ARC: Accumulation for Reed–Solomon Codes

Benedikt Bünz
bb@nyu.edu
New York University

Pratyush Mishra
prat@upenn.edu
University of Pennsylvania

Wilson Nguyen
wdnguyen@stanford.edu
Stanford University

William Wang
ww@priv.pub
New York University

October 22, 2024

Abstract

Proof-Carrying Data (PCD) is a foundational tool for ensuring the correctness of incremental distributed computations that has found numerous applications in theory and practice. The state-of-the-art PCD constructions are obtained via *accumulation* or *folding* schemes. Unfortunately, almost all known constructions of accumulation schemes rely on homomorphic vector commitments (VCs), which results in relatively high computational costs and insecurity in the face of quantum adversaries. A recent work of Bünz, Mishra, Nguyen, and Wang removes the dependence on homomorphic VCs by relying only on the random oracle model, but introduces a bound on the number of consecutive accumulation steps, which in turn bounds the depth of the PCD computation graph and greatly affects prover and verifier efficiency.

In this work, we propose ARC, a novel hash-based accumulation scheme that overcomes this restriction and supports an unbounded number of accumulation steps. The core building block underlying ARC is a new accumulation scheme for claims about proximity of claimed codewords to the Reed–Solomon code. Our approach achieves near-optimal efficiency, requiring a small number of Merkle tree openings relative to the code rate, and avoids the efficiency loss associated with bounded accumulation depth. Unlike prior work, our scheme is also able to accumulate claims up to list-decoding radius, resulting in concrete efficiency improvements.

We use this accumulation scheme to construct two distinct accumulation schemes, again relying solely on random oracles. The first approach accumulates RS proximity claims and can be used as an almost-drop-in replacement in existing PCD deployments based on IOP-based SNARKs. The second approach directly constructs an accumulation scheme for rank-1 constraint systems (and more generally polynomial constraint systems) that is simpler and more efficient than the former and prior approaches.

We introduce the notion of Interactive Oracle Reductions (IORs) to enable a modular and simple security analysis. These extend prior notions of Reductions of Knowledge to the setting of IOPs.

Contents

1 Introduction

Proof-carrying data (PCD) [CT10] is a powerful tool for proving the correctness of distributed computations that unfold incrementally. PCD has enabled numerous theoretical and practical applications, such as enforcing language semantics in distributed settings [CTV13], complexity-preserving [BCCT13; BCTV17] and low-memory [NDCTB24] succinct arguments, verifiable MapReduce computations [CTV15], image provenance [NT16], and consensus protocols and blockchains [Mina; KB20; BMRS20; CCDW20; BCG24].

These applications have motivated numerous constructions of PCD [COS20; BCMS20; BCLMS21; KST22; BC23; KS24]. The state-of-the-art amongst these approaches relies on accumulation [BCMS20; BCLMS21] or folding [KST22] schemes. At a high level, these schemes enable a prover to efficiently accumulate arbitrary NP claims into a running 'accumulator', so that verifying the correctness of each accumulation step can be done cheaply, and furthermore the final accumulator can be checked in time that is independent of the number of accumulation steps. Prior work [BCMS20; BCLMS21] shows how to construct PCD from any accumulation scheme for a *non*-succinct argument (NARK): at each step of the computation, the PCD prover invokes the accumulation prover to accumulate claims about prior steps, and then invokes the argument prover to assert that (a) the current step was performed correctly; and (b) prior claims were accumulated correctly. This PCD construction inherits efficiency and expressivity properties of the underlying accumulation scheme (and NARK), and recent work has made great progress in this regard: the latest schemes achieve, among other benefits, simple constructions that are easy to analyze and implement, low cost for verifying accumulation, and efficient support for claims that use custom gates [GW19]. Unfortunately, existing schemes also suffer from some key drawbacks which we discuss next, categorized by how the schemes are constructed.

Accumulation from homomorphic vector commitments. The vast majority of accumulation scheme constructions [BCLMS21; KST22; BC23; EG23; KS23; KS24] use as a crucial building block homomorphic vector commitments. Unfortunately, all known constructions of the latter rely on one of two kinds of number-theoretic assumptions. The first kind relies on the hardness of the discrete logarithm problem in prime-order groups. This means that the accumulation prover must perform relatively expensive group operations, and furthermore leaves the schemes vulnerable to quantum attacks. The second kind attempts to fix the latter issue by relying on lattice assumptions [BC24; FKNP24], but the resulting accumulation schemes still incur overhead due to their reliance on number-theoretic assumptions.

Furthermore, both kinds of accumulation schemes cannot take advantage of recent advances in the design and implementation of SNARKs based on interactive oracle proofs (IOPs) [BCS16], such as the ability to use small fields [HLP24; Pol; DP23] and reliance on only cryptographic hashes [BCS16].

Accumulation from homomorphism-checkers. To remedy this, a recent work [BMNW24] constructs *hash-based* accumulation schemes that avoid public-key assumptions, achieve plausible post-quantum security, rely on minimal assumptions (just cryptographic hashes), and are able to take advantage of the aforementioned advances in IOP-based SNARKs. Unfortunately, their schemes only support a (small) bounded number of consecutive accumulation steps, and this in turn forces their PCD scheme to declare an *a priori* limit on the depth of the computation graph. Additionally, efficiency of the accumulation prover and verifier worsens as this bound increases; see Section 1.2 for details.

1.1 Our results

In this work, we bypass the aforementioned limitations by constructing efficient hash-based accumulation schemes that support unbounded accumulation depth. At a high level, our schemes work by replacing homomorphic vector commitments with (non-homomorphic) Merkle tree commitments to Reed–Solomon encodings of the NP witnesses being accumulated. Making this high level sketch work requires us to develop a number of new techniques, which we describe next.

New tool: accumulation for Reed–Solomon proximity claims. The key ingredient underlying the foregoing results is a new accumulation scheme for claims about the proximity of a claimed codeword to the Reed–Solomon (RS) code. Our construction makes crucial use of tools that were previously developed for reasoning about properties of Reed–Solomon codes in the context of succinct arguments [BGKS20; ACFY24], and shows how to adapt these tools to the accumulation setting.[1]

In terms of efficiency, our accumulation scheme obtains essentially optimal parameters: asymptotically, for accumulating claims about proximity to the RS code of rate ρ, our scheme requires only $\frac{2\lambda}{\log(1/\rho)}$ Merkle tree openings, and we can get rid of the factor of 2 when assuming common conjectures about list-decoding of RS codes [BBHR18; BGKS20].

In comparison, the accumulation verifier of the prior approach of Bünz et al. [BMNW24], which works for any code but only supports a bounded accumulation depth d, requires $O(d\lambda)$ Merkle tree openings; this is concretely less efficient than our scheme for any non-trivial depth. We are able to avoid this depth bound because the techniques underlying our scheme are *distance-preserving*: if the inputs are at most δ-far from the RS code, then so is the output. In contrast, the approach of Bünz et al. [BMNW24] is not distance-preserving: the output is only guaranteed to be $\delta + \epsilon$-far from the RS code for some parameter ϵ. Furthermore, unlike the approach of Bünz et al. [BMNW24], our scheme extends to list-decoding radius $1 - \sqrt{\rho}$, which enables further efficiency improvements.

Reed–Solomon-based accumulation for NP. We leverage the foregoing RS proximity accumulation scheme to construct two different accumulation schemes for NP that rely solely on random oracles:

- *Accumulation for Polynomial IOPs: (Section 6)* The first approach relies on the observation that numerous prior SNARKs can be viewed as reducing checking NP witnesses to checking proximity of codewords to the Reed–Solomon (RS) code. In more detail, prior work [ACFY24] shows that (the information-theoretic component of) many IOP-based SNARKs for an NP relation $\mathcal{R}$ can be decomposed into three steps: a polynomial interactive oracle proof (PIOP) [CHMMVW20; BFS20] for $\mathcal{R}$ where the verifier checks that the prover's messages (which are guaranteed to be low-degree polynomials) satisfy certain identities, a transformation from these identities to RS proximity claims [KPV19; ACFY24], and a low-degree test (LDT) that enforces these claims.

  We leverage this decomposition to construct an accumulation scheme for $\mathcal{R}$. Our scheme runs the first two steps (PIOP and transformation to RS proximity claims) like above, but then, instead of enforcing the proximity claims via the LDT, accumulates them via our accumulation scheme for RS proximity.

- *Accumulation for R1CS: (Section 7)* Our second approach builds on prior accumulation schemes [BC23; EG23] which reduce claims about an NP relation $\mathcal{R}$ to claims about univariate polynomial identities. Our construction translates these claims into RS proximity claims and then invokes our accumulation scheme for the latter. In more detail, the accumulator in our construction now consists of two codewords: one that corresponds to the RS proximity accumulator and one that contains the accumulated witness to the polynomial identities. The construction maintains the essentially optimal properties of the underlying accumulation for proximity claims. The accumulation verifier checks only $t = \frac{2\lambda}{\log(1/\rho)}$ Merkle path openings per input, and the accumulation is distance-preserving (unlike the scheme of Bünz et al [BMNW24]).

[1] In fact, one can key view (a part of) our scheme as performing the first round of the recent STIR interactive oracle proof of proximity [ACFY24], which means that it will *always* be more efficient than STIR.

scheme	code	IVC overhead per step	IVC verifier	max. IVC length
PIOP + STIR [ACFY24]	RS	$\lambda(\frac{k}{\log(1/\rho)} + \log(\frac{\log n}{\log(1/\rho)}))$	$\lambda(\frac{k}{\log(1/\rho)} + \log(\frac{\log n}{\log(1/\rho)}))\ T_{\mathsf{MT}}$	$\mathrm{poly}(\lambda)$
[BMNW24]	any	$d \cdot \frac{\lambda}{\log(2/(1+\rho))}$	$d \cdot n$	m^d
this paper (PIOP-based)	RS	$k \cdot \frac{\lambda}{\log(1/\rho)}$	$k \cdot n$	$\mathrm{poly}(\lambda)$
this paper (direct)	RS	$\frac{\lambda}{\log(1/\rho)}$	n	$\mathrm{poly}(\lambda)$

Table 1: Comparison of IVC schemes constructed from PCD over a tree of depth d and arity m. All costs omit constant factors. All rows except [BMNW24] assume conjectures about proximity-gaps in the list-decoding radius. IVC overhead per step is measured in number of Merkle tree openings. Above n is the size of the recursive circuit divided by the code rate ρ, and T_{MT} is the time it takes to verify a Merkle Tree opening over n-sized vectors. Finally, k denotes the number of oracles queried by the PIOP verifier. The IVC verifier in the accumulation-based constructions can be outsourced using a SNARK, e.g., using STIR.

The two approaches are useful in different settings. The first approach offers an easy path to improve the efficiency of existing PCD constructions rely on recursive composition of IOP-based SNARKs: simply replace the LDT with our RS proximity accumulation scheme. On the other hand, the second approach, by avoiding PIOPs, is able to attain a design that is simpler and more prover- and verifier- efficient than prior work, and is hence better for new systems.

New model: interactive oracle reductions. Along the way, we formalize a new notion of interactive and probabilistic reduction protocols that we call *interactive oracle reductions* (IORs). Roughly, an IOR from relation $\mathcal{R}_1$ to relation $\mathcal{R}_2$ is an interactive protocol between a prover and a verifier that convinces the verifier that a claimed instance $\mathbb{x}_1$ is in $\mathcal{R}_1$ if and only if another instance $\mathbb{x}_2$ is in $\mathcal{R}_2$. IORs can be seen as the IOP analogues of reductions of knowledge [KP23]. We show how to compile IORs to *non-interactive* reductions by adapting the BCS transformation [BCS16].

We show how to interpret accumulation schemes as applying IORs for specific pairs of relations, and this perspective allows us to construct accumulation schemes in a straightforward manner, and also significantly simplifies our security proofs.

1.2 Related work

Bounded-depth accumulation. As noted in Section 1, the only prior hash-based accumulation scheme is that of Bünz et al [BMNW24]. Unlike our work, their scheme supports any (constant-distance) linear code, including those that enjoy linear-time encoding algorithms [Spi96; DI14; GLSTW23]. However, this benefit comes with a severe drawback: their scheme only supports a bounded number of consecutive accumulation steps. In more detail, they construct a *family* of accumulation schemes that are parameterized by a depth bound d. This bound affects the choice of the code (larger d requires better code distance), and hence also prover efficiency (better distance results in worse rate and hence larger Merkle trees) and verifier efficiency (larger d requires more Merkle tree openings). We also note that the PCD scheme constructed from their accumulation scheme inherits this depth bound, and, even worse, suffers from a concrete attack once the depth of the computation graph exceeds the bound.

In contrast, because our scheme does *not* have a depth bound, we can fix (for each input size) a code with rate and distance that minimizes prover and verifier costs. For instance, we can arbitrarily choose rate $1/2$ to minimize prover costs, or rate $1/4$ or even $1/8$ to reduce verifier costs. We are also not vulnerable to the aforementioned attack.

[BMNW24] also introduce several optimizations for IOP-based-accumulation IVC constructions. These include batch committing to multiple input accumulators, in order to reduce the number of oracle queries. These optimization also apply to our constructions.

Accumulation from hardness of discrete logarithms. As noted in Section 1, most existing accumulation schemes [BGH19; BCMS20; BCLMS21; BDFG21; KST22; KS23; BC23; EG23; KS24] rely on the hardness of computing discrete logarithms over elliptic curve groups. This in turn requires the use of cryptographically large fields both to express the computation, which can incur overheads if the computation does not need such large fields (e.g., it performs arithmetic over small integers). Furthermore, efficient implementations require *cycles of elliptic curves*, which are tricky to use correctly in practice [NBS23].

Accumulation from lattice assumptions. Some recent works [BC24; FKNP24] construct plausibly post-quantum accumulation schemes from lattice-based assumptions such as SIS and Module-SIS. Unlike our work, they depend on additional assumptions beyond random oracles.

PCD from IOP-based SNARKs. A number of recent works have constructed PCD directly from IOP-based SNARKs [COS20; Pol]. These works follow the standard methodology of constructing PCD from succinct arguments [BCCT13; BCTV14]: to prove a t-step computation, the PCD prover invokes the prover for the underlying SNARK to assert that not only was the t-th computation step performed correctly, but also that there exists a valid SNARK proof for the first $t-1$ steps.

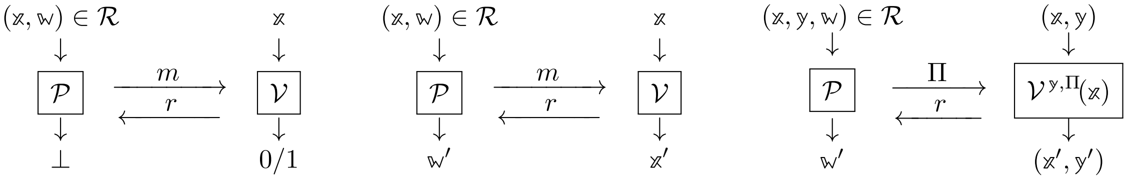
While these PCD schemes inherit the benefits of their underlying SNARKs (e.g., plausible post-quantum security, concretely efficient provers, reliance only on cryptographic hashes, etc.), they incur high asymptotic and concrete PCD overhead due to the need to express the SNARK verifier as an arithmetic circuit. This is problematic, as it lower-bounds the computations for which PCD is effective: for computations that are cheaper than the SNARK verifier, the PCD prover spends most of its time proving the latter instead of the actual computation.

Asymptotically, even incorporating state-of-the-art improvements like STIR [ACFY24] results in a verifier that requires $O(\log n + \lambda \log \log n)$ Merkle tree openings, whereas our accumulation-based approaches would require only $O(\lambda)$ openings. Concretely, when instantiating the Merkle tree with an arithmetization-oriented hash function like Poseidon [GKRRS21], Fractal's verifier circuit is of size at least 1.1 million gates [COS20]. In contrast, using our direct approach to accumulate R1CS claims of 2^{20} constraints requires only roughly $200,000$ gates *without* standard optimizations used by Fractal like proof of work or tree caps, and without using high-degree custom gates which our construction supports cheaply. (We set the rate of the RS code to be $1/16$, which results in $128/\log_2(1/(1/16)) = 32$ Merkle tree openings.)

PCD from other SNARKs. The earliest work on efficient constructions of PCD proceeded by recursive composition of pairing-based SNARKs [BCTV14]. Like accumulation-based PCD that relise no prime-order groups (i.e., without pairings), these constructions also require a cycle of elliptic curves to attain efficient recursion. However, unlike the case for non-pairing curves, cycles of pairing-friendly curves are rare [CCW19; BJS23], and current constructions that meet 128-bit security levels require arithmetic over 1000-bit prime fields [Gui].

2 Techniques

We introduce *interactive oracle reductions*, a notion which extends *interactive oracle proofs* [BCS16; RRR16] to capture *reductions*, a framework recently introduced by Kothapalli and Parno [KP23].



$(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$
$\mathbb{x}$
$\mathcal{P}$
m
r
$\mathcal{V}$
$\perp$
$0/1$
$(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$
$\mathbb{x}$
$\mathcal{P}$
m
r
$\mathcal{V}$
$\mathbb{w}'$
$\mathbb{x}'$
$(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}$
$(\mathbb{x}, \mathbb{y})$
$\mathcal{P}$
Π
r
$\mathcal{V}^{\mathbb{y},\Pi}(\mathbb{x})$
$\mathbb{w}'$
$(\mathbb{x}', \mathbb{y}')$

Argument of Knowledge Reduction of Knowledge Interactive Oracle Reduction

In a *reduction of knowledge*, a prover $\mathcal{P}$ and verifier $\mathcal{V}$ interact to reduce the claim that an instance $\mathbb{x}$ is in a language $\mathcal{L}(\mathcal{R})$ into the claim that a new instance $\mathbb{x}'$ is in a new language $\mathcal{L}(\mathcal{R}')$. Moreover, if $\mathcal{P}$ knows a new witness $\mathbb{w}'$ with $(\mathbb{x}', \mathbb{w}') \in \mathcal{R}'$, then it must also know a witness $\mathbb{w}$ with $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. As an example, a *folding scheme* [BCLMS21; KST22; BC23; KS24] is a reduction from $\mathcal{R} \times \mathcal{R}$ to $\mathcal{R}$.

The language of reductions seems to, in spirit, capture the protocols we construct. However, reductions of knowledge as described by Kothapalli and Parno [KP23] do not capture (1) instances which contain oracle strings $\mathbb{y}$; and (2) verifiers having oracle access to prover messages Π, which are features we need to analyze our protocols. Interactive Oracle Proofs of Proximity (IOPPs) [BCGRS17] roughly capture these features, but are not reductions: in an IOPP, the verifier simply outputs a bit and not a new instance.

Therefore, we define an interactive oracle reduction for a relation $\mathcal{R} := \{(\mathbb{x}, \mathbb{y}, \mathbb{w})\}$ as an interactive protocol between a prover and a verifier, where the verifier is given access oracle access to instance strings $\mathbb{y}$ and prover messages Π. At the end of interaction, the prover outputs a new witness $\mathbb{w}'$ and the verifier outputs a new instance $(\mathbb{x}', \mathbb{y}')$, where $\mathbb{y}'$ is selected from either the input oracle strings or those sent by the prover. Informally, if this new tuple $(\mathbb{x}', \mathbb{y}', \mathbb{w}')$ belongs to $\mathcal{R}'$, then the prover knows of a corresponding witness $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{y}, \mathbb{w})$ belongs to $\mathcal{R}$.

IORs, accumulation, and PCD. By adapting the BCS transformation [BCS16], we show that IORs can be compiled into non-interactive reductions in the random oracle model. We then prove that an accumulation scheme for a relation $\mathcal{R}$ can be constructed from the following (non-interactive) components:

1. A reduction from $\mathcal{R}$ to an intermediate relation $\mathcal{R}_{\mathtt{ACC}}$.
2. A many-to-one reduction from $\mathcal{R}^*_{\mathtt{ACC}}$ to $\mathcal{R}_{\mathtt{ACC}}$. Here, $\mathcal{R}^*_{\mathtt{ACC}}$ is defined to be the multi-instance relation $\{((\mathbb{x}_1, \ldots, \mathbb{x}_m), (\mathbb{w}_1, \ldots, \mathbb{w}_m)) : \forall i \in [m], (\mathbb{x}_i, \mathbb{w}_i) \in \mathcal{R}_{\mathtt{ACC}}\}$.

Assuming that $\mathcal{R}$ is NP-complete, prior work [BCLMS21] has shown how to construct proof-carrying data from such an accumulation scheme.

2.1 Accumulation for Reed–Solomon proximity claims

Let $\mathcal{C} \subset \mathbb{F}^n$ be a Reed–Solomon code. Suppose we have two vectors $f_1, f_2 \in \mathbb{F}^n$. Our goal is to reduce the claim that f_1 and f_2 are δ-close to $\mathcal{C}$ to the claim that a related vector f is δ-close to $\mathcal{C}$.[2] For simplicity, we assume that δ is at most the unique decoding radius of the code.

A natural approach is to take f to be a random linear combination of f_1 and f_2; indeed, proximity gaps for Reed–Solomon codes [BCIKS23] tell us that if *either* f_1 or f_2 is δ-far, then $f := f_1 + r \cdot f_2$ will be δ-far with high probability. However, this fact alone does not give us a many-to-one reduction for proximity

[2] Two vectors $f, g \in \mathbb{F}^n$ are δ-close if they agree on at least a $(1 - \delta)$-fraction of entries. We say that a vector f is δ-close to $\mathcal{C}$ if there exists a codeword which is δ-close to f.

claims. The issue is that f is a "virtual" object defined over two vectors; whenever the verifier queries $f[i]$, it is implicitly querying $f_1[i]$ and $f_2[i]$. This implies that the new claim doubles in size (concretely, $2n+1$ field elements). Ultimately, in order to realize accumulation, the size of the new claim must be independent of the number of old claims.

Prior work. We first recall the approach taken in [BMNW24]. After the verifier samples r, the prover sends a *new* vector f which is claimed to be $f_1 + r \cdot f_2$. The verifier tests this by sampling a random location $i \in [n]$ and checking that the vectors are consistent at the i-th entry: $f[i] = f_1[i] + r \cdot f_2[i]$. By repeating this spot check $\frac{\lambda}{-\log(1-\varepsilon)}$ times, the verifier ensures that f is ε-close to $f_1 + r \cdot f_2$ with high probability. Hence, if either f_1 or f_2 is δ-far, then f is $(\delta - \varepsilon)$-far from the code. Although the size of the new claim is indeed independent of the number of old claims, this is not quite a many-to-one reduction. The issue is that the distance claim degrades from δ to $\delta - \varepsilon$. As a result, [BMNW24] are only able to construct a *bounded-depth accumulation scheme*, where the number of steps must be a small constant fixed in advance.

Background. Let $\mathcal{L}$ be a subset of $\mathbb{F}$ of size n; this is referred to as the evaluation domain. The Reed–Solomon code $\mathsf{RS}[d] \subset \mathbb{F}^n$ is the set of words[3] $f : \mathcal{L} \to \mathbb{F}$ where f is consistent with a polynomial of degree less than d. The *quotient* of a word $f : \mathcal{L} \to \mathbb{F}$ relative to $x, y \in \mathbb{F}$ is defined to be $\mathsf{Quotient}(f, x, y)(X) := \frac{f(X)-y}{X-x}$. We make the following observations:

1. If f is a codeword in $\mathsf{RS}[d]$ with $y = f(x)$, then $\mathsf{Quotient}(f, x, y)$ is a codeword in $\mathsf{RS}[d-1]$. This is because x is a root of $g(X) - y$.

2. If $\mathsf{Quotient}(f, x, y)$ is δ-close to a codeword $w \in \mathsf{RS}[d-1]$, then f is δ-close to a codeword $u \in \mathsf{RS}[d]$ with $u(x) = y$, namely $u(X) := w(X) \cdot (X - x) + y$.

3. If f is δ-far from any codeword $u \in \mathsf{RS}[d]$ with $u(x) = y$, then $\mathsf{Quotient}(f, x, y)$ is δ-far from $\mathsf{RS}[d-1]$. This is essentially the contrapositive of Item 2.

Quotients can be generalized to handle multiple points by defining

$$\mathsf{Quotient}(f, (x_1, y_1), \ldots, (x_t, y_t)) := \frac{f(X) - p(X)}{\prod_{j=1}^{t}(X - x_j)},$$

where p is the Lagrange interpolation of $(x_j, y_j)_{j \in [t]}$. If f is δ-far from any codeword $u \in \mathsf{RS}[d]$ with $u(x_j) = y_j$ for all j, then $\mathsf{Quotient}(f, (x_1, y_1), \ldots, (x_t, y_t))$ is δ-far from $\mathsf{RS}[d-t]$.

This work. We give a many-to-one reduction for Reed–Solomon proximity claims which preserves distance; the resulting accumulation scheme therefore supports an unbounded number of steps. The protocol starts off in the same way as before:

1. Verifier samples a random combination $r \leftarrow \mathbb{F}$.
2. Prover sends a new word $f : \mathcal{L} \to \mathbb{F}$. In the honest case, $f := f_1 + r \cdot f_2$.
3. Verifier samples locations $x_1, \ldots, x_t \leftarrow \mathcal{L}$.

Where we depart is in how the new claim is formulated. The verifier computes $y_j := f_1(x_j) + r \cdot f_2(x_j)$ for each j, and defines the quotient $q := \mathsf{Quotient}(f, (x_1, y_1), \ldots, (x_t, y_t))$. The new claim is that q is δ-close to $\mathsf{RS}[d-t]$. Observe that q is defined over f and a few (specifically, $2t$) auxiliary field elements, and hence the size of the new claim is independent of the number of old claims.

Suppose either f_1 or f_2 is δ-far from $\mathsf{RS}[d]$. We show that q will be δ-far from $\mathsf{RS}[d-t]$ with high probability:

[3] Any word $f : \mathcal{L} \to \mathbb{F}$ can be interpreted as a vector in $\mathbb{F}^n$, and vice versa.

1. The random combination $f' := f_1 + r \cdot f_2$ is δ-far from $\mathsf{RS}[d]$ with high probability.
2. Since δ is at most the unique decoding radius, there is at most one codeword $u \in \mathsf{RS}[d]$ within δ distance of f. Fix u if it exists. Since u is δ-far from f', there exists j such that $u(x_j) \neq f'(x_j) = y_j$ with probability at least $1 - (1-\delta)^t$. Setting $t := \frac{\lambda}{-\log(1-\delta)}$, this is all but negligible.
3. We conclude that f is δ-far from any codeword u with $u(x_j) = y_j$ for all j, which implies that q is δ-far from $\mathsf{RS}[d-t]$.

We are not quite done, because the new claim is about proximity to $\mathsf{RS}[d-t]$, rather than $\mathsf{RS}[d]$. Fortunately, there exist efficient degree correction procedures which allow the verifier to soundly reduce a proximity claim for $\mathsf{RS}[d-t]$ into a proximity claim for $\mathsf{RS}[d]$.

To summarize, we have described a reduction for Reed–Solomon proximity claims which satisfies two key properties. First, the size of the new claim is independent of the number of old claims; this is necessary for accumulation. Second, the reduction is distance-preserving; this is necessary for accumulating an unbounded number of times. Although we focused on combining two claims, our construction can easily be extended to combine many at once.

Theorem 2.1 (informal)**.** *Define the relation $\mathcal{R}_{\mathtt{RS}}$ where $(f, d) \in \mathcal{L}(\mathcal{R}_{\mathtt{RS}})$ if f is δ-close to $\mathsf{RS}[d]$. There exists a many-to-one reduction for $\mathcal{R}_{\mathtt{RS}}$.*

Moving to the list decoding radius. Up to this point we have assumed that the distance parameter δ is at most the unique decoding radius. We would ideally like to support larger δ; this would translate to smaller t and therefore improve query complexity. The key step in the analysis which fails if δ were larger is Item 2; namely, there may be more than one codeword u in the δ-ball of f. To resolve this, we leverage out-of-domain sampling [BGKS20]. In more detail, after the prover sends the new word f, the verifier samples an additional point $x^{\mathtt{out}} \in \mathbb{F}$. The prover responds with a claimed evaluation $y^{\mathtt{out}}$; assuming δ is less than the list decoding radius, with high probability there exists a unique codeword u in the δ-ball satisfying $u(x^{\mathtt{out}}) = y^{\mathtt{out}}$. This point is additionally quotiented to obtain q.

2.2 Accumulation for NP

We describe a highly efficient accumulation scheme for R1CS circuit satisfiability. Recall that an R1CS circuit is defined by matrices $A, B, C \in \mathbb{F}^{\mathsf{M} \times \mathsf{N}}$ and instance length $\mathsf{n} \in \mathbb{N}$. An instance $x \in \mathbb{F}^{\mathsf{n}}$ is in the language if there exists a witness $w \in \mathbb{F}^{\mathsf{N}-\mathsf{n}}$ such that $Az \circ Bz = Cz$ for $z := (x, w) \in \mathbb{F}^{\mathsf{N}}$. Our goal is to accumulate instances of R1CS. Following the accumulation blueprint, it suffices to give (i) a reduction from R1CS to an intermediate relation $\mathcal{R}_{\mathtt{ACC}}$; and (ii) a many-to-one reduction for $\mathcal{R}_{\mathtt{ACC}}$.

Informally, $\mathcal{R}_{\mathtt{ACC}}$ encodes an "algebraic" proximity claim in the sense that f must be δ-close to a codeword u which satisfies an algebraic constraint. Let $d := \mathsf{N} - \mathsf{n}$. Let P be a multivariate polynomial in $k + d$ variables with total degree c. For a codeword $u \in \mathsf{RS}[d]$, let $\vec{u} \in \mathbb{F}^d$ denote its decoding (concretely, its coefficient vector). For a scalar $e \in \mathbb{F}$, vector $v \in \mathbb{F}^k$, and word $f : \mathcal{L} \to \mathbb{F}$, we define $(e, v, f) \in \mathcal{L}(\mathcal{R}_{\mathtt{ACC}})$ if f is δ-close to a codeword $u \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ such that $P(v, \vec{u}) = e$; here, $\vec{u} \in \mathbb{F}^d$ refers to the decoding of u, i.e., its vector of coefficients. We assume that δ is at most the unique decoding radius of the code.

2.2.1 Reduction from R1CS to $\mathcal{R}_{\mathtt{ACC}}$

For simplicity, assume that M is a power of two and define $\mathsf{m} := \log \mathsf{M}$. For each $i = 0, \ldots, \mathsf{M}-1-1$, define the multilinear polynomial $\mathsf{pow}_i(Y_1, \ldots, Y_{\mathsf{m}}) = Y_1^{b_1} \cdots Y_{\mathsf{m}}^{b_{\mathsf{m}}}$, where $b_1, \ldots, b_{\mathsf{m}}$ is the bit representation of

i. Observe that for all $y \in \mathbb{F}$, $\mathsf{pow}_i(y, y^2, y^4, \ldots, y^{2^{\mathsf{m}-1}}) = y^i$. $\mathcal{R}_{\mathsf{ACC}}$ is defined with $k := \mathsf{m} + \mathsf{n}$ and the polynomial

$$P(Y_1, \ldots, Y_{\mathsf{m}}, Z_1, \ldots, Z_{\mathsf{N}}) := \sum_{i=1}^{\mathsf{M}} \mathsf{pow}_{i-1}(Y_1, \ldots, Y_{\mathsf{m}}) \cdot (a_i^T \vec{Z} \cdot b_i^T \vec{Z} - c_i^T \vec{Z}),$$

where a_i, b_i, c_i are the i-th rows of A, B, C. Observe that P total degree $c := \mathsf{m} + 2$. The reduction from R1CS to $\mathcal{R}_{\mathsf{ACC}}$ is as follows.

1. Prover sends a word $f : \mathcal{L} \to \mathbb{F}$. In the honest case, f is the encoding of witness w.
2. Verifier samples a random scalar $r \leftarrow \mathbb{F}$.
3. The new claim is that $(e, v, f) \in \mathcal{L}(\mathcal{R}_{\mathsf{ACC}})$, where $e := 0$ and $v := (r, r^2, r^4, \ldots, r^{2^{\mathsf{m}-1}}, x) \in \mathbb{F}^k$.

