, as constructed in Theorem 5.4. Then $\mathcal{A}$ wins the following game, which we refer to as the *emulated zero finding game*, with probability at least $\delta - \varepsilon_{\mathrm{ARO}}(t + d\ell(m + \lambda + w), \lambda)$:

$$\Pr\left[\begin{array}{l|r} & (\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda) \\ & \mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda) \\ f(X) \not\equiv P(g(X)) & (f, g, \omega) \xleftarrow{\mathsf{tr}} \mathcal{A}^{\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}}(\mathsf{ck}) \\ \wedge\, f(\beta) = P(g(\beta)) & P \leftarrow \mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}|_{\widehat{\mathsf{vo}}}] \\ & \mathsf{cm} := \mathsf{CM.Commit}(\mathsf{ck}, (g, f); \omega) \\ & \beta \in \{0,1\}^\lambda := \mathsf{ro}(\mathsf{cm}) \end{array}\right] \geq \delta - \varepsilon_{\mathrm{ARO}}(t + d\ell(m + \lambda + w), \lambda) \ ,$$

To see this, consider the following. First, an efficient adversary can compute $\widehat{\mathsf{vo}}(g(X))$ with $d\ell(m+\lambda+w)$ queries to $\widehat{\mathsf{vo}}$. Second, by Theorem 5.4, provided it is only receiving queries to $\widehat{\mathsf{vo}}$, $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$ answers each query by freshly sampling $P \leftarrow \mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}|_{\widehat{\mathsf{vo}}}]$ and outputting $P(x)$. This is equivalent to sampling $P \leftarrow \mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}|_{\widehat{\mathsf{vo}}}]$ once and then outputting $P(x)$, as is done in the above experiment.

Thus if the above inequality were not true, an efficient distinguishing adversary can use the zero-finding game to distinguish between the ARO and the emulator, contradicting Theorem 5.4.

**Lower bound on winning probability for forked executions.** We conclude the proof via the general forking lemma (Lemma 3.12).

We define a forking predicate $\mathsf{p}$ that, on input $(x := \mathsf{cm}, \mathsf{o} := (\mathsf{ck}, f, g, \omega), \mathsf{tr})$, checks the following conditions:

**Condition 1.** $\mathsf{cm} = \mathsf{CM.Commit}(\mathsf{ck}, (f, g); \omega)$;

**Condition 2.** either $\overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}} \circ g \colon \mathbb{F} \to \mathbb{F}$ is not total or $f(X) \not\equiv \overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(g(X))$; and

**Condition 3.** $f(\beta) = \overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(g(\beta))$, where $\beta := \mathsf{tr}|_{\mathsf{ro}}(\mathsf{cm})$.

Consider the following (forking lemma) adversary $\mathcal{B}$, which we introduce in order to invoke Lemma 3.12.

$\mathcal{B}^{\mathcal{A},\mathsf{p}}(\mathsf{ck}, y_1, \ldots, y_t; r)$:

1. Run $\mathsf{o} := (\mathsf{ck}, f, g, \omega) \leftarrow \mathcal{A}$ using $r$ as its random tape and simulating its oracle queries as follows:
   (a) answer the $i$-th fresh $\mathsf{ro}$-query with $y_i$.
   (b) answer $\mathsf{wo}$-queries using Construction 6.2, and simulating its $\mathsf{ro}$ queries as above;
   (c) answer $\widehat{\mathsf{vo}}$-queries using $\mathcal{M}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo})}$, simulating its $\mathsf{ro}, \mathsf{wo}$ queries as above.
   Let $\mathsf{tr}$ be the query-answer transcript in this simulation.
2. Compute $\mathsf{cm} \leftarrow \mathsf{CM.Commit}(\mathsf{ck}, \mathsf{o})$.
3. If $\mathsf{p}(\mathsf{cm}, \mathsf{o}, \mathsf{tr}) = 0$, output $(0, \perp)$. Otherwise, define $\mathsf{FP}(\mathsf{tr}, \mathsf{cm})$ to be the index of the query $\mathcal{A}$ made to $\mathsf{ro}$ at $\mathsf{cm}$ and output $(\mathsf{FP}(\mathsf{tr}, \mathsf{cm}), (\mathsf{cm}, \mathsf{o}, \mathsf{tr}))$.

Recall from the assumption that $\mathcal{A}$ wins the zero-finding game with probability $\delta$, which implies that $\mathcal{A}$ wins the emulated zero-finding game with probability $\delta - \varepsilon_{\mathrm{ARO}}(t + d\ell(m + \lambda + w), \lambda)$. We show that $\mathcal{B}$ is

accepting (i.e. outputs $(i, \cdot)$ such that $i \geq 1$) with probability at least $\delta - \varepsilon_{\mathrm{ARO}}(t + d\ell(m + \lambda + w), \lambda) - \frac{1}{|\mathbb{F}|}$. $\mathcal{B}$ is accepting when the output of $\mathcal{A}$ satisfies $\mathsf{p}$.

First, any $\mathcal{A}$ winning the emulated zero-finding game directly satisfies Condition 1.

Second, we show that whenever $\mathcal{A}$ wins the emulated zero-finding game, then Condition 2 is satisfied. We argue this via the contrapositive: if $f(X) \equiv \overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(g(X))$, then $\mathcal{A}$ does not win the emulated zero-finding game. For any input $(x, y, z) \in \mathbb{F}^{m+\lambda+w}$, if $\overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(x, y, z)$ is defined, then it is equal to $P(x, y, z)$ by definition. Thus if $f(X) \equiv \overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(g(X))$, $\overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}$ must be defined over the image of $g$, so $f(X) \equiv P(g(X))$; hence $\mathcal{A}$ cannot win the emulated zero-finding game by definition.

Third, we argue that the probability that $\mathcal{A}$ wins the emulated zero-finding game but fails to satisfy Condition 3 is at most $\frac{1}{|\mathbb{F}|}$. There are two cases: either $\overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(g(\beta))$ is defined, or not. If $\overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(g(\beta))$ is defined then $\overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(g(\beta)) = P(g(\beta))$, and if $\mathcal{A}$ wins the emulated zero-finding game then $P(g(\beta)) = f(\beta)$; hence Condition 3 is satisfied in this case. If $\overline{\mathsf{tr}}_{\widehat{\mathsf{vo}}}(g(\beta)) = \perp$, then the probability that $\mathcal{A}$ wins the emulated zero-finding game is $\frac{1}{|\mathbb{F}|}$, by Claim 7.3, since it must guess the value of $P(g(\beta))$.

Thus, $\mathcal{B}$ accepts with probability at least $\delta - \varepsilon_{\mathrm{ARO}}(t + d\ell(m + \lambda + w), \lambda) - \frac{1}{|\mathbb{F}|}$. We deduce via Lemma 3.12 that

$$\delta - \varepsilon_{\mathrm{ARO}}(t + d\ell(m + \lambda + w), \lambda) - \frac{1}{|\mathbb{F}|} \leq \frac{t}{2^\lambda} + \sqrt{t \cdot \mu} \ ,$$

where

$$\mu := \Pr\left[b = 1 \;\middle|\; \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda) \\ (b, (\mathsf{cm}, \mathsf{o}, \mathsf{tr}), (\mathsf{cm}', \mathsf{o}', \mathsf{tr}')) \leftarrow \mathsf{Fork}_{\mathcal{B}^{\mathcal{A}}, \mathsf{p}}(\mathsf{ck}) \end{array}\right] .$$

We argue that $\mathsf{cm} = \mathsf{cm}'$ in $\mathsf{Fork}_{\mathcal{B}^{\mathcal{A}}, \mathsf{p}}(\mathsf{ck})$'s output. Note that if $b = 1$, then we have $\mathsf{FP}(\mathsf{tr}, \mathsf{cm}) = \mathsf{FP}(\mathsf{tr}', \mathsf{cm}')$. Let $i := \mathsf{FP}(\mathsf{tr}, \mathsf{cm})$. Then by the definition of the forking algorithm, we have that $\mathsf{tr}_{<i} = \mathsf{tr}'_{<i}$ (recall that the notation $\mathsf{tr}_{<\ell}$ refers to the query-answer pairs in $\mathsf{tr}$ up to and excluding the $\ell$-th query). The $i$-th query made by $\mathcal{A}$ only depends on its internal randomness $r$ and $\mathsf{tr}_{<i}$. Since these are the same in both executions, $\mathcal{A}$'s $i$-th query must be the same in both executions. Thus, we have

$$\mu = \Pr\left[b = 1 \;\middle|\; \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda) \\ (b, (\mathsf{cm}, \mathsf{o}, \mathsf{tr}), (\mathsf{cm}, \mathsf{o}', \mathsf{tr}')) \leftarrow \mathsf{Fork}_{\mathcal{B}^{\mathcal{A}}, \mathsf{p}}(\mathsf{ck}) \end{array}\right] .$$

**Upper bound on winning probability for forked executions.** To conclude the proof, we upper bound $\mu$.

We can write $\mu$ as

$$\mu = \Pr\left[\begin{array}{l} \mathsf{cm} \neq \perp \\ \wedge\ \mathsf{p}(\mathsf{cm}, \mathsf{o}, \mathsf{tr}) = 1 \\ \wedge\ \mathsf{p}(\mathsf{cm}, \mathsf{o}', \mathsf{tr}') = 1 \end{array} \;\middle|\; \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda) \\ (b, (\mathsf{cm}, \mathsf{o}, \mathsf{tr}), (\mathsf{cm}, \mathsf{o}', \mathsf{tr}')) \leftarrow \mathsf{Fork}_{\mathcal{B}^{\mathcal{A}}, \mathsf{p}}(\mathsf{ck}) \end{array}\right] . \tag{8}$$

This is because if $b = 1$, then Step 3 of the forking adversary $\mathcal{B}$ implies that $\mathsf{cm} \neq \perp$ and $\mathsf{p}(\mathsf{cm}, \mathsf{o}, \mathsf{tr}) = 1 = \mathsf{p}(\mathsf{cm}, \mathsf{o}', \mathsf{tr}')$.

Denote by $E_1$ the event on the left of Equation 8. By the law of total probability, we have

$$\Pr[E_1] = \Pr[E_1 \wedge (\mathsf{o} = \mathsf{o}')] + \Pr[E_1 \wedge (\mathsf{o} \neq \mathsf{o}')] \ .$$

We compute the value of $\Pr[E_1 \wedge (\mathsf{o} \neq \mathsf{o}')]$. By the definition of $\mathsf{p}$, if $E_1$ holds then $\mathsf{CM.Commit}(\mathsf{ck}, \mathsf{o}) = \mathsf{cm} = \mathsf{CM.Commit}(\mathsf{ck}, \mathsf{o}')$. However, by the binding property of the commitment scheme $\mathsf{CM}$, the probability that this happens and $\mathsf{o} \neq \mathsf{o}'$ occurs is at most $\varepsilon_{\mathsf{CM}}(\lambda)$.

