Beyond Standard Indifferentiability: Simulator Properties for Fiat-Shamir Security

Ashrujit Ghoshal[1] and Sikhar Patranabis[2]

[1] IIT Madras
ashrujit@cse.iitm.ac.in
[2] IBM Research India
sikhar.patranabis@ibm.com

Abstract. The Fiat-Shamir transform converts public-coin interactive arguments into non-interactive arguments (NARGs). Its security is typically analyzed in the random-oracle model, while implementations instantiate the random oracle with structured hash constructions. Standard indifferentiability is the natural abstraction for justifying this replacement. Chiesa and Orrù (TCC '25), however, showed that standard indifferentiability does not generally preserve the information and reset behavior needed by common Fiat-Shamir extraction arguments, and proposed stronger notions of indifferentiability to bridge this gap.

We show that standard indifferentiability can nevertheless be retained as the oracle-replacement abstraction. For knowledge soundness, we give generic transfer theorems for three extraction interfaces: proof-only straight-line extraction follows from standard indifferentiability alone; transcript-dependent extraction additionally requires *transcript reconstruction*; and rewinding extraction requires *reconstruction for rewinding*, preserving simulator behavior across invocations with the same hidden randomness. Our results apply to worst-case polynomial-time extractors. Moreover, we show that the rewinding requirement is inherent: assuming discrete-logarithm hardness, we construct a perfectly indifferentiable construction and fixed prover admitting error-free rewinding extraction in the ideal world, while every polynomial-time construction-world extractor succeeds only with negligible probability.

We instantiate our framework for tagged prefix-free Merkle-Damgård and a fixed-output permutation sponge. The former admits exact transcript and rewinding reconstruction, while the latter admits exact transcript reconstruction and rewinding reconstruction with negligible statistical error incurred only once, independently of the number of restarts.

For zero knowledge, we give a semantic formulation where one persistent simulator jointly generates the proof and public-oracle interaction. Here, *standard indifferentiability alone* yields a generic transfer theorem, requiring neither reconstruction property and incurring only the indifferentiability error. Thus standard indifferentiability provides a common oracle-replacement abstraction for knowledge soundness and semantic zero knowledge, with extraction-specific requirements isolated as separate simulator properties.

Table of Contents

Beyond Standard Indifferentiability: Simulator Properties for Fiat-Shamir Security 1
Ashrujit Ghoshal and Sikhar Patranabis

1 Introduction 3
- 1.1 Our Contributions 4
- 1.2 Paper Organization 6

2 Technical Overview 6
- 2.1 Knowledge Soundness 6
- 2.2 Why Standard Indifferentiability Alone is Insufficient 7
- 2.3 Tagged Prefix-Free Merkle-Damgård 8
- 2.4 Fixed-Output Permutation Sponge 9
- 2.5 Semantic Zero Knowledge 10

3 Preliminaries 10

4 Straight-line Knowledge Soundness 12
- 4.1 Extraction without the Prover Transcript 13
- 4.2 Extraction with the Prover Transcript 14

5 Rewinding Knowledge Soundness 17

6 Tagged Prefix-Free Merkle-Damgård 23
- 6.1 Proof of Indifferentiability 24

7 Fixed-Output Sponge from a Random Permutation 28
- 7.1 Construction and Primitive Interface 28
- 7.2 The Permutation Simulator 29
- 7.3 Proof of Indifferentiability 31
- 7.4 Exact Transcript Reconstruction 32
- 7.5 Reconstruction for Rewinding 33
- 7.6 Consequences 34

8 Semantic Zero Knowledge 34
- 8.1 Definition 34
- 8.2 Transfer through an Indifferentiable Construction 35

1 Introduction

Non-interactive arguments (NARGs) allow a prover to convince a verifier of the validity of a statement without interaction. For a relation R, a proof for a statement x certifies that there exists a witness w such that $R(x, w) = 1$. A basic security requirement is *soundness*, which ensures that a cheating prover cannot convince the verifier of a false statement. Many applications require a NARG to satisfy the stronger guarantee of *knowledge soundness*: whenever a prover produces an accepting proof, an efficient extractor should be able to recover a corresponding witness, except with a suitably small error. Such a NARG is called a NARG of Knowledge (or a NARK). Applications involving private data additionally require *zero knowledge* (ZK): the proof should reveal nothing about the witness beyond the validity of the statement.

Designing NARKs. A standard route to constructing such arguments is the Fiat-Shamir transform [FS87]. Starting from a public-coin interactive proof or argument, the transform replaces the verifier's random challenges with hashes of the statement and the preceding protocol transcript, thereby removing interaction. Security of the resulting non-interactive argument is typically analyzed in the Random Oracle Model (ROM), where the hash function is modeled as a random oracle H. This methodology underlies constructions ranging from digital signatures [Sch91, Lyu09, DKL+20] to modern succinct NARKs [PHGR13, GGPR13, BCG+18, MBKM19, GWC19, XZZ+19, GW20, CHM+20, Set20, BCHO22, EFG22, CBBZ23].

From the Random Oracle to a Hash Construction. The ROM treats H as a monolithic variable-input-length oracle, whereas an implementation uses a structured hash function built from a lower-level primitive. Abstractly, let $\pi \xleftarrow{\$} \mathcal{I}'$ denote this primitive and let C^π denote the resulting hash construction. For example, Merkle-Damgård constructions are built from compression functions [Mer90, Dam90, CDMP05], whereas sponge constructions are built from permutations [BDPV08, BDPV12].

Thus the security proof is carried out relative to the ideal oracle H, whereas the implementation uses C^π. Moreover, a cheating prover may query the underlying primitive π directly, while the honest proof system accesses it only through C^π. This raises the central composition question:

When does security proved relative to H survive the replacement of H by the construction C^π?

Bridging the Gap via Indifferentiability. The indifferentiability framework of Maurer, Renner, and Holenstein [MRH04] provides a natural way to address this oracle-replacement question. Informally, C^π is indifferentiable from H if there exists an efficient simulator S such that no efficient distinguisher can tell apart the real pair (π, C^π) from the ideal pair (S^H, H). The simulator resolves the mismatch between the two worlds: using H, it emulates the primitive interface π that is available to the adversary in the real world. Suitable Merkle-Damgård variants and sponge constructions are known to satisfy such standard indifferentiability guarantees [CDMP05, BDPV08].

This suggests a clean modular approach to Fiat-Shamir security. One would first prove the desired security property for the NARG relative to the random oracle H, and separately prove that the chosen hash construction C^π is indifferentiable from H. If these two guarantees composed generically, the security analysis of the NARG could remain independent of the concrete hash construction used to instantiate its random oracle.

Barrier for Knowledge Soundness. For knowledge soundness, standard indifferentiability alone does not provide such a generic composition theorem. This is especially relevant given that recent CFRG guidance on Fiat-Shamir instantiation calls for *extraction-friendly* and *simulation-friendly* notions to preserve soundness and zero knowledge, respectively [Orr26]. In a recent work, Chiesa and Orrù [CO25] formally identify the barrier for knowledge soundness. After replacing the primitive π by the indifferentiability simulator S^H, a cheating prover P' becomes the composed ideal-world prover $\widehat{P}^H = P'^{\mathsf{S}^H}$. A knowledge extractor for $\widehat{P}$ may require more than its input-output behavior. In particular, it may be given the prover's oracle transcript, or it may rerun the prover with the same randomness while answering its oracle queries differently.

Both forms of access now involve the indifferentiability simulator. The queries of $\widehat{P}$ to H are made internally by S while answering P', and are not in general visible in the external interaction between P' and S^H. Moreover, if an extractor rewinds $\widehat{P}$, then the private randomness of S is part of the fixed randomness of the composed prover being rewound. Replacing it by a fresh simulator execution need not reproduce the prover that generated the original proof.

Thus the difficulty is not that standard indifferentiability fails to replace the oracle interface. Rather, extraction may require *additional information or access to the indifferentiability simulator* that the standard indifferentiability experiment does not expose. This leads to our first question:

Can these missing extraction capabilities be recovered while retaining standard indifferentiability as the oracle-replacement notion?

Barrier for Zero Knowledge. Zero knowledge raises a different issue. ROM zero-knowledge simulators often *program* the random oracle, choosing its answers at particular points as a function of the simulated proof. Chiesa and Orrù [CO25] observe that standard indifferentiability does not provide a corresponding interface for translating such programming into modifications of the underlying primitive. Thus the same modular replacement argument does not directly apply to their ZK formulation. This leads to a second question:

Is this incompatibility with standard indifferentiability inherent to zero knowledge, or does it depend on how the ZK simulation requirement is formulated?

1.1 Our Contributions

Our results give a modular answer to both questions above. For knowledge soundness, we retain standard indifferentiability and isolate the additional capabilities required by extraction as explicit properties of its simulator. We show that the additional requirement depends precisely on the access given to the extractor, and that standard indifferentiability alone is insufficient in general for rewinding extraction. For zero knowledge, we give a semantic formulation under which no additional simulator property is needed: standard indifferentiability alone suffices.

Knowledge-soundness transfer from standard indifferentiability. We distinguish three extraction interfaces. A *proof-only straight-line* extractor receives the statement, proof, and verifier's oracle transcript, but neither the prover's oracle transcript nor access to further prover executions. A *transcript-dependent straight-line* extractor additionally receives the prover's oracle transcript. A *rewinding* extractor may furthermore restart the prover with its original random tape while answering its oracle queries differently.

For the latter two interfaces, we introduce two properties of a standard indifferentiability simulator. *Transcript reconstruction* recovers the simulator's internal H-query transcript from the externally visible primitive interaction, in a manner consistent with the complete extraction experiment. *Reconstruction for rewinding* additionally reproduces later executions of the simulator under adaptively chosen oracle answers, while preserving their dependence on the fixed simulator randomness used in the initial execution. Our generic transfer theorems give the following hierarchy.

Informal Theorem 1.1 (Modular Knowledge-Soundness Transfer) *Let C be a stateless polynomial-time construction such that C^π is indifferentiable from an ideal oracle H via a simulator S. Then replacing H by C^π preserves:*

1. *proof-only straight-line knowledge soundness, with no additional condition on the simulator S;*
2. *transcript-dependent straight-line knowledge soundness, if S admits transcript reconstruction; and*
3. *rewinding knowledge soundness, if S admits reconstruction for rewinding.*

The resulting extraction error is at most $(\varepsilon_{\text{ideal}} + \varepsilon_{\text{ind}} + \varepsilon_{\text{rec}})$ where the three terms are respectively the ideal-model extraction error, the standard-indifferentiability error, and the applicable reconstruction error; $\varepsilon_{\text{rec}} = 0$ in the first case. All extractors in these results have worst-case polynomial running time.

See Theorems 1 to 3 and Corollaries 1 to 3 for the formal statements. The additional assumption on the simulator thus grows with the additional access supplied to the extractor.

Standard indifferentiability alone is insufficient. The reconstruction requirement is not just an artifact of our transfer proof. Assuming the hardness of discrete logarithms, we construct an ideal primitive, a construction that is *perfectly* indifferentiable from a random oracle, and a fixed cheating prover for which the corresponding ideal-world prover admits polynomial-time rewinding extraction with zero failure probability. Nevertheless, every polynomial-time extractor in the construction world recovers a witness only with negligible probability.

The separation uses Fiat-Shamir Schnorr proofs [Sch91]. Its main feature is that the hidden exponent belongs to the simulator's private randomness in the ideal experiment, and hence to the resettable composed prover, whereas in the construction experiment it belongs to the primitive itself. Thus identical external oracle distributions need not give a rewinding extractor the same capabilities. In particular, standard indifferentiability does not imply reconstruction for rewinding. See Lemma 2 and Theorem 4.

Tagged Prefix-Free Merkle-Damgård. We give a simulator for a tagged prefix-free construction whose external compression-query transcript determines both its internal hash-query transcript and every block of private randomness consumed during the initial execution. Replaying this transcript therefore gives exact transcript reconstruction and exact reconstruction for rewinding. These reconstruction guarantees hold for the simulator's actual execution, including executions with collisions or out-of-order queries; the associated bad-event loss occurs only in the indifferentiability proof. Our result is captured by the following (informal) theorem.

Informal Theorem 1.2 (Tagged Prefix-Free Merkle-Damgård) *Replacing the random oracle in a NARG by tagged prefix-free Merkle-Damgård, built from an ideal compression function, preserves knowledge soundness under each of the three extraction interfaces considered in this work, provided the standard indifferentiability error is negligible. Worst-case polynomial-time extraction is preserved, and the only additional security loss is the construction's indifferentiability error: supporting transcript access and rewinding introduces no further loss.*

See Theorems 5 to 7 and Corollary 4 for a more formal exposition.

Fixed-Output Permutation Sponge. We also give a simulator for the stateless sponge that returns one output block and permits adversarial access to both directions of the underlying permutation. Its internal hash transcript is again reconstructed exactly. Rewinding requires more care: permutation constraints cause the simulator to sample from sets that depend on earlier answers. The visible sampling outcomes need not determine the random bits reused after a reset. Our bounded-randomness implementation and conditional sampler (Lemma 3) reconstruct their conditional distribution with negligible statistical error. Our result is captured by the following (informal) theorem (see Theorems 8 to 10 and Corollary 5 for a more formal exposition).

Informal Theorem 1.3 (Fixed-Output Permutation Sponge) *Replacing the random oracle in a NARG by a fixed-output sponge, built from an ideal permutation, preserves knowledge soundness under each of the three extraction interfaces considered here, provided the standard indifferentiability error is negligible. This holds even when the adversary can query the permutation in both directions, and preserves worst-case polynomial-time extraction. Straight-line extraction incurs only the indifferentiability loss; rewinding adds a negligible statistical error that is incurred once for the entire extraction, independently of the number of restarts.*

Our results supply concrete simulator analyses for both compression-based and permutation-based hashing, with guarantees that can be used across NARGs satisfying the corresponding ideal-model extraction premise.

Semantic Zero Knowledge. Finally, we show that the apparent obstruction for ZK is not inherent to standard indifferentiability. We formulate ZK semantically: the simulator jointly produces the simulated proof and answers the adversary's oracle queries, and security requires that the adversary's resulting view be computationally indistinguishable from its view in the real execution. The definition does not prescribe that the simulated oracle be represented as an independently sampled random oracle together with an explicit set of programmed points.

Informal Theorem 1.4 (Semantic Zero Knowledge) *Let C be a stateless polynomial-time construction such that C^π is indifferentiable from an ideal oracle H via S. If a NARG is semantically zero knowledge relative to H via a polynomial-time simulator Z, then its C^π instantiation is semantically zero knowledge via a polynomial-time simulator obtained by composing S and Z. The resulting ZK distinguishing advantage is at most $\varepsilon_{\mathrm{zk,ideal}} + \varepsilon_{\mathrm{ind}}$. No transcript- or rewinding-reconstruction property is required. In particular, semantic ZK is preserved for tagged prefix-free Merkle-Damgård and the fixed-output permutation sponge whenever their standard indifferentiability premises hold.*

The composition in the above theorem is direct. Whenever the indifferentiability simulator S needs an answer from the ideal random oracle, that query is answered by the ROM ZK simulator Z. The same execution of S is retained before and after the simulated proof is produced. Thus the resulting simulation preserves the adversary's entire interaction, and the only additional distinguishing loss is the standard-indifferentiability error.

Scope and Relation to Prior Work. The limitations of standard indifferentiability as a composition tool have been studied beyond the Fiat-Shamir setting. Ristenpart, Shacham, and Shrimpton [RSS11] identified limitations for multistage security and introduced reset indifferentiability.

Our results advance the line of work seeking rigorous foundations for replacing the random oracle in Fiat-Shamir-based NARGs by structured hash constructions. Prior works establish security either directly for particular constructions and oracle models, or through strengthened notions exposing the additional capabilities required by extraction or simulation. We take a different route: we retain standard indifferentiability as the common abstraction for random-oracle replacement, and characterize separately what, if anything, its simulator additionally requires for the security property being transferred. This yields a unified modular picture covering knowledge soundness under different access patterns and semantic zero knowledge, with concrete realizations for both compression-function-based and permutation-based hashing.

Concretely, our modular treatment of Fiat-Shamir-transformed NARGs differs conceptually from [CO25]. Rather than strengthening indifferentiability itself to encode the additional interfaces required by extraction, we isolate the missing extraction capabilities as explicit reconstruction properties of the simulator. This separation handles both transcript-dependent and rewinding extraction while retaining a common indifferentiability framework across hash constructions. For zero knowledge, under our semantic formulation, no extraction-specific strengthening, programming-aware variant of indifferentiability, or construction-specific ZK analysis is needed: standard indifferentiability alone yields a generic transfer theorem. Our analyses of tagged prefix-free Merkle-Damgård and fixed-output permutation sponges can thus be viewed as concrete instantiations of a broader modular methodology, rather than security arguments tied to a particular Fiat-Shamir realization or hash construction.

1.2 Paper Organization

The rest of the paper is organized as follows. We present an overview of our techniques in Section 2. In Section 3, we introduce the necessary preliminaries. We present our results for straight-line and rewinding knowledge soundness in Section 4 and Section 5, respectively. We then turn to concrete hash constructions. In Section 6, we show that our tagged prefix-free Merkle-Damgård construction satisfies both reconstruction properties required by our knowledge-soundness transfer theorems, and in Section 7, we establish the corresponding result for our sponge construction. Section 8 presents our result on semantic zero knowledge.

2 Technical Overview

Our work addresses two tasks that are easy to conflate. Standard indifferentiability is an *oracle-replacement* statement: it lets us replace the real pair (π, C^π) by the ideal pair (S^H, H). For a Fiat-Shamir NARG, however, the security proof may use more than the input-output behavior of the cheating prover. A knowledge extractor may need to see the prover's random-oracle transcript, or it may need to run the same prover again with the same randomness. After the indifferentiability replacement, both of these capabilities pass through the simulator S. The main technical point of our knowledge-soundness results is to isolate exactly what extra information about S is needed, instead of strengthening indifferentiability itself.

Our zero-knowledge (ZK) result has a different flavor. There is no extractor to rewind and no hidden prover transcript to recover. Under our semantic formulation of ZK, the ROM simulator can simply be composed with one persistent execution of the indifferentiability simulator. Thus standard indifferentiability alone suffices.

We now present an overview of these two results, and then describe why the required reconstruction properties hold for tagged prefix-free Merkle-Damgård and a fixed-output permutation sponge. Throughout the discussion of our knowledge-soundness results, extractors have worst-case polynomial running time, including the total work over all prover invocations. We leave it as an interesting open question to realize the corresponding transfer statement for extractors that are only expected polynomial time.

2.1 Knowledge Soundness

Fix a cheating prover $\mathcal{P}'$ that can query the primitive π directly. Let S be a standard indifferentiability simulator for C^π. Replacing the real pair (π, C^π) by the ideal pair (S^H, H) turns $\mathcal{P}'$ into the ordinary

random-oracle prover $\widehat{\mathcal{P}}^{H} = \mathcal{P}'^{\mathsf{S}^{H}}$. The ideal-model knowledge-soundness theorem applies to this *composed* prover. This observation is the source of both our positive transfer theorems and our separation: whatever access the ideal extractor has to $\widehat{\mathcal{P}}$ must somehow be reproduced in the construction world.

Proof-only Straight-Line Extraction. Suppose the extractor receives the statement, the proof, and the verifier's oracle transcript, but it does not receive the prover's oracle transcript and never invokes the prover again. Then there is nothing inside S that the reduction has to expose. One indifferentiability distinguisher can run the prover, verifier, and extractor and directly compare the construction experiment with the ideal random-oracle experiment for $\widehat{\mathcal{P}}$. Hence standard indifferentiability by itself transfers this form of knowledge soundness.

Transcript-Dependent Straight-Line Extraction. If the extractor also receives the prover's oracle transcript, the replacement creates two transcripts:

$$\tau_{\text{ext}} = \tau(P' \leftrightarrow S^{H}) \quad \text{and} \quad \tau_{\text{int}} = \tau(S \leftrightarrow H).$$

Only τ_{ext} is externally visible, whereas the ideal extractor expects τ_{int}. This motivates *transcript reconstruction*: from τ_{ext}, a reconstructor produces

$$\widehat{\tau}_{\text{int}} \leftarrow \mathsf{Rec}^{H}(\tau_{\text{ext}})$$

that can replace τ_{int} in the complete extraction experiment. This is a joint requirement: the reconstructed transcript must remain consistent with the generated proof, the shared state of H, and subsequent oracle queries, rather than merely have the correct marginal distribution. For both concrete simulators considered here, we prove the stronger property that τ_{ext} determines τ_{int} exactly, without querying H.

Rewinding Extraction. Rewinding introduces a stronger requirement. Fixing the tapes ρ of P' and ω of S makes the ideal prover

$$\widehat{P}^{H}[\rho, \omega] = P'[\rho]^{S[\omega]^{H}}.$$

If the extractor later invokes this prover with another oracle K, it must obtain $P'[\rho]^{S[\omega]^{K}}$: the reset therefore reuses both ρ and ω. A fresh execution of S would generally define a different fixed-randomness prover.

Our notion of *reconstruction for rewinding* supplies precisely this missing access. From the initial visible transcript, the reconstructor recovers the required internal transcript and maintains persistent state from which it answers all later simulator invocations under adaptively chosen oracles. Crucially, these invocations must jointly reproduce the dependence induced by one fixed simulator tape ω; correct marginal distributions for individual restarts do not suffice. The tape itself need not be uniquely determined: it is enough to sample once from its correct conditional distribution given the initial execution and reuse that sample thereafter. This distinction is essential for our sponge simulator.

The Three Transfer Theorems. With these interfaces separated, the reductions are easy. Proof-only extraction uses one standard-indifferentiability replacement. For transcript-dependent extraction, we first make the same replacement and then replace the reconstructed internal transcript by the true one. For rewinding, we first make the indifferentiability replacement while implementing all later simulator sessions through the rewinding reconstructor, and then replace this reconstructed access by the true fixed-randomness simulator access.

Consequently, the construction-world extraction error is bounded by the ideal-model extraction error plus the standard-indifferentiability error plus the relevant reconstruction error. The reconstruction term is absent for proof-only extraction. Importantly, the indifferentiability distinguisher itself never needs a resettable-simulator interface: standard indifferentiability is used only for the ordinary oracle replacement, while the extra access needed by extraction is handled separately by reconstruction. This is the modularity we want.

See Theorems 1 to 3 and Corollaries 1 to 3.

2.2 Why Standard Indifferentiability Alone is Insufficient

The rewinding-reconstruction requirement is not merely a proof artifact: we give a perfectly indifferentiable construction for which rewinding extraction is trivial in the ideal world but would solve discrete logarithms in the construction world.