Soundness. Suppose that x is not a valid R1CS instance. We show that $(e, v, f) \notin \mathcal{L}(\mathcal{R}_{\mathsf{ACC}})$ with high probability. Observe that since δ is at most the unique decoding radius, there is at most one codeword u within δ distance of f. Fix u if it exists; otherwise, we immediately have $(e, v, f) \notin \mathcal{L}(\mathcal{R}_{\mathsf{ACC}})$. Define $z := (x, \vec{u})$. Since x is not a valid instance, there exists $i \in [\mathsf{M}]$ such that $a_i^T z \cdot b_i^T z \neq c_i^T z$. Equivalently,

$$F(X) := P(X, X^2, X^4, \ldots, X^{2^{\mathsf{m}-1}}, z) = \sum_{i=1}^{\mathsf{M}} X^{i-1} \cdot (a_i^T z \cdot b_i^T z - c_i^T z)$$

is a non-zero univariate polynomial of degree at most $\mathsf{N} - 1$. Since r is sampled uniformly, we have $P(v, \vec{u}) = F(r) \neq 0$ with probability at least $1 - \frac{\mathsf{N}-1}{|\mathbb{F}|}$. We conclude that $(e, v, f) \notin \mathcal{L}(\mathcal{R}_{\mathsf{ACC}})$.

2.2.2 Many-to-one reduction for $\mathcal{R}_{\mathsf{ACC}}$

Suppose we have many instances $(e_1, v_1, f_1), \ldots, (e_m, v_m, f_m)$. Our goal is to reduce the claim that $(e_i, v_i, f_i) \in \mathcal{R}_{\mathsf{ACC}}$ for all i to the claim that $(e, v, f) \in \mathcal{R}_{\mathsf{ACC}}$ for a new instance (e, v, f). Consider the reduction:

1. Fix a subset $H = \{a_1, \ldots, a_m\} \subset \mathbb{F}$, and define $V(X) = \prod_{i=1}^m (X - a_i)$, which vanishes on H. Let L_i denote the unique Lagrange polynomial of degree less than m satisfying $L_i(a_i) = 1$ and $L_i(a_j) = 0$ for $i \neq j$. The prover sends a univariate polynomial Q of degree at most $c \cdot (m-1) - m$. In the honest case, Q is the unique polynomial which satisfies

$$P\left(\sum_{i=1}^{m} L_i(X) \cdot (v_i, \vec{f_i})\right) - \sum_{i=1}^{m} L_i(X) \cdot e_i = Q(X) \cdot V(X).$$

   This exists because the left side of the above equation vanishes on H.
2. Verifier samples an evaluation point $\alpha \leftarrow \mathbb{F}$.
3. Prover sends a new word $f : \mathcal{L} \to \mathbb{F}$. In the honest case, $f := \sum_{i=1}^m L_i(\alpha) \cdot f_i$.
4. Verifier samples locations $x_1, \ldots, x_t \leftarrow \mathcal{L}$.
5. Verifier computes $e := Q(\alpha) \cdot V(\alpha) + \sum_{i=1}^m L_i(\alpha) \cdot e_i$ and $v := \sum_{i=1}^m L_i(\alpha) \cdot v_i$.
6. Verifier computes $y_j := \sum_{i=1}^m L_i(\alpha) \cdot f_i(x_j)$ for each $j \in [t]$.
7. We have the following new claims:
   - $(e, v, f) \in \mathcal{R}_{\mathsf{ACC}}$.

- $(f_i, d) \in \tilde{\mathcal{R}}_{\texttt{RS}}$ for each $i \in [m]$.
- $(q, d-t) \in \tilde{\mathcal{R}}_{\texttt{RS}}$, where $q := \mathsf{Quotient}(f, (x_1, y_1), \ldots, (x_t, y_t))$.

This is not quite a many-to-one reduction for $\mathcal{R}_{\texttt{ACC}}$, since we also output several proximity claims. We resolve this by keeping track of *two instances*: one for $\mathcal{R}_{\texttt{ACC}}$ and one for $\tilde{\mathcal{R}}_{\texttt{RS}}$. It suffices to construct a many-to-one reduction for $\mathcal{R}_{\texttt{ACC}} \times \tilde{\mathcal{R}}_{\texttt{RS}}$, where the verifier (i) reduces m instances for $\mathcal{R}_{\texttt{ACC}}$ into one instance for $\mathcal{R}_{\texttt{ACC}}$ and $m+1$ instances for $\tilde{\mathcal{R}}_{\texttt{RS}}$; and (ii) reduces $2m+1$ instances for $\tilde{\mathcal{R}}_{\texttt{RS}}$ into one instance for $\tilde{\mathcal{R}}_{\texttt{RS}}$ using Theorem 2.1.

Soundness. We show that if $(e_i, v_i, f_i) \notin \mathcal{R}_{\texttt{ACC}}$ for some i, then at least one of the new instances is invalid with high probability.

1. Assume that $f_1, \ldots, f_m$ are δ-close to $\mathsf{RS}[d]$; otherwise, the many-to-one reduction for $\tilde{\mathcal{R}}_{\texttt{RS}}$ will output an invalid instance and we are done. In fact, the many-to-one reduction will only output a valid instance if there is *correlated agreement*: there exist codewords $u_1, \ldots, u_m \in \mathsf{RS}[d]$ such that $f_1, \ldots, f_m$ respectively agrees with $u_1, \ldots, u_m$ on the same $1-\delta$ fraction of points. This is implied by proximity gaps for Reed–Solomon codes.

2. We are guaranteed that there exists some i such that $P(v_i, \vec{u}_i) \neq e_i$. Observe that

$$F(X) := P\left(\sum_{i=1}^{m} L_i(X) \cdot (v_i, \vec{f}_i)\right) - Q(X) \cdot V(X) - \sum_{i=1}^{m} L_i(X) \cdot e_i$$

   is a non-zero polynomial of degree at most $c \cdot (m-1)$, since $F(a_i) = P(v_i, \vec{u}_i) - e_i$. Define $u' := \sum_{i=1}^{m} L_i(\alpha) \cdot u_i$. With probability $1 - \frac{c \cdot (m-1)}{|\mathbb{F}|}$, $F(\alpha) = P(v, \vec{u}') \neq e$.

3. Define $f' := \sum_{i=1}^{m} L_i(\alpha) \cdot f_i$. By correlated agreement, u' is δ-close to f'.

4. Since δ is at most the unique decoding radius, there exists at most one codeword $u \in \mathsf{RS}[d]$ within δ distance of f. Fix u if it exists and assume that $P(v, \vec{u}) = e$; otherwise, $(e, v, f) \notin \mathcal{R}_{\texttt{ACC}}$ and we are done.

5. Since $P(v, \vec{u}) \neq P(v, \vec{u}')$, we know that $u \neq u'$. Since the distance of the code is double the unique decoding radius, u is 2δ-far from u'. By a triangle inequality, u is δ-far from f'.

6. With probability at least $1 - (1-\delta)^t$, there exists j such that $u(x_j) \neq f'(x_j) = y_j$. Setting $t := \frac{\lambda}{-\log(1-\delta)}$, this is all but negligible.

7. We conclude that f is δ-far from any codeword u with $u(x_j) = y_j$ for all j, which implies that q is δ-far from $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d-t]$. With high probability, the many-to-one reduction for $\tilde{\mathcal{R}}_{\texttt{RS}}$ outputs an invalid instance.

Moving to list decoding radius. As in Section 2.1, we can upgrade δ to be less than the list decoding radius. We use the same technique of out-of-domain samples to bind vectors to a unique codeword within the δ-ball. For the construction, we need to send three separate out-of-domain samples. First, we bind each input f_i to a unique codeword. Then, after the challenge α, we bind the virtual polynomial f'. Finally, we use an additional out-of-domain sample to bind f. We discuss the necessity of these samples in more detail in Remark 7.15.

2.3 Proof-carrying data from reductions

Accumulation from Non-interactive Reductions. As we have seen, interactive oracle reductions and their compiled form, non-interactive reductions, capture natural notions of batching and generalize the existing frameworks of IOPPs and reductions of knowledge. In this work, we show that given a pair of non-interactive reductions matching a particular form, we can naturally construct a corresponding non-interactive argument and accumulation scheme for that non-interactive argument. We state this more clearly in the following informal theorem.

Theorem 2.2 (informal)**.** *Let $\mathcal{R}$ and $\mathcal{R}_{\mathtt{ACC}}$ be indexed relations. Suppose that*

- $\mathsf{RDX}_{\mathtt{CAST}}$ *is a non-interactive reduction from $\mathcal{R}$ to $\mathcal{R}_{\mathtt{ACC}}$.*
- $\mathsf{RDX}_{\mathtt{FOLD}}$ *is a non-interactive reduction from $\mathcal{R}^*_{\mathtt{ACC}}$ to $\mathcal{R}_{\mathtt{ACC}}$.*

Then there exists a non-interactive argument ARG *for $\mathcal{R}$ and an accumulation scheme* ACC *for* ARG.

$$\begin{tikzcd}
(\mathbb{x}, \mathbb{w}) \in \mathcal{R} \arrow[d] & \mathbb{x} \arrow[d] \\
\boxed{\mathcal{P}_{\mathtt{CAST}}} \arrow[r, "\pi_{\mathtt{CAST}}"] \arrow[d] & \boxed{\mathcal{V}_{\mathtt{CAST}}} \arrow[d] \\
\mathsf{acc}.\mathbb{w} & \mathsf{acc}.\mathbb{x}
\end{tikzcd}
\qquad
\begin{tikzcd}
(\mathsf{acc}_i.\mathbb{x}, \mathsf{acc}_i.\mathbb{w})_{i\in[m+n]} \arrow[d] & (\mathsf{acc}_i.\mathbb{x})_{i\in[m+n]} \arrow[d] \\
\boxed{\mathcal{P}_{\mathtt{FOLD}}} \arrow[r, "\pi_{\mathtt{FOLD}}"] \arrow[d] & \boxed{\mathcal{V}_{\mathtt{FOLD}}} \arrow[d] \\
\mathsf{acc}.\mathbb{w} & \mathsf{acc}.\mathbb{x}
\end{tikzcd}$$

Intuitively, the reduction $\mathsf{RDX}_{\mathtt{CAST}}$ *casts* a member $(\mathbb{x}, \mathbb{w})$ of relation $\mathcal{R}$ into a member $(\mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w})$ in the accumulator relation $\mathcal{R}_{\mathtt{ACC}}$. While the reduction $\mathsf{RDX}_{\mathtt{FOLD}}$, folds together multiple members $(\mathsf{acc}_i.\mathbb{x}, \mathsf{acc}_i.\mathbb{w})_i$ of the accumulator relation into a single instance $(\mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w})$. An initial observation is that the reduction $\mathsf{RDX}_{\mathtt{CAST}}$ closely matches the shape of a non-interaction argument $\mathsf{ARG} = (\mathcal{P}, \mathcal{V})$. The argument prover and verifier can internally run the reduction $\mathsf{RDX}_{\mathtt{CAST}}$ to derive a new accumulator instance and witness. The prover can send, along with the reduction proof, the new accumulator witness and the verifier can check if the new accumulator belongs to $\mathcal{R}_{\mathtt{ACC}}$. This immediately gives us an argument for relation $\mathcal{R}$.

$$\begin{tikzcd}
(\mathbb{x}, \mathbb{w}) \in \mathcal{R} \arrow[d] & \mathbb{x} \arrow[d] \\
\boxed{\mathcal{P}: \ (\pi_{\mathtt{CAST}}, \mathsf{acc}.\mathbb{w}) \leftarrow \mathcal{P}_{\mathtt{CAST}}(\mathbb{x}, \mathbb{w})} \arrow[r, "\pi := (\pi_{\mathtt{CAST}}, \mathsf{acc}.\mathbb{w})"] & \boxed{\mathcal{V}: \ \mathsf{acc}.\mathbb{x} \leftarrow \mathcal{V}_{\mathtt{CAST}}(\mathbb{x}, \pi_{\mathtt{CAST}})} \arrow[d] \\
 & (\mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w}) \overset{?}{\in} \mathcal{R}_{\mathtt{ACC}}
\end{tikzcd}$$

All that remains to be shown is how to construct an accumulation scheme for ARG. Naturally, the argument proof π can be partition into $(\pi.\mathbb{x}, \pi.\mathbb{w}) := (\pi_{\mathtt{CAST}}, \mathsf{acc}.\mathbb{w})$. By design, we now have that the accumulation predicate instance $(\mathbb{x}, \pi.\mathbb{x})$ is exactly the input to the reduction verifier $\mathsf{RDX}_{\mathtt{CAST}}$. Thus, given m accumulator instances and n predicate instances, the accumulation prover and verifier can symmetrically run $\mathcal{V}_{\mathtt{CAST}}$ to derive $m + n$ accumulator instances.

$$\begin{tikzcd}
(\mathsf{acc}_1.\mathbb{x}, & \cdots, & \mathsf{acc}_m.\mathbb{x}, & (\pi_{\mathtt{CAST}_1}, \mathbb{x}_1), \arrow[d] & \cdots & (\pi_{\mathtt{CAST}_n}, \mathbb{x}_n)) \arrow[d] \\
 & & & \boxed{\mathcal{V}_{\mathtt{CAST}}} \arrow[d] & \cdots & \boxed{\mathcal{V}_{\mathtt{CAST}}} \arrow[d] \\
(\mathsf{acc}_1.\mathbb{x}, & \cdots, & \mathsf{acc}_m.\mathbb{x}, & \mathsf{acc}_{m+1}.\mathbb{x}, & \cdots & \mathsf{acc}_{m+n}.\mathbb{x})
\end{tikzcd}$$

Now, the accumulation prover can run the reduction $\mathsf{RDX}_{\mathtt{FOLD}}$ to derive the output accumulator $\mathsf{acc} \leftarrow (\mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w})$ which folds together the $m+n$ accumulators and produce an accumulation proof $\mathsf{pf} \leftarrow \pi_{\mathtt{FOLD}}$. The accumulation verifier just has to check this new accumulator instance $\mathsf{acc}.\mathbb{x}$ is identical to what is derived by running the reduction verifier $\mathsf{RDX}_{\texttt{FOLD}}$. Finally, the accumulation decider just checks if an accumulator $\mathsf{acc} := (\mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w})$ belongs to $\mathcal{R}_{\texttt{ACC}}$. We treat this discussion formally in Theorem 4.3 and provide the corresponding argument ARG and accumulation scheme ACC in Construction A.4 and Construction A.6.

From Reductions to Proof Carrying Data and IVC. What we just described is a method to construct accumulation from non-interactive reductions. In prior works [BCMS20; BCLMS21], accumulation schemes for non-interactive arguments can be transformed into IVC and PCD schemes, assuming the non-interactive argument is for an NP-complete relation and the circuit description of the accumulation verifier is succinct (Theorem 5.3 in [BCLMS21]). In our construction, these requirements translate to whether the relation $\mathcal{R}$ is NP-complete and if the reduction verifiers $\mathcal{V}_{\texttt{CAST}}$ and $\mathcal{V}_{\texttt{FOLD}}$ are succinct.

While the PCD construction naturally follows from prior work, the security analysis must be slightly tweaked when considering promise relations, which have both strict and relaxed relations, $\mathcal{R}_{\texttt{ACC}}$ and $\tilde{\mathcal{R}}_{\texttt{ACC}}$ respectively. In particular, the knowledge soundness of both the argument ARG and accumulation scheme ACC hold with respect to a relaxed verifier and decider, $\tilde{\mathcal{V}}$ and $\tilde{\mathrm{D}}$, which check that an accumulator belongs to the relaxed relation $\tilde{\mathcal{R}}_{\texttt{ACC}}$, while in the construction they check pairs belong to the strict relation $\mathcal{R}_{\texttt{ACC}}$. We observe that the knowledge soundness proof of the PCD construction (Theorem 5.3 in [BCLMS21]) can be immediately adapted by replacing the verifier and decider with their relaxed variants. Alternatively, we can also adapt the proof in [BMNW24] which shows how to construct PCD from bounded-depth accumulation. Unlike our work, which has one relaxed verifier and decider, they have a different relaxed verifier and decider for each recursive extraction, up to some depth-bound $s \in \mathbb{N}$. In our setting, we would just maintain the same relaxed verifier and decider regardless of the extraction depth.

3 Preliminaries

Strings and words. For an alphabet Σ, a string $s \in \Sigma^*$ is a tuple of characters in the alphabet. For a finite set S, a word $w : S \to \Sigma$ is a function mapping elements of S to characters in Σ. These objects are somewhat interchangeable; a string $s \in \Sigma^n$ can be viewed as a word over the set of indices $[n]$, and a word $w : S \to \Sigma$ can be viewed as a string of length $|S|$ (assuming S has a fixed ordering).

Restrictions. For a string $s \in \Sigma^n$ and subset of indices $I \subseteq [n]$, the restriction $s|_I : I \to \Sigma$ is defined to be $s|_I(i) = s(i)$. Alternatively, we can treat $s|_I$ as a string of length n over an augmented alphabet $\Sigma \sqcup \{\bot\}$, where the i-th character of $s|_I$ is $s(i)$ if $i \in I$, and $\bot$ otherwise.

Hamming distance. For an alphabet Σ, the relative Hamming distance between two strings $s, s' : s \in \Sigma^n$, denoted $\Delta(s, s')$, is the number of locations where s and s' disagree, divided by n. For a set of strings $S \in \Sigma^n$, we define $\Delta(s, S) := \min_{s' \in S} \Delta(s, s')$.

Polynomials. For a field $\mathbb{F}$, let $\mathbb{F}^{<d}[X]$ denote the set of univariate polynomials over $\mathbb{F}$ of degree less than d. For a set $S \subset \mathbb{F}$, the vanishing polynomial $V_S(X) := \prod_{a \in S}(X - a)$ is the unique non-zero polynomial of degree at most $|S|$ that is zero on S. For an element $a \in S$, let $L_{a,S}$ denote the unique Lagrange polynomial of degree less than $|S|$ such that $L_{a,S}(a) = 1$ and $L_{a,S}(b) = 0$ for all $b \in S \setminus \{a\}$. For a function $f : S \to \mathbb{F}$, let $\hat{f}$ denote the unique extension polynomial of degree less than $|S|$ such that $\hat{f}(a) = f(a)$ for all $a \in S$, i.e., $\hat{f}(X) := \sum_{a \in S} f(a) \cdot L_{a,S}(X)$.

Polynomial quotients. For a field $\mathbb{F}$, polynomial $p \in \mathbb{F}^{<d}[X]$, and set $S \subset \mathbb{F}$, the polynomial quotient $\mathsf{PolyQuotient}(p, S) \in \mathbb{F}^{<d-|S|}[X]$ is defined to be

$$\mathsf{PolyQuotient}(p, S)(x) := \frac{p(x) - r(x)}{V_S(x)},$$

where r is the unique polynomial of degree less than $|S|$ such that $r(a) = p(a)$ for all $a \in S$ (in other words, r is the extension of the restriction of p to S).

Random oracles. Let $\mathcal{U}(\lambda)$ denote the uniform distribution of functions that map $\{0,1\}^*$ to $\{0,1\}^\lambda$. A *random oracle* is a function $\rho : \{0,1\}^* \to \{0,1\}^\lambda$ sampled from $\mathcal{U}(\lambda)$. Our constructions will often use multiple random oracles of varying output sizes; these can be derived from a single random oracle via *domain extension* and *output extension*. For more discussion, see [CY24, Section 2.6].

3.1 Relations

Indexed relations. An *indexed relation* $\mathcal{R}$ is a set of triples $\{(\mathbb{i}, \mathbb{x}, \mathbb{w})\}$ where $\mathbb{i}$ is the index, $\mathbb{x}$ is the instance, and $\mathbb{w}$ is the witness; the corresponding *indexed language* $\mathcal{L}(\mathcal{R})$ is the set of pairs $(\mathbb{i}, \mathbb{x})$ for which there exists a witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$. For example, the indexed relation of satisfiable boolean circuits consists of triples where $\mathbb{i}$ is the description of a boolean circuit, $\mathbb{x}$ is a partial assignment to its input wires, and $\mathbb{w}$ is an assignment to the remaining wires that makes the circuit output 1.

Parameterized relations and R1CS. A *parameterized relation* $\mathcal{R}$ over a (typically implicit) parameter space $\mathbb{P}$ is a set of relations $\{\mathcal{R}(\mathbb{p}) : \mathbb{p} \in \mathbb{P}\}$. The R1CS relation is parameterized by a finite field $\mathbb{F}$; $\mathcal{R}_{\mathtt{R1CS}}(\mathbb{F})$ consists of triples $(\mathbb{i}, \mathbb{x}, \mathbb{w}) = ((A, B, C, n), x, w)$ where A, B, C are $M \times N$ matrices over $\mathbb{F}$, $x \in \mathbb{F}^n$, and $w \in \mathbb{F}^{N-n}$ such that $Az \circ Bz = Cz$ for $z := (x, w)$.

Relations relative to a random oracle. A *relation relative to a random oracle*, denoted $\mathcal{R}^{\mathcal{U}}$, is a set of relations $\{\mathcal{R}^\rho : \rho \in \mathrm{supp}(\mathcal{U})\}$, where $\mathrm{supp}(\mathcal{U})$ denotes $\bigcup_{\lambda \in \mathbb{N}} \mathrm{supp}(\mathcal{U}(\lambda))$.

Promise relations. Some proof systems exhibit a gap between completeness and soundness, i.e., completeness holds for a relation $\mathcal{R}$, but soundness only guarantees membership in a superset relation $\tilde{\mathcal{R}} \supseteq \mathcal{R}$. In this case it is useful to describe $\mathcal{R}$ as a *promise relation*, where soundness holds for the associated *relaxed relation* $\tilde{\mathcal{R}}$.[4]

Putting it all together. A parameterized indexed promise relation $\mathcal{R}^{\mathcal{U}}$ (over parameter space $\mathbb{P}$, relative to a random oracle) is a set of indexed promise relations $\{(\mathcal{R}^{\rho}(\mathbb{p}), \tilde{\mathcal{R}}^{\rho}(\mathbb{p})) : \mathbb{p} \in \mathbb{P}, \rho \in \mathrm{supp}(\mathcal{U})\}$ such that $\mathcal{R}^{\rho}(\mathbb{p}) \subseteq \tilde{\mathcal{R}}^{\rho}(\mathbb{p})$. We say that $\mathcal{R}^{\mathcal{U}}$ is in $\mathsf{NP}^{\mathcal{U}}$ if and only if there exists a polynomial-time oracle Turing machine M such that for every $\mathbb{p} \in \mathbb{P}$ and $\rho \in \mathrm{supp}(\mathcal{U})$, $\mathcal{R}^{\rho}(\mathbb{p}) = \{(\mathbb{i}, \mathbb{x}, \mathbb{w}) : M^{\rho}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \mathbb{w}) = 1\}$.

Multi-instance relations. Let $\mathcal{R}$ be an indexed relation. The *multi-instance relation* is defined to be $\mathcal{R}^* := \{(\mathbb{i}, (\mathbb{x}_1, \ldots, \mathbb{x}_m), (\mathbb{w}_1, \ldots, \mathbb{w}_m)) : \forall i \in [m], (\mathbb{i}, \mathbb{x}_i, \mathbb{w}_i) \in \mathcal{R}\}$. This notion readily extends to the types of relations described above.

3.2 Reed–Solomon codes

For a field $\mathbb{F}$, evaluation domain $\mathcal{L} \subset \mathbb{F}$, and degree $d \in \mathbb{N}$, the Reed–Solomon code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ is the set of words $\mathcal{L} \to \mathbb{F}$ corresponding to polynomials of degree less than d:

$$\mathsf{RS}[\mathbb{F}, \mathcal{L}, d] := \{f : \mathcal{L} \to \mathbb{F} \ : \ \hat{f} \in \mathbb{F}^{<d}[X]\}.$$

The rate of $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ is $\rho := d/|\mathcal{L}|$. For a codeword $f \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$, let $\vec{f} \in \mathbb{F}^d$ denote the coefficient vector of $\hat{f}$.

3.2.1 Rational constraints

Definition 3.1. A *rational function* $\mathfrak{c} = (p, q)$ is a pair of arithmetic circuits, $p : \mathbb{F}^{k+1} \to \mathbb{F}$ and $q : \mathbb{F} \to \mathbb{F}$. For an interleaved word $\mathbf{f} = (f_1, \ldots, f_k)$, $f_i : \mathcal{L} \to \mathbb{F}$, we define $\mathfrak{c}(\mathbf{f}) : \mathcal{L} \to \mathbb{F}$ to be

$$\mathfrak{c}(\mathbf{f})(x) := \frac{p(x, f_1(x), \ldots, f_k(x))}{q(x)}.$$

A *rational constraint* consists of a rational function $\mathfrak{c}$ and a degree bound $d \in \mathbb{N}$. We say that the rational constraint is *satisfied with respect to* $\mathbf{f}$ if $\mathfrak{c}(f) \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$.

3.2.2 List decoding

Definition 3.2. Let $f : \mathcal{L} \to \mathbb{F}$ be a word, $d \in \mathbb{N}$ be a degree, and $\gamma \in (0, 1)$ be a list decoding parameter. We define $\mathsf{List}(f, d, \gamma) := \{g \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d] : \Delta(f, g) \leq \gamma\}$ to be the set of codewords that are γ-close to f. A Reed–Solomon code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ is (γ, ℓ)-*list decodable* if $|\mathsf{List}(f, d, \gamma)| \leq \ell$ for any word $f : \mathcal{L} \to \mathbb{F}$.

Theorem 3.3 (Johnson bound)**.** *The Reed–Solomon code* $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ *is* $(1 - \sqrt{\rho} - \eta, 1/(2\eta\sqrt{\rho}))$*-list-decodable for any choice of* $\eta \in (0, 1 - \sqrt{\rho})$*, where* ρ *is the rate of the code.*

Lemma 3.4 ([ACFY24, Lemma 4.5])**.** *Let* $f : \mathcal{L} \to \mathbb{F}$ *be a word,* $d \in \mathbb{N}$ *be a degree,* $s \in \mathbb{N}$ *be a repetition parameter, and* $\gamma \in (0, 1)$ *be a distance parameter. Suppose that* $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ *is* (γ, ℓ)*-list decodable. Then*

$$\Pr_{x_1, \ldots, x_s \leftarrow \mathbb{F} \setminus \mathcal{L}}[\exists u, u' \in \mathsf{List}(f, d, \gamma), u \neq u', \forall i \in [s], \hat{u}(x_i) = \hat{u}'(x_i)]$$
$$\leq \binom{\ell}{2} \cdot \left(\frac{d-1}{\mathbb{F} - |\mathcal{L}|}\right)^s \leq \frac{\ell^2}{2} \cdot \left(\frac{d-1}{\mathbb{F} - |\mathcal{L}|}\right)^s.$$

[4] Alternatively, a promise relation can be defined as a pair $(\mathcal{R}_{\mathtt{YES}}, \mathcal{R}_{\mathtt{NO}})$, where completeness holds for $\mathcal{R}_{\mathtt{YES}}$ and soundness holds for the complement of $\mathcal{R}_{\mathtt{NO}}$.

3.2.3 Quotients

Definition 3.5 ([ACFY24, Definition 4.2])**.** Let $f : \mathcal{L} \to \mathbb{F}$ be a word, $S \subset \mathbb{F}$ be a set, $\mathsf{Ans} : S \to \mathbb{F}$ be a function, and $\mathsf{Fill} : S \cap \mathcal{L} \to \mathbb{F}$ be a function. We define $\mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill}) : \mathcal{L} \to \mathbb{F}$ to be

$$\mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})(x) := \begin{cases} \mathsf{Fill}(x) & x \in S \\ \frac{f(x) - \hat{\mathsf{Ans}}(x)}{V_S(x)} & x \notin S. \end{cases}$$

Lemma 3.6 ([ACFY24, Lemma 4.4])**.** *Let $f : \mathcal{L} \to \mathbb{F}$ be a word, $d \in \mathbb{N}$ be a degree, $\delta \in (0, 1)$ be a distance parameter, $S \subset \mathbb{F}$ be a subset with $|S| < d$, and $\mathsf{Ans} : S \to \mathbb{F}$ be a function. Suppose that for every $u \in \mathsf{List}(f, d, \delta)$, there exists $x \in S$ such that with $\hat{u}(x) \neq \mathsf{Ans}(x)$. Then for any choice of* Fill,

$$\Delta(\mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill}), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d - |S|]) > \delta - |S \cap \mathcal{L}|/|\mathcal{L}|.$$

3.2.4 Proximity gaps

Definition 3.7 ([BCIKS23])**.** Let $\mathbb{F}$ be a field, $d \in \mathbb{N}$ be a degree, $\rho \in (0, 1)$ be a rate, $\delta \in (0, 1 - \sqrt{\rho})$ be a distance parameter, and $m \in \mathbb{N}$ be an arity. The *proximity error* is defined to be

$$\varepsilon_{\mathtt{prox}}(\mathbb{F}, d, \rho, \delta, m) := \begin{cases} \frac{(m-1) \cdot d}{\rho \cdot |\mathbb{F}|} & \delta \in \left(0, \frac{1-\rho}{2}\right] \\ \frac{(m-1) \cdot d^2}{|\mathbb{F}| \cdot (2 \cdot \min\{1 - \sqrt{\rho} - \delta, \sqrt{\rho}/20\})^7} & \delta \in \left(\frac{1-\rho}{2}, 1 - \sqrt{\rho}\right) \end{cases}$$

Definition 3.8 ([ACFY24, Definition 4.11])**.** Let $d_{\mathtt{max}} \in \mathbb{N}$ be a target degree, $r \in \mathbb{F}$ be a field element, $f_1, \ldots, f_m : \mathcal{L} \to \mathbb{F}$ be words, and $d_1, \ldots, d_m \in [d_{\mathtt{max}}]$ be degrees. We define $\mathsf{Combine}(d_{\mathtt{max}}, r, (f_1, d_1), \ldots, (f_m, d_m)) : \mathcal{L} \to \mathbb{F}$ to be

$$\begin{aligned} &\mathsf{Combine}(d_{\mathtt{max}}, r, (f_1, d_1), \ldots (f_m, d_m))(x) \\ &:= \sum_{i=1}^{m} r_i \cdot f_i(x) \cdot \left(\sum_{j=0}^{d_{\mathtt{max}} - d_i} (rx)^j \right) = \begin{cases} \sum_{i=1}^{m} r_i \cdot f_i(x) \cdot \left(\frac{1 - (rx)^{d_{\mathtt{max}} - d_i + 1}}{1 - rx} \right) & rx \neq 1 \\ \sum_{i=1}^{m} r_i \cdot f_i(x) \cdot (d_{\mathtt{max}} - d_i + 1) & rx = 1. \end{cases} \end{aligned}$$

Lemma 3.9 ([ACFY24, Lemma 4.13])**.** *Let $d_{\mathtt{max}} \in \mathbb{N}$ be a target degree, $f_1, \ldots, f_m : \mathcal{L} \to \mathbb{F}$ be words, $d_1, \ldots, d_m \in [d_{\mathtt{max}}]$ be degrees, and $\delta \in (0, 1 - \sqrt{\rho} - 1/|\mathcal{L}|)$ be a distance parameter, where $\rho := d^*/|\mathcal{L}|$. If*

$$\Pr_{r \leftarrow \mathbb{F}}[\Delta(\mathsf{Combine}(d_{\mathtt{max}}, r, (f_1, d_1), \ldots, (f_m, d_m)), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\mathtt{max}}]) \leq \delta]$$
$$< \varepsilon_{\mathtt{prox}} \left(d_{\mathtt{max}}, \rho, \delta, m \cdot (d_{\mathtt{max}} + 1) - \sum_{i=1}^{m} d_i \right),$$

then there exists a subset $S \subseteq \mathcal{L}$ with $|S| \geq (1 - \delta) \cdot |\mathcal{L}|$ such that for all $i \in [m]$, there exists a codeword $u \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i]$ such that u agrees with f_i on S.

3.3 Merkle trees

We recall the definition of Merkle commitments along with some useful security properties from [CY24, Section 18]. The Merkle commitment scheme is a tuple of deterministic polynomial-time oracle algorithms $\mathsf{MT} = (\mathsf{MT.Commit}, \mathsf{MT.Open}, \mathsf{MT.Check})$ implicitly parameterized by an output size $\sigma \in \mathbb{N}$, alphabet Σ, and string length $\ell \in \mathbb{N}$. All algorithms receive query access to a random oracle $\rho_{\mathsf{MT}} \in \mathcal{U}(\sigma)$.