Let $E_2 := E_1 \wedge (\mathsf{o} = \mathsf{o}')$. From the above, it holds that

$$\mu - \varepsilon_{\mathsf{CM}}(\lambda) \leq \Pr[E_2] \ . \tag{9}$$

Let $i := \mathsf{FP}(\mathsf{tr}, \mathsf{cm})$, and let $\mathsf{tr}|_{i-1}$ denote the truncation of $\mathsf{tr}$ to the first $i-1$ queries. Define $E_3$ as the event that $\overline{\mathsf{tr}|_{i-1}}_{\widehat{\mathsf{vo}}} \circ g$ is total. By the law of total probability, we have

$$\Pr[E_2] = \Pr[E_2 \wedge E_3] + \Pr[E_2 \wedge \overline{E_3}] \ . \tag{10}$$

We show that $E_2 \wedge \overline{E_3}$ occurs with low probability. Define the point $\beta' := \mathsf{tr}'|_{\mathsf{ro}}(\mathsf{cm}')$.

**Claim 7.4.** *The probability that $E_2$ occurs and $\overline{\mathsf{tr}|_{i-1}}_{\widehat{\mathsf{vo}}} \circ g$ is not total is at most $(d\ell(m+\lambda+w)+1)/|\mathbb{F}|$.*

*Proof.* Since $P \in \mathrm{supp}(\mathsf{LDE}_{\mathbb{F},d}[\mathsf{vo}_{\mathsf{tr}} \cup \mathsf{tr}|_{\widehat{\mathsf{vo}}}])$, we have $\deg(P \circ g) \leq d \cdot \ell$. Hence, if $\overline{\mathsf{tr}|_{i-1}}_{\widehat{\mathsf{vo}}} \circ g$ is not total, then there are at most $d \cdot \ell$ points $x \in \mathbb{F}$ such that $\overline{\mathsf{tr}|_{i-1}}_{\widehat{\mathsf{vo}}}(g(x)) \neq \perp$.

Since $\mathsf{tr}|_{i-1}$ contains only queries before the $i$-th query $\mathsf{cm}$, we have $(\mathsf{tr}|_{i-1})|_{\mathsf{ro}}(\mathsf{cm}) = \perp$. Hence $\beta'$ is uniformly random conditioned on $\mathsf{tr}|_{i-1}$. Thus $\Pr[\overline{\mathsf{tr}|_{i-1}}_{\widehat{\mathsf{vo}}}(g(\beta')) \neq \perp] \leq \frac{d\ell(m+\lambda+w)}{|\mathbb{F}|}$. Finally, if $\overline{\mathsf{tr}|_{i-1}}_{\widehat{\mathsf{vo}}}(g(\beta')) = \perp$ then $\Pr[\overline{\mathsf{tr}|_{i-1}}_{\widehat{\mathsf{vo}}}(g(\beta')) = f(\beta')] \leq \frac{1}{|\mathbb{F}|}$, as $f$ and $\mathsf{tr}'$ are independent conditioned on $\mathsf{tr}|_{i-1}$. □

Combining Equations 9 and 10 and Claim 7.4, we get $\mu - (d\ell(m+\lambda+w)+1)/|\mathbb{F}| - \varepsilon_{\mathsf{CM}}(\lambda) \leq \Pr[E_2 \wedge E_3]$. In the event that $E_2 \wedge E_3$, it holds that $\overline{\mathsf{tr}|_{i-1}}_{\widehat{\mathsf{vo}}} \circ g \not\equiv f$, but $\overline{\mathsf{tr}|_{i-1}}_{\widehat{\mathsf{vo}}}(g(\beta')) = f(\beta')$. Since $\beta'$ is drawn independently of $\mathsf{tr}|_{i-1}$, $g$, and $f$, this equality holds with probability at most $(d\ell(m+\lambda+w))/|\mathbb{F}|$.

Rearranging, we conclude that

$$\mu \leq \frac{2d\ell(m+\lambda+w)+1}{|\mathbb{F}|} + \varepsilon_{\mathsf{CM}}(\lambda) \ .$$

Since $\delta - \varepsilon_{\mathrm{ARO}}(t + d\ell(m+\lambda+w), \lambda) - \frac{1}{|\mathbb{F}|} \leq \frac{t}{2^\lambda} + \sqrt{t \cdot \mu}$, we deduce that

$$\delta \leq \sqrt{t \cdot \left( \frac{2d\ell(m+\lambda+w)+1}{|\mathbb{F}|} + \varepsilon_{\mathsf{CM}}(\lambda) \right)} + \frac{1}{|\mathbb{F}|} + \varepsilon_{\mathrm{ARO}}(t + d\ell(m+\lambda+w), \lambda) + \frac{t}{2^\lambda} \ ,$$

which implies the statement.

# 8 Accumulation scheme

We construct an accumulation scheme for ARO queries. See Definition 3.6 for the definition of an accumulation scheme for oracle queries.

**Theorem 8.1.** *Let $\mathscr{F} = \{\mathbb{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of fields, $m\colon \mathbb{N}\to\mathbb{N}$ an arity function, and $d\colon \mathbb{N}\to\mathbb{N}$ a degree function. Let $\mathsf{CM} = (\mathsf{CM.Setup}, \mathsf{CM.Commit})$ be a standard-model commitment scheme that is hiding, $m$-succinct (see Section 3.6), and binding with error $\varepsilon_{\mathsf{CM}}(\lambda)$. Then, $\mathsf{AS}$ described in Construction 8.2 is a zero-knowledge accumulation scheme for $\mathbf{ARO}[\mathscr{F}, m, d]$-queries with the following properties.*

- Soundness error. *When accumulating $n$ queries and $\ell$ old accumulators, the soundness error is*

$$O\left(\sqrt{\frac{t\cdot d(\lambda)\cdot(n+\ell)\cdot(m(\lambda)+\lambda+w(\lambda))}{|\mathbb{F}_\lambda|}+\varepsilon_{\mathsf{CM}}(\lambda)+\varepsilon_{\mathrm{ARO}}(t,\lambda)}\right)$$

  *against any $t$-query adversary, in which $\varepsilon_{\mathrm{ARO}}(t,\lambda)$ is the error for the pass-through stateful $(\mathrm{ARO}[\mathbb{F}, m, \lambda, d, B], \{\widehat{\mathsf{vo}}\})$-emulator for ARO queries from Theorem 5.4.*
- Accumulator size. *The accumulator contains 2 $\widehat{\mathsf{vo}}$ query-answer pairs (i.e., $2(m(\lambda)+\lambda+w(\lambda)+1)$ field elements).*
- Decider efficiency. *The decider consists of checking 2 $\widehat{\mathsf{vo}}$ query-answer pairs.*

## 8.1 Construction

We assume a global ordering of the field $\mathbb{F}$, so that $\mathbb{F} = \{b_1, \ldots, b_{|\mathbb{F}|}\}$. For every $\mathcal{O}\in\mathbf{ARO}[\mathscr{F}, m, d]$, we define $w := w(\lambda)$ to be the size of the output of $B_\lambda$, that is, the witness size function.

**Construction 8.2.** Denote $\mathcal{X} := \mathbf{ARO}[\mathscr{F}, m, d]$. The accumulation scheme $\mathsf{AS} = (\mathrm{G}, \mathrm{I}, \mathrm{P}, \mathrm{V}, \mathrm{D})$ for $\mathcal{X}$-queries is specified below. An accumulator $\mathsf{acc}$ is a tuple $(((x_1, y_1, z_1), \gamma_1), ((x_2, y_2, z_2), \gamma_2))$ where each $((x_i, y_i, z_i), \gamma_i)$ is a $\widehat{\mathsf{vo}}$ query-answer pair in $\mathbb{F}^{m+\lambda+w}\times\mathbb{F}$. Note that a predicate input $\mathsf{q}$ is a triple $(\mathsf{oid}, x, y)\in\mathcal{X}_{\mathsf{oid}}\times\mathrm{Dom}(\mathsf{oid})\times\mathrm{Cod}(\mathsf{oid})$, where $\mathcal{X}_{\mathsf{oid}} = \{\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}\}$, is the set of possible oracle identifiers for $\mathcal{X}$ by abuse of notation. [12]

- $\mathrm{G}(1^\lambda)$: Sample a commitment key $\mathsf{ck}\leftarrow\mathsf{CM.Setup}(1^\lambda)$, and output the public parameters $\mathsf{pp} := \mathsf{ck}$.
- $\mathrm{I}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{pp}=\mathsf{ck})$: Output $(\mathsf{apk}, \mathsf{avk}, \mathsf{dk}) := (\mathsf{ck}, \mathsf{ck}, 1^\lambda)$.
- $\mathrm{P}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{apk}=\mathsf{ck}, [\mathsf{q}_i]_{i=1}^n, [\mathsf{acc}_j]_{j=1}^\ell)$:
  1. Sample a $\widehat{\mathsf{vo}}$ query $(x_{n+2\ell+1}, y_{n+2\ell+1}, z_{n+2\ell+1})\leftarrow\mathbb{F}^{m+\lambda+w}$, and set $\zeta_{n+2\ell+1} := \widehat{\mathsf{vo}}(x_{n+2\ell+1}, y_{n+2\ell+1}, z_{n+2\ell+1})$.
  2. Transform the predicate inputs $[\mathsf{q}_i]_{i=1}^n = [(\mathsf{oid}_k, x_k, y_k)]_{k=1}^n$ into the corresponding $\widehat{\mathsf{vo}}$ query-answer pairs $\hat{\mathsf{q}}_i$ as follows. For every $i\in[n]$:
     (a) If $\mathsf{q}_i = (\mathsf{ro}, x_i, y_i)$, then set $a_i := \mathsf{wo}(x_i)$ and $\hat{\mathsf{q}}_i := ((x_i, y_i, a_i), 1)$.
     (b) If $\mathsf{q}_i = (\mathsf{wo}, x_i, z_i)$, then set $a_i := \mathsf{ro}(x_i)$ and $\hat{\mathsf{q}}_i := ((x_i, a_i, z_i), 1)$.
     (c) If $\mathsf{q}_i = (\widehat{\mathsf{vo}}, (x_i, y_i, z_i), \zeta_i)$, then set $a_i := \perp$ and $\hat{\mathsf{q}}_i := ((x_i, y_i, z_i), \zeta_i)$.
  3. Let $Q = [((x_k, y_k, z_k), \zeta_k)]_{k=1}^{n+2\ell+1}$ be the concatenation of the transformed predicate inputs $[\hat{\mathsf{q}}_i]_{i=1}^n$, old accumulators $[\mathsf{acc}_j]_{j=1}^\ell$, and the $\widehat{\mathsf{vo}}$ query-answer pair $((x_{n+2\ell+1}, y_{n+2\ell+1}, z_{n+2\ell+1}), \zeta_{n+2\ell+1})$ from Step 1.

[12] Earlier we defined oracle identifiers as integers, while here we use the names of the oracles.

