Interactive Oracle Proofs*

Eli Ben-Sasson
eli@cs.technion.ac.il
Technion

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

Nicholas Spooner
spooner@cs.toronto.edu
University of Toronto

February 10, 2016

Abstract

We initiate the study of a proof system model that naturally combines two well-known models: interactive proofs (IPs) and probabilistically-checkable proofs (PCPs). An *interactive oracle proof* (IOP) is an interactive proof in which the verifier is not required to read the prover's messages in their entirety; rather, the verifier has oracle access to the prover's messages, and may probabilistically query them.

IOPs simultaneously generalize IPs and PCPs. Thus, IOPs retain the expressiveness of PCPs, capturing NEXP rather than only PSPACE, and also the flexibility of IPs, allowing multiple rounds of communication with the prover. These degrees of freedom allow for more efficient "PCP-like" interactive protocols, because the prover does not have to compute the parts of a PCP that are not requested by the verifier.

As a first investigation into IOPs, we offer two main technical contributions. First, we give a compiler that maps any public-coin IOP into a non-interactive proof in the random oracle model. We prove that the soundness of the resulting proof is tightly characterized by the soundness of the IOP against *state restoration attacks*, a class of rewinding attacks on the IOP verifier. Our compiler preserves zero knowledge, proof of knowledge, and time complexity of the underlying IOP. As an application, we obtain blackbox unconditional ZK proofs in the random oracle model with quasilinear prover and polylogarithmic verifier, improving on the result of Ishai et al. (2015).

Second, we study the notion of state-restoration soundness of an IOP: we prove tight upper and lower bounds in terms of the IOP's (standard) soundness and round complexity; and describe a simple adversarial strategy that is optimal across all state restoration attacks.

Our compiler can be viewed as a generalization of the Fiat–Shamir paradigm for public-coin IPs (CRYPTO '86), and of the "CS proof" constructions of Micali (FOCS '94) and Valiant (TCC '08) for PCPs. Our analysis of the compiler gives, in particular, a unified understanding of all of these constructions, and also motivates the study of state restoration attacks, not only for IOPs, but also for IPs and PCPs.

Keywords: probabilistically checkable proofs, interactive proofs, Fiat–Shamir paradigm, computationally-sound proofs

*Parts of this paper appear in the third author's master's thesis (April 2015) in the Department of Computer Science at ETH Zurich, supervised by Alessandro Chiesa and Thomas Holenstein. Independent of our work, [RRR16] introduce the notion of *Probabilistically Checkable Interactive Proofs*, which is the same as our notion of Interactive Oracle Proofs.

Contents

1 Introduction **3**
- 1.1 Models of proof systems . . . 3
- 1.2 Compiling proof systems into argument systems . . . 4
- 1.3 Results . . . 5
- 1.4 Techniques . . . 9
- 1.5 Roadmap . . . 11

2 Preliminaries **12**
- 2.1 Basic notations . . . 12
- 2.2 Merkle trees . . . 12
- 2.3 Non-interactive random-oracle proofs . . . 12

3 Extractability and privacy of Merkle trees **14**
- 3.1 Extractability . . . 14
- 3.2 Privacy . . . 15

4 Interactive oracle proofs **18**
- 4.1 Interactive oracle protocols . . . 18
- 4.2 Interactive oracle proof systems . . . 18
- 4.3 Basic complexity-theoretic properties . . . 19

5 State restoration attacks on interactive oracle proofs **20**
- 5.1 State restoration attacks, and soundness and proof of knowledge against these . . . 20
- 5.2 General bounds . . . 21
- 5.3 Tightness of general bounds . . . 22
- 5.4 Restricted state restoration attacks . . . 23

6 From interactive oracle proofs to non-interactive random-oracle proofs **25**

7 Analysis of the transformation T **27**
- 7.1 Upper bound for soundness . . . 28
- 7.2 Lower bound for soundness . . . 30
- 7.3 Proof of knowledge . . . 31
- 7.4 Zero knowledge . . . 32