- $\mathsf{MT.Commit}$ receives as input a string $m \in \Sigma^{\ell}$. It outputs a commitment $\mathsf{cm} \in \{0,1\}^{\sigma}$ and a trapdoor $\mathsf{td} \in \{0,1\}^{O(\sigma\ell)}$.
- $\mathsf{MT.Open}$ receives as input a trapdoor td and subset $I \subseteq [\ell]$. It outputs an opening proof $\mathsf{pf} \in \{0,1\}^{|I| \cdot \sigma \log \ell}$.
- $\mathsf{MT.Check}$ receives as input a commitment cm, restriction $a \in \Sigma^{I}$, and opening proof pf. It outputs a bit indicating whether or not the opening proof authenticates the restriction with respect to the commitment.

Lemma 3.10 (MT is complete)**.** *For every (unbounded) adversary $\mathcal{A}$,*

$$\Pr\left[\mathsf{MT.Check}^{\rho_{\mathtt{MT}}}(\mathsf{cm}, m|_I, \mathsf{pf}) = 1 \;\middle|\; \begin{array}{l} \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\sigma) \\ (m, I) \leftarrow \mathcal{A}^{\rho_{\mathtt{MT}}} \\ (\mathsf{cm}, \mathsf{td}) := \mathsf{MT.Commit}^{\rho_{\mathtt{MT}}}(m) \\ \mathsf{pf} := \mathsf{MT.Open}^{\rho_{\mathtt{MT}}}(\mathsf{td}, I) \end{array} \right] = 1.$$

Lemma 3.11 (MT is multi-extractable)**.** *There exists a deterministic polynomial-time algorithm* $\mathsf{MT.MultiExtract}$ *such that for every query bound $t \in \mathbb{N}$, t-query adversary $\mathcal{A}$, and $k \in \mathbb{N}$,*

$$\Pr\left[\begin{array}{l} \exists i \in [k] : \\ \quad \mathsf{MT.Check}^{\rho_{\mathtt{MT}}}(\mathsf{cm}_i, I_i, a_i, \mathsf{pf}_i) = 1 \\ \quad m_i[I_i] \neq a_i \\ \end{array} \;\middle|\; \begin{array}{l} \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\sigma) \\ (\mathsf{cm}_i, I_i, a_i, \mathsf{pf}_i)_{i \in [k]} \xleftarrow{\mathsf{tr}} \mathcal{A}^{\rho_{\mathtt{MT}}} \\ (m_i, \mathsf{td}_i)_{i \in [k]} := \mathsf{MT.MultiExtract}^{\rho_{\mathtt{MT}}}((\mathsf{cm}_i)_{i \in [k]}, \mathsf{tr}) \\ \forall i \in [k], \mathsf{pf}_i := \mathsf{MT.Open}^{\rho_{\mathtt{MT}}}(\mathsf{td}_i, I_i) \end{array} \right]$$

is at most

$$\kappa_{\mathtt{MT}}(t, \sigma, \ell, k) := \frac{3}{2} \cdot \frac{t^2}{2^{\sigma}} + \frac{k \cdot (\log \ell + 1) \cdot 3t}{2^{\sigma}}.$$

4 Non-interactive reductions

Let $\mathcal{R}_1^{\mathcal{U}}$ and $\mathcal{R}_2^{\mathcal{U}}$ be parameterized indexed promise relations (relative to a random oracle). A (preprocessing) *non-interactive reduction* from $\mathcal{R}_1$ to $\mathcal{R}_2$ in the random oracle model is a tuple of polynomial-time algorithms $\mathsf{RDX} = (\mathcal{G}, \mathcal{I}, \mathcal{P}, \mathcal{V})$, of which $\mathcal{I}$, $\mathcal{P}$, $\mathcal{V}$ have access to the same random oracle, with the following syntax.

- The generator $\mathcal{G}$ receives as input a security parameter λ (in unary) and outputs public parameters pp.
- The indexer $\mathcal{I}$ is a deterministic algorithm which receives as input public parameters pp and index $\mathbb{i}$, and outputs a proving key pk, verification key vk, and new index $\mathbb{i}'$.
- The prover $\mathcal{P}$ receives as input proving key pk, an instance $\mathbb{x}$, and witness $\mathbb{w}$, and outputs a proof π and new witness $\mathbb{w}'$.
- The verifier $\mathcal{V}$ is a deterministic algorithm[5] which receives as input verification key vk, instance $\mathbb{x}$, and proof π, and outputs a new instance $\mathbb{x}'$.

Completeness. RDX is complete if the following holds. For every adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|r} & \rho \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_1^{\rho}(\mathsf{pp}) & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ \Downarrow & (\mathbb{i}, \mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}^{\rho}(\mathsf{pp}) \\ (\mathbb{i}', \mathbb{x}', \mathbb{w}') \in \mathcal{R}_2^{\rho}(\mathsf{pp}) & (\mathsf{pk}, \mathsf{vk}, \mathbb{i}') \leftarrow \mathcal{I}^{\rho}(\mathsf{pp}, \mathbb{i}) \\ & (\pi, \mathbb{w}') \leftarrow \mathcal{P}^{\rho}(\mathsf{pk}, \mathbb{x}, \mathbb{w}) \\ & \mathbb{x}' \leftarrow \mathcal{V}^{\rho}(\mathsf{vk}, \mathbb{x}, \pi) \end{array}\right] = 1.$$

Straightline knowledge soundness. RDX is straightline knowledge sound (with respect to auxiliary input distribution $\mathcal{D}$) if the following holds. There exists a deterministic polynomial-time extractor $\mathcal{E}$ such that for every (non-uniform) polynomial-time adversary $\tilde{\mathcal{P}}$,

$$\Pr\left[\begin{array}{c|r} & \rho \leftarrow \mathcal{U}(\lambda) \\ & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathbb{i}', \mathbb{x}', \mathbb{w}') \in \tilde{\mathcal{R}}_2^{\rho}(\mathsf{pp}) & \mathsf{ai} \leftarrow \mathcal{D}(1^\lambda) \\ \wedge & (\mathbb{i}, \mathbb{x}, \pi, \mathbb{w}'; \mathsf{tr}) \leftarrow \tilde{\mathcal{P}}^{\rho}(\mathsf{pp}, \mathsf{ai}) \\ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \tilde{\mathcal{R}}_1^{\rho}(\mathsf{pp}) & (\mathsf{pk}, \mathsf{vk}, \mathbb{i}') \leftarrow \mathcal{I}^{\rho}(\mathsf{pp}, \mathbb{i}) \\ & \mathbb{x}' \leftarrow \mathcal{V}^{\rho}(\mathsf{vk}, \mathbb{x}, \pi) \\ & \mathbb{w} \leftarrow \mathcal{E}(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \pi, \mathbb{w}', \mathsf{ai}, \mathsf{tr}) \end{array}\right] \leq \operatorname{negl}(\lambda).$$

Remark 4.1. We have defined non-interactive reductions for indexed relations in full generality (parameterized, relative to a random oracle, and relaxed soundness). Sometimes, full generality is not required; for example, suppose $\mathcal{R}$ is a parameterized indexed relation. Then soundness should hold for the same relation $\mathcal{R}$, and the random oracle is not considered when testing membership in $\mathcal{R}$.

4.1 IVC and PCD from non-interactive reductions

We construct a non-interactive argument ARG (Definition A.2) for a relation $\mathcal{R}$ and a corresponding accumulation scheme ACC for ARG (Definition A.3) from two non-interactive reductions,

[5] Proof systems do not typically require verifier to be deterministic. For reductions, deterministic verifiers are useful because it implies that the prover can compute the new instance. Moreover, deterministic verifiers are easy to attain; intuitively, this is because any randomness can be derived from the random oracle.

- $\mathsf{RDX}_{\mathtt{CAST}}$, a non-interactive reduction from $\mathcal{R}$ to some relation $\mathcal{R}_{\mathtt{ACC}}$, and
- $\mathsf{RDX}_{\mathtt{FOLD}}$, a non-interactive reduction from $\mathcal{R}^*_{\mathtt{ACC}}$ to $\mathcal{R}_{\mathtt{ACC}}$,

all of which are in the random oracle model. Crucially, to construct Proof-Carrying Data (PCD), prior work [BCMS20; BCLMS21; COS20] requires a non-interactive argument for an NP-complete relation and an accumulation scheme for that argument in *the standard model*. However, these works only provide secure constructions of such non-interactive arguments and accumulation schemes in the random oracle model. By heuristically instantiating the random oracle with an appropriate hash function, they obtain non-interactive arguments and corresponding accumulation schemes in the standard model with conjectured security (sometimes referred to as *heuristic security*). This is a well-known limitation of the random oracle methodology [CGH04; GK03]. We can follow the same approach to obtain a non-interactive argument and accumulation scheme in the standard model. Furthermore, these prior works also require that the accumulation verifier is succinct (sublinear, Theorem 5.3 in [BCLMS21]) such that the corresponding circuit description does not increase in size after each recursive step. If the reduction verifier of $\mathsf{RDX}_{\mathtt{CAST}}$ and $\mathsf{RDX}_{\mathtt{FOLD}}$ are succinct, then our constructions will trivially satisfy this requirement. What follows is our formal theorem about the transformation from non-interactive reductions to arguments and accumulation. For brevity, we defer the explicit constructions to Appendix A.2; see Section 2.3 for a high-level overview.

Definition 4.2. Let $\mathcal{R}^{\mathcal{U}}$ be a parameterized indexed promise relation. The corresponding *multi-instance relation* $(\mathcal{R}^*)^{\mathcal{U}}$ is defined to be

$$\mathcal{R}^*(\mathbb{p})^{\rho} := \{(\mathbb{i}, (\mathbb{x}_1, \ldots, \mathbb{x}_m), (\mathbb{w}_1, \ldots, \mathbb{w}_m)) : \forall i \in [m], (\mathbb{i}, \mathbb{x}_i, \mathbb{w}_i) \in \mathcal{R}(\mathbb{p})^{\rho}\}.$$

The relaxed relation $\tilde{\mathcal{R}}^*(\mathbb{p})$ is defined analogously. A *many-to-one reduction* is a non-interactive reduction from $\mathcal{R}$ to $\mathcal{R}^*$ which preserves the index: given an index $\mathbb{i}$, the indexer $\mathcal{I}$ outputs the new index $\mathbb{i}' := \mathbb{i}$ (this requirement is necessary for repeatedly composing many-to-one reductions).

Theorem 4.3. *There exists a polynomial-time transformation* T *such that the following holds. Let* $\mathcal{R}$ *be a parameterized indexed relation. Let* $\mathcal{R}^{\mathcal{U}}_{\mathtt{ACC}}$ *be a parameterized indexed promise relation in* $\mathsf{NP}^{\mathcal{U}}$ *with the same parameter space as* $\mathcal{R}$*. Suppose we are given the following non-interactive reductions in the random oracle model:*

- $\mathsf{RDX}_{\mathtt{CAST}} = (\mathcal{G}_{\mathtt{CAST}}, \mathcal{I}_{\mathtt{CAST}}, \mathcal{P}_{\mathtt{CAST}}, \mathcal{V}_{\mathtt{CAST}})$*, a reduction from* $\mathcal{R}$ *to* $\mathcal{R}_{\mathtt{ACC}}$*.*
- $\mathsf{RDX}_{\mathtt{FOLD}} = (\mathcal{G}_{\mathtt{FOLD}}, \mathcal{I}_{\mathtt{FOLD}}, \mathcal{P}_{\mathtt{FOLD}}, \mathcal{V}_{\mathtt{FOLD}})$*, a many-to-one reduction from* $\mathcal{R}^*_{\mathtt{ACC}}$ *to* $\mathcal{R}_{\mathtt{ACC}}$ *with the same generator algorithm as* $\mathsf{RDX}_{\mathtt{CAST}}$ *(i.e.,* $\mathcal{G}_{\mathtt{FOLD}} \equiv \mathcal{G}_{\mathtt{CAST}}$*).*

Then $\mathrm{T}[\mathsf{RDX}_{\mathtt{CAST}}, \mathsf{RDX}_{\mathtt{FOLD}}, \mathcal{R}_{\mathtt{ACC}}] = (\mathsf{ARG}, \mathsf{ACC})$*, where* ARG *is a non-interactive argument for* $\mathcal{R}$ *and* ACC *is an accumulation scheme for* ARG*, both in the random oracle model.*

Proof. We defer the proof to Appendix A. □

5 Interactive oracle reductions

We define *interactive oracle reductions* (IORs), an information-theoretic proof system that adapts interactive oracle proofs to the language of reductions. Intuitively, an IOR is an interactive reduction where the verifier has oracle access to the prover's messages and reads a small number of locations to output the new instance. The key novelty, however, is that an IOR verifier may also want to output claims about proof strings *without fully reading them*.

Formally, we consider *indexed oracle relations*[6] where the instance is split into a *short instance* $\mathbb{x}$ and a tuple of *instance strings* $\vec{\mathbb{y}} = (\mathbb{y}_1, \ldots, \mathbb{y}_{\mathsf{n}})$ written over some alphabet Σ. For notational convenience, we write $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \in \mathcal{R}$ to denote membership in an indexed oracle relation. In the relaxed relation $\tilde{\mathcal{R}}$, we allow instance strings to written over an augmented alphabet $\Sigma \sqcup \{\perp\}$; this will be useful when compiling IORs.

5.1 Definition

Let $\mathcal{R}$ and $\mathcal{R}'$ be indexed oracle promise relations. A (public-coin, holographic) interactive oracle reduction from $\mathcal{R}$ to $\mathcal{R}'$ is a tuple of polynomial-time algorithms $\mathsf{IOR} = (\mathbf{I}, \mathbf{P}, \mathbf{V})$ with the following syntax.

- The indexer $\mathbf{I}$ is a deterministic algorithm which receives as input an index $\mathbb{i}$ (for $\mathcal{R}$). It outputs a *short index* ι, *index string* $\mathbb{I}$, and new index $\mathbb{i}'$ (for $\mathcal{R}'$).
- The prover $\mathbf{P}$ is an interactive algorithm which receives as input the index $\mathbb{i}$, short instance $\mathbb{x}$, instance strings $\vec{\mathbb{y}}$, and witness $\mathbb{w}$. It engages in k rounds of interaction. In the i-th round, it sends a proof string Π_i, then receives a challenge r_i.
- The verifier $\mathbf{V}$ is an interactive algorithm which receives as input the short index ι, short instance $\mathbb{x}$, oracle access to index string $\mathbb{I}$, and oracle access to instance strings $\vec{\mathbb{y}}$. It engages in k rounds of interaction. In each round, it receives oracle access to a proof string Π_i, then sends a uniformly random challenge r_i. At the end of the protocol, it outputs a new instance $(\mathbb{x}', \vec{\mathbb{y}}')$; the new instance oracles are chosen from the old instance oracles or proof oracles received during the interactive protocol.

Without loss of generality, the verifier is split into two phases. In the *interaction phase*, it samples challenges and sends them to the prover. In the *query phase*, it queries the index, instance, and proof oracles. The verifier's output is a deterministic function of its input and the transcript, which we denote

$$(\mathbb{x}', \vec{\mathbb{y}}') := \mathbf{V}^{\mathbb{I}, \vec{\mathbb{y}}, \vec{\Pi}}(\iota, \mathbb{x}, \vec{r}).$$

Completeness. IOR is complete if the following holds. For any $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \in \mathcal{R}$,

$$\Pr\left[(\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}') \in \mathcal{R}' \;\middle|\; \begin{array}{l} (\iota, \mathbb{I}, \mathbb{i}') \leftarrow \mathbf{I}(\mathbb{i}) \\ (\mathbb{w}', (\mathbb{x}', \vec{\mathbb{y}}')) \leftarrow \langle \mathbf{P}(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}), \mathbf{V}^{\mathbb{I}, \vec{\mathbb{y}}}(\iota, \mathbb{x}) \rangle \end{array}\right] = 1.$$

Soundness. IOR has soundness error ε if the following holds. For any $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}) \notin \mathcal{L}(\mathcal{R})$ and adversary $\tilde{\mathbf{P}}$,

$$\Pr\left[(\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}') \in \mathcal{R}' \;\middle|\; \begin{array}{l} (\iota, \mathbb{I}, \mathbb{i}') \leftarrow \mathbf{I}(\mathbb{i}) \\ (\mathbb{w}', (\mathbb{x}', \vec{\mathbb{y}}')) \leftarrow \langle \tilde{\mathbf{P}}, \mathbf{V}^{\mathbb{I}, \vec{\mathbb{y}}}(\iota, \mathbb{x}) \rangle \end{array}\right] \leq \varepsilon(\mathbb{i}, \mathbb{x}).$$

Efficiency measures. We consider the following efficiency measures (these may be functions of the short index and short instance).

[6] We emphasize that is not the same as a relation relative to a (random) oracle.

- *Alphabet:* Σ is the set of symbols used to write the index, instance, and proof strings.
- *Round complexity:* k is the number of back-and-forth interactions in the protocol.
- *Proof length:* $\mathsf{L}^{\mathtt{I}}$ is the length of the index string, $\mathsf{L}^{\mathtt{Y}}_j$ is the length of the j-th instance string, and $\mathsf{L}^{\mathtt{P}}_i$ is the length of the i-th proof string. Let $\mathsf{L}_{\mathtt{max}}$ denote the maximum string length.
- *Query complexity:* the verifier reads $\mathsf{q}^{\mathtt{I}}$ locations from the index string, $\mathsf{q}^{\mathtt{Y}}_j$ locations from the j-th instance string, and $\mathsf{q}^{\mathtt{P}}_i$ locations from the i-th proof string. Let q denote the total number of queries.
- *Randomness:* r_i is the number of bits in the i-th challenge sent by the verifier.

Parameterized reductions. We often parameterize reductions, e.g., by a security parameter. Formally, the prover and indexer will additionally receive as input some parameters $\mathbb{p}$. Completeness must hold for any choice of parameters, and the soundness error is allowed to be a function of the parameters (in addition to the index and short instance).

Remark 5.1 (non-oracle messages)**.** In our constructions, the prover sometimes sends non-oracle messages in the sense that the verifier reads the entire message (and hence oracle access is unnecessary). We do not include this in the IOR definition, but it is straightforward to do so. As an efficiency measure, let s denote the total size (in bits) of the non-oracle messages.

Remark 5.2 (oracle selection)**.** When the verifier outputs new instance oracles $\vec{\mathbb{y}}' = (\mathbb{y}'_1, \ldots, \mathbb{y}'_{\mathsf{n}'})$, this should not blow up its runtime or query complexity. We can formalize this by having the verifier output a tuple of indices $\vec{s} = (s_1, \ldots, s_{\mathsf{n}'})$, $s_j \in [\mathsf{n} + \mathsf{k}]$, which "select" from the old instance and proof oracles. In particular, the j-th new instance string will be $\mathbb{y}'_j := \mathsf{Select}(\vec{\mathbb{y}}, \Pi, s_j)$, where

$$\mathsf{Select}(\vec{\mathbb{y}}, \vec{\Pi}, s) := \begin{cases} \mathbb{y}_s & 1 \leq s \leq \mathsf{n} \\ \Pi_{s-\mathsf{n}} & \mathsf{n} < s \leq \mathsf{n} + \mathsf{k}. \end{cases}$$

5.2 Round-by-round soundness

We define round-by-round soundness and knowledge. These are stronger notions of soundness that allow us to transform IORs into non-interactive reductions.

Definition 5.3. A *state function* for IOR is a function State for which the following holds.

- *Empty transcript:* $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \varnothing) = 0$ unconditionally, where $\varnothing$ denotes the empty transcript.
- *Prover moves:* If $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \tau) = 0$ for a partial transcript $\tau = (\Pi_1, r_i, \ldots, \Pi_{i-1}, r_{i-1})$, $i \in [\mathsf{k}]$, where the prover is about to move, then for any prover message $\Pi_i \in \mathsf{L}_{\mathsf{P},i}$, $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \tau || \Pi_i) = 0$.
- *Full transcript:* If $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \tau) = 0$ for a full transcript $\tau = (\Pi_1, r_1, \ldots, \Pi_{\mathsf{k}}, r_{\mathsf{k}})$, then the verifier outputs $(\mathbb{x}', \vec{\mathbb{y}}') := \mathbf{V}^{\mathbb{I}, \vec{\mathbb{y}}, \vec{\Pi}}(\iota, \mathbb{x}, \vec{r})$ such that $(\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}') \notin \mathcal{L}(\tilde{\mathcal{R}}')$ (where $(\iota, \mathbb{I}, \mathbb{i}') := \mathbf{I}(\mathbb{p}, \mathbb{i})$).

Definition 5.4. IOR has *round-by-round soundness error* $\varepsilon_{\mathtt{rbr}}$ if there exists a state function State such that the following holds. If $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \tau) = 0$ for $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}) \notin \mathcal{L}(\mathcal{R})$ and a partial transcript $\tau = (\Pi_1, r_1, \ldots, \Pi_i)$, $i \in [\mathsf{k}]$, where the verifier is about to move, then

$$\Pr_{r_i \leftarrow \{0,1\}^{\mathsf{r}_i}} [\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \tau || r_i) = 1] \leq \varepsilon_{\mathtt{rbr}}(\mathbb{p}, \mathbb{i}, \mathbb{x}).$$

Definition 5.5. IOR has *round-by-round knowledge error* $\kappa_{\mathtt{rbr}}$ if there exists a state function State and polynomial-time extractor $\mathbf{E}$ such that the following holds. If $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \tau) = 0$ for a partial transcript $\tau = (\Pi_1, r_1, \ldots, \Pi_i)$, $i \in [\mathsf{k}]$, where the verifier is about to move, then

$$\Pr_{r_i \leftarrow \{0,1\}^{\mathsf{r}_i}}[\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \tau || \Pi_i || r_i) = 1] > \kappa_{\mathtt{rbr}}(\mathbb{p}, \mathbb{i}, \mathbb{x})$$

implies the following. For any transcript continuation $(\Pi_{i+1}, \ldots, \Pi_m, \mathbb{w}')$, the extractor outputs $\mathbb{w} \leftarrow \mathbf{E}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \vec{\Pi}, \mathbb{w}')$ such that $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \in \tilde{\mathcal{R}}$.

Remark 5.6 (limitations of round-by-round knowledge)**.** Definition 5.5 is unsatisfactory in the sense that the extractor cannot meaningfully take advantage of the new witness to extract the old witness. It nevertheless suffices because in our constructions, either (a) the prover sends the witness in one shot; or (b) the relation has empty witnesses (hence, round-by-round soundness trivially implies round-by-round knowledge).

5.3 Non-interactive reductions from IORs

We show how to transform IORs into non-interactive reductions.

- In Definition 5.7 we define committed relations. Informally, given an oracle relation $\mathcal{R}$, the committed relation $\mathsf{Com}[\mathcal{R}]$ replaces instance strings with Merkle commitments and adds trapdoors Merkle authentication paths to the witness.

- In Definition 5.8 we define monotone relations. This is a minor technical detail which is required for the transformation to preserve soundness; informally, it ensures that a cheating non-interactive prover cannot gain an advantage by withholding Merkle authentication paths from the new witness. All relations considered in this work are monotone.

- In Theorem 5.9 we give a formal theorem statement for transforming IORs into non-interactive reductions. This is essentially the BCS transformation (with preprocessing), except we also need to handle instance oracles.

Definition 5.7. Let $\mathcal{R}$ be an indexed oracle promise relation and S be a parameterized set. The *committed relation* $\mathsf{Com}[\mathcal{R}, S] = \mathcal{S}^{\mathcal{U}}$ is the parameterized indexed promise relation (relative to a random oracle) defined below.

$$\mathcal{S}^{\rho}(\mathbb{p}) := \left\{ (\mathbb{i}, (\mathbb{x}, \vec{\mathsf{cm}}), (\mathbb{w}, \vec{\mathbb{y}}, \vec{\mathsf{td}})) : \begin{array}{l} (\mathbb{i}, \mathbb{x}) \in S(\mathbb{p}) \\ (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \in \mathcal{R}(\mathbb{p}) \\ \forall j \in [\mathsf{n}], \mathsf{MT.Commit}^{\rho_{\mathsf{MT}}}(\mathbb{y}_j) = (\mathsf{cm}_j, \mathsf{td}_j) \end{array} \right\}$$

$$\tilde{\mathcal{S}}^{\rho}(\mathbb{p}) := \left\{ (\mathbb{i}, (\mathbb{x}, \vec{\mathsf{cm}}), (\mathbb{w}, \vec{\mathbb{y}}, \vec{\mathsf{td}})) : \begin{array}{l} (\mathbb{i}, \mathbb{x}) \in S(\mathbb{p}) \\ (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \in \tilde{\mathcal{R}}(\mathbb{p}) \\ \forall j \in [\mathsf{n}] : \\ \quad \mathsf{pf}_j := \mathsf{MT.Open}^{\rho_{\mathsf{MT}}}(\mathsf{td}_j, \mathrm{Dom}\, \mathbb{y}_j) \\ \quad \mathsf{MT.Check}^{\rho_{\mathsf{MT}}}(\mathsf{cm}_j, \mathbb{y}_j, \mathsf{pf}_j) = 1 \end{array} \right\}$$

Above, $\rho_{\mathtt{MT}}$ is a random oracle in $\mathcal{U}(\sigma)$ derived from ρ. Observe that if $\mathcal{R}$ is in NP and S is efficiently computable, then $\mathcal{S}^{\mathcal{U}}$ is in $\mathsf{NP}^{\mathcal{U}}$.

Definition 5.8. We say that an indexed oracle promise relation $\mathcal{R}$ is *monotone* if the relaxed relation $\tilde{\mathcal{R}}$ satisfies the following property. Suppose $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}) \notin \mathcal{L}(\tilde{\mathcal{R}})$ with $\mathbb{y}_j : I_j \to \Sigma$ (recall that a string written over $\Sigma \sqcup \{\perp\}$ can be interpreted as a restriction to a subset of indices). Then $(\mathbb{i}, \mathbb{x}, (\mathbb{y}_1|_{A_1}, \ldots, \mathbb{y}_{\mathsf{n}}|_{A_{\mathsf{n}}})) \notin \mathcal{L}(\tilde{\mathcal{R}})$ for all subsets $A_1, \ldots, A_{\mathsf{n}}$ with $A_j \subseteq I_j$.

Theorem 5.9. *There exists a polynomial-time transformation* T *such that the following holds. Let* $\mathcal{R}$ *and* $\mathcal{R}'$ *be indexed promise relations (with strings). Let* S *and* S' *be efficiently computable sets parameterized by* $\lambda \in \mathbb{N}$*. Let* IOR *be an interactive oracle reduction (also parameterized by* λ*) from* $\mathcal{R}$ *to* $\mathcal{R}'$ *such that the following holds:*

- IOR *has round-by-round knowledge error* $\kappa_{\mathtt{rbr}}$ *such that*

$$\max_{\substack{\mathbb{i}, \mathbb{x} \in S(\lambda) \\ |\mathbb{i}| + |\mathbb{x}| = \mathrm{poly}(\lambda)}} \kappa_{\mathtt{rbr}}(\lambda, \mathbb{i}, \mathbb{x}) = \mathrm{negl}(\lambda).$$

- *For every parameter* $\lambda \in \mathbb{N}$ *and index* $\mathbb{i} \in S(\lambda)$*, the IOR indexer outputs a new index* $\mathbb{i}' \in S'(\lambda)$*.*

Then $\mathrm{T}[\mathsf{IOR}]$ *is a non-interactive reduction from* $\mathsf{Com}[\mathcal{R}, S]$ *to* $\mathsf{Com}[\mathcal{R}', S']$ *with the following efficiency measures:*

- *Proof size:* $O(\lambda \cdot \mathsf{k} + \mathsf{s} + \mathsf{q} \cdot (\log |\Sigma| + \lambda \cdot \log \mathsf{L}_{\mathtt{max}}))$.
- *Verifier complexity: IOR verifier, plus* $O(\mathsf{k} + \mathsf{q} \cdot \log \mathsf{L}_{\mathtt{max}})$ *queries to the random oracle.*

Proof. We defer the proof to Appendix B. □

6 Accumulation for Reed–Solomon proximity claims

We describe a many-to-one reduction for Reed–Solomon proximity claims. Intuitively, a proximity claim consists of a rational constraint $(\mathfrak{c}, d)$ and interleaved word $\mathbf{f} = (f_1, \ldots, f_k)$ such that $\mathfrak{c}(\mathbf{f})$ is δ-close to $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$. For completeness, we require that $\mathfrak{c}(\mathbf{f})$ is an exact codeword.

Definition 6.1 (Reed–Solomon proximity relation)**.** The index consists of the following:

- Description of a finite field $\mathbb{F}$ and evaluation domain $\mathcal{L} \subset \mathbb{F}$.
- Maximum degree parameter $d_{\mathtt{max}} \in \mathbb{N}$, $d_{\mathtt{max}} < |\mathcal{L}|$. Define $\rho := (d_{\mathtt{max}} + 1)/|\mathcal{L}|$ to be the rate of the corresponding Reed–Solomon code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\mathtt{max}}]$.
- Distance parameter $\delta \in (0, 1)$.

The instance is a rational constraint $(\mathfrak{c}, d)$, where $d \leq d_{\mathtt{max}}$. The instance oracles $\mathbf{f} = (f_1, \ldots, f_k)$ are words $f_i : \mathcal{L} \to \mathbb{F}$, collectively interpreted as an interleaved word in $(\mathbb{F}^k)^{\mathcal{L}}$. The witness is empty. We define the indexed promise relation $\mathcal{R}_{\mathtt{RS}}$ below.

$$\mathcal{R}_{\mathtt{RS}} := \{(\mathbb{i}, (\mathfrak{c}, d), \mathbf{f}, \perp) : \mathfrak{c}(\mathbf{f}) \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]\}$$
$$\tilde{\mathcal{R}}_{\mathtt{RS}} := \{(\mathbb{i}, (\mathfrak{c}, d), \mathbf{f}, \perp) : \Delta(\mathfrak{c}(\mathbf{f}), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]) \leq \delta\}$$

Theorem 6.2. *Consider a security parameter $\lambda \in \mathbb{N}$. Assume the index and instance are such that the following holds:*

- $|\mathbb{F}| \geq 2^{\lambda} \cdot 10^7 \cdot m \cdot {d_{\mathtt{max}}}^3 \cdot \rho^{-3.5}$.
- $\delta \in \left(0, 1 - 1.05 \cdot \sqrt{\rho} - \frac{\lambda}{-\log(1-\delta) \cdot |\mathcal{L}|}\right)$.

Then Construction 6.3, when instantiated with parameters $s = 1$ and $t = \frac{\lambda}{-\log(1-\delta)}$, is an interactive oracle reduction from $\mathcal{R}^{*}_{\mathtt{RS}}$ to $\mathcal{R}_{\mathtt{RS}}$ with round-by-round soundness error $2^{-\lambda}$ and the following efficiency measures:

- *Alphabet: $\mathbb{F}$.*
- *Round complexity: 3. The verifier sends the first message.*
- *Query complexity: $t \cdot \sum_{i=1}^{m} k_i$, where k_i is the number of oracles in the i-th instance.*
- *Proof length: $|\mathcal{L}| + t + 1$ field elements.*
- *Prover time: $O(|\mathcal{L}| \cdot \sum_{i=1}^{m} |\mathfrak{c}_i| + d_{\mathtt{max}} \log d_{\mathtt{max}})$ field operations, where $|\mathfrak{c}_i|$ denotes the circuit size of the i-th instance's constraint.*
- *Verifier time: $O(t \cdot \sum_{i=1}^{m} |\mathfrak{c}_i|)$ field operations.*

Moreover, the size of the new instance is independent of the size of the old instance.

Proof. Follows from instantiating Construction 6.3 with $s := 1$ and $t := \frac{\lambda}{-\log(1-\delta)}$, by Lemma 6.4 and Lemma 6.5.