4. Compute the polynomial $g \in (\mathbb{F}[X])^{m+\lambda+w}$ of degree less than $n+2\ell+1$ that interpolates the $\widehat{\mathsf{vo}}$ queries in $Q$ over the fixed domain $\{b_k\}_{k=1}^{n+2\ell+1}$ (i.e., such that $g(b_k) = (x_k, y_k, z_k)$ for every $k \in [n+2\ell+1]$). Note that $g$ is a vector of $m+\lambda+w$ univariate polynomials from $\mathbb{F}$ to $\mathbb{F}$, where the $i$-th polynomial operates on coordinate $i \in [m+\lambda+w]$.
5. Compute the polynomial $f \in \mathbb{F}[X]$ such that $f(X) \equiv \widehat{\mathsf{vo}}(g(X))$.[13]
6. Sample commitment randomness $\omega$ and then compute the commitment

$$\mathsf{cm} := \mathsf{CM.Commit}(\mathsf{ck}, ((x_1, \ldots, x_{n+2\ell+1}), f); \omega) \in \{0,1\}^m \ .$$

7. Compute $\beta_0 := \mathsf{ro}(\mathsf{cm}) \in \{0,1\}^\lambda$ and $\beta_1 := \mathsf{wo}(\mathsf{cm}) \in \{0,1\}^w$; $\beta_0$ is the Fiat–Shamir challenge.[14] Interpret $\beta_0$ as an element of $\mathbb{F}$. If $\beta_0 \in \{b_1, \ldots, b_{n+2\ell+1}\}$, go to Step 6; except if this has happened $\lambda$ times, in which case we proceed.
8. Compute $(x, y, z) := g(\beta_0)$.
9. Output the new accumulator $\mathsf{acc}$ and accumulation proof $\pi_{\mathrm{V}}$ defined as follows:

$$\mathsf{acc} := \big(((\mathsf{cm}, \beta_0, \beta_1), 1), ((x, y, z), f(\beta_0))\big) \ ,$$
$$\pi_{\mathrm{V}} := \big(((x_{n+2\ell+1}, y_{n+2\ell+1}, z_{n+2\ell+1}), \zeta_{n+2\ell+1}), (a_1, \ldots, a_n), f, \omega\big) \ .$$

- $\mathrm{V}\Big(\mathsf{avk} = \mathsf{ck}, [\mathsf{q}_i]_{i=1}^n, [\mathsf{acc}_j]_{j=1}^\ell, \mathsf{acc} = \big(((\mathsf{cm}, \beta_0, \beta_1), 1), ((x, y, z), \gamma)\big),$
  $\pi_{\mathrm{V}} = \big(((x_{n+2\ell+1}, y_{n+2\ell+1}, z_{n+2\ell+1}), \zeta_{n+2\ell+1}), (a_1, \ldots, a_n), f, \omega\big)\Big)$:
  1. Compute the list $Q = [((x_k, y_k, z_k), \zeta_k)]_{k=1}^{n+2\ell+1}$ and the polynomial $g$ from $[\mathsf{q}_i]_{i=1}^n$, $[\mathsf{acc}_j]_{j=1}^\ell$, and $((x_{n+2\ell+1}, y_{n+2\ell+1}, z_{n+2\ell+1}), \zeta_{n+2\ell+1})$ as $\mathrm{P}$ does, apart from the following: rather than sampling the $(n+2\ell+1)$-th entry of $Q$, use the $\widehat{\mathsf{vo}}$ query-answer pair received in $\pi_{\mathrm{V}}$; and rather than querying $\mathsf{ro}$ and $\mathsf{wo}$ to transform the input predicate $[\mathsf{q}_i]_{i=1}^n$, use $(a_1, \ldots, a_n)$ received in $\pi_{\mathrm{V}}$.
  2. Check that:
     - $\mathsf{cm} = \mathsf{CM.Commit}(\mathsf{ck}, ((x_1, \ldots, x_{n+2\ell+1}), f); \omega)$;
     - $(x, y, z) = g(\beta_0)$; and
     - $\gamma = f(\beta_0)$.
  3. For every $k \in [n+2\ell+1]$, check that $f(b_k) = \zeta_k$.
- $\mathrm{D}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}\Big(\mathsf{dk} = 1^\lambda, \mathsf{acc} = \big(((x_1, y_1, z_1), 1), ((x_2, y_2, z_2), \gamma)\big)\Big)$:
  1. Check that $\widehat{\mathsf{vo}}(x_1, y_1, z_1) = 1$ and $\widehat{\mathsf{vo}}(x_2, y_2, z_2) = \gamma$.

In the next subsections, we prove Theorem 8.1 by analyzing the completeness, soundness, zero knowledge, and efficiency of the construction.

## 8.2 Completeness

The accumulation prover $\mathrm{P}$ receives as input a list of predicate inputs $[\mathsf{q}_i]_{i=1}^n$ and a list of old accumulators $[\mathsf{acc}_j]_{j=1}^\ell$. Suppose that $\Phi^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{q}_i) = 1$ for every $i \in [n]$ and $\mathrm{D}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{acc}_j) = 1$ for every $j \in [\ell]$. Completeness requires showing that $\mathrm{P}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$ outputs a new accumulator $\mathsf{acc}$ and accumulation proof $\pi_{\mathrm{V}}$ that satisfy the following two conditions.

[13] Note that the degree of $f$ is less than $(m+\lambda+w) \cdot d \cdot (n+2\ell+1)$, and so $f$ can be computed by interpolation by evaluating the expression $\widehat{\mathsf{vo}}(g(X))$ at $(m+\lambda+w) \cdot d \cdot (n+2\ell+1)$ points (which involves a corresponding number of queries to $\widehat{\mathsf{vo}}$).

[14] While $\beta_1$ is not used for Fiat–Shamir, the $\beta_1$ value is needed for completeness.

- Condition 1: $\mathrm{V}(\mathsf{avk}, [\mathsf{acc}_j]_{j=1}^{\ell}, \mathsf{acc}, [\mathsf{q}_i]_{i=1}^{n}, \pi_{\mathrm{V}}) = 1$.

  By the construction of V, this holds if the accumulator $\mathsf{acc} = \big(((\mathsf{cm}, \beta_0, \beta_1), 1), ((x, y, z), \gamma)\big)$ and the accumulation proof $\pi_{\mathrm{V}} = \big(((x_{n+2\ell+1}, y_{n+2\ell+1}, z_{n+2\ell+1}), \zeta_{n+2\ell+1}), (a_1, \ldots, a_n), f, \omega\big)$ are such that (i) $\mathsf{cm} = \mathsf{CM.Commit}(\mathsf{ck}, ((x_1, \ldots, x_{n+2\ell+1}), f); \omega)$; (ii) $(x, y, z) = g(\beta_0)$ (where V computes $g$ exactly as P does); (iii) $\gamma = f(\beta_0)$; and (iv) $f(b_k) = \zeta_k$ for every $k \in [n + 2\ell + 1]$. Condition i is satisfied because V computes cm using the same inputs and commmitment key (since $\mathsf{apk} = \mathsf{avk} = \mathsf{ck}$) as $\mathrm{P}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$. Conditions ii and iii are satisfied directly. For Condition iv: $g(b_k) = (x_k, y_k, z_k)$ for all $k \in [n + 2\ell + 1]$ by the definition of $g$. Thus, we have $\widehat{\mathsf{vo}}(g(b_k)) = \widehat{\mathsf{vo}}(x_k, y_k, z_k)$ for every $k \in [n + 2\ell + 1]$. Since we defined $f := \widehat{\mathsf{vo}} \circ g$, we get that $f(b_k) = \zeta_k$ for every $k \in [n + 2\ell + 1]$.

- Condition 2: $\mathrm{D}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{acc}) = 1$.

  This occurs when both elements in acc are valid query-answer pairs for $\widehat{\mathsf{vo}}$. In the first pair, the honest $\mathrm{P}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$ sets $\beta_0 = \mathsf{ro}(\mathsf{cm})$ and $\beta_1 = \mathsf{wo}(\mathsf{cm})$, which means $\widehat{\mathsf{vo}}(\mathsf{cm}, \beta_0, \beta_1) = 1$ by the definition of $\widehat{\mathsf{vo}}$. For the second pair, the honest prover outputs $((x, y, z), f(\beta_0))$, where $g(\beta_0) = (x, y, z)$. This satisfies $\widehat{\mathsf{vo}}(g(\beta_0)) = f(\beta_0)$ because $f \equiv \widehat{\mathsf{vo}} \circ g$.

### 8.3 Soundness

Since AS is an accumulation scheme for $\mathbf{ARO}[\mathscr{F}, m, d]$-queries, we show that the probability below is bounded from above by the expression in the theorem statement:

$$\Pr\left[\begin{array}{c} \mathrm{V}(\mathsf{avk}, [\mathsf{q}_i]_{i=1}^{n}, [\mathsf{acc}_j]_{j=1}^{\ell}, \mathsf{acc}, \pi_{\mathrm{V}}) = 1 \\ \mathrm{D}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{dk}, \mathsf{acc}) = 1 \\ \wedge \\ \exists\, j \in [\ell],\ \mathrm{D}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{dk}, \mathsf{acc}_j) = 0 \ \vee \\ \exists\, i \in [n],\ \Phi^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{q}_i) = 0 \end{array} \middle| \begin{array}{r} (\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O} \\ \mathsf{pp} \leftarrow \mathrm{G}(1^{\lambda}) \\ \begin{pmatrix} [\mathsf{q}_i]_{i=1}^{n} & [\mathsf{acc}_j]_{j=1}^{\ell} \\ \mathsf{acc} & \pi_{\mathrm{V}} \end{pmatrix} \leftarrow \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{pp}) \\ (\mathsf{apk}, \mathsf{avk}, \mathsf{dk}) \leftarrow \mathrm{I}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{pp}) \end{array} \right] . \tag{11}$$

Since $\mathsf{i}_\Phi = \bot$ and $\mathsf{pp}_\Phi = \bot$ in oracle query accumulation schemes, we omit them from the above equation.

In the above equation, let $\mathsf{acc} = \big(((\mathsf{cm}, \beta_0, \beta_1), 1), ((x, y, z), \gamma)\big)$ and $\pi_{\mathrm{V}} = \big(((x_{n+2\ell+1}, y_{n+2\ell+1}, z_{n+2\ell+1}), \zeta_{n+2\ell+1}), (a_1, \ldots, a_n), f, \omega\big)$. We argue that the event in the left side of Equation 11, which we denote by $E$, is equivalent to the condition "$f(X) \not\equiv \widehat{\mathsf{vo}}(g(X))$ and $f(\beta_0) = \widehat{\mathsf{vo}}(g(\beta_0))$".

- First, we show that $f(\beta_0) = \widehat{\mathsf{vo}}(g(\beta_0))$. Note that
  - $\mathrm{D}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{dk}, \mathsf{acc}) = 1$ implies that $\widehat{\mathsf{vo}}(x, y, z) = \gamma$; and
  - $\mathrm{V}(\mathsf{avk}, [\mathsf{q}_i]_{i=1}^{n}, [\mathsf{acc}_j]_{j=1}^{\ell}, \mathsf{acc}, \pi_{\mathrm{V}}) = 1$ implies that $(x, y, z) = g(\beta_0)$ and $\gamma = f(\beta_0)$.

  Then substitution gives $\widehat{\mathsf{vo}}(g(\beta_0)) = f(\beta_0)$.