8 Tree exploration games **34**
- 8.1 Definition of tree exploration games . . . 34
- 8.2 A connection between tree exploration and state restoration . . . 34
- 8.3 The expected value strategy . . . 37

Acknowledgments **39**

A Interactive oracle proofs characterize NEXP **40**

B Soundness error reduction **41**
- B.1 Soundness error reduction for IOPs . . . 41
- B.2 Soundness error reduction for NIROPs . . . 41
- B.3 Strategies for soundness error reduction . . . 43

C Zero knowledge with non-programmable random oracles **44**

References **45**

1 Introduction

The notion of *proof* is central to modern cryptography and complexity theory. The class NP, for example, is the set of languages whose membership can be decided by a deterministic polynomial-time verifier by reading proof strings of polynomial length; this class captures the traditional notion of a mathematical proof. Over the last three decades, researchers have introduced and studied proof systems that generalize the above traditional notion, and investigations from these points of view have led to breakthroughs in cryptography, hardness of approximation, and other areas. In this work we introduce and study a new model of proof system.

1.1 Models of proof systems

We give some context by recalling three of the most well-known among alternative models of proof systems.

Interactive proofs (IPs). Interactive proofs were introduced by Goldwasser, Micali, and Rackoff [GMR89]: in a k-round interactive proof, a probabilistic polynomial-time verifier exchanges k messages with an all-powerful prover, and then accepts or rejects; $\mathsf{IP}[k]$ is the class of languages with a k-round interactive proof. Independently, Babai [Bab85] introduced Arthur–Merlin games: a k-round Arthur–Merlin game is a k-round *public-coin* interactive proof (i.e., the verifier messages are uniformly and independently random); $\mathsf{AM}[k]$ is the class of languages with a k-round Arthur–Merlin game. Goldwasser and Sipser [GS86] showed that the two models are equally powerful: for polynomial k, $\mathsf{IP}[k] \subseteq \mathsf{AM}[k+2]$. Shamir [Sha92], building on the "sum-check" interactive proof of Lund, Fortnow, Karloff, and Nisan [LFKN92], proved that interactive proofs correspond to languages decidable in polynomial space: $\mathsf{IP}[\mathrm{poly}(n)] = \mathsf{PSPACE}$. (Also see [Bab90].)

Multi-prover interactive proofs (MIPs). Multi-prover interactive proofs were introduced by Ben-Or, Goldwasser, Kilian, and Wigderson [BGKW88]: in a k-round p-prover interactive proof, a probabilistic polynomial-time verifier interacts k times with p non-communicating all-powerful provers, and then accepts or rejects; $\mathsf{MIP}[p,k]$ is the class of languages that have a k-round p-prover interactive proof. In [BGKW88], the authors show that two provers always suffice (i.e., $\mathsf{MIP}[p,k] = \mathsf{MIP}[2,k]$), and that all languages in NP have perfect zero knowledge proofs in this model. Fortnow, Rompel, and Sipser [FRS88] show that interaction with two provers is equivalent to interaction with one prover plus oracle access to a proof string, and from there obtain that $\mathsf{MIP}[\mathrm{poly}(n), \mathrm{poly}(n)] \subseteq \mathsf{NEXP}$; Babai, Fortnow and Lund [BFL90] show that NEXP has 1-round 2-prover interactive proofs, thus showing that $\mathsf{MIP}[2,1] = \mathsf{NEXP}$.