Soundness. We analyze the soundness error given the defined size of $\mathbb{F}$:

- For $d_j \geq 1$ we have that $\varepsilon_{\mathtt{prox}}(d_{\mathtt{max}}, \rho, \delta, m \cdot (d_{\mathtt{max}} + 1) - \sum_{j=1}^{m} d_j) \leq \varepsilon_{\mathtt{prox}}(d_{\mathtt{max}}, \rho, \delta, m \cdot d_{\mathtt{max}}) \leq \frac{m \cdot {d_{\mathtt{max}}}^3}{|\mathbb{F}| \cdot (\frac{\sqrt{\rho}}{10})^7}$ (Lemma 3.7)
- $\frac{(m-1) \cdot {d_{\mathtt{max}}}^2}{|\mathbb{F}| \cdot (\frac{\sqrt{\rho}}{10})^7} \leq \frac{(m-1) \cdot {d_{\mathtt{max}}}^2}{2^{\lambda} \cdot (m-1) \cdot 10^7 \cdot {d_{\mathtt{max}}}^2 \cdot \rho^{-3.5} \cdot \frac{\rho^{3.5}}{10^7}} = 2^{-\lambda}$ (Plugging in the value for $\mathbb{F}$)
- $\ell \leq \frac{1}{2 \cdot \frac{\sqrt{\rho}}{20} \cdot \sqrt{\rho}} = \frac{1}{\rho/10}$ (By Theorem 3.3)
- $\varepsilon_{\mathtt{ood}} = \frac{\ell^2}{2} \cdot (\frac{d_{\mathtt{max}}}{|\mathbb{F}| - |\mathcal{L}|})$.
- Assuming that $d_{\mathtt{max}} < |\mathcal{L}| \leq |\mathbb{F}|/2$, we get $\varepsilon_{\mathtt{ood}} \leq \frac{100}{\rho^2} \cdot \frac{d_{\mathtt{max}}}{\mathbb{F}}$

- Since $|\mathbb{F}| \geq 2^{\lambda} \cdot 100 \cdot d_{\mathtt{max}} \cdot \rho^{-2}$ we have that $\varepsilon_{\mathtt{ood}} \leq 2^{-\lambda}$
- $(1-\delta)^t \leq (1-\delta)^{\frac{\lambda}{-\log(1-\delta)}} = 2^{-\lambda}$

Since the RBR-soundness error is the max of these errors, we have that, for the chosen parameters, it is bound by $2^{-\lambda}$.

Efficiency. The protocol has 3 rounds, 3 prover and 3 verifier messages. The number of oracle queries is $t = \frac{\lambda}{-\log(1-\delta)}$. The total proof lengths consists of one oracle of length $|\mathcal{L}|$ and one $\mathbb{F}$ elements, along with Fill of size $t = \frac{\lambda}{-\log(1-\delta)}$. The prover's runtime is dominated by the computation of f, which takes $O(m \cdot |\mathfrak{c}_f| \cdot |\mathcal{L}|) = O(m \cdot t \cdot |\mathcal{L}|)$ field operations, as well as the computation of Fill. Computing Fill can be done using an FFT and takes $O(d_{\mathtt{max}} \cdot \log d_{\mathtt{max}})$ field operations. See Section 6.3 for an optimization that can significantly reduce the number of FFT operations in a repeated invocation of the accumulation scheme. The verifier needs to evaluate f' at t positions. Each evaluation requires $O(m \cdot t)$ field operation. The overall runtime is $O(m \cdot t^2)$ field operations. □

6.1 Construction

We describe an interactive oracle reduction from $\mathcal{R}^*_{\mathtt{RS}}$ to $\mathcal{R}_{\mathtt{RS}}$.

Construction 6.3. The prover and indexer are parameterized by $\mathbb{p} = (s, t)$, where $s, t \in \mathbb{N}$ are the out-of-domain and in-domain repetition parameters. On input index $\mathbb{i}$, the indexer $\mathbf{I}$ outputs short index $\iota := (\mathbb{i}, s, t)$ and new index $\mathbb{i}' := \mathbb{i}$; there is no index oracle string. On input instance $\mathbb{x} = (\mathfrak{c}_i, d_i)_{i \in [m]}$ and instance oracles $\vec{\mathbb{y}} = (\mathbf{f}_i)_{i \in [m]}$, the prover $\mathbf{P}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}})$ and verifier $\mathbf{V}^{\vec{\mathbb{y}}}(\iota, \mathbb{x})$ engage in the following protocol.

Interaction phase.

1. $\mathbf{V}$ sends $r \leftarrow \mathbb{F}$.
2. $\mathbf{P}$ sends $f : \mathcal{L} \to \mathbb{F}$. In the honest case, $\hat{f} := \mathsf{Combine}(d_{\mathtt{max}}, r, (\mathfrak{c}_i(\mathbf{f}_i), d_i)_{i \in [m]})$.
3. $\mathbf{V}$ sends $x^{\mathtt{out}}_1, \ldots, x^{\mathtt{out}}_s \leftarrow \mathbb{F} \setminus \mathcal{L}$.
4. $\mathbf{P}$ sends $y_1, \ldots, y_s \in \mathbb{F}$. In the honest case, $y_j := \hat{f}(x^{\mathtt{out}}_j)$.
5. $\mathbf{V}$ sends $x^{\mathtt{in}}_1, \ldots, x^{\mathtt{in}}_t \leftarrow \mathcal{L}$. Define $S := \{x^{\mathtt{out}}_1, \ldots, x^{\mathtt{out}}_s, x^{\mathtt{in}}_1, \ldots, x^{\mathtt{in}}_t\}$.
6. $\mathbf{P}$ sends $\mathsf{Fill} : \{x^{\mathtt{in}}_j\}_{j \in [t]} \to \mathbb{F}$. In the honest case, $\mathsf{Fill} := \mathsf{PolyQuotient}(\hat{f}, S)$, restricted to $\{x^{\mathtt{in}}_j\}_{j \in [t]}$.

Query phase.

1. Define the virtual function $f' := \mathsf{Combine}(d_{\mathtt{max}}, r, (\mathfrak{c}_i(\mathbf{f}_i), d_i)_{i \in [m]})$.
2. Define $\mathsf{Ans} : S \to \mathbb{F}$ such that $\mathsf{Ans}(x^{\mathtt{out}}_j) := y_j$ and $\mathsf{Ans}(x^{\mathtt{in}}_j) := f'(x^{\mathtt{in}}_j)$.
3. Define the rational function $\mathfrak{c} := \mathsf{Quotient}(\cdot, S, \mathsf{Ans}, \mathsf{Fill})$ and degree constraint $d := d_{\mathtt{max}} - |S|$.
4. $\mathbf{V}$ outputs the new instance $\mathbb{x}' := (\mathfrak{c}, d)$ and instance oracle $\mathbb{y}' := f$.

Lemma 6.4. *Construction 6.3 is complete.*

Proof. Since each $\mathfrak{c}_i(\mathbf{f}_i)$ is a codeword in $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i]$, the combination f' is a codeword in $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\mathtt{max}}]$. The prover computes $f = f'$, so $\hat{f}$ is a polynomial of degree at most $d_{\mathtt{max}}$ and $\mathsf{PolyQuotient}(\hat{f}, S)$ is a polynomial of degree at most $d_{\mathtt{max}} - |S|$. Moreover, $\hat{f}$ agrees with Ans on S; this is because the prover computes the out-of-domain responses honestly, and f agrees with f' (which is virtually computed by the verifier). Finally, $\mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})$ is precisely the evaluations of $\mathsf{PolyQuotient}(\hat{f}, S)$ over $\mathcal{L}$; this is because the prover computes the hole fills honestly. We conclude that $\mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill})$ is a codeword in $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\mathtt{max}} - |S|]$. □

6.2 Soundness analysis

Lemma 6.5. *Construction 6.3 has round-by-round soundness error*

$$\varepsilon_{\mathtt{rbr}}(\mathbb{p}, \mathbb{i}, \mathbb{x}) := \max\left(\varepsilon_{\mathtt{prox}}\left(d_{\mathtt{max}}, \rho, \delta, m \cdot (d_{\mathtt{max}} + 1) - \sum_{i=1}^{m} d_i\right), \varepsilon_{\mathtt{ood}}(d_{\mathtt{max}}, \ell, s), (1-\delta)^t\right),$$

where ℓ is such that $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\mathtt{max}}]$ is $(\delta + t/|\mathcal{L}|, \ell)$-list decodable.

Proof. Define $\gamma := \delta + t/|\mathcal{L}|$. We describe a state function State as follows.

1. *Combined function.* We assign $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, r) = 1$ if and only if $\Delta(f', \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\mathtt{max}}]) \leq \delta$.
2. *Out-of-domain samples.* We assign $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, (r, f, (x_j^{\mathtt{out}})_{j \in [s]})) = 1$ if and only if at least one of the following holds:
   (a) $\Delta(f', \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\mathtt{max}}]) \leq \delta$.
   (b) There exist distinct codewords $u, u' \in \mathsf{List}(f, d_{\mathtt{max}}, \gamma)$ with $\hat{u}(x_i^{\mathtt{out}}) = \hat{u}'(x_i^{\mathtt{out}})$ for all $i \in [s]$.
3. *In-domain queries.* We assign $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, (r, f, (x_j^{\mathtt{out}})_{j \in [s]}, (y_j)_{j \in [s]}, (x_j^{\mathtt{in}})_{j \in [t]})) = 1$ if and only if there exists a codeword $u \in \mathsf{List}(f, d_{\mathtt{max}}, \gamma)$ which agrees with Ans over S.

We analyze the round-by-round soundness errors of State as follows.

1. Suppose that $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \varnothing) = 0$ and $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}) \notin \mathcal{L}(\mathcal{R})$. Since the instance is not in the language, there exists $i \in [m]$ such that $\Delta(\mathfrak{c}_i(\mathbf{f}_i), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_i]) > \delta$. Applying Lemma 3.9, we find that
$$\Pr_r[\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, r) = 1] \leq \varepsilon_{\mathtt{prox}}\left(d_{\mathtt{max}}, \rho, \delta, m \cdot (d_{\mathtt{max}} + 1) - \sum_{i=1}^{m} d_i\right).$$
2. Suppose that $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, (r, f)) = 0$. Clearly, Item 2a cannot hold, so it suffices to bound the probability that Item 2b holds. Applying Lemma 3.4, we find that
$$\Pr_{x_1^{\mathtt{out}}, \ldots, x_s^{\mathtt{out}}}[\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, (r, f, (x_j^{\mathtt{out}})_{j \in [s]})) = 1] \leq \varepsilon_{\mathtt{ood}}(d_{\mathtt{max}}, \ell, s).$$
3. Suppose that $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, (r, f, (x_j^{\mathtt{out}})_{j \in [s]}, (y_j)_{j \in [s]})) = 0$. Then there exists at most one codeword $u \in \mathsf{List}(f, d_{\mathtt{max}}, \gamma)$ which agrees with Ans at the out-of-domain samples $\{x_j^{\mathtt{out}}\}_{j \in [s]}$. In order for State to transition to 1, u must exist and moreover agree with f' at the in-domain queries $\{x_j^{\mathtt{in}}\}_{j \in [t]}$. But since f' is δ-far from the code, we conclude that
$$\Pr_{x_1^{\mathtt{in}}, \ldots, x_t^{\mathtt{in}}}[\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, (r, f, (x_j^{\mathtt{out}})_{j \in [s]}, (y_j)_{j \in [s]}, (x_j^{\mathtt{in}})_{j \in [t]})) = 1] \leq (1-\delta)^t.$$

Finally, we show that State is consistent with the verifier's decision (this is necessary since the protocol ends on a prover message). In particular, let $\tau = (r, f, (x_j^{\mathtt{out}})_{j \in [s]}, (y_j)_{j \in [s]}, (x_j^{\mathtt{in}})_{j \in [t]})$ be a partial transcript containing all but the prover's final message and suppose that $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \tau) = 0$. By the definition of a state function, it must be the case that $\mathsf{State}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \tau || \mathsf{Fill}) = 0$ for any prover message Fill. Since this is a full transcript, we must show that the verifier outputs a bad instance, i.e., $\Delta(\mathsf{Quotient}(f, S, \mathsf{Ans}, \mathsf{Fill}), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\mathtt{max}} - |S|]) > \delta$. This follows from Lemma 3.6. □

6.3 Extensions and optimizations

Delaying FFTs. A computationally expensive step for the prover is computing Fill. Generally, this requires t evaluations of the polynomial, e.g., using an FFT in time $O(|\mathcal{L}| \log |\mathcal{L}|)$. Note that the rest of g can be computed by evaluating the quotient directly. This can be done in time $O(t \cdot |\mathcal{L}|)$. The FFT is particularly expensive if the alphabet of input codewords is some smaller base field $\mathbb{F}$, but the challenge space is over a larger extension field $\mathbb{F}^k$. In this case, the alphabet for g, and thus the domain of the FFT, will be $\mathbb{F}^k$. Given this, we want to delay the FFT for multiple accumulation steps. To do this, the prover can send 0 output values for Fill. Note that g, now is $\frac{t}{|\mathcal{L}|}$ far from the code. Let's refer to these t locations as holes. Whenever the verifier queries any of the holes, the prover aborts and then recommits to the actual codeword. This now requires running an FFT. We now argue that the prover only needs to abort and recommit every once in a while for reasonable parameters.
Assume there are w holes. The probability that the verifier does not query a hole on a single query is $1 - \frac{w}{|\mathcal{L}|}$ or $(1 - \frac{w}{|\mathcal{L}|})^t \approx e^{-\frac{t \cdot w}{|\mathcal{L}|}}$ for t queries. After k steps, each with t queries, there should be at most $k \cdot t$ holes. Thus, the probability that after k steps, no hole has been queried is lower bounded by $\prod_{i=1}^{k}(1 - \frac{i \cdot t}{|\mathcal{L}|})^t \approx e^{-\sum_{i=1}^{k} \frac{i \cdot t^2}{|\mathcal{L}|}} \approx e^{-\frac{k^2 \cdot t^2}{2 \cdot |\mathcal{L}|}}$. As long as $\frac{k^2 \cdot t^2}{2 \cdot |\mathcal{L}|} \leq 1/2$, i.e. $k < \frac{\sqrt{|\mathcal{L}|}}{t}$ the probability of querying even a single hole after k rounds can be upper bounded by $1 - e^{-1/2} \approx 40\%$. For $|\mathcal{L}| = 2^{26}$ and $t = 80$, this implies that with high probability, the FFT needs to be run only every 100 steps or more. To run the FFT efficiently, the prover maintains a representation of the polynomials as evaluation over $\mathcal{L}' \subset \mathbb{F} \setminus \mathcal{L}$. This ensures that there will never be holes within $\mathcal{L}'$.

Conjectured security. In Theorem 6.2, we prove that Construction 6.3 is secure for instances up to $\delta \in (0, 1 - \sqrt{\rho} - \eta)$ for a small value of η. The constraint directly influences t, the key efficiency parameter in the protocol as the query complexity $t = \frac{\lambda}{-\log(1-\delta)} \approx \frac{2 \cdot \lambda}{\log(1/\rho)}$. The constraint on δ is related to the Johnson bound (Theorem 3.3), which proves that there is only a polynomial (in $1/\eta$ and $1/\rho$) number of codewords in a $(1 - \sqrt{\rho} - \eta)$ radius of any string. Prior work [BBHR18; BGKS20; BCIKS23; ACFY24], as well as practical implementations, have taken a conjecture that even within a $(1 - \rho - \eta')$-radius of any string, there exists only a polynomial (in $1/\rho, 1/\eta$) number of codewords, and that the proximity gap lemma (Lemma 3.7) holds up to $1 - \delta - \eta'$. For small η' this results in $t \approx \frac{\lambda}{\log(1/\rho)}$, i.e saving roughly a factor of 2. The conjecture also enables reducing the field size. If we set $s = 2$ and use the Conjecture 5.6 from [ACFY24], we can set the field size $|\mathbb{F}| \geq 2^{\lambda} \cdot \frac{(m-1) \cdot |\mathcal{L}|}{(\eta')}$, and achieve round-by-round soundness $2^{-\lambda}$.

7 Accumulation for NP

We construct an accumulation scheme for R1CS relations.

$$\mathcal{R}_{\texttt{R1CS}}(\mathbb{F}) := \left\{ \begin{array}{l} ((A, B, C \in \mathbb{F}^{M \times N}, M, N), \\ (x \in \mathbb{F}^n, w \in \mathbb{F}^{N-n}), \\ \perp, \perp) \end{array} : \begin{array}{c} \text{For } z := (x, w) \in \mathbb{F}^N, \\ Az \circ Bz = Cz \end{array} \right\}$$

Consider the polynomial over a field $\mathbb{F}$:

$$\hat{P}(Y_1, \ldots, Y_m, Z_1, \ldots, Z_N) := \sum_{i=1}^{M} \mathsf{eq}((i-1), Y_1, \ldots, Y_m) \cdot (a_i^T \vec{Z} \cdot b_i^T \vec{Z} - c_i^T \vec{Z}),$$

for multi-linear $\mathsf{eq}(i, Y_1, \ldots, Y_m) = \begin{cases} 1 & \vec{Y} \in \{0,1\}^m \wedge i = \sum_{i=1}^m 2^{i-1} \cdot Y_i \\ 0 & \vec{Y} \in \{0,1\}^m \wedge i \neq \sum_{i=1}^m 2^{i-1} \cdot Y_i \end{cases}$, such that

$$\hat{P}(Y_1, \ldots, Y_m, \vec{Z}) = 0 \in \mathbb{F}$$

if and only if for $\vec{Z} = x || w$ $((A, B, C, M, N), x, w) \in \mathcal{R}_{\texttt{R1CS}}(\mathbb{F})$. Let $d_P = \log m + 2$ be the total degree of $\hat{P}$.

We first construct a reduction from $\mathcal{R}_{\texttt{R1CS}}(\mathbb{F})$ to an accumulator relation which we define below. The accumulator consists of two oracle strings and constraints on these oracle strings. The first oracle string is equivalent to the oracle in the proximity claim accumulator. The second oracle string, in the honest case, corresponds to the accumulated R1CS witness. We, then, show how to reduce multiple instances of this accumulator relation into one.

Accumulator relation. The accumulator relation is defined as follows. The index $\mathbb{i}$ consists of the following:

- Desciption of a finite field $\mathbb{F}$ and evaluation domain $\mathcal{L} \subset \mathbb{F}$.
- Maximum degree parameter $d_{\texttt{max}} \in \mathbb{N}, d_{\texttt{max}} < |\mathcal{L}|$. Define $\rho := \frac{d_{\texttt{max}}+1}{|\mathcal{L}|}$ to be the rate of the corresponding Reed-Solomon code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\texttt{max}}]$.
- Distance parameter $\delta \in (0, 1)$.
- A second distance parameter $\gamma \in (0, 1)$.

The instance consists of two rational constraints of degree $d_f = d_{\texttt{max}} - t - 2$ and $d_g = d_{\texttt{max}} - t - 1$, as well as $v \in \mathbb{F}^k$ and error term $e \in \mathbb{F}$, $\mathbb{x} = (v, e, \mathfrak{c}_f, \mathfrak{c}_g)$. There are two instance oracle strings $f, g \in \mathbb{F}^{|\mathcal{L}|}$. There is no witness. We formally define the promise relation $\mathcal{R}_{\texttt{ACC}}$ below.

$$\mathcal{R}_{\texttt{ACC}}(\mathbb{p}) := \left\{ (\mathbb{i}, (v, e, \mathfrak{c}_f, \mathfrak{c}_g), (f, g), \perp) : \begin{array}{c} f \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\texttt{max}}] \\ \mathfrak{c}_f(f) \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_f] \\ \mathfrak{c}_g(g) \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_g] \\ \hat{P}(v || \vec{f}) = e \end{array} \right\}$$

$$\tilde{\mathcal{R}}_{\texttt{ACC}}(\mathbb{p}) := \left\{ (\mathbb{i}, (v, e, \mathfrak{c}_f, \mathfrak{c}_g), (f, g), \perp) : \begin{array}{c} \Delta(\mathfrak{c}_g(g), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_g]) \leq \delta \\ \exists u \in \mathsf{List}(f, d_{\texttt{max}}, \gamma), \hat{P}(v || u) = e \\ \wedge \\ \mathfrak{c}_f(u) \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\texttt{max}} - t - 2] \end{array} \right\}$$

7.1 Reduction from $\mathcal{R}_{\texttt{R1CS}}$ to $\mathcal{R}_{\texttt{ACC}}$

Theorem 7.1 (knowledge soundness of accumulation)**.** *Consider a security parameter $\lambda \in \mathbb{N}$. Assume the index and instance are such that the following holds:*

- $|\mathbb{F}| \geq 2^{\lambda} \cdot \frac{10}{\rho} \cdot m.$
- $\delta \in \big(0, 1 - 1.05 \cdot \sqrt{\rho}\big).$
- $\gamma = \delta$

Then Construction 7.2 is an interactive oracle reduction (parameterized by λ with round-by-round knowledge error $2^{-\lambda}$ and the following efficiency measures:

- *Alphabet:* $\mathbb{F}$.
- *Round complexity:* 1
- *Proof length:* $|\mathcal{L}|$
- *Query complexity:* 0

Proof. Set $\eta \geq \sqrt{\rho}/20$, the Johnson bound (Theorem 3.3), gives us that $\ell \leq 1/(2\eta\sqrt{\rho})$, i.e. $\ell < \frac{10}{\rho}$. This implies that $\kappa_{\texttt{rbr}} \leq \ell \cdot m/|\mathbb{F}| \leq 2^{-\lambda}$ □

Construction 7.2.

Interaction phase.

1. $\mathbf{P}$ sends $f : \mathcal{L} \to \mathbb{F}$. In the honest case, $\hat{f} \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\texttt{max}}]$ is the encoding of $w \in \mathbb{F}^{N-n}$.
2. $\mathbf{V}$ sends $r_1, \ldots, r_m \leftarrow \mathbb{F}$.

Output phase.

1. Define the vector $\vec{v} := (r_1, \ldots, r_m, x_1, \ldots, x_n)$.
2. $\mathbf{V}$ outputs the new instance $(v, 0, \bot, \bot)$ and new instance string $(f, [0])$.

7.1.1 Soundness analysis

Lemma 7.3. *Construction 7.2 is complete and is round-by-round knowledge sound with knowledge error* $\kappa_{\texttt{rbr}} = \frac{\ell \cdot m}{\mathbb{F}}$

Proof. Completeness is immediate. Given any transcript the extractor $\mathbf{E}$ decodes the first message f to $\mathsf{List}(f, d_{\texttt{max}}, \gamma)$ and finds a $u \in \mathsf{List}(f, d_{\texttt{max}}, \gamma)$ such that $\hat{P}(Y_1, \ldots, Y_m, \vec{z}) = 0$ for $\vec{z} = \vec{x}||u$. If such a u exits, $\mathbf{E}$ outputs u.

We analyze the round-by-round knowledge error of State as follows. Suppose that $\mathsf{State}(\mathbb{p}, \mathbb{i}, \vec{x}, f) = 0$. Then, for all codewords u in $\mathsf{List}(f, d_{\texttt{max}}, \gamma)$, $\hat{P}(Y_1, \ldots, Y_m, \vec{x}, u) \neq 0$. Since $\hat{P}$ is a multi-linear m-variate non-zero polynomial, we find that by the Schwartz-Zippel lemma $\Pr_{\vec{r}}[\hat{P}(\vec{r}, \vec{x}||\vec{u})] \leq \frac{m}{|\mathbb{F}|}$, and thus taking a union bound over all ℓ polynomials in $\mathsf{List}(f, d_{\texttt{max}}, \gamma)$ we find that,

$$\Pr_{\vec{r} \leftarrow \mathbb{F}^m}[\mathsf{State}(\mathbb{p}, \mathbb{i}, \vec{x}, (f, \vec{r}))] \leq \frac{\ell \cdot m}{|\mathbb{F}|}$$

If the probability is greater than this, then $\hat{P}(\vec{r}, \vec{x}||\vec{u})$ must be a zero-polynomial for some u in the list decoding radius and $\mathbf{E}$ outputs a valid witness to $(\mathbb{i}, \vec{x}) \in \mathcal{L}(\mathcal{R}_{\texttt{R1CS}})$. □

7.2 Reduction from $\mathcal{R}^*_{\mathtt{ACC}}$ to $\mathcal{R}_{\mathtt{ACC}}$

Theorem 7.4 (Security of accumulation). *Consider a security parameter $\lambda \in \mathbb{N}$. Assume the index and instance are such that the following holds:*

- $|\mathbb{F}| \geq 2^{\lambda} \cdot 10^7 \cdot m \cdot d_P \cdot {d_{\mathtt{max}}}^2 \cdot 3t \cdot \rho^{-3.5}$.
- $\delta \in \left(0, 1 - 1.05 \cdot \sqrt{\rho} - \frac{\lambda}{-\log(1-\delta)\cdot|\mathcal{L}|}\right)$.
- $\gamma = \delta + \frac{\lambda}{-\log(1-\delta)\cdot|\mathcal{L}|}$

Then Construction 7.5, when instantiated with $s = 1$ and $t = \frac{\lambda}{-\log(1-\delta)}$, is an interactive oracle reduction (parameterized by λ) with round-by-round soundness error $2^{-\lambda}$ and the following efficiency measures:

- *Alphabet:* $\mathbb{F}$.
- *Round complexity:* 5
- *Query complexity:* $6 \leq t \leq \frac{\lambda}{-\log(1-\delta)}$.
- *Proof length:* $2 \cdot (|\mathcal{L}| + t + m + 1)$ *field elements.*

Proof. **Soundness.** We analyze the soundness error given the defined size of $\mathbb{F}$:

- $\varepsilon_{\mathtt{prox}}(d_{\mathtt{max}}, \rho, \delta, m \cdot (2t+6)) = \frac{m \cdot 3 \cdot t \cdot {d_{\mathtt{max}}}^2}{|\mathbb{F}|(\frac{\sqrt{\rho}}{10})^7} \leq 2^{-\lambda}$ (Plugging in the value for $\mathbb{F}$).
- $\varepsilon_{\mathtt{ood}} = m \cdot \frac{\ell^2}{2} \cdot (\frac{d_{\mathtt{max}}}{|\mathbb{F}|-|\mathcal{L}|})$.
- Assuming that $d_{\mathtt{max}} < |\mathcal{L}| \leq |\mathbb{F}|/2$, we get $\varepsilon_{\mathtt{ood}} \leq m \cdot \frac{100}{\rho^2} \cdot \frac{d_{\mathtt{max}}}{\mathbb{F}}$.
- Since $|\mathbb{F}| \geq 2^{\lambda} \cdot m \cdot 100 \cdot d_{\mathtt{max}} \cdot \rho^{-2}$ we have that $\varepsilon_{\mathtt{ood}} \leq 2^{-\lambda}$
- $\varepsilon_{\mathtt{acc}} = \frac{(m-1)\cdot d_P}{\mathbb{F}} \leq 2^{-\lambda}$ for $|\mathbb{F}| \geq m \cdot d_P \cdot 2^{\lambda}$.
- $(1-\delta)^t \leq (1-\delta)^{\frac{\lambda}{-\log(1-\delta)}} = 2^{-\lambda}$.

Since the round-by-round-soundness error is the maximum of these errors, we have that, for the chosen parameters, it is bounded above by $2^{-\lambda}$.

Efficiency. The protocol has 5 rounds, 6 prover and 5 verifier messages. The number of oracle queries is $t = \frac{\lambda}{-\log(1-\delta)}$ to f and g which can be send as one interleaved codeword over $\mathbb{F}^2$. The total proof length consists of two oracles of length $|\mathcal{L}|$, each as well as $2m + 2$ out of domain responses, along with two Fills of size $t = \frac{\lambda}{-\log(1-\delta)}$. □

Construction 7.5. We describe an interactive oracle reduction from $\mathcal{R}^*_{\mathtt{ACC}}$ to $\mathcal{R}_{\mathtt{ACC}}$. On input index $\mathbb{i}$, the indexer $\mathbf{I}$ outputs short index $\iota := \mathbb{i}$; there is no index oracle string. The prover and verifier are additionally parameterized by $\mathbb{p} = (s, t)$, where $s, t \in \mathbb{N}$ are the out-of-domain and in-domain repetition parameters, respectively. On input instance $\mathbb{x} = (v, e, \mathfrak{c}_{f,i}, \mathfrak{c}_{g,i})_{i\in[m]}$ and instance oracles $\vec{\mathbb{y}} = (f_i, g_i)_{i\in[m]}$, the prover $\mathbf{P}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}})$ and verifier $\mathbf{V}^{\vec{\mathbb{y}}}(\mathbb{p}, \mathbb{i}, \mathbb{x})$ engage in the following protocol.

Interaction phase. Let $H \subset \mathbb{F}$ be an arbitrary sized m subset of $\mathbb{F}$, and $\hat{L}_i(X) \forall i \in [m]$, the corresponding Lagrange polynomials, and $\hat{V}_H(X)$, the vanishing polynomial on H. Define $d_f = d_{\mathtt{max}} - t - 2$ and $d_g = d_{\mathtt{max}} - t - 1$.

1. $\mathbf{P}$ sends $\hat{q}(X) \in \mathbb{F}[X]$, a degree $d_P \cdot (m-1) - m$ polynomial.
   In the honest case $\hat{q}(X) \cdot \hat{V}_H(X) + \sum_{i=1}^{m} \hat{L}_i(X) \cdot e_i = \hat{P}(\sum_{i=1}^{m} \hat{L}_i(X) \cdot (v_i || \hat{f}_i))$
2. $\mathbf{V}$ sends $x^{(1)} \leftarrow (\mathbb{F} \setminus \mathcal{L})^s$
3. $\mathbf{P}$ sends $y_i^{(1)} \in \mathbb{F}^s$ for all $i \in [m]$. In the honest case $y_i^{(1)} := \hat{f}_i(x^{(1)})$. [a]
4. $\mathbf{V}$ sends $\alpha \in \mathbb{F}$ and $x^{(2)} \in (\mathbb{F} \setminus \mathcal{L})^s$

5. $\mathbf{P}$ sends $y_i^{(2)} \in \mathbb{F}^s$ for all $i \in [m]$. In the honest case $y_i^{(2)} := \hat{f}_i(x^{(2)})$.
6. For all $i \in [m]$, let $\mathfrak{c}_{f'_i} := \mathsf{Quotient}(\cdot, \{x^{(1)}, x^{(2)}\}, \mathsf{Ans}'_i, \perp)$
   for $\mathsf{Ans}'_i(x^{(1)}) = y_i^{(1)} \wedge \mathsf{Ans}'_i(x^{(2)}) = y_i^{(2)}$.
7. $\mathbf{V}$ sends $r \leftarrow \mathbb{F}$.
8. $\mathbf{P}$ sends $f, g : \mathcal{L} \to \mathbb{F}$.
   In the honest case, $\hat{f} := \sum_{i=1}^{m} \hat{L}_i(\alpha) \cdot \hat{f}_i$ and
   $\hat{g} := \mathsf{Combine}(d_{\mathtt{max}}, r, ((\mathfrak{c}_{f,i}(\hat{f}_i), d_f), (\mathfrak{c}'_{f,i}(\hat{f}_i), d_{\mathtt{max}} - 2), (\mathfrak{c}_{g,i}(\hat{g}_i), d_g))_{i \in [m]})$
9. $\mathbf{V}$ sends $x^{(3)} \in \mathbb{F}^s$
10. $\mathbf{P}$ sends $y^{((3),g)}, y^{((3),f)} \in \mathbb{F}^s$. In the honest case $y_i^{((3),g)} := \hat{g}(x_i^{((3),g)}) \forall i \in [s]$ and $y_i^{((3),f)} := \hat{f}(x_i^{((3),f)})$.
11. $\mathbf{V}$ sends $x^{\mathtt{in}} \leftarrow \mathbb{F}^t$.
12. $\mathbf{P}$ sends $\mathsf{Fill}_f : S_f \to \mathbb{F}$, where $S_f := \{x^{\mathtt{in}}, x^{(2)}, x^{(3)}\}$.
    In the honest case, $\mathsf{Fill}_f := \mathsf{PolyQuotient}(\hat{f}, S_f)|_{x^{\mathtt{in}}}$.
13. $\mathbf{P}$ sends $\mathsf{Fill}_g : S_g \to \mathbb{F}$, where $S_g := \{x^{\mathtt{in}}, x^{(3)}\}$.
    In the honest case, $\mathsf{Fill}_g := \mathsf{PolyQuotient}(\hat{g}, S_g)|_{x^{\mathtt{in}}}$.

Output phase.