Let $\mathbb{G} = \langle g \rangle$ have prime order p. The primitive samples essentially uniform $w, r \in \mathbb{Z}_p$ (formally, from the negligibly-close distribution μ_p of Section 5) and, besides a hash interface, exposes (Y, A) and z where

$$Y = g^{w}, \quad A = g^{r}, \quad z = r + cw, \quad c = H(\langle Y, A \rangle).$$

The construction C^π exposes only the hash interface. Its simulator samples w, r identically and implements the auxiliary interfaces by the same formulas, making (π, C^π) and (S^H, H) identically distributed and hence perfectly indifferentiable.

Consider the cheating prover that obtains (Y, A, z) from the primitive and outputs the Fiat–Shamir Schnorr proof (A, z) for Y. In the ideal world, (w, r) belong to the simulator's fixed random tape. A rewinding extractor invokes the same prover again with the challenge at $\langle Y, A\rangle$ changed to $(c+1)$ and, reusing the same simulator tape, obtains

$$z' = r + (c+1)w,$$

thus recovering $w = (z' - z)$. All challenge changes and exponent arithmetic here are modulo p.

In the construction world, however, (w, r) belong to the sampled primitive rather than the resettable prover, so rewinding P' does not yield a second response using the same hidden values under a changed challenge. Formally, given a discrete-logarithm challenge $Y = g^w$, the reduction chooses c, z, sets $A = g^z Y^{-c}$, sets $H(\langle Y, A\rangle) = c$, and exposes (Y, A, z) through the auxiliary interfaces. Writing $r = (z - cw)$ for analysis gives

$$A = g^r, \quad z = (r + cw),$$

without requiring the reduction to know w. Thus any construction-world extractor recovering a witness with non-negligible probability solves discrete logarithms; the negligible deviation of μ_p from uniform is accounted for in the formal proof.

Thus even *perfect* standard indifferentiability does not imply a generic rewinding-transfer theorem. The separation also shows, in a direct sense, that the perfectly indifferentiable simulator in this example cannot admit our reconstruction-for-rewinding property. See Lemma 2 and Theorem 4.

2.3 Tagged Prefix-Free Merkle-Damgård

We next show that the reconstruction requirements are not artificial: a natural Merkle-Damgård-style construction satisfies them in their strongest form.

The message encoding consists of tagged blocks $((B_1, 0), \ldots, (B_{\ell-1}, 0), (B_\ell, 1))$, where tag 0 marks a strict encoded prefix and tag 1 marks the final block. The construction iterates a random compression function $F(u, B, t) \in \{0,1\}^n$ from a fixed initial state.

Recognizing a Completed Message. The simulator tracks visible compression-function states representing strict encoded prefixes. A fresh tag-0 query extending such a prefix receives a random n-bit answer and, in the absence of a conflict with an already represented state, records the extended prefix. A fresh tag-1 query completing the encoding of a message m returns $H(m)$. Other fresh queries receive random answers, while repeated queries are answered consistently.

Recovering the Hidden H-transcript without Querying H. The crucial observation is that every fresh $H(m)$ queried by the simulator is used directly as the answer y_i to the primitive query that completes the encoding of m. Hence the corresponding hidden transcript entry

$$(m, H(m)) = (m, y_i)$$

is already exposed in the external transcript. Replaying that transcript with the simulator's deterministic update rules identifies exactly when each such query occurred and the corresponding m. Thus the complete internal H-transcript is recovered exactly, without querying H; see Theorem 6.

Why Rewinding Reconstruction is also Exact. We implement every private random answer of the simulator by reading the next n-bit block of a single ordinary random tape. The whole block is returned as the primitive answer. Hence the external transcript reveals the exact tape block that was consumed. Replay therefore reconstructs the entire prefix of the simulator tape used in the initial execution. The unread suffix is sampled once and retained. Every later simulator session resets its local tables and tape head but reuses this same persistent tape, exactly as the true fixed-randomness simulator does. Consequently reconstruction for rewinding also has error zero; see Theorem 7.

Indifferentiability is Proved Separately. Exact replay does not mean that the construction is perfectly indifferentiable. The indifferentiability coupling can fail if two represented strict prefixes collide, if a nonempty prefix returns to the initial state, or if the distinguisher exposes a state before the represented prefix leading to it has been revealed in order. Bounding these events gives

$$\mathsf{Adv}^{\mathsf{indiff}} \leq O\left(\frac{Q^2}{2^n}\right)$$

in Theorem 5, where Q counts both direct compression queries and compression calls made while answering hash queries. Importantly, these bad events affect only the indifferentiability comparison, not reconstruction itself. Hence all three extraction interfaces transfer with only the standard indifferentiability loss; see Corollary 4.

2.4 Fixed-Output Permutation Sponge

For the fixed-output sponge, transcript reconstruction remains exact, but rewinding is subtler: the simulator samples from admissible sets determined by previously fixed permutation values, and these sets may change under a different oracle after rewinding. We must therefore reconstruct not merely the initial choices, but one persistent random tape governing all later invocations.

Let P be a random permutation on $r + c$ bits, with states $(a, d) \in \{0,1\}^r \times \{0,1\}^c$. For $\mathsf{pad}(m) = B_1 \cdots B_\ell$, the sponge starts from 0^{r+c}, iteratively applies $P(a \oplus B_i, d)$, and outputs the first r bits of the final state. The adversary may query both P and P^{-1}.

Recognizing Sponge Computations. The simulator records fixed permutation values and the sponge computations they determine. From a represented state (α, d), a fresh forward query (a, d) uniquely identifies the next absorbed block $B = \alpha \oplus a$. If this completes $\mathsf{pad}(m)$, the simulator uses $H(m)$ as the first r output bits and samples the remainder from an *admissible set* preserving permutation consistency. Fresh inverse queries choose preimages whose capacity part cannot participate in a represented sponge computation, and therefore require no query to H. See Section 7.2 for the full simulator and its invariant.

Exact Transcript Reconstruction. The external transcript determines every permutation pair fixed during the original execution, regardless of whether it arose from a forward primitive query or an inverse primitive query. Replaying these pairs in order therefore identifies exactly when a forward query first completes $\mathsf{pad}(m)$. Whenever this causes a fresh query $H(m)$, the simulator sets the first r bits b of the corresponding permutation output equal to the returned value $H(m)$. Thus $b = H(m)$ is visible in the external permutation transcript, and the hidden oracle-query transcript entry

$$(m, H(m)) = (m, b)$$

can be recovered directly. Hence the complete internal H-transcript is reconstructed exactly, without querying H; see Theorem 9.

Why Rewinding is More Delicate. Exact transcript reconstruction determines what happened in the initial execution, but rewinding must reproduce the simulator in later executions using the same randomness. Suppose that at some point the simulator chooses uniformly from an admissible set $\{v_0, \ldots, v_{M-1}\}$. In a later invocation, the oracle supplied to the simulator may return different answers. The permutation values fixed before the corresponding choice may therefore differ, and the admissible set may also differ. Remembering only the value selected in the initial execution is consequently insufficient. Even if that value remains admissible, forcing the simulator to select it again need not reproduce the action of the same random tape on the new admissible set.

We therefore implement every private choice using an explicit fixed-length block of the simulator's random tape. Set $\kappa = 2\lambda$. For an ordered admissible set $\{v_0, \ldots, v_{M-1}\}$, the simulator reads a fresh tape block, interprets it as an integer R, and returns $v_{R \bmod M}$. The block length is chosen so that, for every possible M, the distribution of $R \bmod M$ is within statistical distance $2^{-\kappa}$ of uniform on $\{0, \ldots, M-1\}$. Thus this fixed-tape implementation is statistically close to exact uniform sampling, while its behavior in every later invocation is determined by one persistent random tape.

Reconstructing the Persistent Tape. For the distributional analysis, condition on H, on the external algorithm's random tape, and on the initial external transcript. The reconstructor itself still receives only the information specified by the reconstruction experiment. For every private choice made in the initial execution, the external transcript determines the admissible set used at that point and the element returned by the simulator. Suppose the jth such set has size M_j and the returned element has index a_j. If R_j denotes the tape block used for this choice, then the observed execution imposes exactly the constraint

$$R_j \equiv a_j \pmod{M_j}.$$

Different private choices use different tape blocks, and under the above conditioning these constraints concern distinct independent blocks. The conditional distributions of the consumed blocks therefore factor and can be sampled separately.

By Lemma 3, the conditional distribution of each consumed block can be sampled in worst-case polynomial time up to statistical distance $2^{-\kappa}$. Initialization samples these blocks and stores them at

their original tape positions. Tape positions not used in the initial execution remain fresh uniform blocks and are sampled only if a later invocation reaches them. All later reconstructed simulator invocations use this same persistent tape. Thus, if the block R_j is used later when the current admissible set has size M', the simulator selects the element indexed by $R_j \bmod M'$, exactly as the original fixed-tape simulator would when run with the same randomness.

If the initial execution makes at most q_A primitive queries, at most $2q_A$ private tape blocks are consumed. The statistical distance introduced when reconstructing all consumed blocks is therefore at most $2q_A 2^{-\kappa}$. This loss is incurred once, during initialization. Conditioned on the reconstructed tape agreeing with the true conditional tape, all later invocations use exactly the same randomness, including adaptively chosen rewinding invocations. Hence there is no additional reconstruction error per restart. This gives Theorem 10.

Indifferentiability of the Fixed-Tape Simulator. We now show that the same fixed-tape simulator satisfies standard indifferentiability. Consider first the corresponding simulator that samples exactly uniformly from each admissible set. If the distinguisher makes q_{perm} direct forward or inverse permutation queries and its hash queries absorb a total of $\sum_i \ell_i$ padded blocks, let $N = q_{\mathrm{perm}} + \sum_i \ell_i$. The standard permutation-sponge analysis gives an indifferentiability error

$$f_{\mathrm{sp}}(N) = O(N^2/2^c)$$

for this exact-uniform simulator in the relevant parameter range.

Replacing an exact-uniform private choice by the fixed-tape rule changes its distribution by at most $2^{-\kappa}$. Since a direct primitive query causes at most two such private choices, the resulting indifferentiability bound is

$$\mathsf{Adv}^{\mathsf{indiff}} \leq f_{\mathrm{sp}}(N) + 2q_{\mathrm{perm}}2^{-\kappa}.$$

See Theorem 8. The two additional terms arise in different comparisons. The term $2q_{\mathrm{perm}}2^{-\kappa}$ is incurred when exact uniform sampling is replaced by the fixed-tape implementation in the indifferentiability proof. The term $2q_A 2^{-\kappa}$ is incurred when the persistent tape is reconstructed from the initial execution. With $\kappa = 2\lambda$, both are negligible for polynomial query bounds. Together with exact transcript reconstruction, this establishes the properties needed to transfer all three knowledge-soundness interfaces to the fixed-output permutation sponge; see Corollary 5.

2.5 Semantic Zero Knowledge

Finally, for ZK, we bypass the obstruction motivating reconstruction via a semantic formulation. Rather than prescribing how the simulator programs or represents a random oracle (as done in [CO25]), we require only that it jointly simulate the proof and the adversary's complete public-oracle view. Formally, our simulator Z maintains one persistent state supporting both oracle queries and proof generation; in particular, producing the proof may affect subsequent simulated oracle answers. We give the formal definition in Section 8.1.

This formulation composes directly with standard indifferentiability. Given a ROM simulator Z and an indifferentiability simulator S, the construction-world simulator maintains one persistent execution of each. Primitive queries are answered by S, whose ideal-oracle queries are forwarded to Z; proof requests are answered by Z on the same persistent state. After proof generation, both executions simply continue chronologically. Thus S is never reset, rewound, or reconstructed. The resulting hybrid argument bounds the construction-world ZK advantage by the ideal-world ZK advantage plus the standard indifferentiability error. Hence standard indifferentiability alone suffices to preserve semantic zero knowledge: neither transcript nor rewinding reconstruction is required, and the only additional loss is the indifferentiability error. Section 8.2 gives the formal transfer theorem and its proof.

3 Preliminaries

Oracle Executions and Transcripts. The security parameter is λ. Unless explicitly stated otherwise, all algorithms are probabilistic polynomial time with worst-case polynomial running time. In particular, this requirement applies to every extractor in the knowledge-soundness definitions and transfer theorems below. Running-time bounds are polynomial in λ and the explicit input lengths, and may depend on any stated prover-running-time and query bounds.

For an extractor, it bounds the extractor's local computation, the number of prover invocations, and the lengths of its outputs and queries. One polynomial, fixed with the extractor, also bounds the

total work of all oracle algorithms it supplies in terms of λ, the extractor's input length, and their total interaction length. Calls to ideal oracles have unit cost. When proving a reduction efficient, we count the full computation of all prover invocations and all supplied oracle algorithms, as well as transcript handling and the test of whether the extracted witness is valid.

Remark 1 (Expected-time Extractors). Lemma 1 extends an indistinguishability guarantee against polynomial-time tests to a test with polynomial expected running time in both compared experiments. This concerns the difference in output probabilities; it does not show that an extractor remains expected polynomial time after its oracle or invocation interface is replaced.

For example, consider an algorithm that asks an oracle for a bit. If the answer is 1, it performs $2^{2\lambda}$ dummy steps. It then outputs 0, regardless of the answer. With an oracle that always answers 0, the algorithm finishes immediately. Now replace the oracle by one that always answers 1 with probability $2^{-\lambda}$, and otherwise always answers 0. The two oracles behave identically except with probability $2^{-\lambda}$, so no test can distinguish them with advantage greater than $2^{-\lambda}$. Nevertheless, the algorithm's expected running time after the replacement is at least $2^{-\lambda} \cdot 2^{2\lambda} = 2^{\lambda}$. Its output is still 0 in both experiments. Thus a negligible change in oracle behavior can make expected running time exponential, even without changing the output. We therefore restrict the transfer results to worst-case polynomial-time extractors and do not claim an extension to extractors that are only expected polynomial time.

An execution of an oracle begins with the oracle in its initial state and with one random tape. Its state and its position on that tape persist while it answers the queries made during that execution. Unless an experiment explicitly creates a fresh execution or performs a reset, every occurrence of the same oracle symbol refers to this one execution. In particular, a lazily sampled table is shared by all calls to that oracle in the experiment.

For an algorithm A interacting with an oracle O, let $\tau(A \leftrightarrow O)$ be the ordered sequence of query-answer pairs in that interaction. Repeated queries remain in the sequence. An internal simulator transcript includes any underlying-oracle queries made during simulator initialization. If A has several oracle interfaces, $q_A^{(i)}$ denotes the number of queries made to its ith interface in the execution under consideration. We assume that the length of every valid oracle answer is bounded by a polynomial in λ and the length of the corresponding query. Writing a query and reading or copying an answer count toward local computation. For concrete accounting, consider an oracle algorithm B that answers incoming queries by calling one underlying oracle. For integers $Q, L \geq 0$, let $c_B(Q, L)$ be the maximum number of calls that B makes to its underlying oracle, including calls made during initialization, in any execution in which it answers a sequence of at most Q incoming queries whose total bit length is at most L. The maximum is over the random tape of B and over all valid answers of the underlying oracle. Let $t_B(Q, L)$ be the maximum number of local computation steps taken by B in the same executions, including initialization; calls to the underlying oracle have unit cost in this quantity.

Let $\mathcal{I} = (\mathcal{I}_\lambda)_{\lambda \in \mathbb{N}}$ and $\mathcal{I}' = (\mathcal{I}'_\lambda)_{\lambda \in \mathbb{N}}$ be distributions over oracles. We write $H \xleftarrow{\$} \mathcal{I}$ and $\pi \xleftarrow{\$} \mathcal{I}'$ for samples from these distributions. If an experiment explicitly samples both, they are independent unless the experiment states otherwise. Let C be a deterministic stateless oracle algorithm such that C^π has the same query and answer spaces as H. The notation $\mathsf{C}^{\mathcal{I}'}$ denotes the distribution of C^π obtained by sampling $\pi \xleftarrow{\$} \mathcal{I}'$.

Here stateless means that, after C^π finishes answering one query, C retains no local state that affects how it answers a later query. Temporary state used while answering a single query is allowed, and all evaluations of C^π use the same persistent execution of the underlying oracle π.

Non-Interactive Arguments and Oracle Access. Let $\mathcal{R} = (\mathcal{R}_\lambda)_{\lambda \in \mathbb{N}}$ be a uniformly polynomial-time decidable family of relations, where $\mathcal{R}_\lambda \subseteq \{0,1\}^* \times \{0,1\}^*$. When λ is clear from the experiment, we write $(x, w) \in \mathcal{R}$ in place of $(x, w) \in \mathcal{R}_\lambda$. A non-interactive argument for $\mathcal{R}$ relative to an oracle O consists of a prover P^O and a verifier V^O. On input $(x, w) \in \mathcal{R}$, the prover outputs $\mathsf{pf} \xleftarrow{\$} \mathsf{P}^O(x, w)$. On input (x, pf), the verifier outputs $\mathsf{V}^O(x, \mathsf{pf}) \in \{0, 1\}$.

The pair (P, V) is complete relative to $\mathcal{I}$ if, for every polynomial p, there is a negligible function ν_p such that, for every λ and every $(x, w) \in \mathcal{R}$ with $|x| + |w| \leq p(\lambda)$,

$$\Pr\Big[H \xleftarrow{\$} \mathcal{I};\ \mathsf{pf} \xleftarrow{\$} \mathsf{P}^H(x, w) :\ \mathsf{V}^H(x, \mathsf{pf}) = 1\Big] \geq 1 - \nu_p(\lambda)\ .$$

The probability is over H and the random tapes of the prover and verifier.

Suppose we replace H with a construction that is indifferentiable from it. If the indifferentiability guarantee holds only against uniform polynomial-time distinguishers, this replacement preserves completeness

of the new argument for every efficiently generated polynomial-length instance-witness distribution. If the indifferentiability guarantee also holds against non-uniform polynomial-time distinguishers, then the above replacement also preserves completeness of the new argument uniformly over all polynomial-length instance-witness pairs, as required in the definition above. We define each notion of soundness in the section where it is used.

With $H \xleftarrow{\$} \mathcal{I}$, the honest prover and verifier are P^H and V^H. With $\pi \xleftarrow{\$} \mathcal{I}'$, replacing H by C^π gives $\mathsf{P}^{\mathsf{C}^\pi}$ and $\mathsf{V}^{\mathsf{C}^\pi}$. A cheating prover in the latter experiment is written $\mathcal{P}'^\pi$: it may query π directly and may evaluate the public algorithm C^π. The prover, every evaluation of C^π, and the verifier use one shared execution of π. The verifier has no oracle other than C^π.

Standard Indifferentiability. Let S be a stateful oracle algorithm such that S^H has the same query and answer spaces as π. Let $\mathcal{D}$ have two oracle interfaces. In the first experiment its interfaces are π and C^π, in that order; in the second they are S^H and H, in that order. Define

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{C},\mathcal{I}',\mathcal{I}}(\mathcal{D},\mathsf{S}) := \left|\Pr\left[\pi \xleftarrow{\$} \mathcal{I}' :\ \mathcal{D}^{\pi,\mathsf{C}^\pi} = 1\right] - \Pr\left[H \xleftarrow{\$} \mathcal{I} :\ \mathcal{D}^{\mathsf{S}^H,H} = 1\right]\right|.$$

In the first experiment, the first interface and the calls made inside the second interface use one shared execution of π. In the second experiment, one execution of S^H is initialized before $\mathcal{D}$ starts. That execution answers every query made to the first interface of $\mathcal{D}$. The oracle H used by the simulator is the same execution of H that answers queries to the second interface of $\mathcal{D}$.

For functions T, Q_1, Q_2 and $\varepsilon_{\mathrm{ind}}$, the simulator S gives indifferentiability error at most $\varepsilon_{\mathrm{ind}}(T,Q_1,Q_2)$ if $\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{C},\mathcal{I}',\mathcal{I}}(\mathcal{D},\mathsf{S}) \le \varepsilon_{\mathrm{ind}}(T,Q_1,Q_2)$ for every distinguisher $\mathcal{D}$ whose local running time is at most T and that makes at most Q_1 and Q_2 queries to its first and second interfaces. We say that $\mathsf{C}^{\mathcal{I}'}$ is indifferentiable from $\mathcal{I}$ if there is one probabilistic polynomial-time simulator S for which this advantage is negligible for every probabilistic polynomial-time distinguisher $\mathcal{D}$. The description of the simulator is chosen before $\mathcal{D}$. Each separate indifferentiability experiment initializes a fresh simulator execution.

For completeness, the following lemma justifies the observation about expected-time extractors in Remark 1. Both expected running-time bounds are hypotheses of the lemma.

Lemma 1. *Let G_0 and G_1 be two experiments that are computationally indistinguishable against probabilistic polynomial-time tests. Let T be a probabilistic algorithm for which there is a polynomial p such that, in either experiment, the expected running time of T is at most $p(\lambda)$. Then*

$$|\Pr[\mathsf{G}_0(T) = 1] - \Pr[\mathsf{G}_1(T) = 1]|$$

is negligible.

Proof. Suppose otherwise. Then there is a polynomial q and infinitely many λ for which

$$|\Pr[\mathsf{G}_0(T) = 1] - \Pr[\mathsf{G}_1(T) = 1]|$$

is at least $1/q(\lambda)$. Let $T^\star$ run T for at most $4p(\lambda)q(\lambda)$ steps and output 0 if this cutoff is reached. By Markov's inequality, in either experiment the probability that T exceeds this cutoff is at most $1/(4q(\lambda))$. Hence, for infinitely many λ,

$$\begin{aligned}|\Pr[\mathsf{G}_0(T^\star) = 1] - \Pr[\mathsf{G}_1(T^\star) = 1]| \ge\ & |\Pr[\mathsf{G}_0(T) = 1] - \Pr[\mathsf{G}_1(T) = 1]| \\ & - |\Pr[\mathsf{G}_0(T^\star) = 1] - \Pr[\mathsf{G}_0(T) = 1]| \\ & - |\Pr[\mathsf{G}_1(T^\star) = 1] - \Pr[\mathsf{G}_1(T) = 1]| \\ \ge\ & \frac{1}{2q(\lambda)}\end{aligned}$$

Since $T^\star$ is probabilistic polynomial time, this contradicts the assumed computational indistinguishability of the two experiments.

4 Straight-line Knowledge Soundness

Straight-line extraction is the simplest setting in which to study whether knowledge soundness survives replacing the ideal oracle by an indifferentiable construction. The cheating prover is executed only once. After it outputs its statement and proof, the extractor receives the information specified by the

knowledge-soundness experiment and attempts to recover a witness, but it never invokes the prover again. Thus the indifferentiability reduction only has to reproduce this one prover execution.