Probabilistically checkable proofs (PCPs). Probabilistically checkable proofs were introduced by [FRS88, BFLS91, AS98, ALM+98]: in a probabilistically-checkable proof, a probabilistic polynomial-time verifier has oracle access to a proof string; $\mathsf{PCP}[r,q]$ is the class of languages for which the verifier uses at most r bits of randomness, and queries at most q locations of the proof (note that the proof length is at most 2^r). The above results on MIPs imply that $\mathsf{PCP}[\mathrm{poly}(n), \mathrm{poly}(n)] = \mathsf{NEXP}$. Later works "scaled down" this result to NP: Babai, Fortnow, Levin and Szegedy [BFLS91] show that $\mathsf{NP} = \mathsf{PCP}[O(\log n), \mathrm{poly}(\log n)]$; Arora and Safra [AS98] show that $\mathsf{NP} = \mathsf{PCP}[O(\log n), O(\sqrt{\log n})]$; and Arora, Lund, Motwani, Sudan, and Szegedy [ALM+92] show that $\mathsf{NP} = \mathsf{PCP}[O(\log n), O(1)]$. This last is known as the *PCP Theorem*.

Researchers have studied other models of proof systems, and here we name only a few: *linear IPs* [BCI+13], *no-signaling MIPs* [IKM09, Ito10, KRR13, KRR14], *linear PCPs* [IKO07, Gro10, Lip12, BCI+13, GGPR13, PGHR13, BCI+13, SBW11, SMBW12, SVP+12, SBV+13], *interactive PCPs* [KR08, KR09, GIMS10].

We introduce **interactive oracle proofs** (IOPs), a model of proof system that combines aspects of IPs and PCPs. (Later, in Section 1.3.1, we describe how interactive PCPs are a special case of IOPs.) Our work focuses on cryptographic applications of this proof system, as we discuss next.

1.2 Compiling proof systems into argument systems

The proof systems mentioned so far share a common feature: they make no assumptions on the computational resources of a (malicious) prover trying to convince the verifier. Instead, many proof systems make "structural" assumptions on the prover: MIPs assume that the prover is a collection of non-communicating strategies (each representing a "sub-prover"); PCPs assume that the prover is non-adaptive (the answer to a message does not depend on previous messages); linear IPs assume that the prover is a linear function, and so on.

In contrast, in cryptography, one often considers *argument systems* [BC86, BCC88, Kil92, Mic00]: these are proof systems where soundness holds only against provers that have a bound on computational resources (e.g., provers that run in probabilistic polynomial time). The relaxation from statistical soundness to computational soundness allows circumventing various limitations of IPs [BHZ87, GH98, GVW02, PSSV07], while also avoiding "structural" assumptions on the prover, which can be hard to enforce in applications.

Constructing argument systems. A common methodology to construct argument systems with desirable properties (e.g., sublinear communication complexity) follows these two steps: (1) give a proof system that achieves these properties in a model with structural restrictions on (all-powerful) provers; (2) use cryptographic tools to compile that proof system into an argument system, i.e., one where the only restriction on the prover is that it is an efficient algorithm. In other words, the compilation trades any structural assumptions for computational ones. This methodology has been highly productive; see Figure 1 for some examples that follow this paradigm. (On the flip side, many other argument systems are constructed, without the explicit use of underlying proof systems, using cryptographic tools "directly"; for instance see [BCC88]. A partial list of more recent examples is [PR14, GHRW14, BGL$^+$15, CHJV15, CH15, CCC$^+$15, KP15].)

Proofs in the random oracle model. An idealized model for studying computationally-bounded provers is the random oracle model [FS86, BR93], where every party has access to the same random function. A protocol proved secure in this model can potentially be instantiated in practice by replacing the random function with a concrete "random-looking" efficient function. While this intuition fails in the general case [CGH04, BBP04, GK03, BDG$^+$13], the random oracle model is nonetheless a useful testbed for cryptographic primitives. In this paper we focus on proof systems in this model for which the proof consists of a single message from the prover to the verifier. A **non-interactive random-oracle proof** (NIROP) for a relation $\mathscr{R}$ is a pair of probabilistic polynomial-time algorithms, the prover $\mathbb{P}$ and verifier $\mathbb{V}$, that satisfy the following. (1) *Completeness:* for every instance-witness pair $(\mathtt{x}, \mathtt{w})$ in the relation $\mathscr{R}$, $\Pr[\mathbb{V}^\rho(\mathtt{x}, \mathbb{P}^\rho(\mathtt{x}, \mathtt{w})) = 1] = 1$, where the probability is taken over the random oracle ρ as well as any randomness of $\mathbb{P}$ and $\mathbb{V}$. (2) *Soundness:* for every instance $\mathtt{x}$ not in the language of $\mathscr{R}$ and every malicious prover $\tilde{\mathbb{P}}$ that asks at most a polynomial number of queries to the random oracle, it holds that $\Pr[\mathbb{V}^\rho(\mathtt{x}, \tilde{\mathbb{P}}^\rho) = 1]$ is negligible in the security parameter.