1. Define $v := \sum_{i=1}^{m} \hat{L}_i(\alpha) \cdot v_i$
2. Define $e := \hat{V}_H(\alpha) \cdot \hat{q}(\alpha) + \sum_{i=1}^{m} \hat{L}_i(\alpha) \cdot e_i$.
3. Define the virtual functions
   - $g' := \mathsf{Combine}(d_{\mathtt{max}}, r, ((\mathfrak{c}_{f,i}(\hat{f}_i), d_f), (\mathfrak{c}'_{f,i}(\hat{f}_i), d_{\mathtt{max}} - 2), (\mathfrak{c}_{g,i}(\hat{g}_i), d_g))_{i \in [m]})$
   - $f' := \sum_{i=1}^{m} \hat{L}_i(\alpha) \cdot f_i$

   .
4. Define $\mathsf{Ans}_f : S_f \to \mathbb{F}$ such that
   - $\mathsf{Ans}_f(x^{(2)}) := \sum_{i=1}^{m} \hat{L}_i(\alpha) \cdot y_i^{(2)}$
   - $\mathsf{Ans}_f(x^{((3),f)}) := y^{((3),f)}$
   - $\mathsf{Ans}_f(x^{\mathtt{in}}) := f'(x^{\mathtt{in}})$
5. Define the rational constraint $\mathfrak{c}_f := \mathsf{Quotient}(\cdot, S_f, \mathsf{Ans}_f, \mathsf{Fill}_f)$.
6. Define $\mathsf{Ans}_g : S_g \to \mathbb{F}$ such that $\mathsf{Ans}_g(x^{(3)}) := y^{((3),g)}$ and $\mathsf{Ans}_g(x^{\mathtt{in}}) := g'(x^{\mathtt{in}})$.
7. Define the rational constraint $\mathfrak{c}_g := \mathsf{Quotient}(\cdot, S_g, \mathsf{Ans}_g, \mathsf{Fill}_g)$.
8. $\mathbf{V}$ outputs new instance $(v, e, \mathfrak{c}_f, \mathfrak{c}_g)$ and new instance strings (f, g).

[a] We slightly abuse notation and write $f(\vec{x}) = \vec{y}$ for $\vec{x}, \vec{y} \in \mathbb{F}^n$ to denote $f(x_i) = y_i \forall i \in [n]$

7.3 Completeness and soundness of Construction 7.5

Lemma 7.6. *Construction 7.5 is complete.*

Proof. Since $f_i, \mathfrak{c}_{f,i}(f_i)$ and $\mathfrak{c}_{g,i}(g_i)$ are codewords, and all in-domain and out-of-domain query responses correspond to these codewords, we have that both g and f, the outputs of Combine and a random linear combination are codewords as well. Further we have that $\hat{P}(v_i, f_i) = e_i$. $\hat{P}$ is a degree d_P polynomial. Therefore, $\hat{P}(\sum_{i=1}^{m} \hat{L}_i(X)(v_i, f_i)) - \sum_{i=1}^{m} \hat{L}_i(X) e_i$ is 0 on all of H. Therefore, there exists a degree $m \cdot (d_P - 1) - m$ polynomial $\hat{q}(X)$ such that $\hat{P}(\sum_{i=1}^{m} \hat{L}_i(X)(v_i, f_i)) - \sum_{i=1}^{m} \hat{L}_i(X) e_i = \hat{q}(X) \cdot \hat{V}_H(X)$. This implies that $\hat{P}(v, f) = e$. □

Lemma 7.7. *Construction 7.5 has a round-by-round soundness error*

$$\kappa_{\texttt{rbr}} = \max\left(\varepsilon_{\texttt{prox}}(d_{\texttt{max}}, \rho, \delta, m \cdot (2t+6), m \cdot \frac{\ell^2}{2} \cdot \left(\frac{d_{\texttt{max}}}{|\mathbb{F}| - \mathcal{L}}\right)^s, \frac{(m-1) \cdot d_P}{\mathbb{F}}, (1-\delta)^t\right) \quad .$$

Proof. We denote by τ_i the index, the instance and the transcript up to the i-th verifier message, so that $\tau_0 = (\mathbb{p}, \mathbb{i}, \vec{\mathbb{x}})$. We define the following doom sets. D_i is the doom set after the i-th verifier message. We will show that $\tau_0 \in D_0$ implies that there exists an i such that $(\mathbb{i}, \mathbb{x}_i) \notin \mathcal{L}(\mathcal{R}_{\texttt{ACC}})$ with all but negligble probability for all sets. Concretely, we show that the probability $\Pr_{c_j}[\tau_{j-1} \in D_{j-1} \wedge \tau_j = (\tau_{j-1}||c_j) \notin D_j] \leq \kappa^{(j)}_{\texttt{rbr}}$ for round-by-round errors $\kappa^{(1)}_{\texttt{rbr}}, \ldots, \kappa^{(6)}_{\texttt{rbr}}$. The overall round-by-round error will be the maximum of these per-round errors. Let $\gamma = \delta + t/|\mathcal{L}|$, $d_f = d_{\texttt{max}} - t - 2$ and $d_g = d_{\texttt{max}} - t - 1$. Then the following holds:

- $D_0 = D^f_0 \cup D^g_0$ for
  - $D^f_0 = \{\exists i \text{ s.t. } \forall u \in \mathsf{List}(f_i, d_{\texttt{max}}, \gamma) : \mathfrak{c}_{f,i}(u) \notin \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_f] \vee \hat{P}(v_i||u) \neq e_i\}$
  - $D^g_0 = \{\exists i \text{ s.t. } \forall u \in \mathsf{List}(g_i, d_{\texttt{max}}, \gamma) : \mathfrak{c}_{g,i}(u) \notin \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_g]\}$
- $D_1 = D^f_1 \cup D^g_1$ for
  - $D^f_1 = D^f_0 \cap \{\forall i, \forall (u, u') \in \mathsf{List}(f_i, d_{\texttt{max}}, \gamma) : u(x^{(1)}) \neq u'(x^{(1)}) \vee u = u'\}$
  - $D^g_1 = D^g_0$

Claim 7.8 (D_0 to D_1)**.** Let $\Pr_{x^{(1)}}[\tau_0 \in D_0 \wedge (\tau_0, x^{(1)}) \notin D_1] \leq \kappa^{(1)}_{\texttt{rbr}}$. If $\tau_0 \in D_0$ and $(\tau, y^{(1)}) \notin D_1$, then there exists an i such that two polynomials in the list decoding radius of f_i are equal at $x^{(1)}$. The probability that two degree $d_{\texttt{max}}$ polynomials are equal at s points in $\mathbb{F} \setminus \mathcal{L}$ is bounded by $(\frac{d_{\texttt{max}}}{|\mathbb{F}|-|\mathcal{L}|})^s$. A union bound over all $\binom{\ell}{2}$ points in $\mathsf{List}(f_i, d_{\texttt{max}}, \gamma)$ and over i yields that $\kappa^{(1)}_{\texttt{rbr}} = m \cdot \frac{\ell^2}{2} \cdot \left(\frac{d_{\texttt{max}}}{|\mathbb{F}|-|\mathcal{L}|}\right)^s$

- $D_2 = D^{f,\texttt{far}}_2 \cup D^{f,\mathsf{ACC}}_2 \cup D^g_2$ for $D^g_2 = D^g_1$ and
  - $D^{f,\texttt{far}}_2 = \{\exists i \; \forall u \in \mathsf{List}(f_i, d_{\texttt{max}}, \gamma) : \mathfrak{c}_{f,i}(u) \notin \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_f] \vee u(x^{(1)}) \neq y_i\}$
  - $D^{f,\mathsf{ACC}}_2 = \left\{ \begin{array}{c} \hat{P}(v|| \sum_{i=1}^m \hat{L}_i(\alpha) \cdot u_i) \neq \sum_{i=1}^m \hat{L}_i(\alpha) \cdot e_i + \hat{q}(\alpha) \cdot \hat{V}(\alpha) \\ \text{for unique } u_i \in \mathsf{List}(f_i, d_{\texttt{max}}, \gamma) : \mathfrak{c}_{f,i}(u_i) \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_f] \wedge u_i(x^{(1)}) = y^{(1)}_i \end{array} \right\}$

Claim 7.9 (D_1 to D_2)**.** Let $\Pr_\alpha[\tau_1 \in D_1 \wedge (\tau_1, \alpha) \notin D_2] \leq \kappa^{(2)}_{\texttt{rbr}}$, and let $u_i \in \mathsf{List}(f_i, d_{\texttt{max}}, \gamma)$ be the unique polynomial such that $\mathfrak{c}_{f,i}(u_i) \in \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\texttt{max}} - |\mathfrak{c}_{f,i}(u_i)]$, and $u_i(x^{(1)}) = y_i$. If such a u_i does not exist then $(\tau_1, \alpha) \in D^f_2$, if any two u_i agree on $x^{(1)}$ then $\tau_1 \notin D^f_1$ so $\tau_1 \in D^g_1 = D^g_2$. Let $u' := \sum_{i=1}^m \hat{L}_i(\alpha) u_i$. Note that by assumption $\exists i$, s.t. $\hat{P}(v_i||u_i) \neq e_i$. The probability over α that $\hat{P}(v||u') = \hat{q}(\alpha) \cdot \hat{V}(\alpha) + \sum_{i=1}^m \hat{L}_i(\alpha) \cdot e_i$, is equivalent to two $d_P \cdot (m-1)$ polynomials agreeing on a random point. This probability is bounded by $\kappa^{(2)}_{\texttt{rbr}} = \frac{d_P \cdot (m-1)}{\mathbb{F}}$

- $D_3 = D^{f,\mathsf{ACC}}_3 \cup D^{f,\texttt{far}}_3 \cup D^g_3$ for $D^g_3 = D^g_2$, $D^{f,\texttt{far}}_3 = D^{f,\texttt{far}}_2$
  - $D^{f,\mathsf{ACC}}_3 = \{\forall u \in \mathsf{List}(f', d_{\texttt{max}}, \gamma) : u(x^{(2)}) \neq \sum_{i=1}^m \hat{L}_i(\alpha) \cdot y^{(2)}_i \vee \hat{P}(v||u) \neq e\}$

Claim 7.10 (D_2 to D_3)**.** Let $\Pr_{x^{(2)}}[\tau_2 \in D_2 \wedge (\tau_2, x^2) \notin D_3] \leq \kappa^{(3)}_{\mathtt{rbr}}$. Either there exists an i such that all polynomials in the list decoding radius of f_i do not satisfy the constraints $\mathfrak{c}_{f,i}$ or $\mathfrak{c}'_{f,i}$ or we have for all u one of the following two cases:

$\boldsymbol{u = u'}$**.** In this case $P(u) = P(u') \neq e$

$\boldsymbol{u \neq u'}$**.** The probability that u and u' agree on $x^{(2)}$ is bounded by $(\frac{d_{\mathtt{max}}}{|\mathbb{F}|-\mathcal{L}})^s$

Taking a union bound over all ℓ polynomials in $\mathsf{List}[f', d_{\mathtt{max}}, \gamma]$ we get $\kappa^{(3)}_{\mathtt{rbr}} = \ell \cdot (\frac{d_{\mathtt{max}}}{|\mathbb{F}|-\mathcal{L}})^s$

- $D_4 = D_4^f \cup D_4^g$ for
  - $D_4^f = D_3^{f,\mathsf{ACC}}$
  - $D_4^g = \{\Delta(g', \mathsf{RS}[\mathbb{F}, \mathcal{L}, d_{\mathtt{max}}]) > \delta\}$

Claim 7.11 (D_3 to D_4)**.** $\Pr_r[\tau_3 \in D_3 \wedge (\tau_3, r) \notin D_4] \leq \kappa^{(4)}_{\mathtt{rbr}}$. If $\tau_3 \in D_3^{f,\mathtt{far}}$ or $\tau_3 \in D_3^g$ then for some $i \in [m]$ one of the constraints $\mathfrak{c}_{f,i}$, $\mathfrak{c}'_{f,i}$ or $\mathfrak{c}_{g_i}$ won't be satisfied, i.e. the quotiented codeword is far from the code. We can ergo bound the probability that the resulting combined g' is close to the code, using the proximity-gap Lemma 3.9. For this recall that $d_f = d_{\mathtt{max}} - t - 2$ and $d_g = d_{\mathtt{max}} - t - 1$. Since $3m(d_{\mathtt{max}} + 1) - \sum_{i=1}^{m}(d_f + d_g + d_{\mathtt{max}} - 2) = m \cdot (2t + 6)$ by $\kappa^{(4)}_{\mathtt{rbr}} = \varepsilon_{\mathtt{prox}}(d_{\mathtt{max}}, \rho, \delta, m \cdot (2t + 6))$

- $D_5 = D_5^f \cup D_5^g$ for
  - $D_5^f = D_4^f \cap \{\forall (u, u') \in \mathsf{List}(f, d_{\mathtt{max}}, \gamma) : u(x^{(3)}) \neq u'(x^{(3)}) \vee u = u'\}$
  - $D_5^g = D_4^g \cap \{\forall (u, u') \in \mathsf{List}(g, d_{\mathtt{max}}, \gamma) : u(x^{(3)}) \neq u'(x^{(3)}) \vee u = u'\}$

Claim 7.12 (From D_4 to D_5)**.** $\Pr_{x^{(3)}}[\tau_4 \in D_4 \wedge (\tau_4, x^{(3)}) \notin D_5] \leq \kappa^{(5)}_{\mathtt{rbr}}$ We compute the probability over $y^{(3)}$ that $\tau_4 \in D_4$ implies $(\tau_4, x^{(3)}) \notin D_5$. This is the case if two polynomials in the list decoding radius of g or f are equal at independently chosen s points $x^{(3)} \in \mathbb{F} \setminus \mathcal{L}$.

Assume $\boldsymbol{\tau_4 \in D_4^f}$. Using a union bound over all pairs in the list decoding radius, we get that the probability that $\tau_4, x^{(3)} \notin D_5^f$ is bounded by $\frac{\ell^2}{2} \cdot \left(\frac{d_{\mathtt{max}}}{|\mathbb{F}|-|\mathcal{L}|}\right)^s$. This is a necessary condition to escape the doomset and thus suffices as an upper bound.

Otherwise, i.e. $\boldsymbol{\tau_4 \in D_5^g}$. Using the same union bound, we get the bound on the probability that $\tau_4, x^{(3)} \notin D_5^g$.

Overall the probability is bounded by the max of the two cases, i.e. $\kappa^{(5)}_{\mathtt{rbr}} = \frac{\ell^2}{2} \cdot \left(\frac{d_{\mathtt{max}}}{|\mathbb{F}|-|\mathcal{L}|}\right)^s$

- $D_6 = D_6^f \cup D_6^g$
  - $D_6^f = \{\forall u \in \mathsf{List}(f, d_{\mathtt{max}}, \gamma) : \mathfrak{c}_f(u) \notin \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]) \vee \hat{P}(v||u) \neq e\}$
  - $D_6^g = \{\forall u \in \mathsf{List}(g, d_{\mathtt{max}}, \gamma) : \Delta(\mathfrak{c}_g(u), \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]) > \delta\}$

Claim 7.13 (From D_5 to D_6)**.** $\Pr_{x^{\mathtt{in}}}[\tau_5 \in D_5 \wedge (\tau_5, x^{\mathtt{in}}) \notin D_6] \leq \kappa^{(6)}_{\mathtt{rbr}}$.

$\boldsymbol{\tau_5 \in D_5^g}$**.** If $\tau_5 \in D_5^g$ then there exists at most one codeword $u \in \mathsf{List}(g, d_{\mathtt{max}}, \gamma)$ such that $u(x^{(3)}) = y^{((3),g)}$. Additionally $\Delta(u, g') \geq \delta$. The probability that g' and u agree on $x^{\mathtt{in}}$ is bounded by $(1 - \delta)^t$. If they do not agree than $\mathfrak{c}_g(g)$ is at least δ far from $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ by Lemma 3.6.

Else, i.e. $\boldsymbol{\tau_5 \in D_5^f}$. Let u' be the unique polynomial in $\mathsf{List}(f', d_{\mathtt{max}}, \gamma)$ such that $u'(x^{(2)}) = \sum_{i=1}^{m} \hat{L}_i(\alpha) \cdot y_i^{(2)}$. Let u be the unique polynomial in $\mathsf{List}(f, d_{\mathtt{max}}, \gamma)$ such that $u(x^{(3)}) = y^{((3),f)}$. If no such polynomial exists then $\mathfrak{c}_f(f)$ is δ-far from $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$. Otherwise, we have one of three cases:

1. $u \notin \mathsf{List}(f', d_{\mathtt{max}}, \gamma)$. In this case $\Delta(u, f) > \gamma > \delta$ and the probability that u and f' agree on $x^{\mathtt{in}}$ is bounded by $(1-\delta)^t$. If not then $\mathfrak{c}_f(f)$ is δ-far from $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$.

2. $u \in \mathsf{List}(f', d_{\mathtt{max}}, \gamma)$ and $u(x^{(2)}) \neq \sum_{i=1}^{m} \hat{L}_i(\alpha)\cdot y_i^{(2)}$. In this case $\mathfrak{c}_f(u) \notin \mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$ by Lemma 3.6, and thus $\mathfrak{c}_f(f)$ is δ-far from $\mathsf{RS}[\mathbb{F}, \mathcal{L}, d]$.

3. $u = u'$ and $u(x^{(2)}) = \sum_{i=1}^{m} \hat{L}_i(\alpha) \cdot y_i^{(2)}$. Then, by assumption $\hat{P}(v||u) \neq e$.

The error is the max of both cases, i.e., $\kappa^{(6)}_{\mathtt{rbr}} = (1-\delta)^t$.

Note that $\tau_6 \in D_6 \implies (\mathfrak{i}, (v, e, \mathfrak{c}_f, \mathfrak{c}_g)) \notin \mathcal{L}(\mathcal{R}_{\mathtt{ACC}})$.

The overall round-by-round soundness error is

$$\kappa_{\mathtt{rbr}} = \max_{i\in[6]}(\kappa^{(i)}_{\mathtt{rbr}}) = \left(\varepsilon_{\mathtt{prox}}(d_{\mathtt{max}}, \rho, \delta, m\cdot(2t+6)), m \cdot \frac{\ell^2}{2} \cdot (\frac{d_{\mathtt{max}}}{|\mathbb{F}| - \mathcal{L}})^s, \frac{(m-1)\cdot d_P}{\mathbb{F}}, (1-\delta)^t\right).$$

□

Remark 7.14 (Extension to arbitrary $\hat{P}$)**.** Our construction is written for a polynomial $\hat{P}$ that encodes R1CS, but we note that, like ProtoStar [BC23], it directly applies to more general $\hat{P}$, including ones that perform higher-degree checks like those required for encoding a customizable constraint system [STW23] or high-degree Plonk checks [CBBZ23]. This is beneficial because higher degree relations are more expressive and have smaller witnesses; indeed, recent works [Xio+23; DMS24] have shown the concrete benefit of using degree 5 to 7 constraints, and so supporting them can significantly reduce the cost of committing to a new witness. However, this is traded off with a larger $\hat{q}(X)$ (linear in the degree). The protocol does not need to change in order to accommodate different $\hat{P}$. The only difference is that $\hat{q}(X)$ is now a higher degree, $(d_P - 1)\cdot m - m$, polynomial, and that the soundness contains a $\frac{d_P \cdot m}{\mathbb{F}}$ term. As long as $\mathbb{F} > d_P \cdot m \cdot 2^\lambda$, the soundness error remains $2^{-\lambda}$.

Remark 7.15 (Number of out-of-domain samples)**.** Note that the protocol requires three rounds of out-of-domain samples. We give a brief intuitive explanation for these samples and in what scenarios the first can be omitted. The first round binds the prover to use a unique polynomial u_i within the list-decoding radius of each f_i. This is important as otherwise there could be up to ℓ^m possible combination of codewords within the list-decoding radiuses. This is used, when bounding the probability over α that $P(\sum_{i=1}^{m} \hat{L}_i(\alpha)\cdot u_i) = e$. The second out-of-domain challenge, after the choice of α, connects f' to $u' = \sum_{i=1}^{m} \hat{L}_i(\alpha) \cdot u_i$, i.e. a prover can only succeed if u' is in the list-decoding radius of f'. The final out-of-domain challenge forces the prover to use a unique polynomial in f. An astute reader will notice that in accumulation, f becomes an input f_i to the next round reduction-of-knowledge. This means that we ensure that the prover uses a unique polynomial within the list decoding radius of f, twice: Once at the beginning of the reduction and once at the end. This seems superfluous. However, in accumulation, we cannot guarantee that the accumulator was generated in a particular way, e.g., is the output of a previous accumulation step. For completely arbitrary inputs, we cannot guarantee that there is a unique polynomial corresponding to the constraint within the decoding radius of the input. In many practical applications, however, one can safely skip the first round of out-of-domain samples as long as the application checks that input accumulators are the output of an accumulation procedure. This is the case in IVC and PCD.

Remark 7.16 (Conjectured security and delaying FFTs)**.** The optimizations described in Section 6.3, equally apply to Construction 7.5. Importantly, under a commonly taken coding conjecture t, the number of queries can be reduced from roughly $t \approx \frac{2\lambda}{\log(1/\rho)}$ to $t \approx \frac{\lambda}{\log(1/\rho)}$. Additionally, it is possible to delay FFTs by not computing and sending Fill values until the code is queried at one of the holes. At that point, the prover can run an FFT and compute a codeword without holes. This should at most happpen every $\frac{\sqrt{|\mathcal{L}|}}{t}$ steps.

References

[ACFY24] G. Arnon, A. Chiesa, G. Fenzi, and E. Yogev. “STIR: Reed-Solomon Proximity Testing with Fewer Queries”. In: *Proceedings of the 44th Annual International Cryptology Conference*. CRYPTO ’24. 2024, pp. 380–413.

[BBHR18] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. “Fast Reed-Solomon Interactive Oracle Proofs of Proximity”. In: *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming*. ICALP ’18. 2018, 14:1–14:17.

[BC23] B. Bünz and B. Chen. “Protostar: Generic Efficient Accumulation/Folding for Special-Sound Protocols”. In: *Proceedings of the 29th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT ’23. 2023, pp. 77–110.

[BC24] D. Boneh and B. Chen. “LatticeFold: A Lattice-based Folding Scheme and its Applications to Succinct Proof Systems”. Cryptology ePrint Archive, Report 2024/257. 2024.

[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.

[BCG24] E. Boyle, R. Cohen, and A. Goel. “Breaking the $O(\sqrt{n})$-Bit Barrier: Byzantine Agreement with Polylog Bits Per Party”. In: *Journal of Cryptology* 37.1 (2024), p. 2.

[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.

[BCIKS23] E. Ben-Sasson, D. Carmon, Y. Ishai, S. Kopparty, and S. Saraf. “Proximity Gaps for Reed-Solomon Codes”. In: *Journal of the ACM* 70.5 (2023), 31:1–31:57.

[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.

[BCS16] E. Ben-Sasson, A. Chiesa, and N. Spooner. “Interactive Oracle Proofs”. In: *Proceedings of the 14th Theory of Cryptography Conference*. TCC ’16-B. 2016, pp. 31–60.

[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.

[BCTV17] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. “Scalable Zero Knowledge Via Cycles of Elliptic Curves”. In: *Algorithmica* 79.4 (2017), pp. 1102–1160.

[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.

[BFS20] B. Bünz, B. Fisch, and A. Szepieniec. “Transparent SNARKs from DARK Compilers”. In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’20. 2020, pp. 677–706.

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

[BGKS20] E. Ben-Sasson, L. Goldberg, S. Kopparty, and S. Saraf. “DEEP-FRI: Sampling Outside the Box Improves Soundness”. In: *Proceedings of the 11th Innovations in Theoretical Computer Science Conference*. ITCS ’20. 2020, 5:1–5:32.

[BJS23] M. Bellés-Muñoz, J. Jiménez Urroz, and J. Silva. "Revisiting Cycles of Pairing-Friendly Elliptic Curves". In: *Proceedings of the 43rd Annual International Cryptology Conference*. CRYPTO '23. 2023, pp. 3–37.

[BMNW24] B. Bünz, P. Mishra, W. Nguyen, and W. Wang. "Accumulation without Homomorphism". Cryptology ePrint Archive, Report 2024/474. 2024.

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

[CBBZ23] B. Chen, B. Bünz, D. Boneh, and Z. Zhang. "HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates". In: *Proceedings of the 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '23. 2023, pp. 499–530.

[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.

[CCW19] A. Chiesa, L. Chua, and M. Weidner. "On Cycles of Pairing-Friendly Elliptic Curves". In: *SIAM Journal on Applied Algebra and Geometry* 3.2 (2019), pp. 175–192.

[CGH04] R. Canetti, O. Goldreich, and S. Halevi. "The random oracle methodology, revisited". In: *Journal of the ACM* 51.4 (2004), pp. 557–594.

[CHMMVW20] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020.

[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.

[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.

[CY24] A. Chiesa and E. Yogev. *Building Cryptographic Proofs from Hash Functions*. 2024.

[DI14] E. Druk and Y. Ishai. "Linear-Time Encodable Codes Meeting the Gilbert-Varshamov Bound and Their Cryptographic Applications". In: *Proceedings of the 5th Innovations in Theoretical Computer Science Conference*. ITCS '14. 2014, pp. 169–182.

[DMS24] M. Dellepere, P. Mishra, and A. Shirzad. "Garuda and Pari: Faster and Smaller SNARKs via Equifficient Polynomial Commitments". Cryptology ePrint Archive, Report 2024/1245. 2024.

[DP23] B. E. Diamond and J. Posen. "Succinct Arguments over Towers of Binary Fields". Cryptology ePrint Archive, Report 2023/1784. 2023.

[EG23] L. Eagen and A. Gabizon. "ProtoGalaxy: Efficient ProtoStar-style folding of multiple instances". Cryptology ePrint Archive, Report 2023/1106. 2023.

[FKNP24] G. Fenzi, C. Knabenhans, N. K. Nguyen, and D. T. Pham. "Lova: Lattice-Based Folding Scheme from Unstructured Lattices". In: *Proceedings of the 30th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT '24. 2024.

[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.

[GKRRS21] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. “Poseidon: A New Hash Function for Zero-Knowledge Proof Systems”. In: *Proceedings of the 30th USENIX Security Symposium*. USENIX Security ’21. 2021, pp. 519–535.

[GLSTW23] A. Golovnev, J. Lee, S. T. V. Setty, J. Thaler, and R. S. Wahby. “Brakedown: Linear-Time and Field-Agnostic SNARKs for R1CS”. In: *Proceedings of the 43rd Annual International Cryptology Conference*. CRYPTO ’23. 2023, pp. 193–226.

[Gui] A. Guillevic. “Pairing-friendly curves”.

[GW19] A. Gabizon and Z. J. Williamson. “The Turbo-Plonk Program Syntax for Specifying Snark Programs”. In: ZKProof Workshop 3. 2019.

[HLP24] U. Haböck, D. Levit, and S. Papini. “Circle STARKs”. Cryptology ePrint Archive, Report 2024/278. 2024.

[KB20] A. Kattis and J. Bonneau. “Proof of Necessary Work: Succinct State Verification with Fairness Guarantees”. Cryptology ePrint Archive, Report 2020/190. 2020.

[KP23] A. Kothapalli and B. Parno. “Algebraic Reductions of Knowledge”. In: *Proceedings of the 43rd Annual International Cryptology Conference*. CRYPTO ’23. 2023, pp. 669–701.

[KPV19] A. Kattis, K. Panarin, and A. Vlasov. “RedShift: Transparent SNARKs from List Polynomial Commitments”. Cryptology ePrint Archive, Report 2019/1400. 2019.

[KS23] A. Kothapalli and S. Setty. “CycleFold: Folding-scheme-based recursive arguments over a cycle of elliptic curves”. Cryptology ePrint Archive, Report 2023/1192. Aug. 2023.

[KS24] A. Kothapalli and S. T. V. Setty. “HyperNova: Recursive Arguments for Customizable Constraint Systems”. In: *Proceedings of the 44th Annual International Cryptology Conference*. CRYPTO ’24. 2024, pp. 345–379.

[KST22] A. Kothapalli, S. T. V. 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.

[Mina] O(1) Labs. “Mina Cryptocurrency”. `minaprotocol.org`. 2020.

[NBS23] W. D. Nguyen, D. Boneh, and S. T. V. Setty. “Revisiting the Nova Proof System on a Cycle of Curves”. In: *Proceedings of the 5th Conference on Advances in Financial Technologies*. AFT ’23. 2023, 18:1–18:22.

[NDCTB24] W. D. Nguyen, T. Datta, B. Chen, N. Tyagi, and D. Boneh. “Mangrove: A Scalable Framework for Folding-Based SNARKs”. In: *Proceedings of the 44th Annual International Cryptology Conference*. CRYPTO ’24. 2024, pp. 308–344.

[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.

[Pol] Polygon Zero Team. “Plonky2: Fast Recursive Arguments with PLONK and FRI”.

[RRR16] O. Reingold, R. Rothblum, and G. Rothblum. “Constant-Round Interactive Proofs for Delegating Computation”. In: *Proceedings of the 48th ACM Symposium on the Theory of Computing*. STOC ’16. 2016, pp. 49–62.

[Spi96] D. A. Spielman. “Linear-time encodable and decodable error-correcting codes”. In: *IEEE Transactions on Information Theory* 42.6 (1996), pp. 1723–1731.

[STW23] S. Setty, J. Thaler, and R. Wahby. “Customizable constraint systems for succinct arguments”. Cryptology ePrint Archive, Report 2023/552. 2023.

[Xio+23] A. L. Xiong, B. Chen, Z. Zhang, B. Bünz, B. Fisch, F. Krell, and P. Camacho. "VeriZexe: Decentralized Private Computation with Universal Setup". In: *Proceedings of the 32nd USENIX Security Symposium*. USENIX Security '23. 2023, pp. 4445–4462.

A Proof of Theorem 4.3

Theorem 4.3. *There exists a polynomial-time transformation* T *such that the following holds. Let* $\mathcal{R}$ *be a parameterized indexed relation. Let* $\mathcal{R}^{\mathcal{U}}_{\mathtt{ACC}}$ *be a parameterized indexed promise relation in* $\mathsf{NP}^{\mathcal{U}}$ *with the same parameter space as* $\mathcal{R}$*. Suppose we are given the following non-interactive reductions in the random oracle model:*

- $\mathsf{RDX}_{\mathtt{CAST}} = (\mathcal{G}_{\mathtt{CAST}}, \mathcal{I}_{\mathtt{CAST}}, \mathcal{P}_{\mathtt{CAST}}, \mathcal{V}_{\mathtt{CAST}})$*, a reduction from* $\mathcal{R}$ *to* $\mathcal{R}_{\mathtt{ACC}}$*.*
- $\mathsf{RDX}_{\mathtt{FOLD}} = (\mathcal{G}_{\mathtt{FOLD}}, \mathcal{I}_{\mathtt{FOLD}}, \mathcal{P}_{\mathtt{FOLD}}, \mathcal{V}_{\mathtt{FOLD}})$*, a many-to-one reduction from* $\mathcal{R}^*_{\mathtt{ACC}}$ *to* $\mathcal{R}_{\mathtt{ACC}}$ *with the same generator algorithm as* $\mathsf{RDX}_{\mathtt{CAST}}$ *(i.e.,* $\mathcal{G}_{\mathtt{FOLD}} \equiv \mathcal{G}_{\mathtt{CAST}}$*).*

Then $\mathrm{T}[\mathsf{RDX}_{\mathtt{CAST}}, \mathsf{RDX}_{\mathtt{FOLD}}, \mathcal{R}_{\mathtt{ACC}}] = (\mathsf{ARG}, \mathsf{ACC})$*, where* ARG *is a non-interactive argument for* $\mathcal{R}$ *and* ACC *is an accumulation scheme for* ARG*, both in the random oracle model.*

Proof. Follows immediately from Lemma A.5 and Lemma A.7. □

A.1 Accumulation schemes

Here, we include the definition of non-interactive arguments and accumulation schemes. The construction of PCD from our follows exactly as in [BCLMS21]. However, unlike in [BCLMS21], our definitions include a relaxed version of the verifier and decider, which are used in the corresponding knowledge soundness definitions. Theorem 5.3 in [BCLMS21], the generic construction of PCD, follows almost immediately from our definitions. Essentially, it suffices to replace the verifier and decider in the knowledge soundness proof with their relaxed variants. This is similar to how recent work [BMNW24] builds proof-carrying data from bounded-depth accumulation. However, here in our setting, there is only one relaxed verifier and decider rather than up to s for some bound $s \in \mathbb{N}$; hence, the generic construction and proof follows almost immediately.

Definition A.2. A (preprocessing) **non-interactive argument** in the random oracle model is a tuple of polynomial time algorithms $\mathsf{ARG} = (\mathcal{G}, \mathcal{I}, \mathcal{P}, \mathcal{V}, \tilde{\mathcal{V}})$ that satisfy the following properties.