There is nevertheless an important distinction depending on whether the extractor receives the prover's oracle transcript. If it does not, standard indifferentiability gives a direct reduction. If it does, an additional issue arises. After replacing the primitive oracle by S^H, the resulting ideal-oracle prover is $\mathcal{P}'^{\mathsf{S}^H}$. Its queries to H are the queries made by S while answering the queries of $\mathcal{P}'$. These queries are not, in general, determined by the transcript $\tau(\mathcal{P}' \leftrightarrow \mathsf{S}^H)$ that is visible outside the simulator. We therefore treat the two cases separately.

A straight-line extractor is run only after the cheating prover has terminated and has no operation that invokes that prover again. We first consider extractors that do not receive the prover's oracle transcript, and then consider extractors that do.

In both variants below, the extractor may depend on the code of the cheating prover. The quantifier order is that the prover is fixed first, then an extractor for that prover is fixed, and only afterwards are the oracle and the random tapes used in the experiment sampled. Thus the extractor may not depend on the particular sampled execution except through the inputs explicitly supplied to it.

The two variants in this section are straight-line notions: extraction is performed from the completed execution without invoking the prover again. Chiesa-Orrù [CO25], by contrast, use a rewinding extraction interface. Our corresponding rewinding notion is defined in the next section.

4.1 Extraction without the Prover Transcript

An extractor in this subsection has the interface $\mathsf{E}^H(x, \mathsf{pf}, \tau_{\mathsf{V}})$. It receives the statement, the proof, and the ordered transcript of the verifier's oracle queries. It receives neither the prover's oracle transcript nor an operation for invoking the prover.

For an oracle algorithm $\mathcal{P}$ that outputs a pair (x, pf), define

$$\mathsf{Adv}^{\mathrm{po}}_{\mathcal{R},\mathsf{V},\mathcal{I}}(\mathcal{P},\mathsf{E}) := \Pr\left[\begin{array}{l} H \xleftarrow{\$} \mathcal{I};\\ (x,\mathsf{pf}) \xleftarrow{\$} \mathcal{P}^H;\\ b \xleftarrow{\$} \mathsf{V}^H(x,\mathsf{pf}),\ \text{recording } \tau_{\mathsf{V}} = \tau(\mathsf{V} \leftrightarrow H);\\ w \xleftarrow{\$} \mathsf{E}^H(x,\mathsf{pf},\tau_{\mathsf{V}}) :\\ b = 1 \ \wedge\ (x,w) \notin \mathcal{R} \end{array}\right].$$

Here and below, if the extractor outputs $\perp$, then $(x, \perp) \notin \mathcal{R}$. The three algorithms use the same execution of H.

The argument has proof-only straight-line knowledge soundness relative to $\mathcal{I}$ if, for every probabilistic polynomial-time $\mathcal{P}$, there is a probabilistic polynomial-time extractor E for which $\mathsf{Adv}^{\mathrm{po}}_{\mathcal{R},\mathsf{V},\mathcal{I}}(\mathcal{P},\mathsf{E})$ is negligible in λ.

For a cheating prover $\mathcal{P}'$ with direct access to π, define

$$\mathsf{Adv}^{\mathrm{po}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}',\mathsf{E}') := \Pr\left[\begin{array}{l} \pi \xleftarrow{\$} \mathcal{I}';\\ (x,\mathsf{pf}) \xleftarrow{\$} \mathcal{P}'^{\pi};\\ b \xleftarrow{\$} \mathsf{V}^{\mathsf{C}^\pi}(x,\mathsf{pf}),\ \text{recording } \tau_{\mathsf{V}} = \tau(\mathsf{V} \leftrightarrow \mathsf{C}^\pi);\\ w \xleftarrow{\$} \mathsf{E}'^{\pi}(x,\mathsf{pf},\tau_{\mathsf{V}}) :\\ b = 1 \ \wedge\ (x,w) \notin \mathcal{R} \end{array}\right].$$

The transcript τ_{V} contains the queries and answers at the C^π interface. It does not contain the primitive calls made while C^π answers those queries. The argument has proof-only straight-line knowledge soundness relative to $\mathsf{C}^{\mathcal{I}'}$ if, for every probabilistic polynomial-time $\mathcal{P}'$, there is a probabilistic polynomial-time extractor E' for which $\mathsf{Adv}^{\mathrm{po}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}',\mathsf{E}')$ is negligible in λ.

Theorem 1. *Assume that $\mathsf{C}^{\mathcal{I}'}$ is indifferentiable from $\mathcal{I}$ via S. Fix $\mathcal{P}'$, and define the oracle algorithm $\widehat{\mathcal{P}}$ by $\widehat{\mathcal{P}}^H := \mathcal{P}'^{\mathsf{S}^H}$. An execution of $\widehat{\mathcal{P}}^H$ initializes one fresh execution of S^H and then runs $\mathcal{P}'$ using that execution as its oracle.*

For a probabilistic polynomial-time extractor E for $\widehat{\mathcal{P}}$, define $\mathsf{E}'^{\pi}(x,\mathsf{pf},\tau_{\mathsf{V}}) := \mathsf{E}^{\mathsf{C}^\pi}(x,\mathsf{pf},\tau_{\mathsf{V}})$. Then E' is probabilistic polynomial time, and there is a probabilistic polynomial-time distinguisher $\mathcal{D}$ such that

$$\mathsf{Adv}^{\mathrm{po}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}',\mathsf{E}') \le \mathsf{Adv}^{\mathrm{po}}_{\mathcal{R},\mathsf{V},\mathcal{I}}(\widehat{\mathcal{P}},\mathsf{E}) + \mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{C},\mathcal{I}',\mathcal{I}}(\mathcal{D},\mathsf{S}).$$

If $\mathcal{P}'$, V, and E make at most q_P, q_V, and q_E oracle queries, respectively, then $\mathcal{D}$ makes at most q_P queries to its first interface and at most $q_V + q_E$ queries to its second interface.

Proof. We use indifferentiability to replace the primitive and construction interfaces (π, C^π) by (S^H, H).

Define a distinguisher $\mathcal{D}$ with oracle interfaces (O_1, O_2) as follows. It runs $(x, \mathsf{pf}) \xleftarrow{\$} \mathcal{P}'^{O_1}$, then runs $b \xleftarrow{\$} \mathsf{V}^{O_2}(x, \mathsf{pf})$ while recording τ_{V}, and finally runs $w \xleftarrow{\$} \mathsf{E}^{O_2}(x, \mathsf{pf}, \tau_{\mathsf{V}})$. It outputs 1 if and only if $b = 1$ and $(x, w) \notin \mathcal{R}$.

If $(O_1, O_2) = (\pi, \mathsf{C}^\pi)$, the prover is $\mathcal{P}'^{\pi}$, while the verifier and extractor use C^π. By the definition of E', this is exactly the experiment defining $\mathsf{Adv}^{\mathrm{po}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}', \mathsf{E}')$.

If $(O_1, O_2) = (\mathsf{S}^H, H)$, the prover is $\mathcal{P}'^{\mathsf{S}^H} = \widehat{\mathcal{P}}^{\,H}$, while the verifier and extractor use H. This is exactly the experiment defining $\mathsf{Adv}^{\mathrm{po}}_{\mathcal{R},\mathsf{V},\mathcal{I}}(\widehat{\mathcal{P}}, \mathsf{E})$.

The difference between these two probabilities is therefore at most $\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{C},\mathcal{I}',\mathcal{I}}(\mathcal{D}, \mathsf{S})$. The distinguisher runs $\mathcal{P}'$, V, and E once, records a polynomial-size transcript, and tests $(x, w) \in \mathcal{R}$. All these computations have worst-case polynomial running time. The bound on E also holds on the inputs and oracle replies in the construction experiment, by Section 3. Thus standard indifferentiability makes the comparison term negligible. If the ideal-model extraction error is negligible as well, then so is the construction-model extraction error.

If $\mathcal{P}'$, V, and E make at most q_P, q_V, and q_E oracle queries, respectively, then $\mathcal{D}$ makes at most q_P first-interface queries and $q_V + q_E$ second-interface queries. Its local running time includes transcript handling and relation testing. Finally, E' performs at most q_E evaluations of C^π on polynomial-length queries. Since C is stateless and polynomial time, these evaluations have polynomial total cost. Hence E' is probabilistic polynomial time.

Corollary 1. *If the argument has proof-only straight-line knowledge soundness relative to $\mathcal{I}$ and $\mathsf{C}^{\mathcal{I}'}$ is indifferentiable from $\mathcal{I}$, then, for every probabilistic polynomial-time $\mathcal{P}'$, there is a probabilistic polynomial-time extractor E' for which $\mathsf{Adv}^{\mathrm{po}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}', \mathsf{E}')$ is negligible.*

Proof. For the fixed $\mathcal{P}'$, the algorithm $\widehat{\mathcal{P}}$ in Theorem 1 is probabilistic polynomial time. Choose its extractor E using proof-only straight-line knowledge soundness, and apply that theorem. The ideal-model term is negligible by knowledge soundness, and the other term is negligible by indifferentiability. The theorem also proves that E' is probabilistic polynomial time.

If the primitive queries made by $\mathcal{P}'$ have total length at most L_P, then $\widehat{\mathcal{P}}$ makes at most $c_{\mathsf{S}}(q_P, L_P)$ queries to H. Its local running time is the running time of $\mathcal{P}'$ plus at most $t_{\mathsf{S}}(q_P, L_P)$ and the time needed to pass queries and answers between the two algorithms.

Remark 2. The term proof-only refers to the information available to the extractor: after the prover terminates, the extractor receives $(x, \mathsf{pf}, \tau_{\mathsf{V}})$ and oracle access, but it receives neither the prover's oracle transcript nor an operation for invoking the prover again. Thus Theorem 1 is a transfer theorem for arguments that already admit this extraction interface in the ideal-oracle model. For example, if a valid witness can be recovered efficiently from (x, pf) whenever the verifier accepts, then the extractor needs neither the prover's oracle transcript nor another invocation of the prover.

4.2 Extraction with the Prover Transcript

An extractor in this subsection has the interface $\mathsf{E}^H(x, \mathsf{pf}, \tau_{\mathsf{P}}, \tau_{\mathsf{V}})$, where τ_{P} and τ_{V} are the ordered oracle transcripts of the prover and verifier executions.

For $\mathcal{P}$, define

$$\mathsf{Adv}^{\mathrm{tr}}_{\mathcal{R},\mathsf{V},\mathcal{I}}(\mathcal{P}, \mathsf{E}) := \Pr\left[\begin{array}{l} H \xleftarrow{\$} \mathcal{I}; \\ (x, \mathsf{pf}) \xleftarrow{\$} \mathcal{P}^H, \text{ recording } \tau_{\mathsf{P}} = \tau(\mathcal{P} \leftrightarrow H); \\ b \xleftarrow{\$} \mathsf{V}^H(x, \mathsf{pf}), \text{ recording } \tau_{\mathsf{V}} = \tau(\mathsf{V} \leftrightarrow H); \\ w \xleftarrow{\$} \mathsf{E}^H(x, \mathsf{pf}, \tau_{\mathsf{P}}, \tau_{\mathsf{V}}) : \\ b = 1 \ \wedge \ (x, w) \notin \mathcal{R} \end{array}\right].$$

The argument has transcript-dependent straight-line knowledge soundness relative to $\mathcal{I}$ if, for every probabilistic polynomial-time $\mathcal{P}$, there is a probabilistic polynomial-time extractor E for which $\mathsf{Adv}^{\mathrm{tr}}_{\mathcal{R},\mathsf{V},\mathcal{I}}(\mathcal{P}, \mathsf{E})$ is negligible.

For $\mathcal{P}'$ and E', define

$$\mathsf{Adv}^{\mathrm{tr}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}',\mathsf{E}') := \Pr\left[\begin{array}{l} \pi \xleftarrow{\$} \mathcal{I}';\\ (x,\mathsf{pf}) \xleftarrow{\$} \mathcal{P}'^{\pi},\ \text{recording } \tau_{\mathrm{ext}} = \tau(\mathcal{P}' \leftrightarrow \pi);\\ b \xleftarrow{\$} \mathsf{V}^{\mathsf{C}^{\pi}}(x,\mathsf{pf}),\ \text{recording } \tau_{\mathsf{V}} = \tau(\mathsf{V} \leftrightarrow \mathsf{C}^{\pi});\\ w \xleftarrow{\$} \mathsf{E}'^{\pi}(x,\mathsf{pf},\tau_{\mathrm{ext}},\tau_{\mathsf{V}}) :\\ b = 1 \ \wedge\ (x,w) \notin \mathcal{R} \end{array}\right].$$

For the algorithm $\widehat{\mathcal{P}}^{H} = \mathcal{P}'^{\mathsf{S}^{H}}$, the transcript expected by the ideal-oracle extractor is the ordered sequence of queries made by S to H, not the sequence of queries made by $\mathcal{P}'$ to S. The argument has transcript-dependent straight-line knowledge soundness relative to $\mathsf{C}^{\mathcal{I}'}$ if, for every probabilistic polynomial-time $\mathcal{P}'$, there is a probabilistic polynomial-time extractor E' for which $\mathsf{Adv}^{\mathrm{tr}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}',\mathsf{E}')$ is negligible.

We next define the condition used to replace the former sequence by information computed from the latter.

Definition 1 (Transcript Reconstruction). *Fix S. Let A and B be oracle algorithms and let T be a distinguisher. In each of the following experiments, one execution of H is shared by every algorithm that carries the superscript H, and one execution of S^{H} answers all oracle queries made by A. Define*

$$p_{\mathrm{true}}(A,B,T) := \Pr\left[\begin{array}{l} H \xleftarrow{\$} \mathcal{I};\\ v \xleftarrow{\$} A^{\mathsf{S}^{H}},\ \textit{recording } \tau_{\mathrm{ext}} = \tau(A \leftrightarrow \mathsf{S}^{H});\\ \textit{also record } \tau_{\mathrm{int}} = \tau(\mathsf{S} \leftrightarrow H);\\ z \xleftarrow{\$} B^{H}(v) :\\ T^{H}(v,z,\tau_{\mathrm{int}}) = 1 \end{array}\right]$$

and

$$p_{\mathrm{rec}}(A,B,T) := \Pr\left[\begin{array}{l} H \xleftarrow{\$} \mathcal{I};\\ v \xleftarrow{\$} A^{\mathsf{S}^{H}},\ \textit{recording } \tau_{\mathrm{ext}} = \tau(A \leftrightarrow \mathsf{S}^{H});\\ \textit{also record } \tau_{\mathrm{int}} = \tau(\mathsf{S} \leftrightarrow H);\\ z \xleftarrow{\$} B^{H}(v);\\ \widehat{\tau} \xleftarrow{\$} \mathsf{Rec}^{H}(\tau_{\mathrm{ext}}) :\\ T^{H}(v,z,\widehat{\tau}) = 1 \end{array}\right].$$

The output $\widehat{\tau}$ is a sequence of query-answer pairs with the same query and answer spaces as τ_{int}. In each experiment, the occurrence of S^{H} denotes one simulator execution and all occurrences of H denote one shared oracle execution. Set $\mathsf{Adv}^{\mathrm{rec}}_{\mathsf{S},\mathsf{Rec},\mathcal{I}}(A,B,T) := |p_{\mathrm{true}}(A,B,T) - p_{\mathrm{rec}}(A,B,T)|$.

We say that S admits transcript reconstruction if there is a probabilistic polynomial-time algorithm Rec such that for every probabilistic polynomial-time A,B,T, $\mathsf{Adv}^{\mathrm{rec}}_{\mathsf{S},\mathsf{Rec},\mathcal{I}}(A,B,T)$ is negligible.

The algorithm B is included because, in the knowledge-soundness experiment, the verifier runs after the prover and before the extractor. Queries made by Rec occur in p_{rec} but not in p_{true}; thus the definition also accounts for any change to the state of H caused by those queries. Transcript reconstruction requires preservation of the complete joint experiment relevant to extraction; it is not enough for $\widehat{\tau}$ to have the same marginal distribution as τ_{int}. In particular, the transcript must remain interchangeable in the presence of the initial execution, the intervening computation B, the shared state of H, and the final oracle-aided test T. The exact condition below is a stronger sufficient condition: when the true internal transcript is deterministically recoverable from τ_{ext} without oracle queries, the two joint experiments are identical.

Definition 2 (Exact Transcript Reconstruction). *The simulator S admits exact transcript reconstruction if there is one deterministic polynomial-time algorithm Rec that makes no oracle queries and satisfies $\mathsf{Rec}(\tau_{\mathrm{ext}}) = \tau_{\mathrm{int}}$ for every choice of H, every choice of the random tapes of A and S, and every completed execution of $A^{\mathsf{S}^{H}}$. This condition implies Definition 1 with advantage zero.*

Theorem 2. *Assume that $\mathsf{C}^{\mathcal{I}'}$ is indifferentiable from $\mathcal{I}$ via S, and that S admits transcript reconstruction via Rec. Fix $\mathcal{P}'$ and define $\widehat{\mathcal{P}}^{H} := \mathcal{P}'^{\mathsf{S}^{H}}$ as in Theorem 1. For a probabilistic polynomial-time extractor*

E for $\widehat{\mathcal{P}}$, define E' by first computing $\widehat{\tau} \xleftarrow{\$} \mathsf{Rec}^{\mathsf{C}^\pi}(\tau_{\mathrm{ext}})$ and then returning $\mathsf{E}^{\mathsf{C}^\pi}(x, \mathsf{pf}, \widehat{\tau}, \tau_{\mathsf{V}})$. Then E' is probabilistic polynomial time, and there are probabilistic polynomial-time algorithms $B, T, \mathcal{D}$ such that

$$\mathsf{Adv}^{\mathrm{tr}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}', \mathsf{E}') \leq \mathsf{Adv}^{\mathrm{tr}}_{\mathcal{R},\mathsf{V},\mathcal{I}}(\widehat{\mathcal{P}}, \mathsf{E}) + \mathsf{Adv}^{\mathrm{rec}}_{\mathsf{S},\mathsf{Rec},\mathcal{I}}(\mathcal{P}', B, T) + \mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{C},\mathcal{I}',\mathcal{I}}(\mathcal{D}, \mathsf{S}).$$

If $\mathcal{P}'$, V, Rec, and E make at most q_P, q_V, q_{Rec}, and q_E oracle queries, respectively, then $\mathcal{D}$ makes at most q_P queries to its first interface and at most $q_V + q_{\mathsf{Rec}} + q_E$ queries to its second interface. Its local running time includes the computations of all four algorithms, recording and copying τ_{ext} and τ_{V}, and testing $(x, w) \in \mathcal{R}$.

Proof. Fix $\mathcal{P}'$ and the extractor E' defined in the theorem. We consider three hybrids. Let p_i denote the probability in Hyb_i that the verifier accepts but the extractor fails to output a witness.

Hybrid Hyb_0. This is the transcript-dependent knowledge-soundness experiment relative to $\mathsf{C}^{\mathcal{I}'}$. Thus $\pi \xleftarrow{\$} \mathcal{I}'$ is sampled, $(x, \mathsf{pf}) \xleftarrow{\$} \mathcal{P}'^{\pi}$ is run while recording τ_{ext}, and $b \xleftarrow{\$} \mathsf{V}^{\mathsf{C}^\pi}(x, \mathsf{pf})$ is run while recording τ_{V}. The extractor computes $\widehat{\tau} \xleftarrow{\$} \mathsf{Rec}^{\mathsf{C}^\pi}(\tau_{\mathrm{ext}})$ and then runs $w \xleftarrow{\$} \mathsf{E}^{\mathsf{C}^\pi}(x, \mathsf{pf}, \widehat{\tau}, \tau_{\mathsf{V}})$.

By definition, $p_0 = \mathsf{Adv}^{\mathrm{tr}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}', \mathsf{E}')$.

Hybrid Hyb_1. We replace the primitive and construction interfaces by the ideal indifferentiability interfaces. Thus $H \xleftarrow{\$} \mathcal{I}$ is sampled, the prover is run as $\mathcal{P}'^{\mathsf{S}^H}$ while recording τ_{ext}, and the verifier is run as V^H while recording τ_{V}. After the verifier terminates, the extractor computes $\widehat{\tau} \xleftarrow{\$} \mathsf{Rec}^H(\tau_{\mathrm{ext}})$ and runs $w \xleftarrow{\$} \mathsf{E}^H(x, \mathsf{pf}, \widehat{\tau}, \tau_{\mathsf{V}})$.

The transition from Hyb_0 to Hyb_1 is one application of standard indifferentiability. Indeed, define $\mathcal{D}^{O_1, O_2}$ to run $(x, \mathsf{pf}) \xleftarrow{\$} \mathcal{P}'^{O_1}$ while recording τ_{ext}, then run $b \xleftarrow{\$} \mathsf{V}^{O_2}(x, \mathsf{pf})$ while recording τ_{V}, compute $\widehat{\tau} \xleftarrow{\$} \mathsf{Rec}^{O_2}(\tau_{\mathrm{ext}})$, and finally run $w \xleftarrow{\$} \mathsf{E}^{O_2}(x, \mathsf{pf}, \widehat{\tau}, \tau_{\mathsf{V}})$. It outputs 1 exactly when $b = 1$ and $(x, w) \notin \mathcal{R}$.

With $(O_1, O_2) = (\pi, \mathsf{C}^\pi)$ this is Hyb_0, whereas with $(O_1, O_2) = (\mathsf{S}^H, H)$ it is Hyb_1. The initial transcripts have polynomial length. Since Rec is polynomial time, its output also has polynomial length; E is polynomial time on this input under either oracle interface. Recording transcripts and testing $(x, w) \in \mathcal{R}$ have polynomial cost as well. Thus $\mathcal{D}$ is probabilistic polynomial time, and $|p_0 - p_1| \leq \mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{C},\mathcal{I}',\mathcal{I}}(\mathcal{D}, \mathsf{S})$ is negligible by standard indifferentiability.

Hybrid Hyb_2. The prover and verifier executions are exactly as in Hyb_1. During the prover execution, also record $\tau_{\mathrm{int}} = \tau(\mathsf{S} \leftrightarrow H)$. We now change only the transcript given to the extractor: instead of $\widehat{\tau} \xleftarrow{\$} \mathsf{Rec}^H(\tau_{\mathrm{ext}})$, the extractor receives the true transcript τ_{int}. Thus it runs $w \xleftarrow{\$} \mathsf{E}^H(x, \mathsf{pf}, \tau_{\mathrm{int}}, \tau_{\mathsf{V}})$.

The transition from Hyb_1 to Hyb_2 is exactly the transcript-reconstruction guarantee. To instantiate Definition 1, take $A = \mathcal{P}'$. Let $B^H(x, \mathsf{pf})$ run $b \xleftarrow{\$} \mathsf{V}^H(x, \mathsf{pf})$ while recording τ_{V} and return (b, τ_{V}). Given (x, pf), (b, τ_{V}), and a transcript τ, let T^H run $w \xleftarrow{\$} \mathsf{E}^H(x, \mathsf{pf}, \tau, \tau_{\mathsf{V}})$ and output 1 exactly when $b = 1$ and $(x, w) \notin \mathcal{R}$.