Prior NIROPs and our focus. Prior work uses the above 2-step methodology to obtain NIROPs with desirable properties. For example, the Fiat–Shamir paradigm maps 3-message public-coin IPs to corresponding NIROPs [FS86, PS96]; when invoked on suitable IP constructions, this yields very efficient zero knowledge non-interactive proofs. As another example, Micali's "CS proof" construction, building on [Kil92], transforms PCPs to corresponding NIROPs; Valiant [Val08] revisits Micali's construction and proves that it is a proof of knowledge; when invoked on suitable PCPs, these yield non-interactive proofs of knowledge that are extremely short and easy to verify. In this work we study the question of how to compile IOPs, which generalize both IPs and PCPs, into NIROPs. As discussed below, our work ultimately leads to formulating and studying a game-theoretic property of IOPs, which in turn motivates similar questions for IPs and PCPs. We now turn to the discussion of our results. (We do not study the question of avoiding assuming random oracles, as this question continues to be open for the aforementioned prior works and, also, our work.)

proof model	work	crypto tools used by transformation	properties of argument system		
			setup	# of messages	verifiability
IP (public-coin)	[FS86, PS96]	(explicitly-programmable) RO	RO	1	public
	[KR09]	PIR	none	2	private
NIZK w/ hidden bits	[FLS90, PS05]	OWF	CRS	1	public
Σ-protocols	[Lin15]	(non-programmable) RO	CRS + RO	1	public
PCP	[Kil92, BG08]	CRF	none	4	public
	[Mic00, Val08]	(explicitly-programmable) RO	RO	1	public
	[BCCT12, DFH12, GLR11]	extractable CRF	none	2	private
MIP	[BC12]	extractable FHE	none	2	private
MIP (no signaling)	[KRR13, KRR14]	PIR	none	2	private
IPCP (public-coin)	[KR09]	PIR	none	2	private
LIP (any)	[BCI$^+$13]	linear-only encryption	CRS	1	private
LIP (algebraic)	[BCI$^+$13]	linear-only encoding	CRS	1	public
IOP (public-coin)	this work	(explicitly-programmable) RO	RO	1	public

Figure 1: Summary of some works (including this work) that obtain transformations from (information-theoretic) proof systems to corresponding (cryptographic) argument systems. For each, we highlight the main cryptographic tools used by the transformation and properties of the resulting argument system. These properties include: the setup model; the number of messages exchanged by the prover and verifier (non-interactive argument systems are 1 message); and whether the verifier relies on public or private randomness. The abbreviations used in the table are: CRF="collision resistant functions", CRS="common reference string", FHE="fully homomorphic encryption", PIR="private information retrieval", RO="random oracle".

1.3 Results

We present three main contributions: one is definitional and the other two are technical in nature.

1.3.1 Interactive oracle proofs

A new proof system model. We introduce a new proof system model: *interactive oracle proofs* (IOPs).[1] This model naturally combines aspects of interactive and probabilistically checkable proofs; namely, an IOP generalizes an interactive proof as follows: the verifier is not required to read the prover's messages in their entirety; rather, the verifier has oracle access to the prover's messages (viewed as strings), and may probabilistically query these messages. In more detail, a k-round IOP comprises k rounds of interaction. In the i-th round of interaction: the verifier sends a message m_i to the prover, which he reads in full; then the prover replies with a message f_i to the verifier, which he can query (via random access) in this and all later rounds. After the k rounds of interaction, the verifier either accepts or rejects. See Figure 2 for a diagram.