Completeness. ARG is complete if the following holds. For every adversary $\mathcal{A}$,

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

and

$$\Pr\left[\begin{array}{c|r} \mathcal{V}^{\rho}(\mathsf{vk}, \mathbb{x}, \pi) = 1 & \rho \leftarrow \mathcal{U}(\lambda) \\ \Downarrow & \mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}) \\ \tilde{\mathcal{V}}^{\rho}(\mathsf{vk}, \mathbb{x}, \pi) = 1 & (\mathbb{i}, \mathbb{x}, \pi) \leftarrow \mathcal{A}^{\rho}(\mathsf{pp}) \\ & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{I}^{\rho}(\mathsf{pp}, \mathbb{i}) \end{array}\right] = 1.$$

Knowledge soundness. ARG is knowledge sound (with respect to auxiliary input distribution $\mathcal{D}$) if the following holds. There exists a deterministic polynomial-time extractor $\mathcal{E}$ such that for every (non-uniform) polynomial-time adversary $\tilde{\mathcal{P}}$,

$$\Pr\left[\begin{array}{c|r} \begin{array}{c} \tilde{\mathcal{V}}^{\rho}(\mathsf{vk},\mathbb{x},\pi)=1 \\ \wedge \\ (\mathbb{i},\mathbb{x},\mathbb{w})\notin\mathcal{R}^{\rho}(\mathsf{pp}) \end{array} & \begin{array}{r} \rho\leftarrow\mathcal{U}(\lambda) \\ \mathsf{pp}\leftarrow\mathcal{G}(1^{\lambda}) \\ \mathsf{ai}\leftarrow\mathcal{D}(1^{\lambda}) \\ (\mathbb{i},\mathbb{x},\pi;\mathsf{tr})\leftarrow\tilde{\mathcal{P}}^{\rho}(\mathsf{pp},\mathsf{ai}) \\ (\mathsf{pk},\mathsf{vk})\leftarrow\mathcal{I}^{\rho}(\mathsf{pp},\mathbb{i}) \\ \mathbb{w}\leftarrow\mathcal{E}(\mathsf{pp},\mathbb{i},\mathbb{x},\pi,\mathsf{ai},\mathsf{tr}) \end{array} \end{array}\right]\le\mathrm{negl}(\lambda).$$

Definition A.3. Let $\mathsf{ARG}=(\mathcal{G},\mathcal{I},\mathcal{P},\mathcal{V},\tilde{\mathcal{V}})$ be a non-interactive argument in the random oracle model for index relation $\mathcal{R}^{\rho}$ such that the proofs have a canonical partition into pairs $\pi:=(\pi.\mathbb{x},\pi.\mathbb{w})$.

An **accumulation scheme** for ACC for ARG in the random oracle model is a tuple of polynomial-time algorithms $\mathsf{ACC}=(\mathcal{G},\mathrm{I},\mathrm{P},\mathrm{V},\mathrm{D},\tilde{\mathrm{D}})$, which shares the same generator algorithm as ARG. An accumulation scheme must satisfy the following properties.

Completeness. ACC is complete if the following holds. For every adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|r} \begin{array}{c} \forall\, i\in[n],\ \mathcal{V}^{\rho}(\mathsf{vk},\mathbb{x}_i,\pi_i)=1\,\wedge \\ \forall\, j\in[m],\ \mathrm{D}(\mathsf{dk},\mathsf{acc}_j)=1 \\ \Downarrow \\ \mathrm{V}\left(\begin{array}{c}\mathsf{avk},\left[\mathbb{x}_i,\pi_i.\mathbb{x}\right]_{i\in[n]}, \\ \left[\mathsf{acc}_i.\mathbb{x}\right]_{i\in[m]},\mathsf{acc}.\mathbb{x},\mathsf{pf}\end{array}\right)=1 \\ \wedge\,\mathrm{D}(\mathsf{dk},\mathsf{acc})=1 \end{array} & \begin{array}{r} \rho\leftarrow\mathcal{U}(\lambda) \\ \mathsf{pp}\leftarrow\mathcal{G}(1^{\lambda}) \\ \left(\mathbb{i},\ \left[\mathbb{x}_i,(\pi_i.\mathbb{x},\pi_i.\mathbb{w})\right]_{i\in[n]},\ \left[\mathsf{acc}_j\right]_{j\in[m]}\right)\leftarrow\mathcal{A}^{\rho}(\mathsf{pp}) \\ (\mathsf{pk},\mathsf{vk})\leftarrow\mathcal{I}^{\rho}(\mathsf{pp},\mathbb{i}) \\ (\mathsf{apk},\mathsf{avk},\mathsf{dk})\leftarrow\mathrm{I}^{\rho}(\mathsf{pp},\mathbb{i}) \\ (\mathsf{acc},\mathsf{pf})\leftarrow\mathrm{P}^{\rho}\left(\mathsf{apk},\left[\mathbb{x}_i,\pi_i\right]_{i\in[n]},\left[\mathsf{acc}_j\right]_{j\in[m]}\right) \end{array} \end{array}\right]=1,$$

and

$$\Pr\left[\begin{array}{c|r} \begin{array}{c} \mathrm{D}(\mathsf{dk},\mathsf{acc})=1 \\ \Downarrow \\ \tilde{\mathrm{D}}(\mathsf{dk},\mathsf{acc})=1 \end{array} & \begin{array}{r} \rho\leftarrow\mathcal{U}(\lambda) \\ \mathsf{pp}\leftarrow\mathcal{G}(1^{\lambda}) \\ (\mathbb{i},\mathsf{acc})\leftarrow\mathcal{A}^{\rho}(\mathsf{pp}) \\ (\mathsf{apk},\mathsf{avk},\mathsf{dk})\leftarrow\mathrm{I}^{\rho}(\mathsf{pp},\mathbb{i}) \end{array} \end{array}\right]=1.$$

where each accumulator has a canonical partition into pairs $\mathsf{acc}:=(\mathsf{acc}.\mathbb{x},\mathsf{acc}.\mathbb{w})$.

Knowledge soundness. ACC is knowledge sound (with respect to auxiliary input distribution $\mathcal{D}$) if there exists a deterministic polynomial-time extractor E such that for every (non-uniform) polynomial-time adversary $\tilde{\mathrm{P}}$, the following holds

$$\Pr\left[\begin{array}{c|r} \begin{array}{c} \mathrm{V}\left(\begin{array}{c}\mathsf{avk},\left[\mathbb{x}_i,\pi_i.\mathbb{x}\right]_{i\in[n]}, \\ \left[\mathsf{acc}_i.\mathbb{x}\right]_{i\in[m]},\mathsf{acc}.\mathbb{x},\mathsf{pf}\end{array}\right)=1 \\ \wedge\,\tilde{\mathrm{D}}(\mathsf{dk},\mathsf{acc})=1 \\ \wedge \\ \exists\, i\in[n],\ \tilde{\mathcal{V}}^{\rho}(\mathsf{vk},\mathbb{x}_i,\pi_i)\neq 1 \\ \vee\,\exists\, j\in[m],\ \tilde{\mathrm{D}}(\mathsf{dk},\mathsf{acc}_j)\neq 1 \end{array} & \begin{array}{r} \rho\leftarrow\mathcal{U}(\lambda) \\ \mathsf{pp}\leftarrow\mathcal{G}(1^{\lambda}) \\ \mathsf{ai}\leftarrow\mathcal{D}(1^{\lambda}) \\ \left(\begin{array}{c}\mathbb{i},\ \left[\mathbb{x}_i,\pi_i.\mathbb{x}\right]_{i\in[n]}, \\ \left[\mathsf{acc}_i.\mathbb{x}\right]_{i\in[m]},\mathsf{acc},\mathsf{pf}\end{array};\mathsf{tr}\right)\leftarrow\tilde{\mathrm{P}}^{\rho}(\mathsf{pp},\mathsf{ai}) \\ (\mathsf{pk},\mathsf{vk})\leftarrow\mathcal{I}^{\rho}(\mathsf{pp},\mathbb{i}) \\ (\mathsf{apk},\mathsf{avk},\mathsf{dk})\leftarrow\mathrm{I}^{\rho}(\mathsf{pp},\mathbb{i}) \\ \left(\left[\pi_i.\mathbb{w}\right]_{i\in[n]},\ \left[\mathsf{acc}_j.\mathbb{w}\right]_{j\in[m]}\right) \\ \leftarrow\mathrm{E}\left(\mathsf{pp},\begin{array}{c}\mathbb{i},\ \left[\mathbb{x}_i,\pi_i.\mathbb{x}\right]_{i\in[n]}, \\ \left[\mathsf{acc}_i.\mathbb{x}\right]_{i\in[m]},\mathsf{acc},\mathsf{pf}\end{array},\mathsf{ai};\mathsf{tr}\right) \end{array} \end{array}\right]\le\mathrm{negl}(\lambda).$$

A.2 Construction

ARG Construction. The non-interactive argument Construction A.4 can be viewed as a simple wrapper around the non-interactive reduction $\mathsf{RDX}_{\mathtt{CAST}}$. Effectively, the prover and verifier both execute the reduction

$\mathsf{RDX}_{\mathtt{CAST}}$ to reduce the claim about $\mathcal{R}$ to $\mathcal{R}_{\mathtt{ACC}}$. Now, the prover can simply include the output witness $\mathcal{R}_{\mathtt{ACC}}$ as a part of the argument proof. To be specific, the argument proof will simply consist of both the reduction proof $\pi_{\mathtt{CAST}}$ and the output witness $\mathsf{acc}.\mathbb{w}$ for $\mathcal{R}_{\mathtt{ACC}}$. The argument verifier simply derives the corresponding output instance $\mathsf{acc}.\mathbb{x}$ using $\pi_{\mathtt{CAST}}$ and checks the output instance-witness pair belongs to $\mathcal{R}_{\mathtt{ACC}}$.

Construction A.4. We define $\mathsf{ARG} = (\mathcal{G}, \mathcal{I}, \mathcal{P}, \mathcal{V}, \tilde{\mathcal{V}})$ as follows.

```
$\mathcal{G}(1^\lambda)$: Output $\mathsf{pp} \leftarrow \mathcal{G}_{\mathtt{CAST}}(1^\lambda)$.
```

```
$\mathcal{I}^\rho(\mathsf{pp}, \mathbb{i})$:
  1. Compute $(\mathsf{pk}_{\mathtt{CAST}}, \mathsf{vk}_{\mathtt{CAST}}, \mathbb{i}') \leftarrow \mathcal{I}^\rho_{\mathtt{CAST}}(\mathsf{pp}, \mathbb{i})$.
  2. Output $(\mathsf{pk}, \mathsf{vk}) := \big(\mathsf{pk}_{\mathtt{CAST}}, (\mathsf{vk}_{\mathtt{CAST}}, \mathbb{i}', \mathsf{pp})\big)$.
```

```
$\mathcal{P}^\rho(\mathsf{pk}, \mathbb{x}, \mathbb{w})$:
  1. Compute $(\pi_{\mathtt{CAST}}, \mathsf{acc}.\mathbb{w}) \leftarrow \mathcal{P}^\rho_{\mathtt{CAST}}(\mathsf{pk}, \mathbb{x}, \mathbb{w})$.
  2. Assign $(\pi.\mathbb{x}, \pi.\mathbb{w}) := (\pi_{\mathtt{CAST}}, \mathsf{acc}.\mathbb{w})$.
  3. Output $\pi := (\pi.\mathbb{x}, \pi.\mathbb{w})$.
```

```
$\mathcal{V}^\rho\big(\mathsf{vk} := (\mathsf{vk}_{\mathtt{CAST}}, \mathbb{i}', \mathsf{pp}),\ \mathbb{x},\ \pi := (\pi_{\mathtt{CAST}}, \mathsf{acc}.\mathbb{w})\big)$:
  1. Compute $\mathsf{acc}.\mathbb{x} \leftarrow \mathcal{V}^\rho_{\mathtt{CAST}}(\mathsf{vk}_{\mathtt{CAST}}, \mathbb{x}, \pi_{\mathtt{CAST}})$.
  2. Check that $(\mathbb{i}', \mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w}) \in \mathcal{R}^\rho_{\mathtt{ACC}}(\mathsf{pp})$.
```

```
$\tilde{\mathcal{V}}^\rho\big(\mathsf{vk} := (\mathsf{vk}_{\mathtt{CAST}}, \mathbb{i}', \mathsf{pp}),\ \mathbb{x},\ \pi := (\pi_{\mathtt{CAST}}, \mathsf{acc}.\mathbb{w})\big)$:
  1. Compute $\mathsf{acc}.\mathbb{x} \leftarrow \mathcal{V}^\rho_{\mathtt{CAST}}(\mathsf{vk}_{\mathtt{CAST}}, \mathbb{x}, \pi_{\mathtt{CAST}})$.
  2. Check that $(\mathbb{i}', \mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w}) \in \tilde{\mathcal{R}}^\rho_{\mathtt{ACC}}(\mathsf{pp})$.
```

Lemma A.5. *Construction A.4 (*ARG*) is a non-interactive argument for* $\mathcal{R}$*.*

Proof. Since ARG is a trivial wrapper around the non-interactive reduction $\mathsf{RDX}_{\mathtt{CAST}}$, the algorithms remain polynomial-time.

Completeness. Since ARG is a trivial wrapper around the non-interactive reduction $\mathsf{RDX}_{\mathtt{CAST}}$, completeness follows immediately from the completeness of $\mathsf{RDX}_{\mathtt{CAST}}$ and the fact that $\mathcal{R}^\rho_{\mathtt{ACC}}(\mathsf{pp}) \subseteq \tilde{\mathcal{R}}^\rho_{\mathtt{ACC}}(\mathsf{pp})$.

Knowledge soundness. Consider an arbitrary polynomial-time adversary $\tilde{\mathcal{P}}$. We construct an adversary $\tilde{\mathcal{P}}_{\mathtt{CAST}}$ against the non-interactive reduction $\mathsf{RDX}_{\mathtt{CAST}}$ as follows:

```
$\tilde{\mathcal{P}}^{\rho}_{\mathtt{CAST}}(\mathsf{pp}, \mathsf{ai})$:

  1. Compute $(\mathbb{i}, \mathbb{x}, \pi, \pi; \mathsf{tr}) \leftarrow \tilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai})$.
  2. Assign $(\pi_{\mathtt{CAST}}, \mathsf{acc}.\mathbb{w}) := \pi$.
  3. Output $(\mathbb{i}, \mathbb{x}, \pi_{\mathtt{CAST}}, \mathsf{acc}.\mathbb{w}; \mathsf{tr})$.
```

By the knowledge soundness of the non-interactive reduction $\mathsf{RDX}_{\mathtt{CAST}}$, there exists a corresponding extractor $\mathcal{E}_{\mathtt{CAST}}$. We construct an extractor for ARG as follows:

```
$\mathcal{E}(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \pi, \mathsf{ai}, \mathsf{tr})$:

  1. Assign $(\pi_{\mathtt{CAST}}, \mathsf{acc}.\mathbb{w}) := \pi$.
  2. Compute $\mathbb{w} \leftarrow \mathcal{E}_{\mathtt{CAST}}(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \pi_{\mathtt{CAST}}, \mathsf{acc}.\mathbb{w}, \mathsf{ai}, \mathsf{tr})$.
  3. Output $\mathbb{w}$.
```

By the construction of the relaxed argument verifier $\tilde{\mathcal{V}}^{\rho}$, knowledge soundness of RDX, and Remark 4.1, we have that

$$
\Pr\left[\begin{array}{c} \tilde{\mathcal{V}}^{\rho}(\mathsf{vk}, \mathbb{x}, \pi) = 1 \\ \wedge \\ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}^{\rho}(\mathsf{pp}) \end{array} \middle| \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}) \\ \mathsf{ai} \leftarrow \mathcal{D}(1^{\lambda}) \\ (\mathbb{i}, \mathbb{x}, \pi; \mathsf{tr}) \leftarrow \tilde{\mathcal{P}}^{\rho}(\mathsf{pp}, \mathsf{ai}) \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{I}^{\rho}(\mathsf{pp}, \mathbb{i}) \\ \mathbb{w} \leftarrow \mathcal{E}(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \pi, \mathsf{ai}, \mathsf{tr}) \end{array}\right]
$$

$$
\leq \Pr\left[\begin{array}{c} (\mathbb{i}', \mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w}) \in \tilde{\mathcal{R}}^{\rho}_{\mathsf{ACC}}(\mathsf{pp}) \\ \wedge \\ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}^{\rho}(\mathsf{pp}) \end{array} \middle| \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}) \\ \mathsf{ai} \leftarrow \mathcal{D}(1^{\lambda}) \\ (\mathbb{i}, \mathbb{x}, \pi_{\mathtt{CAST}}, \mathsf{acc}.\mathbb{w}; \mathsf{tr}) \leftarrow \tilde{\mathcal{P}}^{\rho}_{\mathtt{CAST}}(\mathsf{pp}, \mathsf{ai}) \\ (\mathsf{pk}_{\mathtt{CAST}}, \mathsf{vk}_{\mathtt{CAST}}, \mathbb{i}') \leftarrow \mathcal{I}^{\rho}_{\mathtt{CAST}}(\mathsf{pp}, \mathbb{i}) \\ \mathsf{acc}.\mathbb{x} \leftarrow \mathcal{V}^{\rho}_{\mathtt{CAST}}(\mathsf{vk}_{\mathtt{CAST}}, \mathbb{x}, \pi_{\mathtt{CAST}}) \\ \mathbb{w} \leftarrow \mathcal{E}_{\mathtt{CAST}}(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \pi_{\mathtt{CAST}}, \mathsf{acc}.\mathbb{w}, \mathsf{ai}, \mathsf{tr}) \end{array}\right] \leq \operatorname{negl}(\lambda).
$$

□

ACC Construction. The accumulation scheme (Construction A.6) for ARG (Construction A.4) can similarly be viewed as a simple wrapper around the non-interactive reduction $\mathsf{RDX}_{\mathtt{FOLD}}$. The first step is to cast the argument verifier claims to claims for $\mathcal{R}_{\mathsf{ACC}}$. This is done by simply calling $\mathcal{V}^{\rho}_{\mathtt{CAST}}$, which is used in the argument verifier to generate an instance $\mathsf{acc}.\mathbb{x}$ for $\mathcal{R}_{\mathsf{ACC}}$. Now that we have $m+n$ claims for $\mathcal{R}_{\mathsf{ACC}}$, the prover and verifier both execute the reduction $\mathsf{RDX}_{\mathtt{FOLD}}$ which exactly reduces $m+n$ claims for $\mathcal{R}_{\mathsf{ACC}}$ to a single claim for $\mathcal{R}_{\mathsf{ACC}}$. The accumulation proof pf is simply the reduction proof $\pi_{\mathtt{FOLD}}$ for $\mathsf{RDX}_{\mathtt{FOLD}}$, and the output accumulator is the output instance-witness pair in $\mathcal{R}_{\mathsf{ACC}}$. The decider (relaxed decider) just check this pair belongs to $\mathcal{R}_{\mathsf{ACC}}$ ($\tilde{\mathcal{R}}_{\mathsf{ACC}}$).

Construction A.6. We define $\mathsf{ACC} = (\mathcal{G}, \mathrm{I}, \mathrm{P}, \mathrm{V}, \mathrm{D}, \tilde{\mathrm{D}})$ as follows. The generator algorithm, $\mathcal{G}$, is defined in Construction A.4.

```
$\mathrm{I}^{\rho}(\mathsf{pp}, \mathbb{i})$:
  1. Compute $(\mathsf{pk}_{\mathtt{CAST}}, \mathsf{vk}_{\mathtt{CAST}}, \mathbb{i}') \leftarrow \mathcal{I}^{\rho}_{\mathtt{CAST}}(\mathsf{pp}, \mathbb{i})$.
  2. Compute $(\mathsf{pk}_{\mathtt{FOLD}}, \mathsf{vk}_{\mathtt{FOLD}}, \mathbb{i}') \leftarrow \mathcal{I}^{\rho}_{\mathtt{FOLD}}(\mathsf{pp}, \mathbb{i}')$.
  3. Output $\mathsf{apk} := (\mathsf{vk}_{\mathtt{CAST}}, \mathsf{pk}_{\mathtt{FOLD}}, \mathsf{vk}_{\mathtt{FOLD}})$, $\mathsf{avk} := (\mathsf{vk}_{\mathtt{CAST}}, \mathsf{vk}_{\mathtt{FOLD}})$, and $\mathsf{dk} := (\mathbb{i}', \mathsf{pp})$.
```

```
$\mathrm{P}^{\rho}\big(\mathsf{apk} := (\mathsf{vk}_{\mathtt{CAST}}, \mathsf{pk}_{\mathtt{FOLD}}, \mathsf{vk}_{\mathtt{FOLD}}), (\mathbb{x}_i, \pi_i)_{i \in [n]}, (\mathsf{acc}_i)_{i \in [m]}\big)$:
  1. For each $i = 1, \ldots, n$,
     (a) Compute $\mathsf{acc}_{(m+i)}.\mathbb{x} \leftarrow \mathcal{V}^{\rho}_{\mathtt{CAST}}(\mathsf{vk}_{\mathtt{CAST}}, \mathbb{x}_i, \pi_i.\mathbb{x})$ and assign $\mathsf{acc}_{(m+i)}.\mathbb{w} := \pi.\mathbb{w}$.
  2. Compute $(\pi_{\mathtt{FOLD}}, \mathsf{acc}.\mathbb{w}) \leftarrow \mathcal{P}^{\rho}_{\mathtt{FOLD}}\big(\mathsf{pk}_{\mathtt{FOLD}}, (\mathsf{acc}_i.\mathbb{x})_{i \in [m+n]}, (\mathsf{acc}_i.\mathbb{w})_{i \in [m+n]}\big)$.
  3. Compute $\mathsf{acc}.\mathbb{x} \leftarrow \mathcal{V}^{\rho}_{\mathtt{FOLD}}(\mathsf{vk}_{\mathtt{FOLD}}, (\mathsf{acc}_i.\mathbb{x})_{i \in [m+n]}, \pi_{\mathtt{FOLD}})$.
  4. Output $\mathsf{acc} \leftarrow (\mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w})$ and $\mathsf{pf} \leftarrow \pi_{\mathtt{FOLD}}$.
```

```
$\mathrm{V}^{\rho}\big(\mathsf{avk} := (\mathsf{vk}_{\mathtt{CAST}}, \mathsf{vk}_{\mathtt{FOLD}}), (\mathbb{x}_i, \pi_i.\mathbb{x})_{i \in [n]}, (\mathsf{acc}_i.\mathbb{x})_{i \in [m]}, \mathsf{acc}.\mathbb{x}, \mathsf{pf}\big)$:
  1. For each $i = 1, \ldots, n$,
     (a) Compute $\mathsf{acc}_{(m+i)}.\mathbb{x} \leftarrow \mathcal{V}^{\rho}_{\mathtt{CAST}}(\mathsf{vk}_{\mathtt{CAST}}, \mathbb{x}_i, \pi_i.\mathbb{x})$.
  2. Check that $\mathsf{acc}.\mathbb{x} = \mathcal{V}^{\rho}_{\mathtt{FOLD}}(\mathsf{vk}_{\mathtt{FOLD}}, (\mathsf{acc}_i.\mathbb{x})_{i \in [m+n]}, \mathsf{pf})$.
```

```
$\mathrm{D}^{\rho}(\mathsf{dk} = (\mathbb{i}', \mathsf{pp}), \mathsf{acc})$:
  1. Check that $(\mathbb{i}', \mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w}) \in \mathcal{R}^{\rho}_{\mathtt{ACC}}(\mathsf{pp})$.
```

```
$\tilde{\mathrm{D}}^{\rho}(\mathsf{dk} = (\mathbb{i}', \mathsf{pp}), \mathsf{acc})$:
  1. Check that $(\mathbb{i}', \mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w}) \in \tilde{\mathcal{R}}^{\rho}_{\mathtt{ACC}}(\mathsf{pp})$.
```

Lemma A.7. *Construction A.6 (*ACC*) is an accumulation scheme for* ARG.

Proof. Since ACC is a trivial wrapper around $\mathsf{RDX}_{\mathtt{CAST}}$ and $\mathsf{RDX}_{\mathtt{ACC}}$, the algorithms remain polynomial-time.

Completeness. By construction of $\mathcal{V}^{\rho}$ and D^{ρ}, if $\forall\, i \in [n]$, $\mathcal{V}^{\rho}(\mathsf{vk}, \mathbb{x}_i, \pi_i) = 1$ and $\forall\, j \in [m]$, $\mathrm{D}(\mathsf{dk}, \mathsf{acc}_j) = 1$, then $[\mathsf{acc}_i.\mathbb{x},\ \mathsf{acc}_i.\mathbb{w}]_{i \in [m+n]} \in \mathcal{R}^{\rho}_{\mathtt{ACC}}(\mathsf{pp})$. Thus, since ACC is a trivial wrapper around $\mathsf{RDX}_{\mathtt{ACC}}$, completeness follows immediately from the completeness of $\mathsf{RDX}_{\mathtt{ACC}}$ and the fact that $\mathcal{R}^{\rho}_{\mathtt{ACC}}(\mathsf{pp}) \subseteq \tilde{\mathcal{R}}^{\rho}_{\mathtt{ACC}}(\mathsf{pp})$.

Knowledge Soundness. Consider an arbitrary polynomial-time adversary $\tilde{\mathrm{P}}$. We construct an adversary $\tilde{\mathcal{P}}_{\mathtt{FOLD}}$ against the non-interactive reduction $\mathsf{RDX}_{\mathtt{FOLD}}$ as follows:

```
$\tilde{\mathcal{P}}^{\rho}_{\texttt{FOLD}}(\mathsf{pp}, \mathsf{ai})$:

    1. Compute $\big(\mathbb{i},\ [\mathbb{x}_i, \pi_i.\mathbb{x}]_{i\in[n]}, [\mathsf{acc}_i.\mathbb{x}]_{i\in[m]}, \mathsf{acc}, \mathsf{pf}; \mathsf{tr}\big) \leftarrow \tilde{\mathrm{P}}^{\rho}(\mathsf{pp}, \mathsf{ai})$.
    2. Assign $(\mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w}) := \mathsf{acc}$ and $\pi_{\texttt{FOLD}} := \mathsf{pf}$.
    3. Compute $(\mathsf{pk}_{\texttt{CAST}}, \mathsf{vk}_{\texttt{CAST}}, \mathbb{i}') \leftarrow \mathcal{I}^{\rho}_{\texttt{CAST}}(\mathsf{pp}, \mathbb{i})$.
    4. For each $i = 1, \ldots, n$,
        (a) Compute $\mathsf{acc}_{(m+i)}.\mathbb{x} \leftarrow \mathcal{V}^{\rho}_{\texttt{CAST}}(\mathsf{vk}_{\texttt{CAST}}, \mathbb{x}_i, \pi_i.\mathbb{x})$.
    5. Output $(\mathbb{i}', [\mathsf{acc}_i.\mathbb{x}]_{i\in[m+n]}, \pi_{\texttt{FOLD}}, \mathsf{acc}.\mathbb{w}; \mathsf{tr})$.
```

By the knowledge soundness of the non-interactive reduction $\mathsf{RDX}_{\texttt{FOLD}}$, there exists a corresponding extractor $\mathcal{E}_{\texttt{FOLD}}$. We construct an extractor for ACC as follows:

```
$\mathrm{E}\big(\mathsf{pp}, \mathbb{i}',\ [\mathbb{x}_i, \pi_i.\mathbb{x}]_{i\in[n]}, [\mathsf{acc}_i.\mathbb{x}]_{i\in[m]}, \mathsf{acc}, \mathsf{pf}, \mathsf{ai}; \mathsf{tr}\big)$:

    1. Assign $(\mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w}) := \mathsf{acc}$ and $\pi_{\texttt{FOLD}} := \mathsf{pf}$.
    2. Compute $[\mathsf{acc}_i.\mathbb{w}]_{i\in[m+n]} \leftarrow \mathcal{E}_{\texttt{FOLD}}(\mathsf{pp}, \mathbb{i}', [\mathsf{acc}_i.\mathbb{x}]_{i\in[m+n]}, \pi_{\texttt{FOLD}}, \mathsf{acc}.\mathbb{w}, \mathsf{ai}, \mathsf{tr})$.
    3. Assign $[\pi_i.\mathbb{w}]_{i\in[n]} := [\mathsf{acc}_i.\mathbb{w}]^{m+n}_{i\in[m+1]}$.
    4. Output $\big([\pi_i.\mathbb{w}]_{i\in[n]}, [\mathsf{acc}_j.\mathbb{w}]_{j\in[m]}\big)$.
```

By construction of the relaxed argument verifier $\tilde{\mathcal{V}}$ and relaxed decider $\tilde{\mathrm{D}}$, construction of the accumulation verifier V and $\tilde{\mathrm{P}}_{\texttt{FOLD}}$, and by knowledge soundness of $\mathsf{RDX}_{\texttt{FOLD}}$, we have that

$$\Pr\left[\begin{array}{c} \mathrm{V}\begin{pmatrix}\mathsf{avk},\ \left[\mathbb{x}_i, \pi_i.\mathbb{x}\right]_{i\in[n]}, \\ [\mathsf{acc}_i.\mathbb{x}]_{i\in[m]}, \mathsf{acc}.\mathbb{x}, \mathsf{pf}\end{pmatrix} = 1 \\ \wedge\, \tilde{\mathrm{D}}(\mathsf{dk},\mathsf{acc}) = 1 \\ \wedge \\ \exists\, i \in [n],\ \tilde{\mathcal{V}}^\rho(\mathsf{vk}, \mathbb{x}_i, \pi_i) \neq 1 \\ \vee \exists\, j\in[m],\ \tilde{\mathrm{D}}(\mathsf{dk},\mathsf{acc}_j)\neq 1 \end{array} \middle| \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \mathsf{pp}\leftarrow\mathcal{G}(1^\lambda) \\ \mathsf{ai}\leftarrow\mathcal{D}(1^\lambda) \\ \left(\begin{array}{c}\mathbb{i},\ [\mathbb{x}_i,\pi_i.\mathbb{x}]_{i\in[n]},\\ [\mathsf{acc}_i.\mathbb{x}]_{i\in[m]},\mathsf{acc},\mathsf{pf}\end{array};\mathsf{tr}\right)\leftarrow \tilde{\mathrm{P}}^\rho(\mathsf{pp},\mathsf{ai}) \\ (\mathsf{pk},\mathsf{vk})\leftarrow\mathcal{I}^\rho(\mathsf{pp},\mathbb{i}) \\ (\mathsf{apk},\mathsf{avk},\mathsf{dk})\leftarrow \mathrm{I}^\rho(\mathsf{pp},\mathbb{i}) \\ \left(\left[\pi_i.\mathbb{w}\right]_{i\in[n]},\ \left[\mathsf{acc}_j.\mathbb{w}\right]_{j\in[m]}\right) \\ \leftarrow \mathrm{E}\left(\begin{array}{c}\mathsf{pp},\mathbb{i},\ [\mathbb{x}_i,\pi_i.\mathbb{x}]_{i\in[n]},\\ [\mathsf{acc}_i.\mathbb{x}]_{i\in[m]},\mathsf{acc},\mathsf{pf}\end{array},\mathsf{ai};\mathsf{tr}\right)\end{array}\right]$$

$$\leq \Pr\left[\begin{array}{c} \mathrm{V}\begin{pmatrix}\mathsf{avk},\ \left[\mathbb{x}_i, \pi_i.\mathbb{x}\right]_{i\in[n]}, \\ [\mathsf{acc}_i.\mathbb{x}]_{i\in[m]}, \mathsf{acc}.\mathbb{x}, \mathsf{pf}\end{pmatrix} = 1 \\ \wedge\, \tilde{\mathrm{D}}(\mathsf{dk},\mathsf{acc}) = 1 \\ \wedge \\ [(\mathbb{i}', \mathsf{acc}_i.\mathbb{x}, \mathsf{acc}_i.\mathbb{w})]_{i\in[m+n]} \notin \tilde{\mathcal{R}}^{\rho,m+n}_{\mathtt{ACC}}(\mathsf{pp}) \end{array} \middle| \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \mathsf{pp}\leftarrow\mathcal{G}(1^\lambda) \\ \mathsf{ai}\leftarrow\mathcal{D}(1^\lambda) \\ \left(\begin{array}{c}\mathbb{i},\ [\mathbb{x}_i,\pi_i.\mathbb{x}]_{i\in[n]},\\ [\mathsf{acc}_i.\mathbb{x}]_{i\in[m]},\mathsf{acc},\mathsf{pf}\end{array};\mathsf{tr}\right)\leftarrow \tilde{\mathrm{P}}^\rho(\mathsf{pp},\mathsf{ai}) \\ (\mathsf{pk},\mathsf{vk})\leftarrow\mathcal{I}^\rho(\mathsf{pp},\mathbb{i}) \\ (\mathsf{apk},\mathsf{avk},\mathsf{dk})\leftarrow \mathrm{I}^\rho(\mathsf{pp},\mathbb{i}) \\ (\mathsf{pk}_{\mathtt{CAST}},\mathsf{vk}_{\mathtt{CAST}},\mathbb{i}')\leftarrow\mathcal{I}^\rho_{\mathtt{CAST}}(\mathsf{pp},\mathbb{i}) \\ \forall\, i\in[n],\ \mathsf{acc}_{(m+i)}.\mathbb{x}\leftarrow\mathcal{V}^\rho_{\mathtt{CAST}}(\mathsf{vk}_{\mathtt{CAST}},\mathbb{x}_i,\pi_i.\mathbb{x}) \\ \left(\left[\mathsf{acc}_{(m+i)}.\mathbb{w}\right]_{i\in[n]},\ \left[\mathsf{acc}_j.\mathbb{w}\right]_{j\in[m]}\right) \\ \leftarrow \mathrm{E}\left(\begin{array}{c}\mathsf{pp},\mathbb{i},\ [\mathbb{x}_i,\pi_i.\mathbb{x}]_{i\in[n]},\\ [\mathsf{acc}_i.\mathbb{x}]_{i\in[m]},\mathsf{acc},\mathsf{pf}\end{array},\mathsf{ai};\mathsf{tr}\right)\end{array}\right]$$

$$\leq \Pr\left[\begin{array}{c} (\mathbb{i}', \mathsf{acc}.\mathbb{x}, \mathsf{acc}.\mathbb{w}) \in \tilde{\mathcal{R}}^{\rho}_{\mathtt{ACC}}(\mathsf{pp}) \\ \wedge \\ [(\mathbb{i}', \mathsf{acc}_i.\mathbb{x}, \mathsf{acc}_i.\mathbb{w})]_{i\in[m+n]} \notin \tilde{\mathcal{R}}^{\rho,m+n}_{\mathtt{ACC}}(\mathsf{pp}) \end{array} \middle| \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \mathsf{pp}\leftarrow\mathcal{G}(1^\lambda) \\ \mathsf{ai}\leftarrow\mathcal{D}(1^\lambda) \\ \left(\begin{array}{c}\mathbb{i}',[\mathsf{acc}_i.\mathbb{x}]_{i\in[m+n]},\\ \pi_{\mathtt{FOLD}},\mathsf{acc}.\mathbb{w}\end{array};\mathsf{tr}\right)\leftarrow \tilde{\mathcal{P}}^\rho_{\mathtt{FOLD}}(\mathsf{pp},\mathsf{ai}) \\ (\mathsf{pk}_{\mathtt{FOLD}},\mathsf{vk}_{\mathtt{FOLD}},\mathbb{i}'')\leftarrow\mathcal{I}^\rho_{\mathtt{FOLD}}(\mathsf{pp},\mathbb{i}') \\ \mathsf{acc}.\mathbb{x}\leftarrow\mathcal{V}^\rho_{\mathtt{FOLD}}(\mathsf{vk}_{\mathtt{FOLD}},[\mathsf{acc}_i.\mathbb{x}]_{i\in[m+n]},\pi_{\mathtt{FOLD}}) \\ [\mathsf{acc}_i.\mathbb{w}]_{i\in[m+n]}\leftarrow\mathcal{E}_{\mathtt{FOLD}}\left(\begin{array}{c}\mathsf{pp},\mathbb{i}',[\mathsf{acc}_i.\mathbb{x}]_{i\in[m+n]},\\ \pi_{\mathtt{FOLD}},\mathsf{acc}.\mathbb{w},\mathsf{ai},\mathsf{tr}\end{array}\right)\end{array}\right]$$

$$\leq \mathrm{negl}(\lambda).$$

□

B Proof of Theorem 5.9

We first recall Theorem 5.9 and give a full description of the transformation T in Construction B.2. To prove knowledge soundness, we follow the modular approach outlined in [CY24, Section 26.1.2]. In more detail, we decompose $\mathrm{T} = \mathsf{FS} \circ \mathrm{T}_{\mathtt{i}}$ into two transformations:

- $\mathrm{T}_{\mathtt{i}}$ takes an interactive oracle reduction from $\mathcal{R}$ to $\mathcal{R}'$ and returns an *interactive reduction* (in the ROM) from $\mathsf{Com}[\mathcal{R}]$ to $\mathsf{Com}[\mathcal{R}']$. In Appendix B.2 we show that if IOR is state-restoration knowledge sound, then so is the interactive reduction.