- Second, we show that $f(X) \not\equiv \widehat{\mathsf{vo}}(g(X))$. Consider the expression

$$\exists\, j \in [\ell],\ \mathrm{D}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{dk}, \mathsf{acc}_j) = 0 \ \vee \exists\, i \in [n],\ \Phi^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{pp}_\Phi, \mathsf{i}_\Phi, \mathsf{q}_i) = 0 \ .$$

  This means there is a query-answer pair $((x_{k^*}, y_{k^*}, z_{k^*}), \zeta_{k^*}) \in Q$ such that $\widehat{\mathsf{vo}}(x_{k^*}, y_{k^*}, z_{k^*}) \neq \zeta_{k^*}$, for some $k^* \in [n + 2\ell + 1]$. By $g$'s definition, we have $(x_k, y_k, z_k) = g(b_k)$ for every $k \in [n + 2\ell + 1]$, so $\widehat{\mathsf{vo}}(g(b_{k^*})) \neq \zeta_{k^*}$. Also since $\mathrm{V}(\mathsf{avk}, [\mathsf{q}_i]_{i=1}^{n}, [\mathsf{acc}_j]_{j=1}^{\ell}, \mathsf{acc}, \pi_{\mathrm{V}}) = 1$, we have $f(b_k) = \zeta_k$ for every $k \in [n + 2\ell + 1]$. Hence, we get $f(b_{k^*}) \neq \widehat{\mathsf{vo}}(g(b_{k^*}))$, and thus conclude that $f(X) \not\equiv \widehat{\mathsf{vo}}(g(X))$.

Next, we wish to invoke our oracle zero-finding game lemma (Lemma 7.1) to bound the probability in Equation 11. We apply Lemma 7.1 with respect to the commitment scheme $\mathsf{CM}'$ obtained by modifying $\mathsf{CM}$ as follows:

$$\mathsf{CM}'.\mathsf{Commit}(\mathsf{ck}, (g, f); \omega) := \mathsf{CM}.\mathsf{Commit}(\mathsf{ck}, ((g(b_1), \ldots, g(b_{n+2\ell+1})), f); \omega) \ .$$

Note that $\mathsf{CM}'$ is binding to $g$ since $g$ has degree less than $n + 2\ell$. Further, since $\mathsf{CM}$ has binding error $\varepsilon_{\mathsf{CM}}(\lambda)$ in the standard model (and hence also in the ROM[15]), Lemma 6.6 states that $\mathsf{CM}$ has binding error $\varepsilon_{\mathsf{CM}}(\lambda) + \varepsilon_{\mathrm{ARO}}(t, \lambda)$ in the AROM.

The winning event in the zero-finding game is that

$$\mathsf{cm} = \mathsf{CM}'.\mathsf{Commit}(\mathsf{ck}, (g, f); \omega) = \mathsf{CM}.\mathsf{Commit}(\mathsf{ck}, ((x_1, \ldots, x_{n+2\ell+1}) \ , f); \omega) \ ,$$

$f(X) \not\equiv \widehat{\mathsf{vo}}(g(X))$, and $f(\beta_0) = \widehat{\mathsf{vo}}(g(\beta_0))$ for $\beta_0 := \mathsf{ro}(\mathsf{cm})$. Since $\mathrm{V}(\mathsf{avk}, [\mathsf{q}_i]_{i=1}^n, [\mathsf{acc}_j]_{j=1}^\ell, \mathsf{acc}, \pi_{\mathrm{V}}) = 1$, we get that $\mathsf{cm} = \mathsf{CM}.\mathsf{Commit}(\mathsf{ck}, ((x_1, \ldots, x_{n+2\ell+1}), f); \omega)$. Also note that $\deg(g) \leq n + 2\ell$. Thus, by applying our oracle zero-finding game lemma, the probability that $f(X) \not\equiv \widehat{\mathsf{vo}}(g(X))$ and $f(\beta_0) = \widehat{\mathsf{vo}}(g(\beta_0))$ is at most

$$\sqrt{t \cdot \left(\frac{2d(n+2\ell)(m+\lambda+w)+1}{|\mathbb{F}|}\right) + \varepsilon_{\mathsf{CM}}(\lambda) + \varepsilon_{\mathrm{ARO}}(t,\lambda)} + \frac{1}{|\mathbb{F}|} + \varepsilon_{\mathrm{ARO}}(t,\lambda) + \frac{t}{2^\lambda} \ .$$

Since the event $E$ is equivalent to $f(X) \not\equiv \widehat{\mathsf{vo}}(g(X))$ and $f(\beta_0) = \widehat{\mathsf{vo}}(g(\beta_0))$, the above probability is an upper bound on the probability in Equation 11, as desired.

## 8.4 Zero knowledge

We describe a zero-knowledge simulator S that has access to $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}})$ and simulates: (i) the scheme's public parameters $\mathsf{pp}$; and (ii) the distribution of P's accumulator $\mathsf{acc}$ without access to P's inputs (the predicate inputs $[\mathsf{q}_i]_{i=1}^n$ and old accumulators $[\mathsf{acc}_j]_{j=1}^\ell$). Then we show that if the commitment $\mathsf{CM}$ is statistically (resp. computationally) hiding, then the simulated joint distribution $((\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}), \mathsf{pp}, \mathsf{acc})$ is statistically (resp. computationally) indistinguishable from $((\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}), \mathsf{pp}, \mathsf{acc})$ produced in the real protocol.

- *Parameter generation:* $\mathrm{S}(1^\lambda) \to \mathsf{pp}$.
  1. Sample $\mathsf{ck} \leftarrow \mathsf{CM}.\mathsf{Setup}(1^\lambda)$.
  2. Output $\mathsf{pp} := \mathsf{ck}$ (and store $\mathsf{ck}$ in the internal state).
- *Proving:* $\mathrm{S}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{pp}_\Phi = \bot, \mathsf{i}_\Phi = \bot) \to \mathsf{acc}$.
  1. Sample commitment randomness $\omega$ and compute a commitment $\mathsf{cm} := \mathsf{CM}.\mathsf{Commit}(\mathsf{ck}, ((0, \ldots, 0), f'); \omega)$ where $f' \in \mathbb{F}[X]$ is the zero polynomial (appropriately padded).
  2. Query $\mathsf{ro}$ and $\mathsf{wo}$ to compute $\beta_0 := \mathsf{ro}(\mathsf{cm})$ and $\beta_1 := \mathsf{wo}(\mathsf{cm})$. If $\beta_0 \in \{b_1, \ldots, b_{n+2\ell}\}$, resample $\omega$ and recompute $\mathsf{cm}$ until this is not the case. Abort if we resample more than $\kappa := \lambda/(\log|\mathbb{F}| - \log(n+2\ell))$ times.[16]
  3. Sample a random $\widehat{\mathsf{vo}}$ query $(x, y, z) \leftarrow \mathbb{F}^{m+\lambda+w}$.
  4. Query the oracle $\widehat{\mathsf{vo}}$ to compute $\gamma := \widehat{\mathsf{vo}}(x, y, z)$.

[15] Otherwise, we can construct a standard-model adversary for $\mathsf{CM}$ via running the adversary for $\mathsf{CM}$ in the ROM and answering $\mathsf{ro}$-queries using a "lazily sampled" evaluation table for $\mathsf{ro}$.

[16] If the field $\mathbb{F}$ is of superpolynomial size (and $n, \ell$ are polynomially bounded) then resampling is not necessary.

5. Output the accumulator $\mathsf{acc} := \big(((\mathsf{cm}, \beta_0, \beta_1), 1), ((x, y, z), \gamma)\big)$.

Observe that the probability that the simulator aborts in Step 2 is at most $\mathrm{negl}(\lambda)$. By Claim 3.7, with all but negligible probability, all of the sampled $\mathsf{cm}$ are distinct; hence the probability that $\mathsf{ro}(\mathsf{cm}) \in \{b_1, \ldots, b_{n+2\ell}\}$ for all sampled $\mathsf{cm}$ is at most $\left(\frac{n+2\ell}{|\mathbb{F}|}\right)^{\kappa} + \mathrm{negl}(\lambda) = \mathrm{negl}(\lambda)$.

We argue that $((\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}), \mathsf{pp}, \mathsf{acc})$ above is indistinguishable from that produced by the accumulation prover. Since $\mathrm{S}$ does not program the oracle, we fix an oracle $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}$, then argue that $\mathsf{pp}$ and $\mathsf{acc}$ output by $\mathrm{S}$ are distributed correctly, given $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}})$.

First, the real and simulated $\mathsf{pp}$ have the same distribution: they are both commitment keys $\mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda)$.

Next, we show that the real and simulated $\mathsf{acc} = \big(((\mathsf{cm}, \beta_0, \beta_1), 1), ((x, y, z), \gamma)\big)$ are indistinguishable. It suffices to argue the indistinguishability of $\mathsf{cm}$ and $(x, y, z)$ (between the real protocol and the simulation), because $\beta_0 = \mathsf{ro}(\mathsf{cm}), \beta_1 = \mathsf{wo}(\mathsf{cm})$ and $\gamma = \widehat{\mathsf{vo}}(x, y, z)$ in both the real protocol and the simulation.

- $\mathsf{cm}$: Since $\mathsf{CM}$ is (statistically/computationally) hiding, $\mathsf{cm}$ is (statistically/computationally) indistinguishable from a commitment to any other message of the same length. This is true even if the adversary is given many independent commitments to the same message. Note that apart from the choice of message, the prover and simulator sample $\mathsf{cm}$ in the same way.
- $(x, y, z)$: Since $\beta_0 \notin \{b_1, \ldots, b_{n+2\ell}\}$ (else the simulator aborts), in the real protocol, as in the simulation, $g(\beta_0)$ is uniformly random in $\mathbb{F}^{m+\lambda+w}$. This follows from a standard algebraic fact stated below.

  *Fact* 8.3. Let $t \in \mathbb{N}$ and let $b_1, \ldots, b_{N+1} \in \mathbb{F}$ be distinct field elements, and let $x_1, \ldots, x_N \in \mathbb{F}^t$. Sample $x_{N+1} \leftarrow \mathbb{F}^t$ uniformly and construct $g$ to be the (unique) interpolation of the points $(b_1, x_1), \ldots, (b_{N+1}, x_{N+1}) \in \mathbb{F} \times \mathbb{F}^t$ of minimal degree. Then for any point $\beta \leftarrow \mathbb{F} \setminus \{b_1, \ldots, b_N\}$, $g(\beta)$ is uniformly random in $\mathbb{F}^t$.

## 8.5 Efficiency

We discuss the efficiency of Construction 8.2.