Both B and T are probabilistic polynomial time: they run the polynomial-time verifier and extractor on polynomial-length inputs, and relation membership is polynomial-time decidable. With these choices, the reconstructed experiment in Definition 1 is Hyb_1, while the true experiment is Hyb_2. Therefore $|p_1 - p_2| \leq \mathsf{Adv}^{\mathrm{rec}}_{\mathsf{S},\mathsf{Rec},\mathcal{I}}(\mathcal{P}', B, T)$, and this quantity is negligible by transcript reconstruction.

Identifying the final hybrid. By definition, the derived ideal-model prover is $\widehat{\mathcal{P}}^H = \mathcal{P}'^{\mathsf{S}^H}$. Its oracle transcript is exactly $\tau_{\mathrm{int}} = \tau(\mathsf{S} \leftrightarrow H)$. Hence Hyb_2 is exactly the transcript-dependent knowledge-soundness experiment relative to $\mathcal{I}$ for $\widehat{\mathcal{P}}$ and E. Thus $p_2 = \mathsf{Adv}^{\mathrm{tr}}_{\mathcal{R},\mathsf{V},\mathcal{I}}(\widehat{\mathcal{P}}, \mathsf{E})$.

Combining the two hybrid transitions proves the probability bound. The extractor E' runs Rec once on a polynomial-size transcript, then runs E on the resulting polynomial-size input. Both perform polynomially many oracle calls of polynomial length. Replacing those calls by evaluations of the stateless polynomial-time construction C^π gives polynomial total running time. This also proves the claimed query accounting.

Corollary 2. *If the argument has transcript-dependent straight-line knowledge soundness relative to $\mathcal{I}$, $\mathsf{C}^{\mathcal{I}'}$ is indifferentiable from $\mathcal{I}$ via S, and S admits transcript reconstruction, then, for every probabilistic polynomial-time $\mathcal{P}'$, there is a probabilistic polynomial-time extractor E' for which $\mathsf{Adv}^{\mathrm{tr}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}', \mathsf{E}')$ is negligible.*

Proof. Apply transcript-dependent straight-line knowledge soundness to $\widehat{\mathcal{P}}$ and choose the resulting polynomial-time extractor. Apply Theorem 2. The ideal-model extraction error is negligible by knowledge soundness, and the two comparison terms are negligible by reconstruction and indifferentiability. The theorem proves that the derived E' is probabilistic polynomial time.

5 Rewinding Knowledge Soundness

We now turn to rewinding extractors, i.e., extractors that may later run the same prover again, possibly while answering its oracle queries differently. We use a reset-and-replay interface related to the state-restoration formulation of Chiesa-Orrù [CO25]. The prover's random tape is fixed once. The extractor may repeatedly restart the prover and answer its oracle queries as it chooses. To reproduce an earlier prover state, the extractor restarts the prover and supplies the same sequence of oracle answers that led to that state. Since the random tape is unchanged, the prover follows the same execution up to that point.

Our goal is to transfer knowledge soundness from the ideal-oracle model to the construction model. To prove this implication, the reduction transforms a cheating prover $\mathcal{P}'$ from the construction model into an ideal-oracle prover $\widehat{\mathcal{P}}$. Thus the security implication and the prover transformation go in opposite directions. In the straight-line setting, the reduction only has to reproduce the prover's initial execution. With rewinding, it must also support every later invocation of the prover made by the extractor. Moreover, these invocations must use the same prover randomness as the initial execution.

After the indifferentiability transition, the primitive oracle available to the construction-world prover is replaced by S^H. Consequently, rerunning the same prover requires more than simply starting a fresh copy of S^H: the new execution must use the same simulator randomness as the initial one. Once the random tapes of the prover and S are fixed, their composition defines the particular ideal-oracle prover that produced the original proof, and this is the prover that the extractor must be able to invoke again.

Standard indifferentiability does not provide such resettable access. It gives a single stateful execution of S^H, but does not expose the simulator's random tape or otherwise provide a way to start a fresh simulator instance with the same randomness. We therefore need an additional property ensuring that, from the information available after the initial execution, the reduction can reproduce the prover behavior required by a rewinding extractor. We formalize this property below and show that, together with standard indifferentiability, it suffices to transfer rewinding knowledge soundness.

Resetting a Prover. For a probabilistic oracle algorithm A, let Ω_A denote the distribution of its ordinary private random tape. A sample $\rho \xleftarrow{\$} \Omega_A$ fixes that tape. We write $A[\rho]$ for the deterministic oracle machine obtained by fixing it. A reset restores the initial local state, including the random-tape head, while retaining ρ. In particular, an ordinary lazy-sampling table is cleared on reset. Two different query histories can consume the same tape positions; in particular, fixing the tape does not assign independent answers to every possible query across all such histories.

Concretely, we use the standard read-only random-tape implementation. Sampling ρ fixes the contents of the random tape once. Every reset restores the work tapes, control state, and random-tape head to their initial positions, but leaves the contents of the random tape equal to ρ. Hence, after ρ is fixed, the behavior of all reset invocations is that of one deterministic oracle machine $A[\rho]$. The ideal-world rewinding knowledge-soundness premise below is understood with this fixed-tape reset semantics.

We write $\langle A[\rho] \rangle$ for resettable black-box access to $A[\rho]$. On each invocation, the caller supplies an oracle algorithm K, after which $A[\rho]$ is reset to its initial state and run with oracle K. Every invocation uses the same fixed randomness ρ. The caller sees the oracle queries made by $A[\rho]$ through its interaction with K, but does not receive the internal state of $A[\rho]$.

We will also compose fixed-randomness oracle algorithms. Let $S[\omega]$ be a deterministic oracle algorithm that implements the oracle interface expected by $A[\rho]$. For any oracle K used by $S[\omega]$, define $(A[\rho] \circ S[\omega])^K := A[\rho]^{S[\omega]^K}$. Thus $A[\rho]$ sends its oracle queries to $S[\omega]$, which may in turn query K. A reset of $A[\rho] \circ S[\omega]$ restores both $A[\rho]$ and $S[\omega]$ to their initial local states while retaining the same fixed randomness ρ and ω.

A rewinding extractor has oracle access to H and invocation access to the fixed-tape prover that produced the initial proof. It may make sequential invocations of the prover, choosing when to stop adaptively, but the number of invocations is polynomially bounded in every execution. On each invocation, it may choose a different oracle algorithm K, and the prover is run with oracle K.

For a cheating prover $\mathcal{P}$ and an extractor E, define

$$\mathsf{Adv}^{\mathrm{rw}}_{\mathcal{R},\mathsf{V},\mathcal{I}}(\mathcal{P},\mathsf{E}) := \Pr\left[\begin{array}{l} H \xleftarrow{\$} \mathcal{I};\ \rho \xleftarrow{\$} \Omega_{\mathcal{P}}; \\ (x,\mathsf{pf}) \leftarrow \mathcal{P}[\rho]^H,\ \text{recording } \tau_{\mathsf{P}} = \tau(\mathcal{P}[\rho] \leftrightarrow H); \\ b \xleftarrow{\$} \mathsf{V}^H(x,\mathsf{pf}),\ \text{recording } \tau_{\mathsf{V}} = \tau(\mathsf{V} \leftrightarrow H); \\ w \xleftarrow{\$} \mathsf{E}^H(x,\mathsf{pf},\tau_{\mathsf{P}},\tau_{\mathsf{V}};\langle \mathcal{P}[\rho] \rangle) : \\ b = 1 \ \wedge \ (x,w) \notin \mathcal{R} \end{array}\right].$$

The argument has rewinding knowledge soundness relative to $\mathcal{I}$ if, for every probabilistic polynomial-time $\mathcal{P}$, there is a probabilistic polynomial-time extractor E for which $\mathsf{Adv}^{\mathrm{rw}}_{\mathcal{R},\mathsf{V},\mathcal{I}}(\mathcal{P},\mathsf{E})$ is negligible.

For a cheating prover $\mathcal{P}'$ and an extractor E', define

$$\mathsf{Adv}^{\mathrm{rw}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}',\mathsf{E}') := \Pr\left[\begin{array}{l}\pi \xleftarrow{\$} \mathcal{I}';\ \rho \xleftarrow{\$} \Omega_{\mathcal{P}'};\\ (x,\mathsf{pf}) \leftarrow \mathcal{P}'[\rho]^{\pi},\ \text{recording } \tau_{\mathrm{ext}} = \tau(\mathcal{P}'[\rho] \leftrightarrow \pi);\\ b \xleftarrow{\$} \mathsf{V}^{\mathsf{C}^{\pi}}(x,\mathsf{pf}),\ \text{recording } \tau_{\mathsf{V}} = \tau(\mathsf{V} \leftrightarrow \mathsf{C}^{\pi});\\ w \xleftarrow{\$} \mathsf{E}'^{\pi}(x,\mathsf{pf},\tau_{\mathrm{ext}},\tau_{\mathsf{V}};\langle\mathcal{P}'[\rho]\rangle):\\ b = 1 \ \wedge\ (x,w) \notin \mathcal{R}\end{array}\right].$$

On every invocation of $\langle\mathcal{P}'[\rho]\rangle$, the supplied oracle K has the query and answer spaces of π. The argument has rewinding knowledge soundness relative to $\mathsf{C}^{\mathcal{I}'}$ if, for every probabilistic polynomial-time $\mathcal{P}'$, there is a probabilistic polynomial-time extractor E' for which $\mathsf{Adv}^{\mathrm{rw}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}',\mathsf{E}')$ is negligible.

Reconstructing the Simulator for Later Invocations After the indifferentiability transition, the prover's primitive queries are answered by $\mathsf{S}[\omega]^H$ for a fixed simulator randomness ω. A rewinding extractor may later invoke the same prover again. To reproduce such an invocation, the reduction must therefore run the prover with a fresh copy of the simulator that uses the same randomness ω as in the initial execution.

The reduction does not know ω. After the initial execution, it has access to H and to the transcript τ_{ext} of the interaction between the prover and $\mathsf{S}[\omega]^H$. We ask whether this information is sufficient both to recover the transcript τ_{int} of the queries made by $\mathsf{S}[\omega]$ to H during the initial execution and to reproduce the simulator behavior needed in later invocations.

Definition 3 (Rewinding Reconstructor). *A rewinding reconstructor* $\mathsf{Rec}_{\mathrm{rw}}$ *is a stateful probabilistic oracle algorithm with three subroutines, namely* Init, Start, *and* Answer.

Let L be one plus the number of calls to these subroutines and the total bit length of their externally supplied inputs, including τ_{ext} at initialization and the primitive queries supplied to Answer*, over all sessions. There must be a single polynomial p such that the reconstructor's total local work, underlying-oracle calls, and bit lengths of generated queries and outputs are at most $p(\lambda, L)$. This bound holds for every request sequence and every well-formed sequence of underlying-oracle answers of the lengths allowed by Section 3. It includes initialization and every subsequent call. The computation of the supplied oracle algorithms is charged separately to the complete experiment. Thus polynomially many calls with polynomial total input length incur polynomial total reconstruction cost.*

The call $\widehat{\tau} \xleftarrow{\$} \mathsf{Rec}_{\mathrm{rw}}.\mathsf{Init}^H(\tau_{\mathrm{ext}})$ initializes $\mathsf{Rec}_{\mathrm{rw}}$ and returns a candidate transcript $\widehat{\tau}$ with the same query and answer spaces as τ_{int}. The randomness and state of $\mathsf{Rec}_{\mathrm{rw}}$ persist after initialization.

A call $\mathsf{Rec}_{\mathrm{rw}}.\mathsf{Start}()$ starts a new simulator session. On a primitive query q, the call $\mathsf{Rec}_{\mathrm{rw}}.\mathsf{Answer}^K(q)$ returns a primitive answer. All calls in one session use the same persistent instance of K, which has the same interface as H. Different sessions may use different oracles K. Calling Start *resets the session state but does not reset the persistent state or randomness of $\mathsf{Rec}_{\mathrm{rw}}$.*

Definition 4 (Reconstructed Simulator Session). *After initialization, define $\widehat{S}_{\mathsf{Rec}_{\mathrm{rw}}}$ to be the oracle algorithm implemented by the initialized reconstructor. When run with oracle K, it first calls $\mathsf{Rec}_{\mathrm{rw}}.\mathsf{Start}()$ and then answers each primitive query q by calling $\mathsf{Rec}_{\mathrm{rw}}.\mathsf{Answer}^K(q)$. Thus $\widehat{S}^K_{\mathsf{Rec}_{\mathrm{rw}}}$ denotes one reconstructed-simulator session using oracle K.*

For a fixed random tape ρ of an oracle algorithm A, define $(A[\rho] \circ \widehat{S}_{\mathsf{Rec}_{\mathrm{rw}}})^K := A[\rho]^{\widehat{S}^K_{\mathsf{Rec}_{\mathrm{rw}}}}$. Therefore every invocation through $\langle A[\rho] \circ \widehat{S}_{\mathsf{Rec}_{\mathrm{rw}}}\rangle$ resets $A[\rho]$ and starts a fresh reconstructed-simulator session, while the initialized state and randomness of $\mathsf{Rec}_{\mathrm{rw}}$ persist across all invocations.

Definition 5 (Rewinding Reconstruction Experiments). *Let A and B be probabilistic oracle algorithms, and let T be a distinguisher.*

The experiment $\mathsf{True}^{\mathrm{rw}}_{\mathsf{S},\mathcal{I}}(A,B,T)$ proceeds as follows.

1. *Sample $H \xleftarrow{\$} \mathcal{I}$, $\rho \xleftarrow{\$} \Omega_A$, and $\omega \xleftarrow{\$} \Omega_{\mathsf{S}}$.*
2. *Run $v \leftarrow A[\rho]^{\mathsf{S}[\omega]^H}$, recording $\tau_{\mathrm{ext}} = \tau(A[\rho] \leftrightarrow \mathsf{S}[\omega]^H)$ and $\tau_{\mathrm{int}} = \tau(\mathsf{S}[\omega] \leftrightarrow H)$ during this execution.*
3. *Run $z \xleftarrow{\$} B^H(v)$.*
4. *Run $T^H(v, z, \tau_{\mathrm{int}}; \langle A[\rho] \circ \mathsf{S}[\omega]\rangle)$ and output its result.*

The experiment $\mathsf{Rec}^{\mathrm{rw}}_{\mathsf{S},\mathsf{Rec}_{\mathrm{rw}},\mathcal{I}}(A,B,T)$ is identical through Step 3. It then computes $\widehat{\tau} \xleftarrow{\$} \mathsf{Rec}_{\mathrm{rw}}.\mathsf{Init}^H(\tau_{\mathrm{ext}})$, runs $T^H(v, z, \widehat{\tau}; \langle A[\rho] \circ \widehat{S}_{\mathsf{Rec}_{\mathrm{rw}}}\rangle)$, and finally outputs the corresponding result.

All occurrences of H in either experiment refer to the same sampled oracle and the same persistent state.

Definition 6 (Reconstruction for Rewinding). *For a simulator* S*, a rewinding reconstructor* $\mathsf{Rec}_{\mathrm{rw}}$*, and probabilistic oracle algorithms* A, B, T*, define*

$$\begin{aligned}&\mathsf{Adv}^{\text{rw-rec}}_{\mathsf{S},\mathsf{Rec}_{\mathrm{rw}},\mathcal{I}}(A,B,T)\\&\quad:=\left|\Pr\left[\mathsf{True}^{\mathrm{rw}}_{\mathsf{S},\mathcal{I}}(A,B,T)=1\right]-\Pr\left[\mathsf{Rec}^{\mathrm{rw}}_{\mathsf{S},\mathsf{Rec}_{\mathrm{rw}},\mathcal{I}}(A,B,T)=1\right]\right|.\end{aligned}$$

We say that S *admits reconstruction for rewinding if there exists a single rewinding reconstructor* $\mathsf{Rec}_{\mathrm{rw}}$ *such that* $\mathsf{Adv}^{\text{rw-rec}}_{\mathsf{S},\mathsf{Rec}_{\mathrm{rw}},\mathcal{I}}(A,B,T)$ *is negligible for every probabilistic polynomial-time* A, B, T*.*

The randomness ρ used by A initially is reused whenever T invokes A. In the true experiment, every simulator session uses the same tape ω. In the reconstructed experiment, every session uses the same initialized reconstructor. The algorithm B permits oracle queries between the initial execution of A and initialization of the reconstructor; in the application below, B runs the verifier.

If T never invokes its final input, the output of $\mathsf{Rec}_{\mathrm{rw}}.\mathsf{Init}$ is a transcript reconstructor in the sense of Definition 1. Hence reconstruction for rewinding implies transcript reconstruction.

Rewinding Knowledge-Soundness Transfer Theorem.

Theorem 3 (Transfer of Rewinding Knowledge Soundness). *Assume that* $\mathsf{C}^{\mathcal{I}'}$ *is indifferentiable from* $\mathcal{I}$ *via* S*, and that* S *admits reconstruction for rewinding via* $\mathsf{Rec}_{\mathrm{rw}}$*. Fix a probabilistic polynomial-time cheating prover* $\mathcal{P}'$*.*

Define the probabilistic oracle algorithm $\widehat{\mathcal{P}}$ *as follows. Its random tape is a pair* $(\rho,\omega)\xleftarrow{\$}\Omega_{\mathcal{P}'}\times\Omega_{\mathsf{S}}$*, and* $\widehat{\mathcal{P}}[\rho,\omega]^H:=\mathcal{P}'[\rho]^{\mathsf{S}[\omega]^H}$*. A reset of* $\widehat{\mathcal{P}}[\rho,\omega]$ *resets both component algorithms and reuses the same pair* (ρ,ω)*.*

Let E *be a probabilistic polynomial-time rewinding extractor for* $\widehat{\mathcal{P}}$*. Define* E' *as follows. On input* $(x,\mathsf{pf},\tau_{\mathrm{ext}},\tau_{\mathsf{V}})$ *and invocation access to* $\langle\mathcal{P}'[\rho]\rangle$*, it computes* $\widehat{\tau}\xleftarrow{\$}\mathsf{Rec}_{\mathrm{rw}}.\mathsf{Init}^{\mathsf{C}^\pi}(\tau_{\mathrm{ext}})$ *and runs* $\mathsf{E}^{\mathsf{C}^\pi}(x,\mathsf{pf},\widehat{\tau},\tau_{\mathsf{V}};\cdot)$*. Whenever* E *invokes its prover with an oracle* K*, the extractor* E' *invokes* $\langle\mathcal{P}'[\rho]\rangle$ *and answers that invocation's primitive queries using a new* $\widehat{S}^K_{\mathsf{Rec}_{\mathrm{rw}}}$ *session.*

Then E' *is probabilistic polynomial time, and there are probabilistic polynomial-time algorithms* $B, T, \mathcal{D}$ *such that*

$$\mathsf{Adv}^{\mathrm{rw}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}',\mathsf{E}')\le\mathsf{Adv}^{\mathrm{rw}}_{\mathcal{R},\mathsf{V},\mathcal{I}}(\widehat{\mathcal{P}},\mathsf{E})+\mathsf{Adv}^{\text{rw-rec}}_{\mathsf{S},\mathsf{Rec}_{\mathrm{rw}},\mathcal{I}}(\mathcal{P}',B,T)+\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{C},\mathcal{I}',\mathcal{I}}(\mathcal{D},\mathsf{S}).$$

Let q_P^{initial} *bound the primitive queries in the initial execution of* $\mathcal{P}'$*, let* q_V *bound the verifier's second-interface queries, and let* q_{Init} *bound those made by* $\mathsf{Rec}_{\mathrm{rw}}.\mathsf{Init}$*. Let* q_E^{all} *bound all second-interface queries during the remaining extraction phase, including queries issued through every supplied oracle* K *and every reconstructed-simulator session. Then*

$$Q_1\le q_P^{\mathrm{initial}},\qquad Q_2\le q_V+q_{\mathrm{Init}}+q_E^{\mathrm{all}}.$$

The running-time accounting includes every prover invocation, every reconstruction call, every supplied oracle algorithm, transcript handling, and relation testing. The proof below shows that all these costs have worst-case polynomial bounds.

Proof. Fix $\mathcal{P}'$ and the extractor E' defined in the theorem. We consider three hybrids. Let p_i denote the probability in Hyb_i that the verifier accepts but the extractor fails to output a witness for the accepted statement.

Hybrid Hyb_0. This is the knowledge-soundness experiment relative to $\mathsf{C}^{\mathcal{I}'}$ with prover $\mathcal{P}'$ and extractor E'. Thus we sample $\pi\xleftarrow{\$}\mathcal{I}'$ and $\rho\xleftarrow{\$}\Omega_{\mathcal{P}'}$, run $\mathcal{P}'[\rho]^\pi$ to obtain (x,pf) and τ_{ext}, and then run $\mathsf{V}^{\mathsf{C}^\pi}(x,\mathsf{pf})$ while recording τ_{V}. The extractor initializes $\mathsf{Rec}_{\mathrm{rw}}$ on τ_{ext} using oracle C^π and then runs E as specified in the definition of E'.

Whenever E invokes its prover with an oracle K, the invocation resets $\mathcal{P}'[\rho]$ and answers its primitive queries using a fresh session of $\widehat{S}^K_{\mathsf{Rec}_{\mathrm{rw}}}$.

By definition, $p_0=\mathsf{Adv}^{\mathrm{rw}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}',\mathsf{E}')$.

HYBRID Hyb_1. We replace the primitive and construction interfaces by the ideal indifferentiability interfaces. Namely, sample $H \xleftarrow{\$} \mathcal{I}$, and let $\mathsf{S}[\omega]^H$ be the simulator instance used at the first interface. The initial prover execution is now $\mathcal{P}'[\rho]^{\mathsf{S}[\omega]^H}$, while the verifier uses H.

Everything after the verifier is unchanged from Hyb_0: $\mathsf{Rec}_{\mathrm{rw}}$ is initialized on τ_{ext}, now using oracle H, and E^H receives the resulting transcript $\widehat{\tau}$ and invocation access obtained from the initialized reconstructor. Thus, when E invokes its prover with an oracle K, a fresh $\widehat{S}^K_{\mathsf{Rec}_{\mathrm{rw}}}$ session is used.