- The Fiat–Shamir transformation FS takes an interactive reduction (in the ROM) and returns a non-interactive reduction (in the ROM) between the same relations. In Appendix B.3 we show that if the interactive reduction is state-restoration knowledge sound, then the non-interactive reduction is knowledge sound.

Theorem 5.9. *There exists a polynomial-time transformation* T *such that the following holds. Let* $\mathcal{R}$ *and* $\mathcal{R}'$ *be indexed promise relations (with strings). Let* S *and* S' *be efficiently computable sets parameterized by* $\lambda \in \mathbb{N}$*. Let* IOR *be an interactive oracle reduction (also parameterized by* λ*) from* $\mathcal{R}$ *to* $\mathcal{R}'$ *such that the following holds:*

- IOR *has round-by-round knowledge error* $\kappa_{\mathtt{rbr}}$ *such that*

$$\max_{\substack{\mathbb{i},\mathbb{x} \in S(\lambda) \\ |\mathbb{i}|+|\mathbb{x}|=\mathrm{poly}(\lambda)}} \kappa_{\mathtt{rbr}}(\lambda, \mathbb{i}, \mathbb{x}) = \mathrm{negl}(\lambda).$$

- *For every parameter* $\lambda \in \mathbb{N}$ *and index* $\mathbb{i} \in S(\lambda)$*, the IOR indexer outputs a new index* $\mathbb{i}' \in S'(\lambda)$*.*

Then $\mathrm{T}[\mathsf{IOR}]$ *is a non-interactive reduction from* $\mathsf{Com}[\mathcal{R}, S]$ *to* $\mathsf{Com}[\mathcal{R}', S']$ *with the following efficiency measures:*

- *Proof size:* $O(\lambda \cdot \mathsf{k} + \mathsf{s} + \mathsf{q} \cdot (\log |\Sigma| + \lambda \cdot \log \mathsf{L}_{\mathtt{max}}))$.
- *Verifier complexity: IOR verifier, plus* $O(\mathsf{k} + \mathsf{q} \cdot \log \mathsf{L}_{\mathtt{max}})$ *queries to the random oracle.*

Construction B.2. Given $\mathsf{IOR} = (\mathbf{I}, \mathbf{P}, \mathbf{V})$, $\mathrm{T}[\mathsf{IOR}] = (\mathcal{G}, \mathcal{I}, \mathcal{P}, \mathcal{V})$ is defined as follows. We use domain separation to split the random oracle ρ into a Merkle tree oracle $\rho_{\mathtt{MT}}$ and Fiat–Shamir oracle $\rho_{\mathtt{FS}}$. The Fiat–Shamir oracle is further split into oracles for each of the verifier's challenges: $\rho_{\mathtt{FS}} = (\rho_i)_{i \in [\mathsf{k}]}$, $\rho_i \sim \mathcal{U}(\mathsf{r}_i)$. Without loss of generality, assume that the verifier's challenges are at least λ bits.

```
$\mathcal{G}(1^\lambda)$:

 1. Output $\mathsf{pp} = 1^\lambda$.
```

```
$\mathcal{I}^\rho(\mathsf{pp}, \mathbb{i})$:

 1. Run the IOR indexer $(\iota, \mathbb{I}) := \mathbf{I}(1^\lambda, \mathbb{i})$.
 2. Commit to the index string $(\mathsf{icm}, \mathsf{itd}) \leftarrow \mathsf{MT.Commit}^{\rho_{\mathtt{MT}}}(\mathbb{I})$.
 3. Output $\mathsf{pk} := (\mathsf{pp}, \mathbb{i}, \iota, \mathbb{I}, \mathsf{icm}, \mathsf{itd})$ and $\mathsf{vk} := (\iota, \mathsf{icm})$.
```

$\mathcal{P}^{\rho}(\mathsf{pk}, (\mathbb{x}, \vec{\mathsf{cm}}), (\mathbb{w}, \vec{\mathbb{y}}, \vec{\mathsf{td}}))$:

1. For $i = 1, \ldots, \mathsf{k}$:
   (a) Compute the i-th proof string
   $$(\mathsf{st}_i, \Pi_i) \leftarrow \begin{cases} \mathbf{P}(1^{\lambda}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) & i = 1 \\ \mathbf{P}(\mathsf{st}_{i-1}, r_{i-1}) & i > 1 \end{cases}$$
   (b) Commit to the i-th proof string $(\mathsf{pcm}_i, \mathsf{ptd}_i) \leftarrow \mathsf{MT.Commit}^{\rho_{\mathtt{MT}}}(\Pi_i)$.
   (c) Derive the i-th challenge
   $$r_i := \begin{cases} \rho_1(\iota, \mathsf{icm}, \mathbb{x}, \mathsf{cm}_1, \ldots, \mathsf{cm}_{\mathsf{n}}, \mathsf{pcm}_1) & i = 1 \\ \rho_i(r_{i-1}, \mathsf{pcm}_i) & i > 1 \end{cases}$$
2. Compute the IOR prover's output $\mathbb{w}' \leftarrow \mathbf{P}(\mathsf{st}_{\mathsf{k}}, r_{\mathsf{k}})$.
3. Compute the IOR verifier's output $(\mathbb{x}', \vec{s}) \leftarrow \mathbf{V}^{\mathbb{I}, \vec{\mathbb{y}}, \vec{\Pi}}(\iota, \mathbb{x}, \vec{r})$. Record the queries made by the IOR verifier to the index, instance, and proof strings.
4. Select new instance strings and trapdoors: for $j = 1, \ldots, \mathsf{n}'$, let $\mathbb{y}'_j := \mathsf{Select}(\vec{\mathbb{y}}, \vec{\Pi}, s_j)$ and $\mathsf{td}'_j := \mathsf{Select}(\vec{\mathsf{td}}, \vec{\mathsf{ptd}}, s_j)$.
5. Set answers and compute opening proofs:
   (a) Let $\mathsf{ans}_1 := \mathbb{I}|_{Q^{\mathtt{I}}}$ and $\mathsf{pf}_1 := \mathsf{MT.Open}(\mathsf{itd}, Q^{\mathtt{I}})$, where $Q^{\mathtt{I}}$ denotes the set of queries made by the IOR verifier to the index string $\mathbb{I}$.
   (b) For $j = 1, \ldots, \mathsf{n}$, let $\mathsf{ans}_{1+j} := \mathbb{y}_j|_{Q^{\mathtt{Y}}_j}$ and $\mathsf{pf}_{1+j} := \mathsf{MT.Open}(\mathsf{td}, Q^{\mathtt{Y}}_j)$, where $Q^{\mathtt{Y}}_j$ denotes the set of queries made by the IOR verifier to the j-th instance string $\mathbb{y}_j$.
   (c) For $i = 1, \ldots, m$, let $\mathsf{ans}_{1+\mathsf{n}+i} := \Pi_i|_{Q^{\mathsf{P}}_i}$ and $\mathsf{pf}_{1+\mathsf{n}+i} := \mathsf{MT.Open}(\mathsf{td}, Q^{\mathsf{P}}_i)$, where Q^{P}_i denotes the set of queries made by the IOR verifier to the i-th proof string Π_i.
6. Output $\pi := (\vec{\mathsf{pcm}}, \vec{\mathsf{ans}}, \vec{\mathsf{pf}})$ and $(\mathbb{w}', \vec{\mathbb{y}}', \vec{\mathsf{td}}')$.

$\mathcal{V}^{\rho}(\mathsf{vk}, (\mathbb{x}, \vec{\mathsf{cm}}), \pi)$:

1. Check opening proofs:
   (a) If $\mathsf{MT.Check}^{\rho_{\mathtt{MT}}}(\mathsf{icm}, \mathsf{ans}_1, \mathsf{pf}_1)$ rejects, output $\bot$.
   (b) For $j = 1, \ldots, \mathsf{n}$, if $\mathsf{MT.Check}^{\rho_{\mathtt{MT}}}(\mathsf{cm}_j, \mathsf{ans}_{1+j}, \mathsf{pf}_{1+j})$ rejects, output $\bot$.
   (c) For $i = 1, \ldots, \mathsf{k}$, if $\mathsf{MT.Check}^{\rho_{\mathtt{MT}}}(\mathsf{pcm}_i, \mathsf{ans}_{1+\mathsf{n}+i}, \mathsf{pf}_{1+\mathsf{n}+i})$ rejects, output $\bot$.
2. For $i = 1, \ldots, \mathsf{k}$: derive the i-th challenge
   $$r_i := \begin{cases} \rho_1(\iota, \mathsf{icm}, \mathbb{x}, \mathsf{cm}_1, \ldots, \mathsf{cm}_{\mathsf{n}}, \mathsf{pcm}_1) & i = 1 \\ \rho_i(r_{i-1}, \mathsf{pcm}_i) & i > 1 \end{cases}$$
3. Compute the IOR verifier's output $(\mathbb{x}', \vec{s}) \leftarrow \mathbf{V}^{\vec{\mathsf{ans}}}(\iota, \mathbb{x}, \vec{r})$.
4. Select new commitments: for $j = 1, \ldots, \mathsf{n}'$, let $\mathsf{cm}'_j := \mathsf{Select}(\vec{\mathsf{cm}}, \vec{\mathsf{pcm}}, s_j)$.

5. Output $(\mathbb{x}', \vec{\mathsf{cm}}')$.

Proof of Theorem 5.9. Let $\mathbf{E}$ be the IOR state-restoration extractor, $\mathcal{E}_{\mathtt{i}} := \mathcal{E}_{\mathtt{i}}[\mathbf{E}]$ be the IR state-restoration extractor from Lemma B.6, and $\mathcal{E} := \mathcal{E}[\mathcal{E}_{\mathtt{i}}]$ be the non-interactive reduction extractor from Lemma B.9; these are all straightline extractors. Let $\tilde{\mathcal{P}}$ be a non-interactive reduction prover which outputs at most n bits, makes at most $t_{\mathtt{MT}}$ queries to $\rho_{\mathtt{MT}}$, and makes at most $t_{\mathtt{FS}}$ queries to $\rho_{\mathtt{FS}}$. Let $\bar{\mathbb{x}} = (\mathbb{x}, \vec{\mathsf{cm}})$ and $\bar{\mathbb{w}} = (\mathbb{w}, \vec{\mathbb{y}}, \vec{\mathsf{td}})$ denote an instance and witness in the committed relations. The knowledge error of $\tilde{\mathcal{P}}$ is

$$\kappa := \Pr\left[\begin{array}{c|l} & \rho := \mathcal{U}(\lambda) \\ & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathbb{i}, \bar{\mathbb{x}}, \bar{\mathbb{w}}) \notin \mathsf{Com}[\tilde{\mathcal{R}}|_S]^\rho(\mathsf{pp}) & (\mathbb{i}, \bar{\mathbb{x}}, \pi, \bar{\mathbb{w}}') \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}^\rho(\mathsf{pp}) \\ (\mathbb{i}', \bar{\mathbb{x}}', \bar{\mathbb{w}}') \in \mathsf{Com}[\tilde{\mathcal{R}}'|_{S'}]^\rho(\mathsf{pp}) & \bar{\mathbb{w}} := \mathcal{E}(\mathbb{i}, \bar{\mathbb{x}}, \pi, \bar{\mathbb{w}}', \mathsf{tr}) \\ & (\mathsf{pk}, \mathsf{vk}, \mathbb{i}') := \mathcal{I}^\rho(\mathsf{pp}, \mathbb{i}) \\ & \bar{\mathbb{x}}' := \mathcal{V}^\rho(\mathsf{vk}, \bar{\mathbb{x}}, \pi) \end{array}\right].$$

Since committed relations do not access $\rho_{\mathtt{FS}}$, it will be convenient to define the relations relative to $\rho_{\mathtt{MT}}$ and explicitly sample the individual oracles. Define $Z := \{(\mathbb{i}, \bar{\mathbb{x}} = (\mathbb{x}, \vec{\mathsf{cm}}) : (\mathbb{i}, \mathbb{x}) \in S(\lambda), |\mathbb{i}| + |\mathbb{x}| \leq n\}$. Rewriting, we get

$$\kappa \leq \Pr\left[\begin{array}{c|l} & \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\lambda) \\ & \rho_{\mathtt{FS}} \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{i}, \bar{\mathbb{x}}) \in Z & \rho := (\rho_{\mathtt{MT}}, \rho_{\mathtt{FS}}) \\ (\mathbb{i}, \bar{\mathbb{x}}, \bar{\mathbb{w}}) \notin \mathsf{Com}[\tilde{\mathcal{R}}]^{\rho_{\mathtt{MT}}} & (\mathbb{i}, \bar{\mathbb{x}}, \pi, \bar{\mathbb{w}}') \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}^\rho \\ (\mathbb{i}', \bar{\mathbb{x}}', \bar{\mathbb{w}}') \in \mathsf{Com}[\tilde{\mathcal{R}}']^{\rho_{\mathtt{MT}}} & \bar{\mathbb{w}} := \mathcal{E}(\mathbb{i}, \bar{\mathbb{x}}, \pi, \bar{\mathbb{w}}', \mathsf{tr}) \\ & (\mathsf{pk}, \mathsf{vk}, \mathbb{i}') := \mathcal{I}^\rho(1^\lambda, \mathbb{i}) \\ & \bar{\mathbb{x}}' := \mathcal{V}^\rho(\mathsf{vk}, \bar{\mathbb{x}}, \pi) \end{array}\right].$$

From Lemma B.9, we obtain

$$\kappa \leq \Pr\left[\begin{array}{c|l} & \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{i}, \bar{\mathbb{x}}) \in Z & \rho_{\mathtt{SR}} \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{i}, \bar{\mathbb{x}}, \bar{\mathbb{w}}) \notin \mathsf{Com}[\tilde{\mathcal{R}}]^{\rho_{\mathtt{MT}}} & (\mathbb{i}, \bar{\mathbb{x}}, \vec{\alpha}, \bar{\mathbb{w}}', \vec{r}) \xleftarrow{\mathsf{tr}} \mathsf{Game}_{\mathtt{IR}}(\rho_{\mathtt{SR}}, \tilde{\mathcal{P}}_{\mathtt{i}}[\tilde{\mathcal{P}}]^{\rho_{\mathtt{MT}}}) \\ (\mathbb{i}', \bar{\mathbb{x}}', \bar{\mathbb{w}}') \in \mathsf{Com}[\tilde{\mathcal{R}}']^{\rho_{\mathtt{MT}}} & \mathbb{w} := \mathcal{E}_{\mathtt{i}}(\mathbb{i}, \bar{\mathbb{x}}, \vec{\alpha}, \bar{\mathbb{w}}', \mathsf{tr}) \\ & (\mathsf{pk}, \mathsf{vk}, \mathbb{i}') := \mathcal{I}_{\mathtt{i}}(1^\lambda, \mathbb{i}) \\ & \mathbb{x}' := \mathcal{V}_{\mathtt{i}}^{\rho_{\mathtt{MT}}}(\mathsf{vk}, \mathbb{x}, \vec{\alpha}, \vec{r}) \end{array}\right] + \frac{t_{\mathtt{FS}}^2}{2^\lambda}.$$

From Lemma B.6, we obtain

$$\kappa \leq \Pr\left[\begin{array}{c|l} & \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\lambda) \\ |\mathbb{i}| + |\mathbb{x}| \leq n & \rho_{\mathtt{SR}} \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{i}, \mathbb{x}) \in S(\lambda) & (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \vec{\Pi}, \mathbb{w}', \vec{r}) \xleftarrow{\mathsf{tr}} \mathsf{Game}_{\mathtt{IOR}}(\rho_{\mathtt{SR}}, \tilde{\mathbf{P}}[\tilde{\mathcal{P}}_{\mathtt{i}}[\tilde{\mathcal{P}}]]^{\rho_{\mathtt{MT}}}) \\ (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \notin \tilde{\mathcal{R}} & \mathbb{w} := \mathbf{E}(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \vec{\Pi}, \mathbb{w}', \mathsf{tr}) \\ (\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}') \in \tilde{\mathcal{R}}' & (\iota, \mathbb{I}, \mathbb{i}') := \mathbf{I}(1^\lambda, \mathbb{i}) \\ & (\mathbb{x}', \vec{\mathbb{y}}') := \mathbf{V}^{\mathbb{I}, \vec{\mathbb{y}}, \vec{\Pi}}(\iota, \mathbb{x}, \vec{r}) \end{array}\right]$$

$$+ \kappa_{\mathtt{MT}}(\lambda, t_{\mathtt{MT}}, \mathsf{L}_{\mathtt{max}}, 1 + \mathsf{n} + \mathsf{k}) + \frac{t_{\mathtt{FS}}^2}{2^\lambda}.$$

Observe that $\tilde{\mathbf{P}}[\tilde{\mathcal{P}}_{\mathtt{i}}[\tilde{\mathcal{P}}]]$ makes at most $t_{\mathtt{FS}}$ moves. By straightline state-restoration knowledge soundness of IOR, we have

$$\kappa \leq \kappa_{\mathtt{SR}}(\lambda, t_{\mathtt{FS}}, n) + \kappa_{\mathtt{MT}}(\lambda, t_{\mathtt{MT}}, \mathsf{L}_{\mathtt{max}}, 1 + \mathsf{n} + \mathsf{k}) + \frac{t_{\mathtt{FS}}^2}{2^\lambda}.$$

Setting $n = \mathrm{poly}(\lambda)$, $t_{\mathtt{FS}} = \mathrm{poly}(\lambda)$, and $t_{\mathtt{MT}} = \mathrm{poly}(\lambda)$, $\mathsf{n} = \mathrm{poly}(\lambda)$, and $\mathsf{k} = \mathrm{poly}(\lambda)$, we get $\kappa = \mathrm{negl}(\lambda)$. □

B.1 State-restoration soundness

We define state-restoration soundness for IORs, and show that round-by-round knowledge soundness implies state-restoration soundness.

Definition B.3. The *IOR state restoration game* $\mathsf{Game}_{\mathtt{IOR}}$ with functions $\rho = (\rho_i)_{i \in [\mathsf{k}]}$ and adversary $\tilde{\mathbf{P}}$ is defined below.

```
$\mathsf{Game}_{\mathtt{IOR}}(\rho, \tilde{\mathbf{P}})$:
  1. Repeat the following until $\tilde{\mathbf{P}}$ exists the loop:
     (a) $\tilde{\mathbf{P}}$ sends a move $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, (\Pi_1, \ldots, \Pi_i))$ with $i \in [\mathsf{k}]$.
     (b) Set $r_i := \rho_i(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, (\Pi_1, \ldots, \Pi_i))$.
     (c) Return $r_i$ to $\tilde{\mathbf{P}}$.
  2. $\tilde{\mathbf{P}}$ outputs $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, (\Pi_1, \ldots, \Pi_{\mathsf{k}}), \mathbb{w}', (A_1, \ldots, A_{\mathsf{n}'}))$.
  3. For each $i = 1, \ldots, \mathsf{k}$, set $r_i := \rho_i(\mathbb{i}, \mathbb{x}, (\Pi_1, \ldots, \Pi_i))$.
  4. Output $(\mathbb{i}, \mathbb{x}, (\Pi_i)_{i \in [\mathsf{k}]}, \mathbb{w}', (A_j)_{j \in [\mathsf{n}']}, (r_i)_{i \in [\mathsf{k}]})$.
```

Let tr denote the list of move-response pairs performed in the loop. We use the following notation to describe an execution of the state-restoration game:

$$(\mathbb{i}, \mathbb{x}, \vec{\Pi}, \mathbb{w}', \vec{A}, \vec{r}) \xleftarrow{\mathsf{tr}} \mathsf{Game}_{\mathtt{IOR}}(\rho, \tilde{\mathbf{P}}).$$

Definition B.4. IOR has *straightline state-restoration knowledge error* $\kappa_{\mathtt{sr}}$ if the following holds. There exists a polynomial-time extractor $\mathbf{E}$ such that for every move budget $t \in \mathbb{N}$, t-move deterministic adversary $\tilde{\mathbf{P}}$, parameters $\mathbb{p}$, and set S,

$$\Pr\left[\begin{array}{c|l} & \rho = (\rho_i)_{i \in [\mathsf{k}]} \leftarrow \mathcal{U}((\mathsf{r}_i)_{i \in [\mathsf{k}]}) \\ (\mathbb{i}, \mathbb{x}) \in S & (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \vec{\Pi}, \mathbb{w}', \vec{A}, \vec{r}) \xleftarrow{\mathsf{tr}} \mathsf{Game}_{\mathtt{IOR}}(\rho, \tilde{\mathbf{P}}) \\ (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \notin \tilde{\mathcal{R}} & (\iota, \mathbb{I}, \mathbb{i}') := \mathbf{I}(\mathbb{p}, \mathbb{i}) \\ (\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}^*, \mathbb{w}') \in \tilde{\mathcal{R}}' & (\mathbb{x}', \vec{s}) := \mathbf{V}^{\mathbb{I}, \vec{\mathbb{y}}, \vec{\Pi}}(\iota, \mathbb{x}, \vec{r}) \\ & \forall j \in [\mathsf{n}'], \mathbb{y}_j^* := \mathsf{Select}(\vec{\mathbb{y}}, \vec{\Pi}, s_j)|_{A_j} \\ & \mathbb{w} \leftarrow \mathbf{E}(\mathbb{p}, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \vec{\Pi}, \mathbb{w}', \vec{A}, \mathsf{tr}) \end{array}\right] \leq \kappa_{\mathtt{sr}}(t, \mathbb{p}, S).$$

Theorem B.5. *Suppose that $\mathcal{R}'$ is monotone. If* IOR *has round-by-round knowledge error $\kappa_{\mathtt{rbr}}$, then* IOR *has straightline state-restoration knowledge error*

$$\kappa_{\mathtt{sr}}(t, \mathbb{p}, S) := \max_{(\mathbb{i}, \mathbb{x}) \in S} (t + \mathsf{k}) \cdot \kappa_{\mathtt{rbr}}(\mathbb{p}, \mathbb{i}, \mathbb{x}). \tag{B.1}$$

Proof. Since $\mathcal{R}'$ is monotone, the adversary's restrictions $\vec{A}$ cannot improve its advantage. It therefore suffices to bound the following probability:

$$\Pr\left[\begin{array}{c|l} & \rho = (\rho_i)_{i\in[\mathsf{k}]} \leftarrow \mathcal{U}((\mathsf{r}_i)_{i\in[\mathsf{k}]}) \\ (\mathbb{i},\mathbb{x})\in S & (\mathbb{i},\mathbb{x},\vec{\mathbb{y}},\vec{\Pi},\mathbb{w}',\vec{r}) \xleftarrow{\mathsf{tr}} \mathsf{Game}_{\mathtt{IOR}}(\rho,\tilde{\mathbf{P}}) \\ (\mathbb{i},\mathbb{x},\vec{\mathbb{y}},\mathbb{w})\notin\tilde{\mathcal{R}} & (\iota,\mathbb{I},\mathbb{i}') := \mathbf{I}(\mathbb{p},\mathbb{i}) \\ (\mathbb{i}',\mathbb{x}',\vec{\mathbb{y}}',\mathbb{w}')\in\tilde{\mathcal{R}}' & (\mathbb{x}',\vec{\mathbb{y}}') := \mathbf{V}^{\mathbb{I},\vec{\mathbb{y}},\vec{\Pi}}(\iota,\mathbb{x},\vec{r}) \\ & \mathbb{w}\leftarrow \mathbf{E}(\mathbb{p},\mathbb{i},\mathbb{x},\vec{\mathbb{y}},\vec{\Pi},\mathbb{w}',\mathsf{tr}) \end{array}\right]$$

Rearranging, we find that this is identical to the state-restoration knowledge error of an IOP where the prover sends the new witness as an additional message and the verifier performs all of the highlighted steps:

$$\Pr\left[\begin{array}{c|l} & \rho = (\rho_i)_{i\in[\mathsf{k}]} \leftarrow \mathcal{U}((\mathsf{r}_i)_{i\in[\mathsf{k}]}) \\ & (\mathbb{i},\mathbb{x},\vec{\mathbb{y}},(\Pi_1,\ldots,\Pi_{\mathsf{k}},\mathbb{w}'),\vec{r}) \xleftarrow{\mathsf{tr}} \mathsf{Game}_{\mathtt{IOR}}(\rho,\tilde{\mathbf{P}}) \\ (\mathbb{i},\mathbb{x})\in S & (\iota,\mathbb{I},\mathbb{i}') := \mathbf{I}(\mathbb{p},\mathbb{i}) \\ (\mathbb{i},\mathbb{x},\vec{\mathbb{y}},\mathbb{w})\notin\tilde{\mathcal{R}} & (\mathbb{x}',\vec{\mathbb{y}}') := \mathbf{V}^{\mathbb{I},\vec{\mathbb{y}},\vec{\Pi}}(\iota,\mathbb{x},\vec{r}) \\ b=1 & b := (\mathbb{i}',\mathbb{x}',\vec{\mathbb{y}}',\mathbb{w}')\in\tilde{\mathcal{R}}' \\ & \mathbb{w}\leftarrow \mathbf{E}(\mathbb{p},\mathbb{i},\mathbb{x},\vec{\mathbb{y}},\vec{\Pi},\mathbb{w}',\mathsf{tr}) \end{array}\right]$$

Observe that the round-by-round knowledge error of this IOP is precisely $\kappa_{\mathtt{rbr}}$. We conclude by appealing to [CY24, Theorem 31.2.1], which relates IOP state-restoration knowledge to IOP round-by-round knowledge precisely as in Equation (B.1).[7] □

B.2 Replacing oracles with Merkle commitments

Lemma B.6. *Let* $\mathbf{E}$ *be an IOR state-restoration straightline extractor. There exists an IOR state-restoration prover* $\tilde{\mathbf{P}}[\cdot]$ *and deterministic polynomial-time IR state-restoration straightline extractor* $\mathcal{E}_{\mathtt{i}} := \mathcal{E}_{\mathtt{i}}[\mathbf{E}]$ *such that, for every* $t_{\$}$*-move IR state-restoration prover* $\tilde{\mathcal{P}}_{\mathtt{i}}$ *that makes at most* $t_{\mathtt{MT}}$ *queries to* $\rho_{\mathtt{MT}}$*,* $\tilde{\mathbf{P}}[\tilde{\mathcal{P}}_{\mathtt{i}}]$ *makes at*

[7] There are a few technical details which we address here. First, [CY24] restricts index-instance pairs by imposing a size bound, whereas we test membership in S. This only changes how we compute the SR knowledge error, i.e., what index-instance pairs we quantify over to find the maximum RBR knowledge error. Second, a direct application of [CY24, Theorem 31.2.1] would say that the SR knowledge error is $(t+\mathsf{k}+1)\cdot\kappa_{\mathtt{rbr}}$, since the IOP has $\mathsf{k}+1$ rounds. However, a close reading shows that the SR knowledge error is in fact $(t+\mathsf{k})\cdot\kappa_{\mathtt{rbr}}$; this is because the final round only contains a prover message, and the RBR state function cannot change after the verifier's last message.

most $t_\$$ moves and the following holds for any set Z:

$$
\begin{aligned}
&\Pr\left[\begin{array}{c|l}
 & \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\lambda) \\
 & \rho_{\mathtt{SR}} \leftarrow \mathcal{U}(\lambda) \\
(\mathbb{i}, \mathbb{x}) \in Z & (\mathbb{i}, (\mathbb{x}, \vec{\mathsf{cm}}), \vec{\alpha}, (\mathbb{w}', \vec{\mathbb{y}}', \vec{\mathsf{td}}'), \vec{r}) \\
(\mathbb{i}, (\mathbb{x}, \vec{\mathsf{cm}}), (\mathbb{w}, \vec{\mathbb{y}}, \mathsf{td})) \in \mathsf{Com}[\tilde{\mathcal{R}}]^{\rho_{\mathtt{MT}}} & \quad \xleftarrow{\mathsf{tr}} \mathsf{Game}_{\mathtt{IR}}(\rho_{\mathtt{SR}}, \tilde{\mathcal{P}_{\mathtt{i}}}^{\rho_{\mathtt{MT}}}) \\
(\mathbb{i}', (\mathbb{x}', \vec{\mathsf{cm}}'), (\mathbb{w}', \vec{\mathbb{y}}', \mathsf{td}')) \in \mathsf{Com}[\tilde{\mathcal{R}}']^{\rho_{\mathtt{MT}}} & (\mathbb{w}, \vec{\mathbb{y}}, \vec{\mathsf{td}}) := \mathcal{E}_{\mathtt{i}}(\mathbb{i}, (\mathbb{x}, \vec{\mathsf{cm}}), \vec{\alpha}, (\mathbb{w}', \vec{\mathbb{y}}', \vec{\mathsf{td}}'), \mathsf{tr}) \\
 & (\mathsf{pk}, \mathsf{vk}, \mathbb{i}') := \mathcal{I}_{\mathtt{i}}(1^\lambda, \mathbb{i}) \\
 & (\mathbb{x}', \vec{\mathsf{cm}}') := \mathcal{V}_{\mathtt{i}}^{\rho_{\mathtt{MT}}}(\mathsf{vk}, (\mathbb{x}, \vec{\mathsf{cm}}), \vec{\alpha}, \vec{r})
\end{array}\right] \\
&\leq \Pr\left[\begin{array}{c|l}
 & \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\lambda) \\
 & \rho_{\mathtt{SR}} \leftarrow \mathcal{U}(\lambda) \\
(\mathbb{i}, \mathbb{x}) \in Z & (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \vec{\Pi}, \mathbb{w}', \vec{r}) \xleftarrow{\mathsf{tr}} \mathsf{Game}_{\mathtt{IOR}}(\rho_{\mathtt{SR}}, \tilde{\mathbf{P}}[\tilde{\mathcal{P}_{\mathtt{i}}}]^{\rho_{\mathtt{MT}}}) \\
(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \notin \tilde{\mathcal{R}} & \mathbb{w} := \mathbf{E}(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \vec{\Pi}, \mathbb{w}', \mathsf{tr}) \\
(\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}') \in \tilde{\mathcal{R}}' & (\iota, \mathbb{I}, \mathbb{i}') := \mathbf{I}(1^\lambda, \mathbb{i}) \\
 & (\mathbb{x}', \vec{\mathbb{y}}') := \mathbf{V}^{\mathbb{I}, \vec{\mathbb{y}}, \vec{\Pi}}(\iota, \mathbb{x}, \vec{r})
\end{array}\right] \\
&\quad + \kappa_{\mathtt{MT}}(\lambda, t_{\mathtt{MT}}, \mathsf{L}_{\mathtt{max}}, 1 + \mathsf{n} + \mathsf{k}).
\end{aligned}
\tag{B.2}
$$

Construction B.7. Given a state-restoration prover $\tilde{\mathcal{P}_{\mathtt{i}}}$ for $\mathrm{T}_{\mathtt{i}}[\mathsf{IOR}]$, the state-restoration prover $\tilde{\mathbf{P}}[\tilde{\mathcal{P}_{\mathtt{i}}}]$ for IOR is defined as follows.