- *Generator.* Efficiency follows from the efficiency of $\mathsf{CM.Setup}$, which takes $\mathrm{poly}(\lambda)$ time.
- *Indexer.* This takes $\mathrm{poly}(\lambda)$ time.
- *Accumulation prover.* Running $\mathrm{P}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$ involves making at most $n$ queries (where each query is either to $\mathsf{ro}$ or $\mathsf{wo}$), computing the polynomial $g$ via polynomial interpolation, committing to $(f, g)$, and computing the polynomial $f$ via evaluating $g$ at $(((m + \lambda + w) \cdot d) - 1) \cdot (n + 2\ell + 1)$ query points,[17] making $(m + \lambda + w) \cdot d \cdot (n + 2\ell + 1)$ queries to $\mathsf{ro}$, then interpolating. These operations can be accomplished in polynomial time in $\lambda$.
- *Accumulator size.* Consider the first query-answer pair $((\mathsf{cm}, \mathsf{ro}(\mathsf{cm}), \mathsf{wo}(\mathsf{cm})), 1)$ in $\mathsf{acc}$. Since the commitment scheme $\mathsf{CM}$ has commitment size $m$, we know that $\mathsf{cm} \in \{0,1\}^m \subseteq \mathbb{F}^m$. Then, the query-answer pair $((\mathsf{cm}, \mathsf{ro}(\mathsf{cm}), \mathsf{wo}(\mathsf{cm})), 1)$ is in $\mathbb{F}^{m+\lambda+w} \times \mathbb{F}$. The second query-answer pair in $\mathsf{acc}$ is $((x, y, z), f(\beta_0))$, which is in $\mathbb{F}^{m+\lambda+w} \times \mathbb{F}$ due to the domain of $\widehat{\mathsf{vo}}$ and the definition of $f$.
- *Accumulation proof size.* The accumulation proof $\pi_{\mathrm{V}}$ includes (i) a $\widehat{\mathsf{vo}}$ query-answer pair $((x_{n+2\ell+1}, y_{n+2\ell+1}, z_{n+2\ell+1}), \zeta_{n+2\ell+1})$, which can be represented with $m + \lambda + w + 1$ elements of $\mathbb{F}$; (ii) advice values $(a_1, \ldots, a_n)$, which can be represented with $\leq n \cdot \max\{\lambda, w\}$ elements of $\mathbb{F}$; (iii) a single-variate polynomial $f$ of degree

[17] This assumes that we reuse the $g$ query-answer pairs $(b_i, (x_i, y_i, z_i))$ from the prover's Step 4.

at most $(m + \lambda + w) \cdot d \cdot (n + 2\ell + 1)$, which can be represented with $(m + \lambda + w) \cdot d \cdot (n + 2\ell + 1) + 1$ elements of $\mathbb{F}$; and (iv) commitment randomness $\omega$, which is a bitstring of $\mathrm{poly}(\lambda)$ length.

- *Accumulation verifier.* The accumulation verifier $\mathrm{V}$ computes the polynomial $g$ via interpolation, the commitment $\mathsf{CM.Commit}(\mathsf{ck}, ((x_1, \ldots, x_{n+2\ell+1}), f); \omega)$, a single evaluation of $g$, and $n + 2\ell + 2$ evaluations of $f$. Note that $\mathrm{V}$ makes no oracle queries.
- *Decider.* $\mathrm{D}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$ makes 2 queries to $\widehat{\mathsf{vo}}$.

# 9 PCD in the AROM

We describe how to combine our results to construct PCD in the AROM, proving our main theorem.

## 9.1 SNARKs in the AROM

We argue that a (zk)SNARK in the ROM is also a (zk)SNARK (where honest parties only query the random oracle) in the AROM. In Lemma 9.1 we prove that straightline knowledge soundness is preserved in the AROM. Afterwards, in Lemma 9.2 we prove that zero knowledge of a SNARK in the ROM is preserved in the AROM.

### 9.1.1 Knowledge soundness

**Lemma 9.1.** *Let $\mathscr{F} = \{\mathbb{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of fields, $m\colon \mathbb{N}\to\mathbb{N}$ an arity function, $d\colon \mathbb{N}\to\mathbb{N}$ a degree function such that $d(\lambda)\geq 2$ for a security parameter $\lambda\in\mathbb{N}$, and $\mathcal{X} := \mathbf{ARO}[\mathscr{F}, m, d]$.*

*Let $\mathsf{ARG} = (\mathcal{G}, \mathcal{I}, \mathcal{P}, \mathcal{V})$ be a SNARK in the ROM for a relation $\mathcal{R}$, where the random oracle has arity $\leq m(\lambda)$. Suppose $\mathsf{ARG}$ has straightline knowledge extraction error $\kappa(\lambda, t)$, in which $t\in\mathbb{N}$.*

*Then, $\mathsf{ARG}$ is a SNARK relative to $\mathcal{X}$ for $\mathcal{R}$ (unconditionally) with straightline knowledge extraction error at most $\kappa(\lambda, t\cdot\mathrm{poly}(\lambda)) + \frac{t+\mathrm{poly}(\lambda,|\mathbb{x}|)}{2^\lambda}$, where $t$ is the number of queries made by the adversary in the AROM and $|\mathbb{x}|$ is the size of an instance in $\mathcal{L}(\mathcal{R})$.*

*Proof.* Let $\mathsf{ARG} = (\mathcal{G}, \mathcal{I}, \mathcal{P}, \mathcal{V})$ be a SNARK in the ROM for $\mathcal{R}$ with straightline knowledge extraction error $\kappa(\lambda, t)$. We argue that $\mathsf{ARG}$, with access to $\mathcal{O}\in\mathbf{ARO}[\mathscr{F}, m, d]$, has straightline knowledge extraction error at most $\kappa(\lambda, O(t^2)\cdot\mathrm{poly}(\lambda)) + \frac{t}{2^\lambda}$.

By the definition of straightline knowledge soundness for $\mathsf{ARG}$, there exists a knowledge extractor $\mathcal{E}$ such that for every malicious prover $\tilde{\mathcal{P}}$:

$$\Pr\left[\begin{array}{c|r} \mathcal{V}^{\mathsf{ro}}(\mathsf{ivk}, \mathbb{x}, \pi) = 1 \\ \wedge \\ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R} \end{array} \middle| \begin{array}{r} \mathsf{ro} \leftarrow \mathcal{U}(m(\lambda), \lambda) \\ \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ \mathsf{ai} \leftarrow \mathcal{D}(\mathsf{pp}) \\ (\mathbb{i}, \mathbb{x}, \pi) \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}^{\mathsf{ro}}(\mathsf{pp}, \mathsf{ai}) \\ (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}^{\mathsf{ro}}(\mathsf{pp}, \mathbb{i}) \\ \mathbb{w} \leftarrow \mathcal{E}(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \pi, \mathsf{tr}) \end{array}\right] \leq \kappa(\lambda, t_{\tilde{\mathcal{P}}}) \ , \tag{12}$$

in which $t_{\tilde{\mathcal{P}}} = |\mathsf{tr}|$.

Define a predicate $\mathsf{p}^{\mathsf{ro}}(x)$ corresponding to the game that $\tilde{\mathcal{P}}$ attempts to win in Equation 12:

1. Parse $x$ as $(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \pi, \mathsf{tr})$.
2. Run $(\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}^{\mathsf{ro}}(\mathsf{pp}, \mathbb{i})$.
3. Run $\mathcal{E}(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \pi, \mathsf{tr}|_{\mathsf{ro}})$, in which $\mathsf{tr}|_{\mathsf{ro}}$ denotes $\mathsf{tr}$ restricted to $\mathsf{ro}$ queries.
4. Output 1 if $\mathcal{V}^{\mathsf{ro}}(\mathsf{ivk}, \mathbb{x}, \pi) = 1$ and $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}$. Otherwise, output 0.

Let $t_{\mathsf{p}}$ denote the query complexity of $\mathsf{p}^{\mathsf{ro}}$. By the succinctness of $\mathsf{ARG}$ (Section 3.3) and the efficiency of $\mathcal{I}$, we have $t_{\mathsf{p}} = \mathrm{poly}(\lambda, |\mathbb{x}|)$.

Next, let $\tilde{\mathcal{P}}_{\mathrm{ARO}}$ be any $t$-query malicious prover in the AROM. For some $\delta \in [0,1)$, we have

$$\Pr\left[\begin{array}{l|r} \mathsf{p}^{\mathsf{ro}}(\mathsf{pp},\mathbb{i},\mathbb{x},\pi,\mathsf{tr})=1 & (\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}}) \leftarrow \mathcal{O} \\ & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ & \mathsf{ai} \leftarrow \mathcal{D}(\mathsf{pp}) \\ & (\mathbb{i},\mathbb{x},\pi) \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{pp},\mathsf{ai}) \end{array}\right] > \delta \ . \tag{13}$$

Note that the probability in Equation 13 is equivalent to the straightline knowledge soundness property for ARG in the AROM.

Now, we invoke Theorem 6.5 and the knowledge soundness property of ARG to upper bound $\delta$. Define an adversary $\mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$ that runs the right side of Equation 13, excluding the first line, and outputs $(\mathsf{pp},\mathbb{i},\mathbb{x},\pi,\mathsf{tr})$. Note that $\mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$ makes $t_{\mathcal{A}} = t$ oracle queries because only $\tilde{\mathcal{P}}_{\mathrm{ARO}}$ makes oracle queries. By Theorem 6.5, there exists an adversary $\mathcal{C}$, with access to ro and straightline access to $\mathcal{A}$, so that

$$\Pr\left[\mathsf{p}^{\mathsf{ro}}(x)=1 \;\middle|\; \begin{array}{c} \mathsf{ro} \leftarrow \mathcal{U}(m(\lambda),\lambda) \\ x \leftarrow \mathcal{C}^{(\mathsf{ro},\mathcal{A})} \end{array}\right] \geq \delta - \varepsilon_{\mathrm{ARO}}(t+t_{\mathsf{p}},\lambda) \ ,$$

in which $\varepsilon_{\mathrm{ARO}}(t+t_{\mathsf{p}},\lambda)$ is the emulation error of a pass-through stateful $(\mathcal{O},\{\mathsf{wo},\widehat{\mathsf{vo}}\})$-emulator $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}$. Specifically, $\mathcal{C}^{(\mathsf{ro},\mathcal{A})} := \mathcal{A}^{\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}}$, and $\mathcal{C}^{(\mathsf{ro},\mathcal{A})}$ makes at most $t \cdot t_{\mathcal{M}}$ queries, in which $t_{\mathcal{M}}$ denotes the per-query query complexity of $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}$.

Observe that both $\mathcal{A}$ and $\mathcal{C}$ run $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$ and $\mathsf{ai} \leftarrow \mathcal{D}(\mathsf{pp})$, then differ afterwards; interpret the differing code in $\mathcal{C}$ as a malicious prover $\tilde{\mathcal{P}}^{\mathsf{ro}}$ (which is in the ROM). Then, Equation 12 implies $\kappa(\lambda, t\cdot t_{\mathcal{M}}) \geq \delta - \varepsilon_{\mathrm{ARO}}(t+t_{\mathsf{p}},\lambda)$. Rearranging gives

$$\delta \leq \kappa(\lambda, t\cdot t_{\mathcal{M}}) + \varepsilon_{\mathrm{ARO}}(t+t_{\mathsf{p}},\lambda) \ . \tag{14}$$

By Theorem 5.4, there exists a pass-through stateful $(\mathcal{O},\{\widehat{\mathsf{vo}}\})$-emulator with query complexity $O(1)$ and with emulation error $\varepsilon_{\mathrm{ARO}}(t+t_{\mathsf{p}},\lambda) = \frac{t+t_{\mathsf{p}}}{2^\lambda}$. Thus by Lemma 6.3, $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}$ makes $t_{\mathcal{M}} := O(t_B) = \mathrm{poly}(\lambda)$ oracle queries per emulated query, where the equality is due to the definition of $\mathbf{ARO}[\mathscr{F},m,d]$ (Definition 4.3). Plugging these values into Equation 14 yields the desired result. □

### 9.1.2 Zero-knowledge

We show that zero-knowledge SNARKs in the ROM remain zero-knowledge in the AROM.