Like the PCP model, two fundamental measures of efficiency in the IOP model are the *proof length* p, which is the total number of bits in all of the prover's messages, and the *query complexity* q, which is the total number of locations queried by the verifier across all of the prover's messages. Unlike the PCP model, another fundamental measure of efficiency is the round complexity k; the PCP model can then be viewed as a special case where $k = 1$ (and the first verifier message is empty).

Basic complexity-theoretic properties. We show that IOPs characterize NEXP (like PCPs); both sequential and parallel repetition of IOPs yield (perfect) exponential soundness error reduction (like IPs); and any IOP can be converted into a public-coin one (like IPs). These basic complexity-theoretic properties confirm that our definition of IOP is a natural way to combine aspects of PCPs and IPs.

[1] Independent of our work, [RRR16] introduce *Probabilistically Checkable Interactive Proofs*, which are equivalent to our IOPs.

Motivation: efficiency. IOPs generalize both IPs, by treating the prover's messages as oracle strings, and PCPs, by allowing for more than 1 round. These degrees of freedom enable IOPs to retain the expressive power of PCPs (supporting all languages in NEXP, unlike IPs), while also allowing for additional efficiency.

For example, [BCGV16] obtain unconditional zero knowledge via a 2-round IOP with quasilinear proof length; in comparison, such a result is not known for PCPs (or even IPCPs [KR08]). Moreover, when combined with our compiler (see next contribution) we obtain blackbox unconditional zero-knowledge with quasilinear prover and polylogarithmic verifier in the random-oracle model, improving on results of [IMSX15].

As another example, [BCG$^+$16] obtain 3-round IOPs for circuit satisfiability *with linear proof length and constant query complexity*, while for PCPs prior work only achieves sublinear query complexity [BKK$^+$13]. To do so, [BCG$^+$16] show that *sumcheck* [LFKN92, Sha92] and *proof composition* [AS98] (used in many PCP constructions such as [ALM$^+$98, HS00, BGH$^+$04]) have more efficient "IOP analogues", which in turn imply a number of probabilistic checking results that are more efficient than corresponding ones that only rely on PCPs. We briefly sketch the intuition for why interactive proof composition, via IOPs, is more efficient. In a composed proof, the prover first writes a part π_0 of the proof (e.g., in [ALM$^+$98] π_0 is an evaluation of a low-degree multivariate polynomial, and in [BS08] it is an evaluation of a low-degree univariate polynomial). Then, to demonstrate that π_0 has certain good properties (e.g., it is low degree), the prover also appends a (long) sequence of sub-proofs, where each sub-proof allegedly demonstrates to the verifier that a subset of entries of π_0 is "good". Afterwards, in another invocation of the recursion, the prover appends to each sub-proof a sequence of sub-sub-proofs, and so on. A crucial observation is that the verifier typically queries locations of only a small number of such sub-proofs; moreover, once the initial proof π_0 is fixed, soundness is not harmed if the verifier randomly selects the set of sub-proofs he wants to see and tells this to the prover. In sum, in many PCP constructions (including the aforementioned ones), *the proof length can be greatly reduced via interaction between the prover and verifier*, via an IOP.

As yet another example, [RRR16] use IOPs to obtain doubly-efficient constant-round IPs for polynomial-time bounded-space computations. The result relies on an "amortization theorem" for IOPs that states that, for a so-called *unambiguous* IOPs, batch verification of multiple statements can be more efficient than simply running an independent IOP for each statement.

Remark 1.1 (comparison with IPCP)**.** Kalai and Raz [KR08] introduce and study *interactive PCPs* (IPCPs), a model of proof system that also combines aspects of IPs and PCPs, but in a different way: an IPCP is an IP in which the verifier additionally has oracle access to a PCP provided before the start of the interactive proof. An IPCP can be viewed as a special case of an IOP, i.e., it is an IOP in which the verifier has oracle access to the first prover message, but must read in full subsequent prover messages. The works of [KR08, GKR08] show that boolean formulas with n variables, size m, and depth d have IPCPs where the PCP's size is polynomial in d and n and the communication complexity of the subsequent IP is polynomial in d and $\log m$. This shows that even IPCPs give efficiency advantages over both IPs and PCPs given separately.