The transition from Hyb_0 to Hyb_1 is exactly one application of standard indifferentiability. To see this, define a distinguisher $\mathcal{D}^{O_1,O_2}$ that samples $\rho \xleftarrow{\$} \Omega_{\mathcal{P}'}$, runs $\mathcal{P}'[\rho]^{O_1}$, runs V^{O_2}, initializes $\mathsf{Rec}_{\mathrm{rw}}$ using O_2, runs E^{O_2} with the resulting reconstructed invocation access, and outputs 1 exactly when the verifier accepts and extraction fails. With $(O_1, O_2) = (\pi, \mathsf{C}^\pi)$ this is Hyb_0, whereas with $(O_1, O_2) = (\mathsf{S}^H, H)$ it is Hyb_1. Hence $|p_0 - p_1| \le \mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{C},\mathcal{I}',\mathcal{I}}(\mathcal{D}, \mathsf{S})$. The running-time argument below proves that $\mathcal{D}$ is probabilistic polynomial time, so this quantity is negligible by standard indifferentiability.

HYBRID Hyb_2. The initial prover execution and the verifier execution are exactly as in Hyb_1. In particular, we run $\mathcal{P}'[\rho]^{\mathsf{S}[\omega]^H}$ and record $\tau_{\mathrm{ext}} = \tau(\mathcal{P}'[\rho] \leftrightarrow \mathsf{S}[\omega]^H)$ and $\tau_{\mathrm{int}} = \tau(\mathsf{S}[\omega] \leftrightarrow H)$, and then run the verifier using the same oracle H.

We now change only what is given to E. Instead of initializing $\mathsf{Rec}_{\mathrm{rw}}$ and giving E the reconstructed transcript and reconstructed invocation access, we give it the true transcript τ_{int} and invocation access to $\langle \mathcal{P}'[\rho] \circ \mathsf{S}[\omega] \rangle$.

The transition from Hyb_1 to Hyb_2 is exactly the reconstruction-for-rewinding guarantee. To instantiate Definition 5, take $A = \mathcal{P}'$. Let $B^H(x, \mathsf{pf})$ run $b \xleftarrow{\$} \mathsf{V}^H(x, \mathsf{pf})$ while recording τ_{V} and return (b, τ_{V}). Given (x, pf), (b, τ_{V}), a transcript τ, and invocation access to a prover, let T^H run $\mathsf{E}^H(x, \mathsf{pf}, \tau, \tau_{\mathsf{V}}; \cdot)$ with that invocation access and output 1 exactly when $b = 1$ and extraction fails.

With these choices, the reconstructed experiment in Definition 5 is precisely Hyb_1, while the true experiment is precisely Hyb_2. Therefore

$$|p_1 - p_2| \le \mathsf{Adv}^{\text{rw-rec}}_{\mathsf{S},\mathsf{Rec}_{\mathrm{rw}},\mathcal{I}}(\mathcal{P}', B, T).$$

The algorithms B and T are probabilistic polynomial time by the running-time argument below. Thus this quantity is negligible by reconstruction for rewinding.

IDENTIFYING THE FINAL HYBRID. Recall that $\widehat{\mathcal{P}}$ has random tape (ρ, ω) and is defined by $\widehat{\mathcal{P}}[\rho, \omega]^H = \mathcal{P}'[\rho]^{\mathsf{S}[\omega]^H}$. Hence the initial prover execution in Hyb_2 is exactly the initial execution of $\widehat{\mathcal{P}}[\rho, \omega]^H$.

Moreover, the oracle transcript of this ideal-model prover is exactly $\tau_{\mathrm{int}} = \tau(\mathsf{S}[\omega] \leftrightarrow H)$, and resetting $\widehat{\mathcal{P}}[\rho, \omega]$ gives exactly invocation access to $\langle \mathcal{P}'[\rho] \circ \mathsf{S}[\omega] \rangle$. Thus Hyb_2 is exactly the ideal-oracle rewinding knowledge-soundness experiment for $\widehat{\mathcal{P}}$ and E. Consequently, $p_2 = \mathsf{Adv}^{\mathrm{rw}}_{\mathcal{R},\mathsf{V},\mathcal{I}}(\widehat{\mathcal{P}}, \mathsf{E})$.

Combining the two hybrid transitions gives

$$\begin{aligned}\mathsf{Adv}^{\mathrm{rw}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}', \mathsf{E}') \le\ & \mathsf{Adv}^{\mathrm{rw}}_{\mathcal{R},\mathsf{V},\mathcal{I}}(\widehat{\mathcal{P}}, \mathsf{E}) \\ & + \mathsf{Adv}^{\text{rw-rec}}_{\mathsf{S},\mathsf{Rec}_{\mathrm{rw}},\mathcal{I}}(\mathcal{P}', B, T) + \mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{C},\mathcal{I}',\mathcal{I}}(\mathcal{D}, \mathsf{S}),\end{aligned}$$

as claimed.

RUNNING TIME. The initial execution of $\mathcal{P}'$ and the verifier have polynomial cost and produce polynomial-size transcripts. Initialization of $\mathsf{Rec}_{\mathrm{rw}}$ therefore has polynomial cost and returns a polynomial-size transcript. By Section 3, E has a worst-case polynomial bound for every such input and sequence of well-formed replies. In particular, it makes only polynomially many prover invocations, even with reconstructed invocation access.

Each invocation runs the fixed polynomial-time prover $\mathcal{P}'$ and hence supplies only polynomially many primitive queries of polynomial length. Over all invocations, the external request size L in Definition 3 is therefore polynomial. That definition bounds all reconstruction work and all queries to the supplied oracles by a polynomial in λ, L. The supplied oracle algorithms have polynomial total cost on this polynomial interaction, as required in Section 3. In the true reconstruction experiment, the same argument uses the polynomial-time simulator S instead. Finally, every direct call to C^π can be evaluated in polynomial time, and all transcript copying and relation tests have polynomial cost. Thus E', $\mathcal{D}$, and T have worst-case polynomial running time; B, which runs the verifier, does as well. Counting calls to the two external interfaces gives the stated query bounds.

Corollary 3. *If the argument has rewinding knowledge soundness relative to $\mathcal{I}$, $\mathsf{C}^{\mathcal{I}'}$ is indifferentiable from $\mathcal{I}$ via S, and S admits reconstruction for rewinding, then, for every probabilistic polynomial-time $\mathcal{P}'$, there is a probabilistic polynomial-time extractor E' for which $\mathsf{Adv}^{\mathrm{rw}}_{\mathcal{R},\mathsf{V},\mathsf{C},\mathcal{I}'}(\mathcal{P}', \mathsf{E}')$ is negligible.*

Proof. For the fixed $\mathcal{P}'$, apply rewinding knowledge soundness to the ordinary probabilistic polynomial-time prover $\widehat{\mathcal{P}}$ defined in Theorem 3. This gives a polynomial-time extractor E fixed independently of the sampled tapes (ρ, ω). Apply that theorem. The ideal-model extraction error is negligible by knowledge soundness, and the two comparison errors are negligible by reconstruction and indifferentiability. The theorem also proves E' is probabilistic polynomial time.

Standard Indifferentiability is Insufficient. We give an example in which the construction is perfectly indifferentiable from a random oracle, but extraction does not transfer for a particular cheating prover $\mathcal{P}'$. An extractor for $\widehat{\mathcal{P}}^{H} = \mathcal{P}'^{\mathsf{S}^{H}}$ always recovers a witness in polynomial time. In contrast, assuming that discrete logarithms are hard, every polynomial-time extractor for $\mathcal{P}'^{\pi}$ recovers a witness only with negligible probability.

Let Grp be a deterministic polynomial-time algorithm that, on input 1^{λ}, returns a description of a cyclic group $\mathbb{G}$ of prime order p together with a generator g. Group elements have canonical polynomial-length binary encodings, and group operations and membership testing are polynomial time. Write $(\mathbb{G}, p, g) = \mathsf{Grp}(1^{\lambda})$. Assume that, for every probabilistic polynomial-time B,

$$\Pr\Big[w \xleftarrow{\$} \mathbb{Z}_p;\ Y = g^{w} :\ B(\mathbb{G}, p, g, Y) = w\Big]$$

is negligible in λ. Fix a polynomial-time injective encoding of pairs of group elements as bit strings and denote it by $\langle \cdot, \cdot \rangle$.

Let $\ell_p = \lceil \log_2 p \rceil + 2\lambda$. Choose U uniformly from $\{0, \ldots, 2^{\ell_p} - 1\}$, and let μ_p denote the distribution of $U \bmod p$.

For each $a \in \{0, \ldots, p-1\}$, there are either $\lfloor 2^{\ell_p}/p \rfloor$ or $\lceil 2^{\ell_p}/p \rceil$ integers $u \in \{0, \ldots, 2^{\ell_p} - 1\}$ satisfying $u \bmod p = a$. Hence $\Pr[U \bmod p = a]$ differs from $1/p$ by at most $2^{-\ell_p}$. Summing these differences shows that the statistical distance between μ_p and the uniform distribution on $\mathbb{Z}_p$ is at most $p/2^{\ell_p} \le 2^{-2\lambda}$.

Let $H : \{0,1\}^{*} \to \mathbb{Z}_p$ be a random oracle. A primitive $\pi \xleftarrow{\$} \mathcal{I}'$ is sampled by choosing H and independent $w, r \xleftarrow{\$} \mu_p$. It has the interfaces $\pi.\mathsf{Hash}(m) = H(m)$, $\pi.\mathsf{Com}() = (Y, A) = (g^{w}, g^{r})$, and $\pi.\mathsf{Resp}() = r + H(\langle Y, A \rangle) w \pmod p$. Define $\mathsf{C}^{\pi}(m) := \pi.\mathsf{Hash}(m)$.

The simulator S^{H} samples independent $w, r \xleftarrow{\$} \mu_p$ using the first two ℓ_p-bit blocks of its tape, forwards Hash queries to H, returns (g^{w}, g^{r}) on Com queries, and returns $r + H(\langle g^{w}, g^{r} \rangle) w \pmod p$ on Resp queries.

Lemma 2. *For every distinguisher $\mathcal{D}$, $\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{C},\mathcal{I}',\mathcal{I}}(\mathcal{D}, \mathsf{S}) = 0$, where $\mathcal{I}$ is the random-oracle distribution just specified.*

Proof. In both experiments, H, w, r are independent with the same distributions, the second interface is H, and every answer at the first interface is computed by the same formula from these three values. The two pairs of oracle interfaces therefore have the same distribution. □

For each λ, consider the Fiat-Shamir Schnorr proof [Sch91, PS00] for $\mathcal{R}_{\mathrm{dl},\lambda} = \{(Y, w) \in \mathbb{G} \times \mathbb{Z}_p : Y = g^{w}\}$. When λ is clear, we write $\mathcal{R}_{\mathrm{dl}}$. A proof is (A, z). The verifier first checks that $Y, A \in G$ and that z is a canonical encoding of an element of $\mathbb{Z}_p$; it rejects otherwise. Next, it sets $c = H(\langle Y, A \rangle)$ and accepts exactly when $g^{z} = A Y^{c}$.

Let $\mathcal{P}'$ be the deterministic oracle algorithm that queries Com to obtain (Y, A), queries Resp to obtain z, and returns $(Y, (A, z))$.

Theorem 4. *Under the discrete-logarithm assumption, the following hold.*

(i) The probabilistic ideal-oracle prover $\widehat{\mathcal{P}}$ obtained by running $\mathcal{P}'$ through S^{H} has a single probabilistic polynomial-time rewinding extractor whose failure probability is zero.

(ii) In the experiment with $\pi \xleftarrow{\$} \mathcal{I}'$, the prover $\mathcal{P}'$ produces an accepting proof with probability one, but every probabilistic polynomial-time extractor has negligible probability of outputting w such that $Y = g^{w}$.

(iii) The simulator S does not admit reconstruction for rewinding: for $A = \mathcal{P}'$ and the algorithms B, T defined in the proof, every probabilistic polynomial-time rewinding reconstructor $\mathsf{Rec}_{\mathrm{rw}}$ has reconstruction advantage at least $1 - \mathrm{negl}(\lambda)$.

Proof. All arithmetic on exponents is modulo p.

(i) Extraction with the simulator. The prover outputs $(Y, (A, z))$, where $Y = g^w$, $A = g^r$, and $z = r + cw$ for $c = H(\langle Y, A\rangle)$. The extractor E reads c from the recorded query-answer pair $(\langle Y, A\rangle, c)$ in τ_{P}. It then reruns the prover with the same random tape, using the oracle

$$K(m) = \begin{cases} c+1, & m = \langle Y, A\rangle, \\ H(m), & \text{otherwise.} \end{cases}$$

The simulator uses the same w, r in this second execution. Consequently, the prover returns the same Y, A, but now its response is $z' = r + (c+1)w$. The extractor outputs $z' - z = (r + (c+1)w) - (r + cw) = w$. This procedure takes polynomial time and succeeds for every choice of w, r. To define it on all inputs, let E return $\perp$ if the required query-answer pair is missing or any required value is malformed.

(ii) Extraction with the primitive. The proof produced by $\mathcal{P}'^{\pi}$ always verifies, since $g^z = g^{r+cw} = AY^c$. Fix a polynomial-time extractor E', and let α be its probability of recovering the witness. We construct an algorithm that uses E' to solve discrete logarithms. Given a challenge $Y = g^w$ for uniform $w \in \mathbb{Z}_p$, the algorithm does not know w. It chooses independent uniform $c, z \in \mathbb{Z}_p$ and sets $A = g^z Y^{-c}$. It answers $H(\langle Y, A\rangle)$ with c. For every other input to H, it chooses an independent uniform answer on the first query and returns that same answer on subsequent queries. It simulates the primitive by answering Com with (Y, A), Resp with z, and Hash queries with this H. Using these answers, the algorithm runs $\mathcal{P}'$ and the verifier, records $\tau_{\text{ext}}, \tau_{\mathsf{V}}$, and gives the resulting inputs to E'. It also answers all direct primitive queries made by E'. Whenever E' requests another execution of $\mathcal{P}'$ with an oracle K, it runs the deterministic code of $\mathcal{P}'$ with that K. None of these operations requires knowledge of w. If E' returns a valid witness, that witness is the discrete logarithm of Y.

We now compare the distribution of these inputs and answers with the extraction experiment. For this calculation, let $r = z - cw$; the algorithm itself need not compute r. For every fixed w, c, the value r is uniform because z is uniform. Thus w, r, c are independent and uniform, and $A = g^r$, $z = r + cw$. In particular, the value c assigned to $H(\langle Y, A\rangle)$ is uniform and independent of Y, A. All other values of H are independent and uniform as well. The simulation therefore has exactly the distribution of the extraction experiment with uniform w, r, as desired.

The actual primitive samples w, r independently from μ_p. Since μ_p is within statistical distance $2^{-2\lambda}$ of uniform, the joint distributions of H, Y, A, z in these two experiments differ by at most $\delta_\lambda = 2^{(1-2\lambda)}$. The reduction with exact uniform sampling therefore succeeds with probability at least $\alpha - \delta_\lambda$.

To implement the reduction in worst-case polynomial time, use μ_p for each uniform sample described above. Let $q_{\text{samp}}(\lambda)$ bound the number of these samples, including c, z and all new answers to H. Such a polynomial bound exists because E' is polynomial time. Replacing each sample changes its distribution by at most $2^{-2\lambda}$, so all replacements together change the success probability by at most $q_{\text{samp}}(\lambda)2^{-2\lambda}$. The resulting polynomial-time algorithm succeeds with probability at least $\alpha - \delta_\lambda - q_{\text{samp}}(\lambda)2^{-2\lambda}$. The discrete-logarithm assumption implies that α is negligible.

(iii) Impossibility of reconstruction. Use $\mathcal{P}'$ as the first algorithm in the reconstruction experiments. Let B run the verifier and return its decision and query transcript. Let T run the extractor E from part (i), using the transcript and prover access supplied in the experiment. It outputs 1 exactly when the verifier accepts and E returns a valid witness.

With the actual simulator and its actual transcript, part (i) shows that T always outputs 1. Thus $\Pr\left[\mathsf{True}^{\text{rw}}_{\mathsf{S},\mathcal{I}}(\mathcal{P}', B, T) = 1\right] = 1$.

Fix any polynomial-time reconstructor Rec_{rw}, and let β be the probability that T outputs 1 in the reconstructed experiment. We show that β is negligible by constructing an extractor for $\mathcal{P}'^{\pi}$.

This extractor initializes Rec_{rw} on τ_{ext}, answering its queries to H through $\pi.\mathsf{Hash}$. It then runs E with the reconstructed transcript, again answering its queries to H through $\pi.\mathsf{Hash}$. When E requests a prover execution with an oracle K, the extractor calls $\mathsf{Rec}_{\text{rw}}.\mathsf{Start}()$ and reruns $\mathcal{P}'$, answering each primitive query q by $\mathsf{Rec}_{\text{rw}}.\mathsf{Answer}^K(q)$. It returns the witness produced by E.

This is a polynomial-time extractor: initialization has polynomial cost, and E makes only one further prover invocation. Moreover, its success probability is exactly β. Indeed, π and S^H generate H, Y, A, z with the same joint distribution, and every subsequent operation is the same as in the reconstructed experiment. Part (ii) therefore implies that β is negligible. The reconstruction advantage is $\mathsf{Adv}^{\text{rw-rec}}_{\mathsf{S},\mathsf{Rec}_{\text{rw}},\mathcal{I}}(\mathcal{P}', B, T) = 1 - \beta \geq 1 - \text{negl}(\lambda)$ as required. □

Finally, we remark that this is a prover-specific separation: it compares extraction for the fixed prover P' with extraction for its composition with S. In particular, it does not assert that the Schnorr argument satisfies our worst-case polynomial-time knowledge-soundness definition against every ideal-world prover.

6 Tagged Prefix-Free Merkle-Damgård

Throughout, $\mathcal{I}'$ is the distribution of the random compression function F defined below, $\mathcal{I}$ is the distribution of a random oracle $H : \{0,1\}^* \to \{0,1\}^n$, and the construction is $\mathsf{C}^F = \mathsf{PFMD}^F$. Thus the first interface in the indifferentiability experiment is the primitive interface F, and the second is the hash interface PFMD^F; in the ideal experiment these are replaced by $\mathsf{S}_{\mathrm{pf}}^H$ and H, respectively. We give an explicit simulator S_{pf}, prove indifferentiability, and prove the two reconstruction properties used by the straight-line and rewinding transfer theorems.

Construction. Let $n = n(\lambda)$ and $b = b(\lambda)$ be polynomially bounded positive integers. Fix $\mathsf{iv} \in \{0,1\}^n$ and let $\mathcal{B} = \{0,1\}^b$. Let $\mathsf{Pad} : \{0,1\}^* \to (\mathcal{B} \times \{0,1\})^+$ be injective, and suppose every encoding has the form $\mathsf{Pad}(m) = ((B_1,0),\ldots,(B_{\ell-1},0),(B_\ell,1))$. Define $\mathsf{Prefix}(s) = 1$ exactly when s is a strict prefix of $\mathsf{Pad}(m)$ for some m. Define $\mathsf{RemovePad}(s) = m$ when $s = \mathsf{Pad}(m)$, and $\mathsf{RemovePad}(s) = \perp$ when s is not a complete encoding. We assume that Pad, Prefix, and $\mathsf{RemovePad}$ run in polynomial time. For the cumulative running-time requirement in Definition 3, we additionally assume that the padding can be processed one tagged block at a time: after a strict padded prefix s has been processed, deciding whether $s \,\|\, (B,0)$ is again a strict padded prefix and retaining the information needed for the next block takes time polynomial in λ; if $s \,\|\, (B,1)$ is a complete encoding, recovering the corresponding message m takes time at most $p_{\mathrm{pad}}(\lambda)(1+|m|)$ for some polynomial p_{pad}. Standard length-prefix and delimiter-based paddings have this property. The tags make the encoding prefix-free: a complete encoding ends in a pair whose tag is 1, whereas every nonempty strict prefix of an encoding ends in a pair whose tag is 0. Hence $\mathsf{Pad}(m)$ cannot be a strict prefix of $\mathsf{Pad}(m')$ for any m, m'.

Let $F : \{0,1\}^n \times \mathcal{B} \times \{0,1\} \to \{0,1\}^n$ be a random function. For $\mathsf{Pad}(m) = ((B_1,t_1),\ldots,(B_\ell,t_\ell))$, set $h_0 = \mathsf{iv}$ and $h_i = F(h_{i-1}, B_i, t_i)$ for $1 \le i \le \ell$. Define $\mathsf{PFMD}^F(m) = h_\ell$.

The Simulator. Let $H : \{0,1\}^* \to \{0,1\}^n$ be a random oracle. Write the simulator's ordinary random tape as independent uniform n-bit blocks $\omega = (\omega_1, \omega_2, \ldots)$, independently of H. A call to $\mathsf{Next}()$ returns ω_j and increments a session counter j, initially 1. No other step consumes private random bits. In particular, the same tape position can be used for different primitive queries in different reset invocations.

A simulator session has three tables and the counter j. A reset clears these tables, restores the initial values below, and sets $j = 1$, retaining the tape ω.

- $T[q]$ is the answer already returned on the primitive query q.
- $R[u]$ is one tagged block sequence s such that starting from iv and following the previously answered tag-0 primitive queries for the blocks of s reaches the state u. Initially $R[\mathsf{iv}]$ is the empty sequence.
- $U[m]$ is the value returned by $H(m)$ once the simulator has queried H on m.

An entry of R or U is never changed after it is set. On a primitive query $q = (u, B, t)$, the simulator proceeds as follows.

S1. If $q \in \mathsf{Dom}(T)$, return $T[q]$ and make no change.
S2. Suppose $q \notin \mathsf{Dom}(T)$ and $t = 1$. If $u \in \mathsf{Dom}(R)$ and

$$m = \mathsf{RemovePad}(R[u] \,\|\, (B,1)) \neq \perp,$$

then set $y = U[m]$ if $m \in \mathsf{Dom}(U)$; otherwise query $y \leftarrow H(m)$ and set $U[m] = y$. In every other case set $y = \mathsf{Next}()$. Set $T[q] = y$ and return y.
S3. Suppose $q \notin \mathsf{Dom}(T)$ and $t = 0$. Set $y = \mathsf{Next}()$ and $T[q] = y$. If $u \in \mathsf{Dom}(R)$, $\mathsf{Prefix}(R[u] \,\|\, (B,0)) = 1$, and $y \notin \mathsf{Dom}(R)$, then set $R[y] = R[u] \,\|\, (B,0)$. Finally, return y.

Thus, within one simulator session, every first occurrence of a primitive query whose answer is not determined by H consumes exactly one call to Next, and hence exactly one n-bit block of the ordinary tape. Branch decisions, table updates, and repeated queries consume no private randomness.

The update in Step S3 is performed only when the query is first answered. Thus, if $(u, B, 0)$ is queried before u enters $\mathsf{Dom}(R)$, that earlier query is not reconsidered later. This point matters in the indifferentiability proof below: reaching such a previously queried state out of order is one of the low-probability events that the proof must exclude.