**Lemma 9.2.** *Let $\mathscr{F} = \{\mathbb{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of fields, $m\colon \mathbb{N}\to\mathbb{N}$ an arity function, and $d\colon \mathbb{N}\to\mathbb{N}$ a degree function such that $d(\lambda)\geq 2$ for a security parameter $\lambda\in\mathbb{N}$.*

*Let $\mathsf{ARG} = (\mathcal{G},\mathcal{I},\mathcal{P},\mathcal{V})$ be a zero-knowledge SNARK in the ROM for a relation $\mathcal{R}$, where the random oracle has arity $\leq m(\lambda)$, with zero-knowledge simulation error at most $\varepsilon_{\mathsf{ZK}}(\lambda)$. Then, $\mathsf{ARG}$ is a SNARK relative to $\mathcal{X} := \mathbf{ARO}[\mathscr{F},m,d]$ for $\mathcal{R}$ with zero-knowledge simulation error at most $\varepsilon_{\mathsf{ZK}}(\lambda) + \frac{t}{2^{\lambda-1}}$, where $t$ is the number of oracle queries made by a stateful adversary in the AROM.*

*Proof.* Let $\mathsf{ARG} = (\mathcal{G},\mathcal{I},\mathcal{P},\mathcal{V})$ be a zero-knowledge SNARK in the ROM. We argue that ARG maintains zero-knowledge in the AROM; for $\mathcal{O} = \mathrm{ARO}[\mathbb{F},m,\lambda,d,B] \in \mathcal{X}$, we show that there exists an efficient simulator $\mathcal{S}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$ such that for all stateful honest $t$-query adversaries $\mathcal{A}$, the following distributions are

$\varepsilon_{\mathsf{ZK}}(\lambda) + 2 \cdot \varepsilon_{\mathrm{ARO}}(t, \lambda)$-close, where $\varepsilon_{\mathrm{ARO}}(t, \lambda)$ is the $\mathcal{O}$-emulation error:

$$\mathcal{D}_{\mathrm{ARO}} := \left\{ \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\pi) \;\middle|\; \begin{array}{r} (\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathbb{i},\mathbb{x},\mathbb{w}) \leftarrow \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{pp}) \\ (\mathsf{ipk},\mathsf{ivk}) \leftarrow \mathcal{I}^{\mathsf{ro}}(\mathsf{pp},\mathbb{i}) \\ \pi \leftarrow \mathcal{P}^{\mathsf{ro}}(\mathsf{ipk},\mathbb{x},\mathbb{w}) \end{array} \right\}$$

$$\text{and} \quad \mathcal{D}_{\mathrm{ZK}} := \left\{ \mathcal{A}^{\mathcal{S}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}}(\pi) \;\middle|\; \begin{array}{r} (\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathcal{S}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(1^\lambda) \\ (\mathbb{i},\mathbb{x},\mathbb{w}) \xleftarrow{\mathsf{tr}} \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{pp}) \\ \pi \leftarrow \mathcal{S}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathbb{i},\mathbb{x},\mathsf{tr}) \end{array} \right\}.$$

We may assume without loss of generality that the second stage of $\mathcal{A}$ outputs a single bit. Note that in $\mathcal{D}_{\mathrm{ARO}}$, the indexer $\mathcal{I}$ and prover $\mathcal{P}$ only access $\mathsf{ro}$ because we want to argue that ARG, which is defined in the ROM, remains zero-knowledge in the AROM. Further, since $\mathcal{A}$ is honest, we have $(\mathbb{i},\mathbb{x},\mathbb{w}) \in \mathcal{R}$ in both $\mathcal{D}_{\mathrm{ARO}}$ and $\mathcal{D}_{\mathrm{ZK}}$.

Let $\mathcal{S}$ be the zero-knowledge simulator for ARG. Further, let $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}$ be a pass-through stateful $(\mathcal{O}, \{\mathsf{wo}, \widehat{\mathsf{vo}}\})$-emulator with error $\varepsilon_{\mathrm{ARO}}(t, \lambda)$, which exists due to Lemma 6.3. Define the (stateful) zero-knowledge simulator $\mathcal{S}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$ for ARG in the AROM as follows:

- *Parameter generation:* $\mathcal{S}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(1^\lambda) \to \mathsf{pp}$.
  1. Output $\mathsf{pp} \leftarrow \mathcal{S}(1^\lambda)$.
- *Proving:* $\mathcal{S}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathbb{i},\mathbb{x},\mathsf{tr}) \to \pi$.
  1. Run $\pi \leftarrow \mathcal{S}^{\mathsf{ro}}(\mathbb{i},\mathbb{x},\mathsf{tr}|_{\mathsf{ro}})$, in which $\mathsf{tr}|_{\mathsf{ro}}$ denotes the restriction of $\mathsf{tr}$ to $\mathsf{ro}$ query-answer pairs.
  2. Output $\pi$.
- *Query responses:* $\mathcal{S}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{tr}_i, (\mathsf{oid}, x)) \to y$.
  1. Run $(\mathsf{tr}_{i+1}, y) \leftarrow \mathcal{M}_{\mathrm{ARO}}^{\mathcal{S}^{\mathsf{ro}}(\mathsf{tr}_i|_{\mathsf{ro}})}(\mathsf{tr}_i, (\mathsf{oid}, x))$. Note that $\mathcal{S}^{\mathsf{ro}}$ simulates responses to the $\mathsf{ro}$ queries of $\mathcal{M}_{\mathrm{ARO}}$, conditioned on $\mathsf{tr}_i|_{\mathsf{ro}}$, which is $\mathsf{tr}_i$ restricted to $\mathsf{ro}$ queries.
  2. Output $y$.

Note that $\mathcal{S}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$ never queries $\mathsf{wo}$ or $\widehat{\mathsf{vo}}$, so below we write $\mathcal{S}_{\mathrm{ARO}}^{\mathsf{ro}} := \mathcal{S}_{\mathrm{ARO}}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$. Below, we write $\mathcal{A}^{\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}(\mathsf{tr})}$ to mean that $\mathcal{A}$ has query access to an emulator $\mathcal{M}_{\mathrm{ARO}}^{\mathsf{ro}}(\mathsf{tr})$ that is initialized with $\mathsf{tr}$; if $\mathcal{A}$ makes more than one query, then subsequent runs of the emulator are initialized with a transcript containing all query-answer pairs that $\mathcal{A}$ has seen.

Next, we use a hybrid argument to argue the indistinguishability of $\mathcal{D}_{\mathrm{ARO}}$ and $\mathcal{D}_{\mathrm{ZK}}$.

The hybrids are as follows. Below, we use blue text to denote changes from the previous hybrid (i.e. blue text in hybrid $\mathsf{Hyb}_{i+1}$ are the changes from hybrid $\mathsf{Hyb}_i$. Further, we write $\mathsf{tr}$ to denote the $\mathcal{O}$-query-answer transcript of $\mathcal{A}$.

- $\mathbf{H}_0$: $\mathcal{A}$'s view is defined as in $\mathcal{D}_{\mathrm{ARO}}$.

$$\mathcal{D}_{\mathrm{ARO}} := \left\{ \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\pi) \;\middle|\; \begin{array}{r} (\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathbb{i},\mathbb{x},\mathbb{w}) \leftarrow \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{pp}) \\ (\mathsf{ipk},\mathsf{ivk}) \leftarrow \mathcal{I}^{\mathsf{ro}}(\mathsf{pp},\mathbb{i}) \\ \pi \leftarrow \mathcal{P}^{\mathsf{ro}}(\mathsf{ipk},\mathbb{x},\mathbb{w}) \end{array} \right\}.$$

- $\mathbf{H}_1$: $\mathcal{A}$'s view is defined as in $\mathcal{D}_{\mathrm{ARO}}$, except that $\mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$ is replaced by $\mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}(\perp)}$. That is, the distribution is:

$$\mathcal{D}_1 := \left\{ \mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}(\mathsf{tr})}(\pi) \;\middle|\; \begin{array}{r} \mathsf{ro} \leftarrow \mathcal{U}(m(\lambda),\lambda) \\ \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathbb{i},\mathbb{x},\mathbb{w}) \xleftarrow{\mathsf{tr}} \mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}(\perp)}(\mathsf{pp}) \\ (\mathsf{ipk},\mathsf{ivk}) \leftarrow \mathcal{I}^{\mathsf{ro}}(\mathsf{pp},\mathbb{i}) \\ \pi \leftarrow \mathcal{P}^{\mathsf{ro}}(\mathsf{ipk},\mathbb{x},\mathbb{w}) \end{array} \right\} .$$

- $\mathbf{H}_2$: $\mathcal{A}$'s view is defined as:

$$\mathcal{D}_2 := \left\{ \mathcal{A}^{\mathcal{M}^{\mathcal{S}^{\mathsf{ro}}(\mathsf{tr}|_{\mathsf{ro}})}_{\mathrm{ARO}}(\mathsf{tr})}(\pi) \;\middle|\; \begin{array}{r} \mathsf{ro} \leftarrow \mathcal{U}(m(\lambda),\lambda) \\ \mathsf{pp} \leftarrow \mathcal{S}(1^\lambda) \\ (\mathbb{i},\mathbb{x},\mathbb{w}) \xleftarrow{\mathsf{tr}} \mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}(\perp)}(\mathsf{pp}) \\ \pi \leftarrow \mathcal{S}^{\mathsf{ro}}(\mathbb{i},\mathbb{x},\mathsf{tr}|_{\mathsf{ro}}) \end{array} \right\} ,$$

  in which $\mathcal{S}^{\mathsf{ro}}$ is the zero-knowledge simulator for ARG. Observe that $\mathsf{tr}|_{\mathsf{ro}}$ is the query-answer transcript of the composed adversary $\mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}(\perp)}$.
- $\mathbf{H}_3$: $\mathcal{A}$'s view is:

$$\mathcal{D}_3 := \left\{ \mathcal{A}^{\mathcal{S}^{\mathsf{ro}}_{\mathrm{ARO}}(\mathsf{tr})}(\pi) \;\middle|\; \begin{array}{r} \mathsf{ro} \leftarrow \mathcal{U}(m(\lambda),\lambda) \\ \mathsf{pp} \leftarrow \mathcal{S}_{\mathrm{ARO}}(1^\lambda) \\ (\mathbb{i},\mathbb{x},\mathbb{w}) \xleftarrow{\mathsf{tr}} \mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}(\perp)}(\mathsf{pp}) \\ \pi \leftarrow \mathcal{S}^{\mathsf{ro}}_{\mathrm{ARO}}(\mathbb{i},\mathbb{x},\mathsf{tr}) \end{array} \right\} .$$

- $\mathbf{H}_4$: $\mathcal{A}$'s view is defined as in $\mathcal{D}_{\mathrm{ZK}}$ running with $\mathcal{S}^{\mathsf{ro}}_{\mathrm{ARO}}$.

$$\mathcal{D}_{\mathrm{ZK}} := \left\{ \mathcal{A}^{\mathcal{S}^{\mathsf{ro}}_{\mathrm{ARO}}(\mathsf{tr})}(\pi) \;\middle|\; \begin{array}{r} (\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda) \\ \mathsf{pp} \leftarrow \mathcal{S}_{\mathrm{ARO}}(1^\lambda) \\ (\mathbb{i},\mathbb{x},\mathbb{w}) \xleftarrow{\mathsf{tr}} \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{pp}) \\ \pi \leftarrow \mathcal{S}^{\mathsf{ro}}_{\mathrm{ARO}}(\mathbb{i},\mathbb{x},\mathsf{tr}) \end{array} \right\} .$$

Next, we bound the statistical distance between the hybrids.