$\tilde{\mathbf{P}}[\tilde{\mathcal{P}_{\mathtt{i}}}]^{\rho_{\mathtt{MT}}}$:

1. Initialize an empty query-answer trace $\mathsf{tr}_{\mathtt{MT}}$.
2. Simulate $\tilde{\mathcal{P}_{\mathtt{i}}}$ while answering its queries:
   (a) When $\tilde{\mathcal{P}_{\mathtt{i}}}$ makes a query to the oracle $\rho_{\mathtt{MT}}$, answer according to $\rho_{\mathtt{MT}}$ and append the query-answer pair to $\mathsf{tr}_{\mathtt{MT}}$.
   (b) When $\tilde{\mathcal{P}_{\mathtt{i}}}$ sends a move $(\mathbb{i}, (\mathbb{x}, \vec{\mathsf{cm}}), (\mathsf{pcm}_1, \ldots, \mathsf{pcm}_i))$ with $i \in [\mathsf{k}]$:
      i. Let $\mathsf{tr} \subset \mathsf{tr}_{\mathtt{MT}}$ be the query-answer pairs for $\rho_{\mathtt{MT}}$ since the previous move (or the beginning if this is the first move).
      ii. Extract commitment openings:
      $$\begin{aligned} &((\mathbb{y}_1, \mathsf{td}_1), \ldots, (\mathbb{y}_{\mathsf{n}}, \mathsf{td}_{\mathsf{n}}), (\Pi_1, \mathsf{ptd}_1), \ldots, (\Pi_i, \mathsf{ptd}_i)) \\ &:= \mathsf{MT.MultiExtract}((\mathsf{cm}_1, \ldots, \mathsf{cm}_{\mathsf{n}}, \mathsf{pcm}_1, \ldots, \mathsf{pcm}_i), \mathsf{tr}) \end{aligned}$$
      iii. Send the move $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \Pi_1, \ldots, \Pi_i)$ for the IOR state-restoration game.
      iv. Receive the challenge r_i from the game.
      v. Return r_i to $\tilde{\mathcal{P}_{\mathtt{i}}}$.
3. Finally, $\tilde{\mathcal{P}_{\mathtt{i}}}$ halts and outputs $(\mathbb{i}, (\mathbb{x}, \vec{\mathsf{cm}}), (\mathsf{pcm}_1, \ldots, \mathsf{pcm}_{\mathsf{k}}, (\vec{\mathsf{ans}}, \vec{\mathsf{pf}})), (\mathbb{w}', \vec{\mathbb{y}}', \vec{\mathsf{td}}'))$.
4. Let $\mathsf{tr} \subset \mathsf{tr}_{\mathtt{MT}}$ be the query-answer pairs for $\rho_{\mathtt{MT}}$ since the last move while simulating $\tilde{\mathcal{P}_{\mathtt{i}}}$.
5. Extract commitment openings:
$$\begin{aligned} &((\mathbb{y}_1, \mathsf{td}_1), \ldots, (\mathbb{y}_{\mathsf{n}}, \mathsf{td}_{\mathsf{n}}), (\Pi_1, \mathsf{ptd}_1), \ldots, (\Pi_{\mathsf{k}}, \mathsf{ptd}_{\mathsf{k}})) \\ &:= \mathsf{MT.MultiExtract}((\mathsf{cm}_1, \ldots, \mathsf{cm}_{\mathsf{n}}, \mathsf{pcm}_1, \mathsf{pcm}_{\mathsf{k}}), \mathsf{tr}) \end{aligned}$$

6. For each $j = 1, \ldots, \mathsf{n}'$, set the restriction $A_j := \operatorname{Dom} \mathbb{y}'_j$.
7. Output $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \vec{\Pi}, \mathbb{w}', \vec{A})$. Also, implicitly output $\vec{\mathsf{cm}}$, $\mathsf{p}\vec{\mathsf{cm}}$, ans, pf, $\vec{\mathbb{y}}'$, $\vec{\mathsf{td}}'$ (these are only used in the analysis).

Construction B.8. The IR state-restoration extractor $\mathcal{E}_{\mathtt{i}}$ is defined as follows.

$\mathcal{E}_{\mathtt{i}}(\mathbb{i}, (\mathbb{x}, \vec{\mathsf{cm}}), (\mathsf{pcm}_1, \ldots, \mathsf{pcm}_{\mathsf{k}}, (\mathsf{a}\vec{\mathsf{n}}\mathsf{s}, \vec{\mathsf{pf}})), (\mathbb{w}', \vec{\mathbb{y}}', \vec{\mathsf{td}}'), \mathsf{tr})$:

1. Split the prover's trace tr into an IR state-restoration game trace $\mathsf{tr}_{\mathtt{IR}}$ and oracle trace $\mathsf{tr}_{\mathtt{MT}}$.
2. Set $\mathsf{tr}_{\mathtt{IOR}} := \mathsf{IORTrace}(\mathsf{tr}_{\mathtt{IR}}, \mathsf{tr}_{\mathtt{MT}})$, where $\mathsf{IORTrace}$ is the deterministic algorithm which computes the IOR state-restoration trace corresponding to $\tilde{\mathbf{P}}[\tilde{\mathcal{P}}_{\mathtt{i}}]$, as defined in Construction B.7.
3. Extract commitment openings:

$$\begin{aligned} &((\mathbb{y}_1, \mathsf{td}_1), \ldots, (\mathbb{y}_{\mathsf{n}}, \mathsf{td}_{\mathsf{n}}), (\Pi_1, \mathsf{ptd}_1), \ldots, (\Pi_{\mathsf{k}}, \mathsf{ptd}_{\mathsf{k}})) \\ &\quad := \mathsf{MT.MultiExtract}((\mathsf{cm}_1, \ldots, \mathsf{cm}_{\mathsf{n}}, \mathsf{pcm}_1, \mathsf{pcm}_{\mathsf{k}}), \mathsf{tr}_{\mathtt{MT}}) \end{aligned}$$

4. Run the IOR state-restoration extractor: $\mathbb{w} := \mathbf{E}(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \vec{\Pi}, \mathbb{w}', \mathsf{tr}_{\mathtt{IOR}})$.
5. Output $(\mathbb{w}, \vec{\mathbb{y}}, \vec{\mathsf{td}})$.

Proof of Lemma B.6. Define the random variable X as follows:

$$\left\{ (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, b, \mathsf{tr}_{\mathtt{IOR}}, \mathsf{tr}_{\mathtt{MT}}, \vec{r}) \;\middle|\; \begin{array}{l} \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\lambda) \\ \rho_{\mathtt{SR}} \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{i}, (\mathbb{x}, \vec{\mathsf{cm}}), \mathsf{p}\vec{\mathsf{cm}} || (\mathsf{a}\vec{\mathsf{n}}\mathsf{s}, \vec{\mathsf{pf}}), (\mathbb{w}', \vec{\mathbb{y}}', \vec{\mathsf{td}}'), \vec{r}) \\ \quad \xleftarrow{\mathsf{tr}_{\mathtt{IR}}, \mathsf{tr}_{\mathtt{MT}}} \mathsf{Game}_{\mathtt{IR}}(\rho_{\mathtt{SR}}, \tilde{\mathcal{P}}_{\mathtt{i}}^{\,\rho_{\mathtt{MT}}}) \\ (\mathbb{y}_j, \mathsf{td}_j)_{j=1}^{\mathsf{n}} || (\Pi_i, \mathsf{ptd}_i)_{i=1}^{\mathsf{k}} \\ \quad := \mathsf{MT.MultiExtract}(\vec{\mathsf{cm}} || \mathsf{p}\vec{\mathsf{cm}}, \mathsf{tr}_{\mathtt{MT}}) \\ (\mathsf{pk}, \mathsf{vk}) := \mathcal{I}_{\mathtt{i}}^{\rho_{\mathtt{MT}}}(1^\lambda, \mathbb{i}) \\ (\mathbb{x}', \vec{\mathsf{cm}}') := \mathcal{V}_{\mathtt{i}}^{\rho_{\mathtt{MT}}}(\mathsf{vk}, (\mathbb{x}, \vec{\mathsf{cm}}), \mathsf{p}\vec{\mathsf{cm}} || (\mathsf{a}\vec{\mathsf{n}}\mathsf{s}, \vec{\mathsf{pf}}), \vec{r}) \\ b := (\mathbb{i}, (\mathbb{x}', \vec{\mathsf{cm}}'), (\mathbb{w}, \vec{\mathbb{y}}', \vec{\mathsf{td}}')) \in \mathsf{Com}[\tilde{\mathcal{R}}_2]^{\rho_{\mathtt{MT}}} \\ \mathsf{tr}_{\mathtt{IOR}} := \mathsf{IORTrace}(\mathsf{tr}_{\mathtt{IR}}, \mathsf{tr}_{\mathtt{MT}}) \end{array} \right\} \tag{B.3}$$

Define the random variable Y as follows:

$$\left\{ (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, b, \mathsf{tr}_{\mathtt{IOR}}, \mathsf{tr}_{\mathtt{MT}}, \vec{r}) \;\middle|\; \begin{array}{l} \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\lambda) \\ \rho \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \vec{\Pi}, \mathbb{w}', \vec{A}, \vec{r}; \vec{\mathsf{cm}}, \mathsf{p}\vec{\mathsf{cm}}, \mathsf{a}\vec{\mathsf{ns}}, \mathsf{p}\vec{\mathsf{f}}, \vec{\mathbb{y}}', \mathsf{t}\vec{\mathsf{d}}') \\ \quad \xleftarrow{\mathsf{tr}_{\mathtt{IOR}}, \mathsf{tr}_{\mathtt{MT}}} \mathsf{Game}_{\mathtt{IOR}}(\rho, \tilde{\mathbf{P}}[\tilde{\mathcal{P}}_{\mathtt{i}}]^{\rho_{\mathtt{MT}}}) \\ (\iota, \mathbb{I}) := \mathbf{I}(1^{\lambda}, \mathbb{i}) \\ (\mathbb{x}', \vec{s}) := \mathbf{V}^{\mathbb{I}, \vec{\mathbb{y}}, \vec{\Pi}}(\iota, \mathbb{x}, \vec{r}) \\ \forall j \in [\mathsf{n}'], \mathbb{y}^* := \mathsf{Select}(\vec{\mathbb{y}}, \vec{\Pi}, s_j)|_{A_j} \\ b_1 := (\mathbb{i}, \mathbb{x}', \vec{\mathbb{y}}^*, \mathbb{w}') \in \tilde{\mathcal{R}}_2 \\ (\mathsf{icm}, \mathsf{itd}) := \mathsf{MT.Commit}^{\rho_{\mathtt{MT}}}(\mathbb{I}) \\ b_2 := \mathsf{CheckAll}^{\rho_{\mathtt{MT}}}(\mathsf{icm}, \vec{\mathsf{cm}}, \mathsf{p}\vec{\mathsf{cm}}, \mathsf{a}\vec{\mathsf{ns}}, \mathsf{p}\vec{\mathsf{f}}) \\ \forall j \in [\mathsf{n}'], \mathsf{cm}'_j := \mathsf{Select}(\vec{\mathsf{cm}}, \mathsf{p}\vec{\mathsf{cm}}, s_j) \\ b_3 := \bigwedge_{j=1}^{\mathsf{n}'} \mathsf{MT.Verify}^{\rho_{\mathtt{MT}}}(\mathsf{cm}'_j, \mathbb{y}'_j, \mathsf{td}'_j) \\ b := b_1 \wedge b_2 \wedge b_3 \end{array} \right\}$$

It suffices to show that X and Y are $\kappa_{\mathtt{MT}}(\dots)$-statistically close. Unwrapping $\mathcal{I}_{\mathtt{i}}$, $\mathcal{V}_{\mathtt{i}}$, and $\mathsf{Com}[\tilde{\mathcal{R}}_2]$ in the definition of X, we get the following (differences from Equation (B.3) are highlighted):

$$\left\{ (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, b, \mathsf{tr}_{\mathtt{IOR}}, \mathsf{tr}_{\mathtt{MT}}, \vec{r}) \;\middle|\; \begin{array}{l} \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\lambda) \\ \rho \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{i}, (\mathbb{x}, \vec{\mathsf{cm}}), \mathsf{p}\vec{\mathsf{cm}} || (\mathsf{a}\vec{\mathsf{ns}}, \mathsf{p}\vec{\mathsf{f}}), \vec{r}, (\mathbb{w}', \vec{\mathbb{y}}', \mathsf{t}\vec{\mathsf{d}}')) \\ \quad \xleftarrow{\mathsf{tr}_{\mathtt{IR}}, \mathsf{tr}_{\mathtt{MT}}} \mathsf{Game}_{\mathtt{IR}}(\rho, \tilde{\mathcal{P}_{\mathtt{i}}}^{\rho_{\mathtt{MT}}}) \\ (\mathbb{y}_j, \mathsf{td}_j)_{j=1}^{\mathsf{n}} || (\Pi_i, \mathsf{ptd}_i)_{i=1}^{\mathsf{k}} \\ \quad := \mathsf{MT.MultiExtract}(\vec{\mathsf{cm}} || \mathsf{p}\vec{\mathsf{cm}}), \mathsf{tr}_{\mathtt{MT}}) \\ (\iota, \mathbb{I}) := \mathbf{I}(1^{\lambda}, \mathbb{i}) \\ (\mathsf{icm}, \mathsf{itd}) := \mathsf{MT.Commit}^{\rho_{\mathtt{MT}}}(\mathbb{I}) \\ b_2 := \mathsf{CheckAll}^{\rho_{\mathtt{MT}}}(\mathsf{icm}, \vec{\mathsf{cm}}, \mathsf{p}\vec{\mathsf{cm}}, \mathsf{a}\vec{\mathsf{ns}}, \mathsf{p}\vec{\mathsf{f}}) \\ (\mathbb{x}', \vec{s}) := \mathbf{V}^{\mathsf{a}\vec{\mathsf{ns}}}(\iota, \mathbb{x}, \vec{r}) \\ \forall j \in [\mathsf{n}'], \mathsf{cm}'_j := \mathsf{Select}(\vec{\mathsf{cm}}, \mathsf{p}\vec{\mathsf{cm}}, s_j) \\ b_1 := (\mathbb{i}, \mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}') \in \tilde{\mathcal{R}}_2 \\ b_3 := \bigwedge_{j=1}^{\mathsf{n}'} \mathsf{MT.Verify}^{\rho_{\mathtt{MT}}}(\mathsf{cm}'_j, \mathbb{y}'_j, \mathsf{td}'_j) \\ b := b_1 \wedge b_2 \wedge b_3 \\ \mathsf{tr}_{\mathtt{IOR}} := \mathsf{IORTrace}(\mathsf{tr}_{\mathtt{IR}}, \mathsf{tr}_{\mathtt{MT}}) \end{array} \right\} \tag{B.4}$$

Define the event E as follows (differences from Equation (B.4) are highlighted):

$$\left[\begin{array}{c|l}
 & \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\lambda) \\
 & \rho \leftarrow \mathcal{U}(\lambda) \\
 & (\mathring{\mathbb{i}}, (\mathbb{x}, \vec{\mathsf{cm}}), \vec{\mathsf{pcm}} || (\vec{\mathsf{ans}}, \vec{\mathsf{pf}}), \vec{r}, (\mathbb{w}', \vec{\mathbb{y}}', \vec{\mathsf{td}}')) \\
 & \quad \xleftarrow{\mathsf{tr}_{\mathtt{MT}}} \mathsf{Game}_{\mathtt{IR}}(\rho, \tilde{\mathcal{P}}_{\mathtt{i}}^{\rho_{\mathtt{MT}}}) \\
\neg(b_2 \wedge b_3) & (\mathbb{y}_j, \mathsf{td}_j)_{j=1}^{\mathsf{n}} || (\Pi_i, \mathsf{ptd}_i)_{i=1}^{\mathsf{k}} \\
\vee & \quad := \mathsf{MT.MultiExtract}(\vec{\mathsf{cm}} || \vec{\mathsf{pcm}}), \mathsf{tr}_{\mathtt{MT}}) \\
\mathbb{I}|_{\mathrm{Dom}\,\mathsf{ans}_1} = \mathsf{ans}_1 & (\iota, \mathbb{I}) := \mathbf{I}(1^\lambda, \mathring{\mathbb{i}}) \\
\bigwedge_{j=1}^{\mathsf{n}} \mathbb{y}_j|_{\mathrm{Dom}\,\mathsf{ans}_{1+j}} = \mathsf{ans}_{1+j} & (\mathsf{icm}, \mathsf{itd}) := \mathsf{MT.Commit}^{\rho_{\mathtt{MT}}}(\mathbb{I}) \\
\bigwedge_{i=1}^{\mathsf{k}} \Pi_i|_{\mathrm{Dom}\,\mathsf{ans}_{1+\mathsf{n}+i}} = \mathsf{ans}_{1+\mathsf{n}+i} & b_2 := \mathsf{CheckAll}^{\rho_{\mathtt{MT}}}(\mathsf{icm}, \vec{\mathsf{cm}}, \vec{\mathsf{pcm}}, \vec{\mathsf{ans}}, \vec{\mathsf{pf}}) \\
\bigwedge_{j=1}^{\mathsf{n}'} \mathbb{y}_j^* = \mathbb{y}_j' & (\mathbb{x}', \vec{s}) := \mathbf{V}^{\vec{\mathsf{ans}}}(\iota, \mathbb{x}, \vec{r}) \\
 & \forall j \in [\mathsf{n}'], \mathsf{cm}_j' := \mathsf{Select}(\vec{\mathsf{cm}}, \vec{\mathsf{pcm}}, s_j) \\
 & b_3 := \bigwedge_{j=1}^{\mathsf{n}'} \mathsf{MT.Verify}^{\rho_{\mathtt{MT}}}(\mathsf{cm}_j', \mathbb{y}_j', \mathsf{td}_j') \\
 & \forall j \in [\mathsf{n}'], \mathbb{y}_j^* := \mathsf{Select}(\vec{\mathbb{y}}, \vec{\Pi}, s_j)|_{\mathrm{Dom}\,\mathbb{y}_j'}
\end{array}\right]$$

Observe that $(X|E)$ is equivalent to the following distribution (differences from Equation (B.4) are highlighted):

$$\left\{\begin{array}{c|l}
 & \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\lambda) \\
 & \rho \leftarrow \mathcal{U}(\lambda) \\
 & (\mathring{\mathbb{i}}, (\mathbb{x}, \vec{\mathsf{cm}}), \vec{\mathsf{pcm}} || (\vec{\mathsf{ans}}, \vec{\mathsf{pf}}), \vec{r}, (\mathbb{w}', \vec{\mathbb{y}}', \vec{\mathsf{td}}')) \\
 & \quad \xleftarrow{\mathsf{tr}_{\mathtt{IR}}, \mathsf{tr}_{\mathtt{MT}}} \mathsf{Game}_{\mathtt{IR}}(\rho, \tilde{\mathcal{P}}_{\mathtt{i}}^{\rho_{\mathtt{MT}}}) \\
 & (\mathbb{y}_j, \mathsf{td}_j)_{j=1}^{\mathsf{n}} || (\Pi_i, \mathsf{ptd}_i)_{i=1}^{\mathsf{k}} \\
 & \quad := \mathsf{MT.MultiExtract}(\vec{\mathsf{cm}} || \vec{\mathsf{pcm}}), \mathsf{tr}_{\mathtt{MT}}) \\
 & (\iota, \mathbb{I}) := \mathbf{I}(1^\lambda, \mathring{\mathbb{i}}) \\
(\mathring{\mathbb{i}}, \mathbb{x}, \vec{\mathbb{y}}, b, \mathsf{tr}_{\mathtt{IOR}}, \mathsf{tr}_{\mathtt{MT}}, \vec{r}) & (\mathsf{icm}, \mathsf{itd}) := \mathsf{MT.Commit}^{\rho_{\mathtt{MT}}}(\mathbb{I}) \\
\text{conditioned on } E & b_2 := \mathsf{CheckAll}^{\rho_{\mathtt{MT}}}(\mathsf{icm}, \vec{\mathsf{cm}}, \vec{\mathsf{pcm}}, \vec{\mathsf{ans}}, \vec{\mathsf{pf}}) \\
 & (\mathbb{x}', \vec{s}) := \mathbf{V}^{\mathbb{I}, \vec{\mathbb{y}}, \vec{\Pi}}(\iota, \mathbb{x}, \vec{r}) \\
 & \forall j \in [\mathsf{n}'], \mathbb{y}_j^* := \mathsf{Select}(\vec{\mathbb{y}}, \vec{\Pi}, s_j)|_{\mathrm{Dom}\,\mathbb{y}_j'} \\
 & \forall j \in [\mathsf{n}'], \mathsf{cm}_j' := \mathsf{Select}(\vec{\mathsf{cm}}, \vec{\mathsf{pcm}}, s_j) \\
 & b_1 := (\mathring{\mathbb{i}}, \mathbb{x}', \vec{\mathbb{y}}^*, \mathbb{w}') \in \tilde{\mathcal{R}}_2(\mathbb{p}) \\
 & b_3 := \bigwedge_{j=1}^{\mathsf{n}'} \mathsf{MT.Verify}^{\rho_{\mathtt{MT}}}(\mathsf{cm}_j', \mathbb{y}_j', \mathsf{td}_j') \\
 & b := b_1 \wedge b_2 \wedge b_3 \\
 & \mathsf{tr}_{\mathtt{IOR}} := \mathsf{IORTrace}(\mathsf{tr}_{\mathtt{IR}}, \mathsf{tr}_{\mathtt{MT}})
\end{array}\right\} \tag{B.5}$$

This is equivalent to the following distribution (differences from Equation (B.5) are highlighted):

$$\left\{ \begin{array}{c|l} & \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\lambda) \\ & \rho \leftarrow \mathcal{U}(\lambda) \\ & (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \vec{\Pi}, \mathbb{w}', \vec{A}, \vec{r}; \vec{\mathsf{cm}}, \vec{\mathsf{pcm}}, \vec{\mathsf{ans}}, \vec{\mathsf{pf}}, \vec{\mathbb{y}}', \vec{\mathsf{td}}') \\ & \quad \xleftarrow{\mathsf{tr}_{\mathtt{IOR}}, \mathsf{tr}_{\mathtt{MT}}} \mathsf{Game}_{\mathtt{IOR}}(\rho, \tilde{\mathbf{P}}[\tilde{\mathcal{P}}_{\mathtt{i}}]^{\rho_{\mathtt{MT}}}) \\ & (\iota, \mathbb{I}) := \mathbf{I}(1^{\lambda}, \mathbb{i}) \\ (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, b, \mathsf{tr}_{\mathtt{IOR}}, \mathsf{tr}_{\mathtt{MT}}, \vec{r}) & (\mathsf{icm}, \mathsf{itd}) := \mathsf{MT.Commit}^{\rho_{\mathtt{MT}}}(\mathbb{I}) \\ \text{conditioned on } E & b_2 := \mathsf{CheckAll}^{\rho_{\mathtt{MT}}}(\mathsf{icm}, \vec{\mathsf{cm}}, \vec{\mathsf{pcm}}, \vec{\mathsf{ans}}, \vec{\mathsf{pf}}) \\ & (\mathbb{x}', \vec{s}) := \mathbf{V}^{\mathbb{I}, \vec{\mathbb{y}}, \vec{\Pi}}(\iota, \mathbb{x}, \vec{r}) \\ & \forall j \in [\mathsf{n}'], \mathbb{y}^*_j := \mathsf{Select}(\vec{\mathbb{y}}, \vec{\Pi}, s_j) \, |_{A_j} \\ & \forall j \in [\mathsf{n}'], \mathsf{cm}'_j := \mathsf{Select}(\vec{\mathsf{cm}}, \vec{\mathsf{pcm}}, s_j) \\ & b_1 := (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}^*, \mathbb{w}') \in \tilde{\mathcal{R}}_2(\mathbb{p}) \\ & b_3 := \bigwedge_{j=1}^{\mathsf{n}'} \mathsf{MT.Verify}^{\rho_{\mathtt{MT}}}(\mathsf{cm}'_j, \mathbb{y}'_j, \mathsf{td}'_j) \\ & b := b_1 \wedge b_2 \wedge b_3 \end{array} \right\}$$

Some rearranging shows that this is identical to the definition of $(Y|E)$. It remains to give an upper bound for the probability of $\overline{E}$. From Lemma 3.11, we get $\kappa_{\mathtt{MT}}(\lambda, t_{\mathtt{MT}}, \mathsf{L}_{\mathtt{max}}, 1 + \mathsf{n} + \mathsf{k})$.

□

B.3 Fiat–Shamir transformation

Lemma B.9. *Let $\mathcal{E}_{\mathtt{i}}$ be an IR state-restoration straightline extractor. There exists an IR state-restoration prover $\tilde{\mathcal{P}}_{\mathtt{i}}[\cdot]$ such that the following holds and a non-interactive reduction straightline extractor $\mathcal{E} := \mathcal{E}[\mathcal{E}_{\mathtt{i}}]$ such that, for every non-interactive reduction prover $\tilde{\mathcal{P}}$ that makes at most $\mathsf{t}_{\mathtt{MT}}$ queries to $\rho_{\mathtt{MT}}$ and $\mathsf{t}_{\mathtt{FS}}$ queries to $\rho_{\mathtt{FS}}$, $\tilde{\mathcal{P}}_{\mathtt{i}}[\tilde{\mathcal{P}}]$ makes at most $t_{\mathtt{FS}}$ moves and the following holds for any set Z:*

$$\begin{aligned} &\Pr\left[\begin{array}{c|l} & \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\lambda) \\ & \rho_{\mathtt{FS}} \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{i}, \bar{\mathbb{x}}) \in Z & \rho := (\rho_{\mathtt{MT}}, \rho_{\mathtt{FS}}) \\ (\mathbb{i}, \bar{\mathbb{x}}, \bar{\mathbb{w}}) \notin \mathsf{Com}[\tilde{\mathcal{R}}]^{\rho_{\mathtt{MT}}} & (\mathbb{i}, \bar{\mathbb{x}}, \pi, \bar{\mathbb{w}}') \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}^{\rho} \\ (\mathbb{i}', \bar{\mathbb{x}}', \bar{\mathbb{w}}') \in \mathsf{Com}[\tilde{\mathcal{R}}']^{\rho_{\mathtt{MT}}} & \bar{\mathbb{w}} := \mathcal{E}(\mathbb{i}, \mathbb{x}, \pi, \mathbb{w}, \mathsf{tr}) \\ & (\mathsf{pk}, \mathsf{vk}, \mathbb{i}') := \mathcal{I}^{\rho}(1^{\lambda}, \mathbb{i}) \\ & \bar{\mathbb{x}}' := \mathcal{V}^{\rho}(\mathsf{vk}, \bar{\mathbb{x}}, \pi) \end{array} \right] \\ \leq &\Pr\left[\begin{array}{c|l} & \rho_{\mathtt{MT}} \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{i}, \bar{\mathbb{x}}) \in Z & \rho_{\mathtt{SR}} \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{i}, \bar{\mathbb{x}}, \bar{\mathbb{w}}) \notin \mathsf{Com}[\tilde{\mathcal{R}}]^{\rho_{\mathtt{MT}}} & (\mathbb{i}, \bar{\mathbb{x}}, \vec{\alpha}, \bar{\mathbb{w}}', \vec{r}) \xleftarrow{\mathsf{tr}} \mathsf{Game}_{\mathtt{IR}}(\rho_{\mathtt{SR}}, \tilde{\mathcal{P}}_{\mathtt{i}}[\tilde{\mathcal{P}}]^{\rho_{\mathtt{MT}}}) \\ (\mathbb{i}', \bar{\mathbb{x}}', \bar{\mathbb{w}}') \in \mathsf{Com}[\tilde{\mathcal{R}}']^{\rho_{\mathtt{MT}}} & \bar{\mathbb{w}} := \mathcal{E}_{\mathtt{i}}(\mathbb{i}, \bar{\mathbb{x}}, \vec{\alpha}, \bar{\mathbb{w}}', \mathsf{tr}) \\ & (\mathsf{pk}, \mathsf{vk}, \mathbb{i}') := \mathcal{I}_{\mathtt{i}}(1^{\lambda}, \mathbb{i}) \\ & \bar{\mathbb{x}}' := \mathcal{V}_{\mathtt{i}}^{\rho_{\mathtt{MT}}}(\mathsf{vk}, \bar{\mathbb{x}}, \vec{\alpha}, \vec{r}) \end{array} \right] + \frac{t_{\$}^2}{2^{\lambda}}. \end{aligned} \tag{B.6}$$

Proof. This is a straightforward adaptation of [CY24, Theorem 16.1.1], which transforms interactive proofs (or arguments) into non-interactive arguments. Observe that any reduction can be viewed as an argument where the prover additionally sends the new witness and the verifier additionally tests membership in the new relation. From this perspective, the experiment on the right side of Equation (B.6) is equivalent to the state-restoration experiment for an interactive argument, and the experiment on the left side is equivalent to the knowledge soundness experiment for the Fiat–Shamir transformation of the interactive argument.

There are a few minor technical differences which we address here. First, we give the interactive prover $\tilde{\mathcal{P}}_{\mathtt{i}}$ access to a random oracle $\rho_{\mathtt{MT}}$; this is justified since the reduction only makes black-box use of $\tilde{\mathcal{P}}_{\mathtt{i}}$. Second, the relations $\mathcal{R}, \mathcal{R}'$ are defined relative to $\rho_{\mathtt{MT}}$ and we additionally test membership in an arbitrary set Z; this is justified since the distributions of index, instance, and witness in the experiments are statistically close [CY24, Lemma 16.3.3]. □