The simulator is polynomial time. Conceptually, $R[u]$ denotes the full tagged block sequence above, but an implementation stores each such sequence by a pointer to the preceding sequence together with the newly appended block and the information retained by the block-by-block padding procedure just assumed. Hence extending a tag-0 path does not require copying or rescanning the whole prefix. Recovering a complete message on a tag-1 query costs at most $p_{\mathrm{pad}}(\lambda)(1+|m|)$, and the message m is then used as the corresponding query to H.

6.1 Proof of Indifferentiability

We use the interface order fixed in Section 3. In the real experiment the distinguisher receives (F, PFMD^F), with the primitive interface first and the hash interface second. In the ideal experiment it receives $(\mathsf{S}_{\mathrm{pf}}^H, H)$ in the same order.

Theorem 5 (Indifferentiability). *Let $\mathcal{D}$ be any distinguisher. Suppose that, on every execution, if its second-interface queries are $m_1, \ldots, m_{q_H}$ and it makes q_F queries to its first interface, then*

$$q_F + \sum_{j=1}^{q_H} |\mathsf{Pad}(m_j)| \le Q,$$

where $|\mathsf{Pad}(m_j)|$ denotes the number of tagged blocks in the encoding. Assume $Q \le 2^{n-3}$. Then

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{PFMD},\mathcal{I}',\mathcal{I}}(\mathcal{D}, \mathsf{S}_{\mathrm{pf}}) \le \frac{6Q^2}{2^n}.$$

Consequently, if $Q^2/2^n$ is negligible for every polynomially bounded Q, then $\mathsf{PFMD}^{\mathcal{I}'}$ is indifferentiable from $\mathcal{I}$ via S_{pf} in the sense of Section 3.

Proof. We couple the real and ideal experiments and show that their answers are identical unless one of two kinds of unlikely events occurs. For this single-execution proof only, we may generate the unforced answers with an independent lazy random function $G : \{0,1\}^n \times \mathcal{B} \times \{0,1\} \to \{0,1\}^n$. Each unforced new primitive input uses one previously unread value of G, and repeated queries use T. Thus this produces exactly the one-session distribution of the sequential-tape simulator. Hidden hash evaluations in the coupling may inspect further values of G, without changing the simulator's visible session tables. Indeed, in a single session, list the fresh primitive inputs whose answers are not forced by H in the order in which they first occur. The sequential-tape simulator assigns to them the independent uniform blocks $\omega_1, \omega_2, \ldots$, while the lazy-function view assigns the independent uniform values $G(q)$ to those same first occurrences; repeated inputs are answered from T in either description. The two descriptions therefore have the same joint distribution for one session. This identification is used only here: after a reset, the fixed-randomness simulator deliberately reuses the same tape positions for possibly different primitive inputs, which would not be captured by keeping one lazy function G across sessions.

Consider a strict prefix

$$s = ((B_1, 0), \ldots, (B_i, 0))$$

of some padded message. Starting from iv, evaluating these i compression calls produces an n-bit state. Such states can arise in two ways during the coupled execution: the distinguisher may follow the corresponding primitive queries starting from iv, or the real hash interface may encounter the prefix while evaluating $\mathsf{PFMD}^F(m)$. Across the whole execution there are at most Q compression-function evaluations, and hence at most Q occasions on which a new state of this form can first be fixed.

We declare *bad* if either of the following happens.

(1) Two different explored strict padded prefixes reach the same n-bit state, or a nonempty explored strict padded prefix reaches iv.
(2) Let u be the state reached after some nonempty strict padded prefix. Before the distinguisher has obtained u by querying the blocks of that prefix in order starting from iv, the same n-bit value u appears elsewhere in the distinguisher's transcript: either as the first component of a primitive query, or as an oracle answer other than the answer that reveals u by following that prefix in order.

Write $N = 2^n$. The case $Q = 0$ is immediate, so assume $Q \ge 1$. For a nonempty strict padded prefix s, say that s is *explored* at the first moment when the coupled execution fixes the state reached after processing s from iv. This can happen either while the real hash interface evaluates a padded message containing s, or when the distinguisher itself queries the blocks of s consecutively and in order from iv. We call that state the *label* of s. An explored prefix is *revealed* once the distinguisher completes this in-order primitive traversal of s, so that the primitive answer returning its label appears in the visible transcript; until that moment its label is *hidden*. These notions are monotone: a prefix is explored at most once, and a revealed label never becomes hidden again. At most Q nonempty strict prefixes are explored. Before bad occurs, a hidden label is neither iv nor any value already visible as a primitive-query first component or an oracle answer.

A new strict-prefix label is obtained from a fresh entry of G before bad occurs. To see this, an earlier evaluation along the same prefix would already have stored that label. An earlier evaluation along a different prefix would require a collision between their predecessor labels, unless the prefixes and next blocks coincide. An earlier out-of-order primitive evaluation would already have used the unrevealed predecessor as a query's first component. These alternatives are either the same prefix or a preceding bad event. A tag-1 entry cannot be this entry because the new strict prefix ends in tag 0.

We also need a conditional statement for a label that was sampled earlier but is still hidden. Fix the visible history, the explored prefix strings, which of them are hidden, and the labels of all explored prefixes except one hidden label v. Do not fix v or unexposed random-function entries. Condition also on the absence of bad so far. The conditional law of v is uniform over the values not among those other labels, iv, and the visible first components and answers. Indeed, interchange any two such values by a permutation σ of $\{0,1\}^n$, fixing all excluded values. Transform the auxiliary functions by $G(u,B,t) \mapsto \sigma(G(\sigma^{-1}(u),B,t))$ and $H(m) \mapsto \sigma(H(m))$. Their joint random-function law is invariant. The visible history is unchanged, as are the explored block strings and every equality test already performed. Prefix labels are relabelled by σ. The simulator uses state labels only for table lookup and equality, so its branch decisions are unchanged. This is a probability-preserving bijection between the conditioned histories having either candidate value for v. The argument fixes only the visible oracle values, not an entire sampled H.

There are at most Q other prefix labels, Q visible first components, and Q visible answer occurrences. Thus the number of excluded values is at most $3Q+1 \le 4Q \le N/2$. Any next query's first component, chosen from the visible history, equals any particular still-hidden label with conditional probability at most $2/N$.

We charge the first bad event in one of three ways.

(a) When a new strict-prefix label is sampled, charge equality with iv, another strict-prefix label, or a previously visible first component or answer. There are at most $3Q$ forbidden values and at most Q such fresh draws. The contribution is at most $3Q^2/N$.
(b) When a primitive query is chosen, charge equality of its first component with an already sampled, still-hidden label. The conditional symmetry argument gives probability at most $2Q/N$ per query and hence at most $2Q^2/N$ in total.
(c) When a fresh visible answer that is not the legitimate revelation of a prefix label is sampled, charge equality with a still-hidden label. Before the first bad event it is a fresh uniform entry of G or H, independent of the already sampled hidden labels. The contribution is at most Q^2/N. A repeated answer cannot cause the first bad event: if its matching hidden label existed earlier, the earlier occurrence would already have been bad; if that label was sampled later, case (a) would have been bad. A legitimate prefix revelation is excluded by definition.

These cases also cover an out-of-order query made before its target prefix is explored: its first component is then a previously visible value in case (a). Summing the conditional first-event bounds gives

$$\Pr[\mathsf{bad}] \le \frac{3Q^2+2Q^2+Q^2}{N} \le \frac{6Q^2}{2^n}.$$

It remains to show that, as long as bad has not occurred, the real and ideal experiments return exactly the same answers.

We use the following invariant. Whenever $R[u]=s$ in the ideal experiment, the real compression function, started from iv and evaluated on the tagged blocks of s, reaches the same state u. Moreover, if the distinguisher has queried the blocks of a strict padded prefix s in order starting from iv, then the ideal table contains $R[u]=s$ for the state u returned after the last block of s. Event (1) guarantees that two different strict padded prefixes never compete for the same entry of R.

We remark here that a primitive query that is new to the simulator need not be fresh in the real compression table. In particular, an earlier hash-interface evaluation may already have fixed its value. In that case, we reuse the value fixed by the coupling. For a non-final entry, this is the corresponding value of G, while for a final entry completing m, it is $H(m)$. We sample a new coupled value only when the relevant entry has not previously been fixed.

The invariant is immediate initially because $R[\mathsf{iv}]$ is the empty sequence. Suppose first that the distinguisher makes a new primitive query $q=(u,B,0)$. If $u \in \mathsf{Dom}(R)$ and $R[u] \parallel (B,0)$ is a strict padded prefix, then the query is exactly the next real compression call after that prefix. If the entry was fixed by an earlier hash evaluation, both experiments reuse the corresponding value of $G(q)$. Otherwise we assign the same fresh uniform value to $F(q)$ and $G(q)$. Both experiments therefore return the same

answer. Since bad has not occurred, this state is not already represented, and the ideal experiment adds the corresponding entry to R.

Event (1) guarantees that the new state is not already assigned to a different strict prefix. If $u \notin \mathsf{Dom}(R)$, or the extended sequence is not a strict padded prefix, the ideal simulator answers with $G(q)$. We reuse the same value for $F(q)$ if it was already fixed; otherwise, we assign that value to the fresh real entry. If this query were later to become a query lying strictly inside the evaluation of a padded message without the distinguisher first following the preceding blocks in order, event (2) would occur. Thus, while bad is false, answering such a query with $G(q)$ can never create a later inconsistency.

Now suppose the distinguisher makes a new primitive query $q = (u, B, 1)$. If

$$u \in \mathsf{Dom}(R) \qquad \text{and} \qquad \mathsf{RemovePad}(R[u] \parallel (B, 1)) = m,$$

then, by the invariant, q is exactly the last compression call in the real evaluation of m. We couple its real answer with $H(m)$, which is exactly the answer returned by the simulator. If an earlier hash evaluation already fixed this final compression entry, its value is already coupled to $H(m)$, and both experiments reuse that value. Otherwise, before bad occurs, both the final compression entry and $H(m)$ are fresh: an earlier hash-interface evaluation of m would already have fixed the final entry, while an earlier simulator query on m would already have placed this same primitive query in T. We then assign the final compression entry and $H(m)$ the same fresh uniform n-bit value, preserving their required marginal distributions.

If the displayed condition fails, the simulator answers with $G(q)$ and we use the same value for $F(q)$ in the real experiment. As in the tag-0 case, if this primitive query could later become the last call of a padded message without the preceding state first being obtained in order, event (2) would occur. Hence no inconsistency arises while bad is false.

Finally suppose the distinguisher queries the hash interface on a message m. In the ideal experiment the answer is $H(m)$. In the real experiment evaluate the blocks of $\mathsf{Pad}(m)$ from iv. Every nonfinal compression entry that was already fixed has, by the preceding cases and the invariant, the same value in both experiments. For every fresh nonfinal entry we use the corresponding fresh value of G; evaluating an as-yet unused value of the lazy random function in the coupling does not change the ideal distribution. If one of the resulting internal states had already appeared in the distinguisher's transcript without being obtained by following the preceding blocks in order, event (2) would occur. Thus the final compression input is either already fixed consistently to $H(m)$ or is fresh. In the latter case we set its real value to the fresh value $H(m)$. The real hash answer is therefore exactly $H(m)$.

By induction over the distinguisher's queries, the two transcripts are identical until bad occurs. The distinguishing advantage is therefore at most $\Pr[\mathsf{bad}]$, which is at most $6Q^2/2^n$.

Exact Transcript Reconstruction. Fix a probabilistic oracle algorithm A, a random tape ρ for A, and a random tape ω for S_{pf}. For a completed execution of $A[\rho]^{\mathsf{S}_{\mathrm{pf}}[\omega]^H}$, write

$$\tau_{\mathrm{ext}} = \tau(A[\rho] \leftrightarrow \mathsf{S}_{\mathrm{pf}}[\omega]^H), \quad \tau_{\mathrm{int}} = \tau(\mathsf{S}_{\mathrm{pf}}[\omega] \leftrightarrow H).$$

Define $\mathsf{Rec}_{\mathrm{pf}}$ as follows. On input $\tau_{\mathrm{ext}} = ((q_1, y_1), \ldots, (q_s, y_s))$, it initializes T' and U' as empty tables, initializes τ' as the empty sequence, and sets $R'[\mathsf{iv}]$ to the empty sequence. It scans the entries in order. At entry (q_i, y_i) it performs the same tests as the simulator, using T', R', U', but it uses the recorded answer y_i instead of reading the next private tape block. Whenever the simulator would make its first query $H(m)$, the replay algorithm appends (m, y_i) to τ' and sets $U'[m] = y_i$. It performs the same updates to T' and R' as the simulator. The output is τ'. This description specifies the behavior needed on transcripts arising from genuine simulator executions. To make $\mathsf{Rec}_{\mathrm{pf}}$ a total deterministic algorithm, on an arbitrary input it additionally checks query and answer lengths, consistency of repeated queries, and consistency of cached answers while scanning; if any check fails, it returns the empty transcript. Any fixed efficient completion on malformed inputs would suffice for the theorem below, whose correctness claim concerns only genuine simulator transcripts.

Theorem 6 (Exact Replay). *For every H, every random tape ρ of A, every random tape ω of S_{pf}, and every completed execution of $A[\rho]^{\mathsf{S}_{\mathrm{pf}}[\omega]^H}$, $\mathsf{Rec}_{\mathrm{pf}}(\tau_{\mathrm{ext}}) = \tau_{\mathrm{int}}$. At the end of the scan, the replay tables (T', R', U') also equal the session tables (T, R, U) of $\mathsf{S}_{\mathrm{pf}}[\omega]^H$ after that execution.*

Proof. We prove by induction on i that, after processing the first i entries of τ_{ext}, the replay tables equal the simulator tables after the corresponding primitive query, and τ' equals the internal H-query transcript up to that point.

The claim is immediate for $i = 0$. Suppose the claim holds before processing (q_i, y_i), where $q_i = (u_i, B_i, t_i)$. If q_i is repeated, the induction hypothesis puts the same entry in T and T'. Both computations return that entry and make no other change.

Suppose q_i is new and $t_i = 1$. By the induction hypothesis, the two computations agree on whether $u_i \in \mathsf{Dom}(R)$ and, when it is, on the value $R[u_i]$. They therefore obtain the same result from $\mathsf{RemovePad}(R[u_i] \| (B_i, 1))$. If the result is $\perp$, or if $u_i \notin \mathsf{Dom}(R)$, both store y_i only in T. If the result is a message $m \in \mathsf{Dom}(U)$, both use the already stored value and add the same entry to T. If the result is a message $m \notin \mathsf{Dom}(U)$, then the simulator obtained y_i by querying $H(m)$. The replay algorithm appends exactly (m, y_i) to τ' and stores the same entries in U' and T'.

Conversely, suppose q_i is new and $t_i = 0$. Both computations store $T[q_i] = y_i$. By the induction hypothesis, they agree on whether $u_i \in \mathsf{Dom}(R)$, on $R[u_i]$ when it exists, and on whether $y_i \in \mathsf{Dom}(R)$. They therefore give the same result to the test in Step S3 and either both add the same entry to R or both leave R unchanged. Thus the induction statement holds after entry i. After the final entry, $\tau' = \tau_{\mathrm{int}}$ and all three replay tables equal the simulator tables. □

The algorithm $\mathsf{Rec}_{\mathrm{pf}}$ is deterministic and makes no oracle queries. Hence Theorem 6 proves exact transcript reconstruction in the sense of Definition 2.

Rewinding Reconstruction. We now define a rewinding reconstructor $\mathsf{Rec}_{\mathrm{pf,rw}}$ with the three entry points from Definition 3. The subroutine $\mathsf{Rec}_{\mathrm{pf,rw}}.\mathsf{Init}^{H}(\tau_{\mathrm{ext}})$ first performs the replay above. Each time replay reaches a call to Next, it appends the recorded answer to a persistent sequence W. Thus, if k private draws were consumed initially, initialization stores $W[1], \ldots, W[k]$ in draw order. It returns the reconstructed internal transcript and makes no queries to H.

The subroutine $\mathsf{Rec}_{\mathrm{pf,rw}}.\mathsf{Start}()$ resets T_s, U_s, R_s to the simulator's initial tables and resets the draw counter to $j = 1$. It does not clear W. On a primitive query, Answer^{K} runs the actual simulator steps with K in place of H. At a call to Next it returns $W[j]$, sampling and appending a fresh uniform n-bit block if this position has never been read in any reconstructed session, and then increments j. All sessions therefore use one fixed ordinary tape. The primitive query at position j need not be the same in different sessions.

Initialization checks query and answer lengths and consistency of repeated queries and cached answers during replay. If a check fails, it returns the empty transcript and initializes W as empty; later sessions still run the same simulator with a fresh tape. This branch never occurs on a true simulator transcript.

Theorem 7 (Exact Reconstruction for Rewinding). *For every probabilistic polynomial-time A, B, T, $\mathsf{Adv}^{\text{rw-rec}}_{\mathsf{S}_{\mathrm{pf}},\mathsf{Rec}_{\mathrm{pf,rw}},\mathcal{I}}(A, B, T) = 0$.*

Proof. Condition on H, on the random tape ρ of A, and on an external transcript τ_{ext} of positive probability in the initial execution. By Theorem 6, $\mathsf{Rec}_{\mathrm{pf,rw}}.\mathsf{Init}$ returns the true internal transcript τ_{int}. The simulator consumes private randomness only through Next. Replay identifies every such call and its output. Consequently, conditioning on $H, \rho, \tau_{\mathrm{ext}}$ fixes exactly the consumed prefix $\omega_1, \ldots, \omega_k$ to the values stored in W. The termination of the initial execution and each of its branches are determined by the supplied oracle answers and this prefix; they impose no condition on the unread suffix. That suffix remains independent and uniform. This is exactly the distribution implemented by persistent W. The key point is that every call to Next uses its entire n-bit block directly as the simulator's private answer, and that answer is visible in τ_{ext}. There is no rejection sampling, truncation, or other hidden use of a consumed block. Hence conditioning on τ_{ext} fixes each consumed raw tape block exactly. This is the feature that fails for the sponge simulator of Section 7: there the transcript determines the selected rank of a draw, but not the entire raw tape block that produced it.

Now fix the coins of B and T as well, and couple the true tape and the tape represented by W to be equal. In every later invocation, both implementations reset the session tables and the draw counter, and run the same simulator code with the same supplied K. By induction on the entire interaction they return the same answers and make the same queries to K, including when changes to K alter the query sequence or the number of private draws. This establishes equality of the joint conditional laws, not just equality of individual session marginals.

Finally, $\mathsf{Rec}_{\mathrm{pf,rw}}.\mathsf{Init}$ makes no query to H, so it does not alter the shared ideal-oracle execution between the initial prover run and the later test. Thus, conditioned on H, ρ, and τ_{ext}, the true and reconstructed rewinding experiments are identically distributed. Averaging over the conditioned values therefore yields zero advantage. □

The reconstructor satisfies the cost requirement of Definition 3. With external interaction size L, there are at most L primitive requests and private draws to process. Each represented prefix has at most L blocks.

Polynomial-time padding recognition and inversion therefore produce messages of length bounded by a polynomial in λ, L; there are at most L such decoding operations and oracle calls. Table operations, tape storage, and all transcript copying are polynomial in the same parameters. This proves one bound $p(\lambda, L)$ on total work and generated traffic. More specifically, for invocations of a fixed worst-case polynomial-time $\mathcal{P}'$, the number of requests in each session is polynomially bounded, so both the session tables and the largest tape position used are polynomially bounded independently of the number of sessions. Finally, a polynomial-time extractor makes only polynomially many such invocations, so the total reconstruction cost is polynomial, as desired.

Corollary 4. *Assume that the parameters are such that $Q^2/2^n$ is negligible for every polynomially bounded Q. Then S_{pf} gives standard indifferentiability of $\mathsf{PFMD}^{\mathcal{I}'}$ from $\mathcal{I}$, admits exact transcript reconstruction, and admits reconstruction for rewinding with advantage 0. Consequently, proof-only straight-line, transcript-dependent straight-line, and rewinding knowledge soundness transfer by Corollaries 1 to 3. For a transfer reduction with worst-case aggregate bound Q as in Theorem 5, its indifferentiability term is at most $6Q^2/2^n$. The transcript- and rewinding-reconstruction terms are exactly 0. The polynomial-time transfer reductions have polynomial worst-case aggregate bounds.*

Proof. Use Theorems 5 to 7 and the three transfer corollaries. Their reductions have polynomial worst-case aggregate bounds, satisfying the query regime in Theorem 5 for all sufficiently large λ under the parameter assumption. The indifferentiability term is $6Q^2/2^n$, and both reconstruction terms are zero. □

7 Fixed-Output Sponge from a Random Permutation

Throughout this section, $\mathcal{I}'$ is the distribution of the tagged forward/inverse interface π_P of a uniformly random permutation P on $\{0,1\}^{r+c}$, $\mathcal{I}$ is the distribution of a random oracle $H : \{0,1\}^* \to \{0,1\}^r$, and the construction in the notation of Section 3 is $\mathsf{C}^{\pi_P} = \mathsf{Sp}^P$. Thus the first interface in the indifferentiability experiment is the primitive interface π_P and the second is the hash interface Sp^P; in the ideal experiment these are replaced by $\mathsf{S}_{\mathrm{sp}}^H$ and H, respectively. We give one simulator S_{sp}, prove a sponge indifferentiability bound for it, and then prove the two reconstruction properties used by the straight-line and rewinding transfer theorems.

7.1 Construction and Primitive Interface

Fix polynomially bounded positive integers $r = r(\lambda)$ and $c = c(\lambda)$, and let $w = r + c$. A string $x \in \{0,1\}^w$ is written $x = (x^A, x^C)$ with $x^A \in \{0,1\}^r$ and $x^C \in \{0,1\}^c$; these are simply the first r bits and the last c bits of x.

Let $\mathsf{pad} : \{0,1\}^* \to (\{0,1\}^r)^+$ be an injective polynomial-time encoding such that the final block of every $\mathsf{pad}(m)$ is nonzero. We assume that membership in the image of pad and inversion on that image are polynomial time. We also assume the following incremental implementation, which is used only for the cumulative running-time statement in the rewinding reconstruction theorem. While a block string is extended one block at a time, the information needed to determine whether it has the form $\mathsf{pad}(m) \,\|\, (0^r)^j$ can be updated with time polynomial in λ per appended block; when a complete encoding $\mathsf{pad}(m)$ is first recognized, recovering m takes at most $p_{\mathrm{pad}}(\lambda)(1 + |m|)$ steps for some polynomial p_{pad}. The usual delimiter- or length-based injective paddings satisfy this condition.

Let P be a uniformly random permutation on $\{0,1\}^w$. We regard the primitive as one tagged oracle π_P defined by

$$\pi_P(+1, x) = P(x), \quad \pi_P(-1, y) = P^{-1}(y).$$

Thus a primitive query specifies whether it is a forward or inverse permutation query. The construction itself uses only the forward direction.