**$\mathbf{H_0}$ vs. $\mathbf{H_1}$.** We argue that $\mathbf{H}_0$ and $\mathbf{H}_1$ have distance at most $\varepsilon_{\mathrm{ARO}}(t,\lambda)$. Given $(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda)$, define a stateful adversary $\mathcal{A}_{\mathsf{Thm}}$, as follows:

1. Run $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$.
2. Run $(\mathbb{i},\mathbb{x},\mathbb{w}) \leftarrow \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}$.
3. Run $(\mathsf{ipk},\mathsf{ivk}) \leftarrow \mathcal{I}^{\mathsf{ro}}(\mathsf{pp},\mathbb{i})$.
4. Run $\pi \leftarrow \mathcal{P}^{\mathsf{ro}}(\mathsf{ipk},\mathbb{x},\mathbb{w})$.
5. Output $b \leftarrow \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\pi)$.

Observe that $\mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}_{\mathsf{Thm}}$ is exactly $\mathcal{D}_{\mathrm{ARO}}$ and $\mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}}_{\mathsf{Thm}}$ is exactly $\mathcal{D}_1$. Hence the statistical distance between $\mathbf{H}_0$ and $\mathbf{H}_1$ is at most

$$\left| \Pr\left[ \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}_{\mathsf{Thm}} = 1 \;\middle|\; (\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda) \right] - \Pr\left[ \mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}}_{\mathsf{Thm}} = 1 \;\middle|\; \mathsf{ro} \leftarrow \mathcal{U}(m,\lambda) \right] \right| . \tag{15}$$

Invoking Corollary 6.4, Equation 15 is upper bounded by $\varepsilon_{\mathrm{ARO}}(t,\lambda)$.

**$\mathbf{H_1}$ vs. $\mathbf{H_2}$.** We argue that $\mathbf{H}_1$ and $\mathbf{H}_2$ have statistical distance at most $\varepsilon_{\mathsf{ZK}}(\lambda)$. Observe that the composed adversary $\mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}(\bot)}$ only queries $\mathsf{ro}$.

Next, we argue that, in $\mathbf{H}_1$, $\mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}(\bot)}(\mathsf{pp})$ is honest, i.e. outputs $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ with probability $\geq 1-\mathrm{negl}(\lambda)$. Since $\mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{pp})$ is honest, it outputs $x = (\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ with probability $\delta \geq 1-\mathrm{negl}(\lambda)$. Next, we invoke Theorem 6.5, with a predicate $\mathsf{p}(x)$ checking that $x \in \mathcal{R}$, on $\mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{pp})$ to get the transformed adversary $\mathcal{C}^{(\mathsf{ro},\mathcal{A})} := \mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}(\bot)}(\mathsf{pp})$; the output $x$ of $\mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}(\bot)}(\mathsf{pp})$ satisfies $x \in \mathcal{R}$ with probability $\geq \delta - \varepsilon_{\mathrm{ARO}}(t,\lambda) \geq (1 - \mathrm{negl}(\lambda)) - \varepsilon_{\mathrm{ARO}}(t,\lambda)$. Setting the emulation error $\varepsilon_{\mathrm{ARO}}(t,\lambda) := \frac{t}{2^\lambda}$, as computed in Theorem 5.4, implies that $\mathcal{C}^{(\mathsf{ro},\mathcal{A})}$ outputs a valid $x \in \mathcal{R}$ with probability $\geq 1 - \mathrm{negl}(\lambda)$.

Thus, we can invoke $\mathsf{ARG}$'s zero-knowledge property with respect to $\mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}(\bot)}$; given the zero-knowledge simulator $\mathcal{S}^{\mathsf{ro}}$ for $\mathsf{ARG}$, we can syntactically update distribution $\mathcal{D}_1$ to (a) replace $\mathcal{P}^{\mathsf{ro}}$ with $\mathcal{S}^{\mathsf{ro}}$; and (b) after $\pi$ is produced, have $\mathcal{S}^{\mathsf{ro}}(\mathsf{tr}|_{\mathsf{ro}})$ answer the composed adversary's $\mathsf{ro}$ queries. This updated distribution is exactly the distribution $\mathcal{D}_2$ in $\mathbf{H}_2$, which proves the claim.

**$\mathbf{H_2}$ vs. $\mathbf{H_3}$.** $\mathbf{H}_2$ is obtained from $\mathbf{H}_3$ by expanding the definition of $\mathcal{S}^{\mathsf{ro}}_{\mathrm{ARO}}$, so the two hybrids are identical.

**$\mathbf{H_3}$ vs. $\mathbf{H_4}$.** The difference between the hybrids is that, in $\mathbf{H}_4$, $\mathcal{A}$ accesses the real $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}})$ when outputting $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ versus, in $\mathbf{H}_3$, $\mathcal{A}$ accesses the emulator $\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}(\bot)$. We bound the statistical distance between $\mathbf{H}_3$ and $\mathbf{H}_4$.

Given $(\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda)$, define a stateful adversary $\mathcal{A}_{\mathsf{Thm}}$ that does the following:

1. Run $\mathsf{pp} \leftarrow \mathcal{S}^{\mathsf{ro}}_{\mathrm{ARO}}(1^\lambda)$.
2. Run $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \xleftarrow{\mathsf{tr}} \mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}(\mathsf{pp})$.
3. Run $\pi \leftarrow \mathcal{S}^{\mathsf{ro}}_{\mathrm{ARO}}(\mathbb{i}, \mathbb{x}, \mathsf{tr})$.
4. Output $b \leftarrow \mathcal{A}^{\mathcal{S}^{\mathsf{ro}}_{\mathrm{ARO}}(\mathsf{tr})}(\pi)$.

Observe that $\mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}_{\mathsf{Thm}}$ is exactly $\mathcal{D}_{\mathrm{ZK}}$ and $\mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}}_{\mathsf{Thm}}$ is exactly $\mathcal{D}_3$. Hence the statistical distance between $\mathbf{H}_3$ and $\mathbf{H}_4$ is at most

$$\left| \Pr\left[\mathcal{A}^{(\mathsf{ro},\mathsf{wo},\widehat{\mathsf{vo}})}_{\mathsf{Thm}} = 1 \;\middle|\; (\mathsf{ro}, \mathsf{wo}, \widehat{\mathsf{vo}}) \leftarrow \mathcal{O}(\lambda) \right] - \Pr\left[\mathcal{A}^{\mathcal{M}^{\mathsf{ro}}_{\mathrm{ARO}}(\bot)}_{\mathsf{Thm}} = 1 \;\middle|\; \mathsf{ro} \leftarrow \mathcal{U}(m,\lambda) \right] \right| . \tag{16}$$

By Corollary 6.4, Equation 16 is upper bounded by $\varepsilon_{\mathrm{ARO}}(t,\lambda)$.

**Overall bound.** By the triangle inequality, we conclude that the statistical distance between $\mathbf{H}_0$ and $\mathbf{H}_4$ is at most $\varepsilon_{\mathsf{ZK}}(\lambda) + 2 \cdot \varepsilon_{\mathrm{ARO}}(t,\lambda)$. Further, setting the emulation error $\varepsilon_{\mathrm{ARO}}(t,\lambda) := \frac{t}{2^\lambda}$, as computed in Theorem 5.4, yields the statement. □

As a consequence of Lemmas 9.1 and 9.2, the Micali SNARK [Mic00] is secure in the AROM.

**Corollary 9.3.** *Let $\mathscr{F} = \{\mathbb{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of fields, $m\colon \mathbb{N} \to \mathbb{N}$ be an arity function, $d\colon \mathbb{N} \to \mathbb{N}$ be a degree function, and $\lambda \in \mathbb{N}$ be the security parameter.*

*If $m(\lambda) \geq 2\lambda$ and $d(\lambda) \geq 2$, then the Micali SNARK [Mic00], instantiated with a (holographic) PCP that is honest-verifier zero knowledge and has knowledge soundness, is a zero-knowledge SNARK with straightline knowledge extraction relative to $\mathbf{ARO}[\mathscr{F}, m, d]$.*

The requirement $m(\lambda) \geq 2\lambda$ comes from the Micali SNARK construction, which uses the random oracle $\mathsf{ro}$ to compute the Merkle tree. Thus, setting $m(\lambda) \geq 2\lambda$ ensures that $\mathsf{ro}$ can parse inputs containing two $\mathsf{ro}$ outputs.

## 9.2 PCD from SNARKs in the AROM

**Theorem 9.4** (formal restatement of Theorem 1)**.** *Let $\mathscr{F} = \{\mathbb{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of fields, $m\colon \mathbb{N}\to\mathbb{N}$ an arity function, and $d\colon \mathbb{N}\to\mathbb{N}$ a degree function such that $d(\lambda)\geq 2$, $m(\lambda)\geq 2\lambda$, and $|\mathbb{F}_\lambda| = \lambda^{\omega(1)}$.*

*There exists a zero-knowledge PCD scheme relative to $\mathbf{ARO}[\mathscr{F}, m, d]$ (see Definition 4.2) for polynomial-time compliance predicates (of unbounded depth) with access to the sampled oracle, assuming the existence of (standard-model) collision-resistant hash functions.*

*Proof.* We recall a lemma from [CCS22] on SNARKs for oracle computations, with the following minor strengthening: if the given SNARK $\mathsf{ARG}_{\mathsf{in}}$ has straightline knowledge extraction, then so does the resulting SNARK $\mathsf{ARG}_{\mathsf{out}}$. This is straightforward from the construction of the extractor for $\mathsf{ARG}_{\mathsf{out}}$ in [CCS22].