1.3.2 From interactive oracle proofs to non-interactive random-oracle proofs

We give a polynomial-time transformation that maps any public-coin interactive oracle proof (IOP) to a corresponding non-interactive random-oracle proof (NIROP). We prove that the soundness of the output proof is tightly characterized by the soundness of the IOP verifier against *state restoration attacks*, a class of rewinding attacks on the verifier that we now describe.

At a high level, a state restoration attack against an IOP verifier works as follows: the malicious prover and the verifier start interacting, as they normally would in an IOP; at any moment, however, the prover can choose to set the verifier to any state at which the verifier has previously been, and the verifier then continues

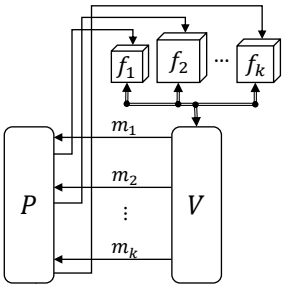

Figure 2: An interactive oracle proof system.

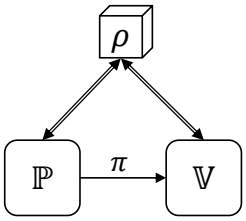

Figure 3: A non-interactive random-oracle proof system.

onwards from that point with fresh randomness. Of course, if the prover could restore the verifier's state an unbounded number of times, the prover would eventually succeed in making the verifier accept. We thus only consider malicious provers that interact with the verifier for at most a certain number of rounds: for $b \in \mathbb{N}$, we say a prover is b-*round* if it plays at most b rounds during any interaction with any verifier. Then, we say that an IOP has state restoration soundness $s_{\mathrm{sr}}(\mathbb{x}, b)$ if every b-round state-restoring prover cannot make the IOP verifier accept an instance $\mathbb{x}$ (not in the language) with probability greater than $s_{\mathrm{sr}}(\mathbb{x}, b)$.

Informally, our result about transforming IOPs into NIROPs can be stated as follows.

Theorem 1.2 (IOP $\to$ NIROP)**.** *There exists a polynomial-time transformation T such that, for every relation $\mathscr{R}$, if (P, V) is a public-coin interactive oracle proof system for $\mathscr{R}$ with state restoration soundness $s_{\mathrm{sr}}(\mathbb{x}, b)$, then $(\mathbb{P}, \mathbb{V}) := T(P, V)$ is a non-interactive random-oracle proof system for $\mathscr{R}$ with soundness*

$$s_{\mathrm{sr}}(\mathbb{x}, m) + O(m^2 2^{-\lambda}) \ ,$$

where m is an upper bound on the number of queries to the random oracle that a malicious prover can make, and λ is a security parameter. The aforementioned soundness is tight up to small factors. (Good state restoration soundness can be obtained, e.g., via parallel repetition as in Remark 1.6.)

Moreover, we prove that the transformation T is benign in the sense that it preserves natural properties of the IOP. Namely, (1) the runtimes of the NIROP prover and verifier are linear in those of the IOP prover and verifier (up to a polynomial factor in λ); (2) the NIROP is a proof of knowledge if the IOP is a proof of knowledge; and (3) the NIROP is (malicious-verifier) statistical zero knowledge if the IOP is honest-verifier statistical zero knowledge. See Theorem 7.1 for the formal statement; the statement employs the notion of *restricted state restoration soundness* (see Section 5.4) as it allows for a tighter lower bound on soundness.

An immediate application is obtaining blackbox unconditional ZK in the random oracle model with quasilinear prover and polylogarithmic verifier, improving on results of [IMSX15], by plugging the work of [BCGV16] into our compiler.