For $\mathsf{pad}(m) = p_1 \cdots p_\ell$, set $s_0 = (0^r, 0^c)$. If $s_{i-1} = (a_{i-1}, d_{i-1})$, define

$$s_i = P(a_{i-1} \oplus p_i, d_{i-1}).$$

Writing $s_\ell = (a_\ell, d_\ell)$, define $\mathsf{Sp}^P(m) := a_\ell$.

For a block string $p \in (\{0,1\}^r)^*$, remove all trailing zero blocks and write uniquely

$$p = p' \,\|\, (0^r)^j,$$

where either p' is empty or its final block is nonzero. Define $\mathsf{SDec}(p) = (m, j)$ when $p' = \mathsf{pad}(m)$, and define $\mathsf{SDec}(p) = \perp$ otherwise. The assumptions on pad make SDec well defined and polynomial time. Indeed, whenever $p = \mathsf{pad}(m) \parallel (0^r)^j$, the integer j is exactly the number of trailing zero blocks of p, because the final block of $\mathsf{pad}(m)$ is nonzero. Removing those blocks therefore recovers $\mathsf{pad}(m)$ uniquely, and injectivity of pad then determines m uniquely.

7.2 The Permutation Simulator

Let $H : \{0,1\}^* \to \{0,1\}^r$ be a random oracle. The private randomness of S_{sp} fixes all random choices made by the simulator. In the fixed-randomness notation of Section 5, a tape $\omega \xleftarrow{\$} \Omega_{\mathsf{S}_{\mathrm{sp}}}$ therefore determines a deterministic oracle algorithm $\mathsf{S}_{\mathrm{sp}}[\omega]$.

Set $\kappa = 2\lambda$ and $\ell_{\mathrm{rnd}} = w + \kappa$. Write $\omega = (\omega_1, \omega_2, \ldots)$ as independent uniform ℓ_{rnd}-bit blocks. A session counter ν starts at 1. For a nonempty set $\mathcal{A}$ of at most 2^w strings, list its elements in lexicographic order as $\mathcal{A} = \{v_0, \ldots, v_{M-1}\}$. The operation $\mathsf{Draw}(\mathcal{A})$ reads the integer R encoded by ω_ν, increments ν, and returns $v_{R \bmod M}$. Every private sampling operation below uses this rule, even when $|\mathcal{A}| = 1$ or $\mathcal{A} = \{0,1\}^r$. Cached values consume no tape block. A reset clears all session tables and resets $\nu = 1$; it retains only the contents of the ordinary tape. In particular, it does not retain a private random function indexed by messages or histories.

We will use two facts about this sampling rule. First, Draw is statistically close to choosing a uniformly random element of its input set. Second, in the rewinding reconstruction we will observe which element was returned and will need to sample a tape block consistent with that observed choice. The following lemma gives both facts.

Lemma 3 (Sampling from a Fixed Tape Block). *Let $D_{\mathrm{rnd}} = 2^{\ell_{\mathrm{rnd}}}$. For $1 \le M \le 2^w$, if R is uniform in $\{0, \ldots, D_{\mathrm{rnd}} - 1\}$, the distribution of $R \bmod M$ is at statistical distance at most $2^{-\kappa}$ from uniform on $\{0, \ldots, M-1\}$. Suppose the returned element has index a, so that $R \bmod M = a$, and put*

$$t_a = 1 + \left\lfloor \frac{D_{\mathrm{rnd}} - 1 - a}{M} \right\rfloor .$$

Conditioned on $R \bmod M = a$, the integer R is uniform on $\{a + Mu : 0 \le u < t_a\}$. This conditional distribution can be sampled within statistical distance $2^{-\kappa}$ in worst-case polynomial time by taking a fresh uniform $(\ell_{\mathrm{rnd}} + \kappa)$-bit integer V and returning $a + M(V \bmod t_a)$.

Proof. For each $a \in \{0, \ldots, M-1\}$, there are either $\lfloor D_{\mathrm{rnd}}/M \rfloor$ or $\lceil D_{\mathrm{rnd}}/M \rceil$ integers R satisfying $R \bmod M = a$. Hence the probability of each possible value of $R \bmod M$ differs from $1/M$ by at most $1/D_{\mathrm{rnd}}$. Summing these differences and dividing by two gives statistical distance at most $M/D_{\mathrm{rnd}} \le 2^{-\kappa}$. Conditioned on $R \bmod M = a$, all integers R satisfying this congruence are equally likely, which gives the stated conditional distribution. Apply the same calculation to $V \bmod t_a$: its error is at most $t_a/2^{\ell_{\mathrm{rnd}}+\kappa} \le 2^{-\kappa}$. The map $u \mapsto a + Mu$ preserves this distance. All the integers have polynomial bit length, so the specified arithmetic and sampling have worst-case polynomial cost.

A simulator session keeps two tables F and F^{-1} describing a partial bijection on $\{0,1\}^w$, a table L, and tables U, V. Initially F, F^{-1}, U, V are empty and $L[0^c] = ((0^r, 0^c), \epsilon)$. An entry $L[d] = ((\alpha, d), p)$ means that, starting from $(0^r, 0^c)$ and using the already fixed forward permutation values corresponding to the blocks of p, the sponge computation reaches the state (α, d). The table U stores $H(m)$ after the simulator first queries H on m, and $V[m, j]$ stores the private r-bit value first chosen for that pair in the current session.

For the current partial permutation, let $\mathsf{Rep} = \mathsf{Dom}(L)$ and let Out be the set of c-bit strings that occur as the last c bits of some element of $\mathsf{Dom}(F)$. Thus $d \in \mathsf{Out}$ exactly when some permutation input already stored in F has last c bits equal to d.

A session also has a fallback flag, initially false. Repeated queries always return their stored answer. Before answering a new query, let $i = |\mathsf{Dom}(F)|$. If $4(i+1) \ge 2^c$, set the fallback flag to true. Once the flag is true, answer a new forward query by the lexicographically first unused output, or a new inverse query by the lexicographically first unused input. Store the resulting input-output pair in F, F^{-1}, make no query to H, consume no private bits, and do not update L, U, V. A new query always has an unused partner, because F is a partial bijection. Resetting clears this flag as well.

The fallback is used only to make the simulator a total oracle algorithm on arbitrary executions. In the query regime of Theorem 8 it is never reached, as verified in the proof of that theorem. Moreover, once

reached, the fallback is deterministic, makes no query to H, and consumes no private tape block. It is therefore replayed exactly by both reconstruction procedures and introduces no additional reconstruction error.

When the flag is false, the simulator $\mathsf{S}_{\mathrm{sp}}^{H}$ answers a tagged primitive query as follows.

S1. On a forward query $(+1, x)$, if $x \in \mathsf{Dom}(F)$, return $F[x]$.

S2. Suppose $(+1, x)$ is a new forward query, write $x = (a, d)$, and suppose $d \in \mathsf{Rep}$ and $\mathsf{Rep} \cup \mathsf{Out} \neq \{0,1\}^c$. Write $L[d] = ((\alpha, d), p)$ and set

$$p^+ = p \parallel (\alpha \oplus a).$$

If $\mathsf{SDec}(p^+) = (m, 0)$, set $z = U[m]$ when $m \in \mathsf{Dom}(U)$; otherwise query $z \leftarrow H(m)$ and set $U[m] = z$. If $\mathsf{SDec}(p^+) = (m, j)$ with $j > 0$, set $z = V[m, j]$ when this entry is already defined; otherwise set $z = \mathsf{Draw}(\{0,1\}^r)$ and $V[m, j] = z$. If $\mathsf{SDec}(p^+) = \perp$, set $z = \mathsf{Draw}(\{0,1\}^r)$.
Let $\mathcal{A}$ be the strings $e \in \{0,1\}^c \setminus (\mathsf{Rep} \cup \mathsf{Out})$ for which $(z, e) \notin \mathsf{Dom}(F^{-1})$, and set $e = \mathsf{Draw}(\mathcal{A})$. Set $y = (z, e)$, set $F[x] = y$ and $F^{-1}[y] = x$, set $L[e] = (y, p^+)$, and return y.
Because e is chosen outside $\mathsf{Rep} = \mathsf{Dom}(L)$, the assignment $L[e] = (y, p^+)$ always creates a new entry of L; it never overwrites an already represented state or block string. This is the freshness property used in Lemma 4.

S3. On every other new forward query $(+1, x)$, set

$$y = \mathsf{Draw}(\{0,1\}^w \setminus \mathsf{Dom}(F^{-1})).$$

Set $F[x] = y$ and $F^{-1}[y] = x$, and return y.

S4. On an inverse query $(-1, y)$, if $y \in \mathsf{Dom}(F^{-1})$, return $F^{-1}[y]$.

S5. Suppose $(-1, y)$ is a new inverse query. Set $a = \mathsf{Draw}(\{0,1\}^r)$. Let $\mathcal{A}$ be the strings $d \in \{0,1\}^c \setminus \mathsf{Rep}$ for which $(a, d) \notin \mathsf{Dom}(F)$, and set $d = \mathsf{Draw}(\mathcal{A})$. Set $x = (a, d)$, set $F[x] = y$ and $F^{-1}[y] = x$, and return x.

If the fallback flag is false, then $i + 1 < 2^{c-2}$. There are at most $i + 1$ represented capacity values and at most i values in Out. For fixed z, at most i additional capacity values are excluded by the range of F. Thus the set in Step S2 has at least $2^c - 3i - 1 > 0$ elements. The set in Step S5 has at least $2^c - 2i - 1 > 0$ elements. Consequently every call to Draw is on a nonempty set. Each set is a bit-string space minus a list of at most $3i + 1$ forbidden values. Sort and deduplicate that list. The position of an admissible value in the lexicographically ordered set is its integer value minus the number of smaller forbidden values. Conversely, the value at a specified position can be found by skipping the forbidden values in sorted order. These operations require time polynomial in w, i, not in 2^w. The same method implements the deterministic fallback.

The simulator must be able to determine from a forward query whether it continues a sponge computation already recorded in L. It must also ensure that a value used earlier as the last c bits of a permutation input cannot later acquire a conflicting entry in L. The following lemma establishes these two facts.

Lemma 4 (Uniqueness of Recorded Sponge Computations). *Before the fallback branch is reached, every $d \in \mathsf{Dom}(L)$ has exactly one represented state and one represented block string. Moreover, if a c-bit value occurs as the last c bits of a stored permutation input before it is represented, then it is never represented later.*

Proof. Initially only 0^c is represented. A new represented state can be created only in Step S2. Its last c bits e are chosen outside both Rep and Out, so this value was neither represented nor already used as the last c bits of a stored permutation input. Hence $L[e]$ is a new entry, and it uniquely records the state reached and the block string that reaches it.

A forward query from an unrepresented c-bit value places that value in Out. Likewise, a new inverse query returns a new preimage (a, d) and stores it as a permutation input; Step S5 chooses its last c bits d outside Rep. Once a value belongs to Out, Step S2 never chooses it for a new represented state. Repeated queries do not change these tables. In particular, if a new inverse query returns a preimage whose last c bits are d, then storing this preimage places d in Out. From that point onward Step S2 excludes d when choosing the last c bits of a newly recorded sponge state. Thus a value first used in a permutation input cannot later acquire an entry in L. Induction over the primitive queries proves the claim.

In particular, whenever a forward query has input (a, d) with $d \in \mathsf{Dom}(L)$, the block string p^+ in Step S2 is determined uniquely by the ordered primitive transcript up to that query.

7.3 Proof of Indifferentiability

We use the interface order fixed in Section 3. In the real experiment the distinguisher receives (π_P, Sp^P), with the primitive interface first and the hash interface second. In the ideal experiment it receives $(\mathsf{S}_{\mathrm{sp}}^H, H)$ in the same order.

Let q_{perm} bound the number of forward and inverse first-interface queries in every execution. Let σ bound $\sum_i \ell_i$ in every execution, where the second-interface messages are m_i and $\mathsf{pad}(m_i)$ has ℓ_i blocks. Set $N = q_{\mathsf{perm}} + \sigma$. Define

$$f_{\mathrm{sp}}(N) := 1 - \prod_{i=0}^{N-1} \frac{1-(i+1)/2^c}{1-i/2^{r+c}}.$$

Theorem 8 (Indifferentiability of the Fixed-Output Permutation Sponge). *For every distinguisher $\mathcal{D}$ with $N < 2^{c-2}$,*

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{Sp},\mathcal{I}',\mathcal{I}}(\mathcal{D}, \mathsf{S}_{\mathrm{sp}}) \le f_{\mathrm{sp}}(N) + 2q_{\mathsf{perm}}2^{-\kappa}.$$

In particular, $f_{\mathrm{sp}}(N) = O(N^2/2^c)$ for $N = o(2^c)$. Consequently, if $N^2/2^c$ is negligible for every polynomially bounded N, then $\mathsf{Sp}^{\mathcal{I}'}$ is indifferentiable from $\mathcal{I}$ via S_{sp} in the sense of Section 3.

Proof. First replace every call to Draw by an exact uniform choice from the same admissible set, only for this comparison argument. Call this reference simulator $\mathsf{S}_{\mathrm{sp}}^{\mathrm{unif}}$. We use the random-permutation analysis from [BDPV08, Lemma 5 and Theorem 2], specialized to a sponge interface that returns only the first r output bits. Their permutation simulator exposes both the forward and inverse permutation interfaces, exactly as our tagged primitive interface does. Their padded-sponge proof requires the padded input to be injectively decodable and to end in a nonzero block; these are precisely the properties assumed of pad above.

There is one small bookkeeping difference from the variable-output formulation in [BDPV08]. For our fixed-output interface, the returned value is the first r bits of the state immediately after the last padded block is absorbed. Hence a hash query with ℓ_i padded blocks is replaced by exactly ℓ_i forward permutation queries; no additional squeezing query is needed. Thus the relevant cost for the specialization is $q_{\mathsf{perm}} + \sum_i \ell_i = N$. With this specialization, the permutation-transcript bound is exactly the function $f_{\mathrm{sp}}(N)$ displayed above. The early fallback is never reached under the budget $N < 2^{c-2}$.

Consider first an ideal oracle that, for every message m, supplies an unbounded sequence of independent r-bit strings $Z(m,0), Z(m,1), \ldots$. Its first string $Z(m,0)$ is a random function $H(m)$, while the strings $Z(m,j)$ for $j > 0$ form an independent random function $J(m,j)$. The permutation simulator in the standard sponge proof maintains the partial permutation. On a new forward query whose last c input bits are represented, it reconstructs the corresponding block string p^+; if $p^+ = \mathsf{pad}(m) \,\|\, (0^r)^j$, it uses $Z(m,j)$ as the first r bits of the response, and otherwise uses a fresh uniform r-bit string. It chooses the last c bits of the output outside the represented values and outside the values already occurring as the last c bits of a stored permutation input, subject to the permutation constraint. On a new inverse query it chooses the last c bits of the preimage outside the represented values, again subject to the permutation constraint.

These are exactly the reference simulator's Steps $\mathsf{S1}$–$\mathsf{S5}$ after writing $Z(m,0) = H(m)$ and $Z(m,j) = J(m,j)$ for $j > 0$. Within one session, sampling $V[m,j]$ uniformly on its first use gives the same distribution as querying an independent random function J at (m,j). Caching these values does not change the distribution of the simulator's answers.

Now let $\mathcal{D}$ distinguish the two systems in the theorem. We use the query-elimination step from the proof of [BDPV08, Theorem 2], but with our fixed-output interface. Replace a hash query m by the ℓ_i successive forward permutation queries obtained by processing the blocks of $\mathsf{pad}(m)$. In the real system, the first r bits of the final response are exactly $\mathsf{Sp}^P(m)$. In the ideal system, sponge consistency of the reference simulator gives the same conclusion: when the final query completes the processing of $\mathsf{pad}(m)$, Step $\mathsf{S2}$ uses $Z(m,0) = H(m)$ as the first r bits of its response. Thus the hash response is determined by those ℓ_i primitive responses, with no extra query for a squeezing step.

After eliminating all hash queries in this way, the resulting primitive-query sequence has cost at most $q_{\mathsf{perm}} + \sum_i \ell_i = N$ and gives at least the information available in the original interaction. The permutation-transcript bound of [BDPV08, Lemma 5] therefore bounds the distinguishing advantage for $\mathsf{S}_{\mathrm{sp}}^{\mathrm{unif}}$ by $f_{\mathrm{sp}}(N)$.

It remains to note that our explicit sampling rules are well defined throughout this query regime. After $i < N$ primitive input-output pairs have been fixed, $|\mathsf{Rep}| \le i+1$ and $|\mathsf{Out}| \le i$. For any fixed r-bit value z, at most i choices of e are excluded by the requirement $(z,e) \notin \mathsf{Dom}(F^{-1})$. Hence Step $\mathsf{S2}$ has at

least $2^c - 3i - 1 > 0$ admissible choices when $N < 2^{c-2}$. Step $\mathsf{S5}$ has at least $2^c - 2i - 1 > 0$ admissible choices by the same counting. Thus the fallback branch is never reached.

The random-permutation bound in [BDPV08] is $f_{\mathrm{sp}}(N)$ for $N < 2^c$, and therefore applies to the reference simulator in our range. Couple this reference simulator with S_{sp} until their first different choice. Conditional on an identical history, their next admissible sets are the same. By Lemma 3, a maximal coupling of that uniform choice and the next tape-based choice fails with probability at most $2^{-\kappa}$. At most $2q_{\mathsf{perm}}$ private choices occur: every fresh primitive query invokes Draw at most twice (Step $\mathsf{S2}$ uses at most two draws, Step $\mathsf{S3}$ one, and Step $\mathsf{S5}$ two), while repeated queries and fallback replies use none. The two ideal experiments consequently have statistical distance at most $2q_{\mathsf{perm}}2^{-\kappa}$. The triangle inequality gives the stated bound.

7.4 Exact Transcript Reconstruction

Fix a probabilistic oracle algorithm A, a random tape ρ for A, and a random tape ω for S_{sp}. For a completed execution of $A[\rho]^{\mathsf{S}_{\mathrm{sp}}[\omega]^H}$, write

$$\tau_{\mathrm{ext}} = \tau(A[\rho] \leftrightarrow \mathsf{S}_{\mathrm{sp}}[\omega]^H), \quad \tau_{\mathrm{int}} = \tau(\mathsf{S}_{\mathrm{sp}}[\omega] \leftrightarrow H).$$

Define the deterministic replay algorithm $\mathsf{Rec}_{\mathrm{sp}}$ by scanning τ_{ext} in order. It starts from the same initial session tables F', F'^{-1}, L', U', V' as the simulator and performs the same case distinction on every query. Every forward or inverse permutation input-output pair fixed by the simulator is visible in the corresponding query and answer in τ_{ext}, so replay uses that recorded pair instead of consulting the simulator's private randomness.

For a new forward query whose last c input bits have an entry in L', replay computes the same block string p^+. If $\mathsf{SDec}(p^+) = (m, 0)$ and $m \notin \mathsf{Dom}(U')$, the first r bits z of the recorded permutation answer are exactly the answer returned by the simulator's query $H(m)$. Replay therefore appends (m, z) to its reconstructed transcript and sets $U'[m] = z$. If the entry already exists, replay records no new query to H. When $\mathsf{SDec}(p^+) = (m, j)$ with $j > 0$, replay stores the observed first r bits in $V'[m, j]$. It then performs the same updates to F', F'^{-1} and L' as the simulator. The other forward and inverse cases make no query to H and are replayed directly from the visible query and answer.

Replay also maintains the fallback flag and applies the same fallback test and deterministic updates as the simulator. It checks query and answer formats, consistency of the partial bijection and cached values, and membership of each sampling outcome in its prescribed admissible set. If a check fails, it returns the empty transcript.

Theorem 9 (Exact Replay for the Permutation Sponge). *For every H, every random tape ρ of A, every random tape ω of S_{sp}, and every completed execution of $A[\rho]^{\mathsf{S}_{\mathrm{sp}}[\omega]^H}$,*

$$\mathsf{Rec}_{\mathrm{sp}}(\tau_{\mathrm{ext}}) = \tau_{\mathrm{int}}.$$

After each entry of τ_{ext} is processed, the replay tables equal the simulator's session tables at the same point of the execution.

Proof. We induct over the ordered entries of τ_{ext}. The tables agree initially. Assume that they agree before the next primitive query.

For a repeated query, both executions return the same stored answer and make no update. For a new query, the value $|\mathsf{Dom}(F)|$ determines in both executions whether the fallback flag is set. If it is, the recorded query and answer determine the updates of F, F^{-1}, and there is no update to L, U, V or to the oracle transcript. Otherwise equality of the partial permutation tables and L implies that both take the same one of Steps $\mathsf{S2}$, $\mathsf{S3}$, and $\mathsf{S5}$. In Step $\mathsf{S2}$ they reconstruct the same represented state, the same p^+, and the same value of $\mathsf{SDec}(p^+)$.

If $\mathsf{SDec}(p^+) = (m, 0)$ and $m \notin \mathsf{Dom}(U)$, the simulator queries $H(m)$ and uses the answer as the first r bits of its externally visible permutation answer. Replay therefore recovers exactly that hidden query–answer pair. If $m \in \mathsf{Dom}(U)$, neither execution makes a new query to H. For $j > 0$, the value placed in $V[m, j]$ is again visible as the first r bits of the permutation answer.

Every other sampling outcome is likewise contained in the visible query and answer. The raw bits that produced those outcomes need not be uniquely determined.

Thus the current transcript entry determines every session-table update made by the simulator. Replay makes the same updates, preserving the invariant. Induction proves both the table equality and the equality of the reconstructed and true H transcripts.

The algorithm $\mathsf{Rec}_{\mathrm{sp}}$ is deterministic and makes no oracle queries. Thus Theorem 9 gives exact transcript reconstruction in the sense of Definition 2.

7.5 Reconstruction for Rewinding

We define $\mathsf{Rec}_{\mathrm{sp,rw}}$ with the three entry points from Definition 3. Its persistent state W is a sequence of raw ℓ_{rnd}-bit tape blocks indexed by draw position. It is not indexed by the primitive query or session history.

The subroutine $\mathsf{Rec}_{\mathrm{sp,rw}}.\mathsf{Init}^{H}(\tau_{\mathrm{ext}})$ performs the exact replay above. For every private draw, in order, it determines the admissible set $\mathcal{A}$, its size M, and the index a of the returned element in the lexicographic ordering of $\mathcal{A}$, as determined from the recorded query and answer. For Step S2 these outcomes are the private rate component, when one is drawn, and the capacity component; for Step S3 it is the whole output; for Step S5 they are the two preimage components, in order. The fallback and cached replies consume no block. At each draw position, initialization uses the conditional sampler in Lemma 3 to create a raw block $W[j]$. Different conditional samples use independent fresh randomness. Initialization returns the reconstructed H-transcript and makes no query to H.