**Lemma 9.5** ([CCS22, Lemma 8.2])**.** *Let $\mathcal{O}$ be an oracle distribution. Suppose that we are given:*

*(i) a SNARK $\mathsf{ARG}_{\mathsf{in}}$ in the $\mathcal{O}$-oracle model for an (oracle-free) relation $\mathcal{R}$; and*

*(ii) an accumulation scheme $\mathsf{AS} = (\mathrm{G}, \mathrm{I}, \mathrm{P}, \mathrm{V}, \mathrm{D})$ for $\mathcal{O}$-queries (in particular, $\mathrm{V}$ makes no oracle query).*

*Then we can construct a SNARK $\mathsf{ARG}_{\mathsf{out}}$ relative to $\mathcal{O}$ for $\mathcal{R}^{\mathcal{O}}$.*

*Moreover: (i) if $\mathsf{ARG}_{\mathsf{in}}$ is zero-knowledge and $\mathsf{AS}$ is zero-knowledge, then $\mathsf{ARG}_{\mathsf{out}}$ is zero-knowledge; and (ii) if $\mathsf{ARG}_{\mathsf{in}}$ has straightline knowledge extraction then $\mathsf{ARG}_{\mathsf{out}}$ has straightline knowledge extraction.*

We invoke Lemma 9.5 with $\mathcal{O}\in\mathbf{ARO}[\mathscr{F}, m, d]$, in which $\mathsf{ARG}_{\mathsf{in}}$ is the Micali SNARK run in the AROM (Corollary 9.3) and $\mathsf{AS}$ is the accumulation scheme for AROM queries from Theorem 8.1. This gives a SNARK $\mathsf{ARG}_{\mathsf{out}}$ in the AROM for AROM computations.

At this point, we could invoke [CCS22, Theorem 9.2] to obtain PCD for constant-depth compliance predicates. However, since $\mathsf{ARG}_{\mathsf{out}}$ has straightline knowledge extraction, we can obtain a stronger result: PCD for arbitrary-depth compliance predicates.

Given the output of a (cheating) prover $\tilde{\mathbb{P}}$, the PCD knowledge extractor receives $\tilde{\mathbb{P}}$'s oracle transcript $\mathsf{tr}$, applies $\mathsf{ARG}_{\mathsf{out}}$'s straightline knowledge extractor to get the witness $\mathbb{w}$, then reconstructs a transcript of the computation based on $\mathbb{w}$. Since $\mathsf{ARG}_{\mathsf{out}}$'s extractor is straightline, each extraction step has incurs an additive cost. It follows that we obtain PCD in the AROM for all arbitrary-depth polynomial-time compliance predicates. We omit further details about the knowledge extractor as this essentially follows from [CT10]; the main difference is that the SNARK in [CT10] satisfies a stronger knowledge extraction property called "list extraction". However, our notion of straightline extraction suffices for the [CT10] analysis. □

## Acknowledgments

We thank Giacomo Fenzi for pointing out some inaccuracies in an earlier draft of this paper.

Tom Gur is supported by the UKRI Future Leaders Fellowship MR/S031545/1 and EPRSC New Horizons Grant EP/X018180/1. Jack O'Connor is supported by the Engineering and Physical Sciences Research Council through the Mathematics of Systems Centre for Doctoral Training at the University of Warwick (reference EP/S022244/1). Megan Chen is supported by DARPA under Agreement No. HR00112020023.

## References

[Alo99] N. Alon. "Combinatorial Nullstellensatz". In: *Combinatorics, Probability and Computing* 8 (1999), pp. 7–29.

[AW09] S. Aaronson and A. Wigderson. "Algebrization: A New Barrier in Complexity Theory". In: *ACM Transactions on Computation Theory* 1.1 (2009), 2:1–2:54.

[BCCT13] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. "Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data". In: *Proceedings of the 45th ACM Symposium on the Theory of Computing*. STOC '13. 2013, pp. 111–120.

[BCFGRS17] E. Ben-Sasson, A. Chiesa, M. A. Forbes, A. Gabizon, M. Riabzev, and N. Spooner. "Zero Knowledge Protocols from Succinct Constraint Detection". In: *Proceedings of the 15th Theory of Cryptography Conference*. TCC '17. 2017, pp. 172–206.

[BCGRS17] E. Ben-Sasson, A. Chiesa, A. Gabizon, M. Riabzev, and N. Spooner. "Interactive Oracle Proofs with Constant Rate and Query Complexity". In: *Proceedings of the 44th International Colloquium on Automata, Languages and Programming*. ICALP '17. 2017, 40:1–40:15.

[BCLMS21] B. Bünz, A. Chiesa, W. Lin, P. Mishra, and N. Spooner. "Proof-Carrying Data Without Succinct Arguments". In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO '21. 2021, pp. 681–710.

[BCMS20] B. Bünz, A. Chiesa, P. Mishra, and N. Spooner. "Proof-Carrying Data from Accumulation Schemes". In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC '20. 2020, pp. 1–18.

[BCTV14] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. "Scalable Zero Knowledge via Cycles of Elliptic Curves". In: *Proceedings of the 34th Annual International Cryptology Conference*. CRYPTO '14. 2014, pp. 276–294.

[BDFG21] D. Boneh, J. Drake, B. Fisch, and A. Gabizon. "Halo Infinite: Proof-Carrying Data from Additive Polynomial Commitments". In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO '21. 2021, pp. 649–680.

[BGH19] S. Bowe, J. Grigg, and D. Hopwood. *Halo: Recursive Proof Composition without a Trusted Setup*. Cryptology ePrint Archive, Report 2019/1021. 2019.

[BGV11] S. Benabbas, R. Gennaro, and Y. Vahlis. "Verifiable Delegation of Computation over Large Datasets". In: *Proceedings of the 31st Annual International Cryptology Conference*. CRYPTO '11. 2011, pp. 111–131.

[BMRS20] J. Bonneau, I. Meckler, V. Rao, and E. Shapiro. *Coda: Decentralized Cryptocurrency at Scale*. Cryptology ePrint Archive, Report 2020/352. 2020.

[BN06] M. Bellare and G. Neven. "Multi-signatures in the plain public-Key model and a general forking lemma". In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. CCS '06. 2006, pp. 390–399.

[BR06] M. Bellare and P. Rogaway. “The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs”. In: *Proceedings of the 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’06. 2006, pp. 409–426.

[BR93] M. Bellare and P. Rogaway. “Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols”. In: *Proceedings of the 1st ACM Conference on Computer and Communications Security*. CCS ’93. 1993, pp. 62–73.

[CCDW20] W. Chen, A. Chiesa, E. Dauterman, and N. P. Ward. *Reducing Participation Costs via Incremental Verification for Ledger Systems*. Cryptology ePrint Archive, Report 2020/1522. 2020.

[CCS22] M. Chen, A. Chiesa, and N. Spooner. “On Succinct Non-interactive Arguments in Relativized Worlds”. In: *Proceedings of the 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’22. 2022, pp. 336–366.

[CFS17] A. Chiesa, M. A. Forbes, and N. Spooner. *A Zero Knowledge Sumcheck and its Applications*. Cryptology ePrint Archive, Report 2017/305. 2017.

[CL20] A. Chiesa and S. Liu. “On the Impossibility of Probabilistic Proofs in Relativized Worlds”. In: *Proceedings of the 11th Innovations in Theoretical Computer Science Conference*. ITCS ’20. 2020, 57:1–57:30.

[COS20] A. Chiesa, D. Ojha, and N. Spooner. “Fractal: Post-Quantum and Transparent Recursive Proofs from Holography”. In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’20. 2020, pp. 769–793.

[CT10] A. Chiesa and E. Tromer. “Proof-Carrying Data and Hearsay Arguments from Signature Cards”. In: *Proceedings of the 1st Symposium on Innovations in Computer Science*. ICS ’10. 2010, pp. 310–331.

[CTV13] S. Chong, E. Tromer, and J. A. Vaughan. *Enforcing Language Semantics Using Proof-Carrying Data*. Cryptology ePrint Archive, Report 2013/513. 2013.

[CTV15] A. Chiesa, E. Tromer, and M. Virza. “Cluster Computing in Zero Knowledge”. In: *Proceedings of the 34th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT ’15. 2015, pp. 371–403.

[GK03] S. Goldwasser and Y. T. Kalai. “On the (In)security of the Fiat-Shamir Paradigm”. In: *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. FOCS ’03. 2003, pp. 102–113.

[Gro16] J. Groth. “On the Size of Pairing-Based Non-interactive Arguments”. In: *Proceedings of the 35th Annual International Conference on Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’16. 2016, pp. 305–326.

[GW11] C. Gentry and D. Wichs. “Separating Succinct Non-Interactive Arguments From All Falsifiable Assumptions”. In: *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*. STOC ’11. 2011, pp. 99–108.

[HN23] M. Hall-Andersen and J. B. Nielsen. “On Valiant’s Conjecture: Impossibility of Incrementally Verifiable Computation from Random Oracles”. In: *Proceedings of the 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’23. 2023, pp. 438–469.

[JKRS09] A. Juma, V. Kabanets, C. Rackoff, and A. Shpilka. “The Black-Box Query Complexity of Polynomial Summation”. In: *Computational Complexity* 18.1 (2009), pp. 59–79.

[JLLW22] A. Jain, H. Lin, J. Luo, and D. Wichs. *The Pseudorandom Oracle Model and Ideal Obfuscation*. Cryptology ePrint Archive, Paper 2022/1204. 2022.

[KB23] A. Kattis and J. Bonneau. “Proof of Necessary Work: Succinct State Verification with Fairness Guarantees”. In: *Proceedings of the 27th Financial Cryptography and Data Security*. FC ’23. 2023.

[KR08] Y. Kalai and R. Raz. “Interactive PCP”. In: *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*. ICALP ’08. 2008, pp. 536–547.

[KST22] A. Kothapalli, S. Setty, and I. Tzialla. “Nova: Recursive Zero-Knowledge Arguments from Folding Schemes”. In: *Proceedings of the 42nd Annual International Cryptology Conference*. CRYPTO ’22. 2022, pp. 359–388.

[Mic00] S. Micali. “Computationally Sound Proofs”. In: *SIAM Journal on Computing* 30.4 (2000). Preliminary version appeared in FOCS ’94., pp. 1253–1298.

[Mina] O(1) Labs. *Mina Cryptocurrency*. `https://minaprotocol.com/`. 2017.

[NT16] A. Naveh and E. Tromer. “PhotoProof: Cryptographic Image Authentication for Any Set of Permissible Transformations”. In: *Proceedings of the 37th IEEE Symposium on Security and Privacy*. S&P ’16. 2016, pp. 255–271.

[TFZBT22] N. Tyagi, B. Fisch, A. Zitek, J. Bonneau, and S. Tessaro. “VeRSA: Verifiable Registries with Efficient Client Audits from RSA Authenticated Dictionaries”. In: *Proceedings of the 29th ACM Conference on Computer and Communications Security*. CCS ’22. 2022, pp. 2793–2807.

[Val08] P. Valiant. “Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency”. In: *Proceedings of the 5th Theory of Cryptography Conference*. TCC ’08. 2008, pp. 1–18.

[Zha22] M. Zhandry. “Augmented Random Oracles”. In: *Proceedings of the 42nd Annual International Cryptology Conference*. CRYPTO ’22. 2022, pp. 35–65.