During replay, initialization checks formats, consistency of repeated queries, partial-bijection consistency, cached answers, membership in each prescribed admissible set, and deterministic fallback replies. On a failure, it returns the empty transcript and clears W. Subsequent sessions still run the actual simulator with a fresh ordinary tape. This failure branch has probability zero on a true simulator transcript. These checks do not query H; an uncached answer attributed to $H(m)$ is recorded from the visible rate component.

The subroutine $\mathsf{Start}()$ resets all session tables, the fallback flag, and the draw counter to the simulator's initial values, but does not clear W. The subroutine $\mathsf{Answer}^{K}(q)$ runs the actual simulator code with K in place of H. Each private draw reads the next raw block $W[\nu]$; a previously unread position is filled with an independent uniform ℓ_{rnd}-bit block and retained thereafter. It applies that block to the *current* admissible set using Draw and increments ν. In particular, a reset that changes the admissible set reuses the same raw integer, not the same previously selected element.

Theorem 10 (Reconstruction for Rewinding). *Let A, B, T be probabilistic polynomial-time algorithms, and let q_A bound the number of primitive queries in the initial execution of A. Then*

$$\mathsf{Adv}^{\text{rw-rec}}_{\mathsf{S}_{\mathrm{sp}},\mathsf{Rec}_{\mathrm{sp,rw}},\mathcal{I}}(A,B,T) \le 2q_A 2^{-\kappa}.$$

There is no additional error per later invocation. In particular, $\kappa = 2\lambda$ gives reconstruction for rewinding in the sense of Definition 6.

Proof. Fix H and the tape ρ of A, and condition on a positive-probability external transcript τ_{ext}. By Theorem 9, initialization returns the true internal transcript. It also identifies the number k of private draws and each pair (M_j, a_j), for $1 \le j \le k$. Here $k \le 2q_A$. For these fixed data, the condition on the raw block ω_j is exactly that its integer value has residue a_j modulo M_j. All branches, set choices, and session-table updates are determined by the visible sampling outcomes and H; none inspect other bits of a raw block. Hence the conditional distributions of the first k raw blocks are independent, with precisely the distributions described in Lemma 3. Termination and the recorded history impose no constraint on the unread suffix, which remains independent and uniform.

For each of the first k positions, the initialized block $W[j]$ is within statistical distance $2^{-\kappa}$ of that conditional distribution. Couple these blocks one at a time; the probability that any coupling fails is at most $k2^{-\kappa}$. Couple every unread suffix block identically. Except with that probability, the true tape and the reconstructed tape are now equal at every position.

Fix the coins of B and T and use the same shared H. On the event of tape equality, every reset initializes the same session state, and both implementations run the identical simulator with the same supplied K. Induction over all subsequent interactions gives identical primitive answers and identical calls to the supplied oracles, even when their answers change the admissible sets or draw counts. The deterministic fallback is also identical. Therefore the conditional output probabilities differ by at most $k2^{-\kappa}$, independently of the number of later invocations. Initialization does not query H, so there is no intervening state change. Averaging over $H, \rho, \tau_{\mathrm{ext}}$ proves the bound $2q_A 2^{-\kappa}$.

The coupling above is a coupling of the single persistent reconstructed tape W with the simulator's single fixed tape, conditioned on the initial transcript; it is not a separate coupling for each later rewind session. On the event that these tapes agree, every future $\mathsf{Start}/\mathsf{Answer}$ session—including sessions chosen adaptively from earlier answers—uses the same tape blocks in the same positions. All such sessions therefore agree simultaneously, so no union bound over the number of later invocations is needed.

The reconstructor meets the cost requirement of Definition 3. If the external interaction size is L, it processes at most L primitive requests and at most $2L$ draws. Its tables contain at most polynomially many

bits in λ, L. Computing the position of an allowed value and finding the value at a specified position use the finite forbidden lists described above, and the conditional sampler uses only polynomial-bit arithmetic and bounded random draws. Each represented block string has at most L blocks, so polynomial-time padding inversion bounds both the time and the length of every generated message by a polynomial in λ, L. At most L such operations occur. This bounds all work and generated traffic by a single polynomial $p(\lambda, L)$. For invocations of any fixed worst-case polynomial-time $\mathcal{P}'$, each session has a polynomial query bound; the largest tape position and session-table sizes are then polynomial independently of the number of sessions.

Since a polynomial-time extractor makes only polynomially many invocations, the total reconstruction cost in the transfer is polynomial.

7.6 Consequences

Corollary 5. *Assume $N^2/2^c$ is negligible for every polynomially bounded N. With $\kappa = 2\lambda$, S_{sp} gives standard indifferentiability of $\mathsf{Sp}^{\mathcal{I}'}$ from $\mathcal{I}$. The algorithm $\mathsf{Rec}_{\mathrm{sp}}$ reconstructs the internal transcript exactly, and $\mathsf{Rec}_{\mathrm{sp,rw}}$ has rewinding-reconstruction error at most $2q_A 2^{-\kappa}$ for an initial-query bound q_A. Consequently, proof-only straight-line, transcript-dependent straight-line, and rewinding knowledge soundness transfer by Corollaries 1 to 3.*

For a reduction with worst-case aggregate budget $N < 2^{c-2}$, the indifferentiability term is at most $f_{\mathrm{sp}}(N) + 2q_{\mathsf{perm}}2^{-\kappa}$. The transcript-reconstruction term is zero. The rewinding-reconstruction term is at most $2q_A 2^{-\kappa}$, where q_A counts only the initial prover's primitive queries, not later reset invocations. For a rewinding transfer whose two initial-query bounds both equal $q_{\mathsf{perm}}^{\mathrm{initial}}$, the total additional error is therefore at most $f_{\mathrm{sp}}(N) + 4q_{\mathsf{perm}}^{\mathrm{initial}}2^{-\kappa}$. The two finite-tape contributions in the coefficient 4 arise at different steps. The first term $2q_{\mathsf{perm}}^{\mathrm{initial}}2^{-\kappa}$ is the statistical loss from replacing the exact-uniform draws of $\mathsf{S}_{\mathrm{sp}}^{\mathrm{unif}}$ by the bounded-tape implementation in the indifferentiability proof. The second term $2q_{\mathsf{perm}}^{\mathrm{initial}}2^{-\kappa}$ is the loss from sampling the simulator tape conditioned on the initial transcript in the rewinding reconstruction theorem. They are therefore additive losses from two distinct comparisons, not the same error counted twice. The polynomial-time transfer reductions have polynomial worst-case aggregate budgets.

Proof. Combine Theorems 8 to 10 with the generic transfer corollaries. Their reductions have polynomial worst-case aggregate budgets. Every such budget is less than 2^{c-2} for all sufficiently large λ under the capacity assumption. Both finite-tape terms are negligible because $\kappa = 2\lambda$. The total is the sum of the indifferentiability and rewinding-reconstruction terms.

8 Semantic Zero Knowledge

The zero-knowledge formulation of [CO25] is tailored to their application and imposes a specific simulator interface. We do not try to preserve that particular simulator syntax here. Instead we use a semantic notion: the simulator may generate the proof and the public oracle process in any efficient way, and security asks only that the resulting joint view be computationally indistinguishable from the real one. With this formulation, zero knowledge is preserved directly by standard indifferentiability; none of the transcript-reconstruction or rewinding-reconstruction properties from the preceding sections are needed.

A zero-knowledge adversary is a pair $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. The first stage outputs a statement x, a witness w, and auxiliary information aux that is passed to the second stage. Both stages may query the public oracle available in the experiment.

8.1 Definition

Definition 7 (Semantic Zero-Knowledge Simulator). *A semantic zero-knowledge simulator is a stateful triple $\mathsf{Z} = (\mathsf{Z.Init}, \mathsf{Z.Qry}, \mathsf{Z.Prove})$. The call $\sigma \xleftarrow{\$} \mathsf{Z.Init}(1^\lambda)$ creates its initial state. Given the current state σ, a call $(y, \sigma') \xleftarrow{\$} \mathsf{Z.Qry}[\sigma](u)$ answers one public-oracle query and updates the state, and a call $(\mathsf{pf}, \sigma') \xleftarrow{\$} \mathsf{Z.Prove}[\sigma](x)$ produces the simulated proof and updates the same state. Thus the adversary sees one persistent simulated oracle process before and after the proof is produced.*

Importantly, $\mathsf{Z.Prove}$ may modify the same internal state that governs later $\mathsf{Z.Qry}$ calls. Hence producing the simulated proof may change the simulator's future public-oracle behavior. We do not require this change to be represented by an explicit list of programmed points, or by any other particular representation of the simulated oracle state.

The simulator is probabilistic polynomial time if, against every probabilistic polynomial-time two-stage adversary making polynomially many queries of polynomial total length, its total running time and state size are polynomial in λ and the explicit input lengths.

For the construction model, define

$$p^{\mathrm{zk,real}}_{\mathcal{R},(\mathsf{P},\mathsf{V}),\mathsf{C},\mathcal{I}'}(\mathcal{A}) := \Pr\left[\begin{array}{l} \pi \xleftarrow{\$} \mathcal{I}';\\ (x,w,\mathsf{aux}) \xleftarrow{\$} \mathcal{A}_1^{\pi};\\ \text{if } (x,w)\notin\mathcal{R}\text{, return } 0;\\ \mathsf{pf} \xleftarrow{\$} \mathsf{P}^{\mathsf{C}^{\pi}}(x,w);\\ b \xleftarrow{\$} \mathcal{A}_2^{\pi}(\mathsf{aux},\mathsf{pf}) :\\ b=1 \end{array}\right].$$

All occurrences of π refer to one shared execution. In particular, the two adversarial stages and every primitive call made while evaluating C^{π} use the same sampled primitive.

For a semantic simulator Z, define

$$p^{\mathrm{zk,ideal}}_{\mathcal{R},(\mathsf{P},\mathsf{V})}(\mathcal{A},\mathsf{Z}) := \Pr\left[\begin{array}{l} \sigma \xleftarrow{\$} \mathsf{Z.Init}(1^{\lambda});\\ (x,w,\mathsf{aux}) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{Z.Qry}[\sigma]};\\ \text{if } (x,w)\notin\mathcal{R}\text{, return } 0;\\ (\mathsf{pf},\sigma) \xleftarrow{\$} \mathsf{Z.Prove}[\sigma](x);\\ b \xleftarrow{\$} \mathcal{A}_2^{\mathsf{Z.Qry}[\sigma]}(\mathsf{aux},\mathsf{pf}) :\\ b=1 \end{array}\right].$$

The witness w is used only for the relation-membership test and is not given to $\mathsf{Z.Prove}$.

Definition 8 (Semantic Zero Knowledge). *Define*

$$\mathsf{Adv}^{\mathrm{zk}}_{\mathcal{R},(\mathsf{P},\mathsf{V}),\mathsf{C},\mathcal{I}'}(\mathcal{A},\mathsf{Z}) := \left|p^{\mathrm{zk,real}}_{\mathcal{R},(\mathsf{P},\mathsf{V}),\mathsf{C},\mathcal{I}'}(\mathcal{A}) - p^{\mathrm{zk,ideal}}_{\mathcal{R},(\mathsf{P},\mathsf{V})}(\mathcal{A},\mathsf{Z})\right|.$$

The instantiated argument is semantically zero knowledge if there is a probabilistic polynomial-time semantic simulator Z for which this advantage is negligible for every probabilistic polynomial-time two-stage adversary $\mathcal{A}$.

When the public oracle itself is $H \xleftarrow{\$} \mathcal{I}$, we write $\mathsf{Adv}^{\mathrm{zk}}_{\mathcal{R},(\mathsf{P},\mathsf{V}),\mathcal{I}}(\mathcal{A},\mathsf{Z})$ for the analogous advantage. Its real experiment samples one $H \xleftarrow{\$} \mathcal{I}$, runs $\mathcal{A}_1^H$, produces the honest proof with P^H, and then runs $\mathcal{A}_2^H$, all using the same execution of H.

The experiment above concerns a single proof request. We emphasize that the word *semantic* is important here: the definition constrains only the joint distribution seen by the adversary. It does not require the simulator to realize its oracle by starting from an honestly sampled oracle and programming a prescribed set of points, nor does it require any other particular representation of the simulated oracle state.

8.2 Transfer through an Indifferentiable Construction

Let Z be a semantic zero-knowledge simulator for the NARG relative to $\mathcal{I}$, and let S witness indifferentiability of $\mathsf{C}^{\mathcal{I}'}$ from $\mathcal{I}$. From these two algorithms we construct a semantic simulator $\widehat{\mathsf{Z}}$ for the target NARG relative to $\mathsf{C}^{\mathcal{I}'}$. At initialization, $\widehat{\mathsf{Z}}$ starts one persistent execution of Z and one persistent execution of S. On a primitive-oracle query q, $\widehat{\mathsf{Z}}.\mathsf{Qry}$ gives q to the current execution of S. Whenever S makes a query to its $\mathcal{I}$-oracle, that query is answered by invoking $\mathsf{Z.Qry}$ on the current state of the same persistent execution of Z. The answer produced by S is returned to the adversary.

On the proof request x, $\widehat{\mathsf{Z}}.\mathsf{Prove}$ invokes $\mathsf{Z.Prove}(x)$ on that same execution of Z and returns the resulting proof. This call does not reset or otherwise change the state of S. Hence, after the proof is produced, later primitive-oracle queries continue the same execution of S, and every $\mathcal{I}$-oracle query made by S is answered using the possibly updated state of $\mathsf{Z.Qry}$. Thus, $\widehat{\mathsf{Z}}$ maintains one chronological execution of both algorithms throughout. It never resets, rewinds, or reconstructs the indifferentiability simulator.

Theorem 11 (Semantic Zero Knowledge from Standard Indifferentiability). *Assume that* $\mathsf{C}^{\mathcal{I}'}$ *is indifferentiable from* $\mathcal{I}$ *via* S. *Let* Z *be a semantic zero-knowledge simulator for the NARG relative to* $\mathcal{I}$, *and let* $\widehat{\mathsf{Z}}$ *be the simulator defined above.*

For every probabilistic polynomial-time adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$, *there are a probabilistic polynomial-time indifferentiability distinguisher* $\mathcal{D}$ *and a probabilistic polynomial-time adversary* $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ *such that*

$$\mathsf{Adv}^{\mathsf{zk}}_{\mathcal{R},(\mathsf{P},\mathsf{V}),\mathsf{C},\mathcal{I}'}(\mathcal{A}', \widehat{\mathsf{Z}}) \leq \mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{C},\mathcal{I}',\mathcal{I}}(\mathcal{D}, \mathsf{S}) + \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{R},(\mathsf{P},\mathsf{V}),\mathcal{I}}(\mathcal{B}, \mathsf{Z}).$$

Consequently, semantic zero knowledge relative to $\mathcal{I}$ *is preserved when* $\mathcal{I}$ *is replaced by an indifferentiable construction* $\mathsf{C}^{\mathcal{I}'}$.

If the two stages of $\mathcal{A}'$ *make at most* q_A *primitive queries of total length at most* L_A, *and the honest prover makes at most* q_P *queries to its public oracle, then* $\mathcal{D}$ *makes at most* q_A *queries to its first interface and at most* q_P *queries to its second interface. The derived adversary* $\mathcal{B}$ *makes at most* $c_{\mathsf{S}}(q_A, L_A)$ *queries to its ideal-oracle interface and incurs at most* $t_{\mathsf{S}}(q_A, L_A)$ *additional local computation from running* S.

The query accounting in the theorem refers to calls issued by $\mathcal{D}$ to its two indifferentiability interfaces. For each primitive query made by $\mathcal{A}'$, the distinguisher makes one call to its first interface, and for each public-oracle query made by the honest prover it makes one call to its second interface. Thus these numbers are at most q_A and q_P, respectively. In the real indifferentiability world the second interface is C^{π}, so evaluating one of its calls may itself invoke the primitive π. Those primitive calls are internal to the implementation of the second interface supplied by the indifferentiability experiment; they are not additional first-interface calls made by $\mathcal{D}$.

Proof. Let p_{hyb} be the success probability of the following experiment. Sample $H \xleftarrow{\$} \mathcal{I}$ and initialize one execution of S^H. Run $(x, w, \mathsf{aux}) \xleftarrow{\$} \mathcal{A}_1'^{\mathsf{S}^H}$; if $(x, w) \notin \mathcal{R}$, return 0; otherwise run $\mathsf{pf} \xleftarrow{\$} \mathsf{P}^H(x, w)$ and then $b \xleftarrow{\$} \mathcal{A}_2'^{\mathsf{S}^H}(\mathsf{aux}, \mathsf{pf})$, using the same simulator execution in both adversarial stages, and return b.

Given interfaces (O_1, O_2), define $\mathcal{D}$ to execute this same code with O_1 in place of the primitive/simulator interface and O_2 in place of the construction/ideal-oracle interface. With $(O_1, O_2) = (\pi, \mathsf{C}^{\pi})$ this is exactly the real construction experiment. With $(O_1, O_2) = (\mathsf{S}^H, H)$ it is exactly the experiment defining p_{hyb}. Therefore the difference between the two probabilities is at most $\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{C},\mathcal{I}',\mathcal{I}}(\mathcal{D}, \mathsf{S})$.

It remains to compare p_{hyb} with the semantic ideal experiment using $\widehat{\mathsf{Z}}$. Define $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ as follows. Given an oracle O, the first stage $\mathcal{B}_1^O$ initializes one execution of S^O, runs $(x, w, \mathsf{aux}) \xleftarrow{\$} \mathcal{A}_1'^{\mathsf{S}^O}$, and outputs $(x, w, (\mathsf{aux}, \sigma_{\mathsf{S}}))$, where σ_{S} is the current state of that simulator execution. Given $(\mathsf{aux}, \sigma_{\mathsf{S}}, \mathsf{pf})$, the second stage resumes that simulator state, runs $b \xleftarrow{\$} \mathcal{A}_2'^{\mathsf{S}^O}(\mathsf{aux}, \mathsf{pf})$, and returns b.

Here σ_{S} denotes the complete resumable state of the copy of S that $\mathcal{B}_1$ is running, including whatever execution information is needed to continue that copy with the same randomness. Passing σ_{S} to $\mathcal{B}_2$ is therefore just the ordinary continuation of one stateful execution across the two stages. It gives $\mathcal{B}$ no reset or rewinding access to S, and $\mathcal{B}$ never needs to recover or resample the simulator's private random tape separately.

In the real ideal-oracle zero-knowledge experiment for $\mathcal{B}$, the oracle O is one sampled H, the proof is generated by P^H, and the same saved simulator state is resumed afterward. This is exactly p_{hyb}. In the semantic ideal experiment for $\mathcal{B}$, every ideal-oracle query made by the saved copy of S is answered by the current $\mathsf{Z.Qry}$ state, the proof is produced by $\mathsf{Z.Prove}$, and the same saved simulator state is then resumed. This is exactly the target semantic ideal experiment with $\widehat{\mathsf{Z}}$. The second difference is therefore at most $\mathsf{Adv}^{\mathsf{zk}}_{\mathcal{R},(\mathsf{P},\mathsf{V}),\mathcal{I}}(\mathcal{B}, \mathsf{Z})$.

The theorem follows by the triangle inequality. The query and running-time bounds follow directly from the code of $\mathcal{D}$ and from the definitions of c_{S} and t_{S} in the preliminaries.

Corollary 6. *If the NARG is semantically zero knowledge relative to the random oracle* $\mathcal{I}$, *then both the tagged-PFMD instantiation of Section 6 and the fixed-output sponge instantiation of Section 7 are semantically zero knowledge whenever the corresponding standard indifferentiability premise holds.*

Proof. Apply Theorem 11 with the simulator S_{pf} from Section 6 or with the simulator S_{sp} from Section 7.2, respectively. The transcript-reconstruction and rewinding-reconstruction properties of these simulators are not used.

Quantitatively, in either instantiation the target semantic-zero-knowledge advantage is bounded by the ideal-model semantic-zero-knowledge advantage of the derived adversary $\mathcal{B}$ plus the corresponding standard-indifferentiability advantage of the distinguisher $\mathcal{D}$, with the query parameters stated in Theorem 11.

Consequently, substituting the tagged-PFMD or sponge indifferentiability bound proved earlier gives exactly the ideal-model zero-knowledge error plus that indifferentiability error. There is no additional loss from transcript reconstruction or rewinding reconstruction.

Acknowledgements

The work of Ashrujit Ghoshal was supported by an ANRF Early Career Research Grant (Grant No ANRF/ECRG/2025/000556/ENS), a New Faculty Initiation Grant at IIT Madras, the Arvind Raghunathan Center for Theoretical CS (aRtCS), the Mphasis F1 Foundation, the National Quantum Mission initiative of Dept of Science & Technology, India and CyStar, IIT Madras.

AI Usage

We used ChatGPT 5.6 Sol and GPT-6 Astra exclusively for editorial assistance with the presentation, structure, and organization of the main body and appendices. All technical ideas, results, and proofs were developed by the authors without AI assistance.

References

BCG+18. Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune K. Jakobsen, and Mary Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 595–626. Springer, Cham, December 2018.

BCHO22. Jonathan Bootle, Alessandro Chiesa, Yuncong Hu, and Michele Orrù. Gemini: Elastic SNARKs for diverse environments. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 427–457. Springer, Cham, May / June 2022.

BDPV08. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the Sponge construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, Berlin, Heidelberg, April 2008.

BDPV12. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-pass authenticated encryption and other applications. In Ali Miri and Serge Vaudenay, editors, *SAC 2011*, volume 7118 of *LNCS*, pages 320–337. Springer, Berlin, Heidelberg, August 2012.

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

CDMP05. Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Berlin, Heidelberg, August 2005.

CHM+20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Cham, May 2020.

CO25. Alessandro Chiesa and Michele Orrù. A fiat-shamir transformation from duplex sponges. In Benny Applebaum and Huijia (Rachel) Lin, editors, *TCC 2025, Part I*, volume 16268 of *LNCS*, pages 452–474. Springer, Cham, December 2025.

Dam90. Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 416–427. Springer, New York, August 1990.

DKL+20. Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 64–93. Springer, Cham, December 2020.

EFG22. Liam Eagen, Dario Fiore, and Ariel Gabizon. cq: Cached quotients for fast lookups. Cryptology ePrint Archive, Report 2022/1763, 2022.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Berlin, Heidelberg, May 2013.

GW20. Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Report 2020/315, 2020.

GWC19. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019.

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

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

Mer90. Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 428–446. Springer, New York, August 1990.

MRH04. Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Berlin, Heidelberg, February 2004.

Orr26. Michele Orrù. Fiat-shamir transformation. Internet-Draft draft-irtf-cfrg-fiat-shamir-03, August 2026. Work in Progress.

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

RSS11. Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Berlin, Heidelberg, May 2011.

Sch91. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

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

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