Our compiler can be viewed as a generalization of the Fiat–Shamir paradigm for public-coin IPs [FS86, PS96], and of the "CS proof" constructions of Micali [Mic00] and Valiant [Val08] for PCPs. Our analysis of the compiler gives, in particular, a unified understanding of all of these constructions, and motivates the study of state restoration attacks, not only for IOPs, but also for IPs and PCPs. (Indeed, we are not aware of works that study the security of the Fiat–Shamir paradigm, in the random oracle model, applied to a public-coin IP with arbitrary number of rounds; the analyses that we are aware of focus on the case of 2 rounds.)

Our next contribution is a first set of results about such kinds of attacks, as described in the next section.

Remark 1.3 (resetting, backtracking)**.** We compare state restoration soundness with other soundness notions:

- State restoration attacks generalize resetting attacks [BGGL01], in which the prover invokes multiple verifier incarnations with the *same randomness prefix*. Resettable soundness is thus bounded from above by state restoration soundness (and so bounds on the former do not always imply bounds on the latter).
- State restoration is closely related to backtracking [BD16]. The two notions differ in that: (1) backtracking "charges" more for restoring verifier states that are further into the past, and (2) backtracking also considers the case of the verifier restoring states of the prover (as part of the completeness property of the protocol); backtracking soundness is thus polynomially related to state restoration soundness.
  Bishop and Dodis [BD16] give a compiler from a public-coin IP to an error-resilient IP, whose soundness is related to the backtracking soundness of the original IP; essentially, they use hashing techniques to limit a malicious prover impersonating an adversarial channel to choosing when to backtrack the protocol. Their setting is a completely different example in which backtracking, and thus state restoration, plays a role.

Remark 1.4 (zero knowledge and programmability)**.** There are several notions of zero knowledge in the random oracle model, depending on "how programmable" the random oracle is (see Remark 2.2). The notion that we use is zero knowledge in the explicitly-programmable random oracle (EPRO) model; the stronger notion in the non-programmable random oracle model is not achievable for NIROPs (see Appendix C). (And, as in prior works, soundness and proof of knowledge do not rely on any programming of the random oracle.)

1.3.3 State restoration attacks on interactive oracle proofs

The analysis of our transformation from public-coin IOPs to NIROPs highlights state restoration soundness as a notion that merits further study. We provide two results in this direction. First, we prove tight upper and lower bounds on state restoration soundness in terms of the IOP's (standard) soundness and round complexity.

Theorem 1.5. *For any relation $\mathscr{R}$, public-coin k-round IOP for $\mathscr{R}$, and instance $\mathbb{x}$ not in the language of $\mathscr{R}$,*

$$\forall\, b \geq k(\mathbb{x}) + 1, \quad \left\lfloor \frac{b}{k(\mathbb{x})+1} \right\rfloor s(\mathbb{x})(1 - o(1)) \leq s_{\mathrm{sr}}(\mathbb{x}, b) \leq \binom{b}{k(\mathbb{x})+1} s(\mathbb{x}) \ ,^{2}$$

where $s_{\mathrm{sr}}(\mathbb{x}, b)$ is the state restoration soundness of IOP and $s(\mathbb{x})$ its (standard) soundness for the instance $\mathbb{x}$. Also, the bounds are tight: there are IOPs that meet the lower bound and IOPs that meet the upper bound.

Remark 1.6 (good state restoration soundness)**.** One way to obtain state restoration soundness $2^{-\lambda}$ in the general case is to apply r-fold parallel repetition to the IOP with $r = \Omega(\frac{k \log b + \lambda}{\log s(\mathbb{x})})$; note that r is polynomially bounded for natural choices of k, b, λ. This choice of r is pessimistic, because for IOPs that do not meet the upper bound (e.g., they are "robust" against such attacks) a smaller choice of r suffices.

Second, we study the structure of optimal state restoration attacks: we prove that, for any public-coin IOP, there is a simple state restoration attack that has optimal expected cost, where cost is the number of rounds until the prover wins. This result relies on a correspondence that we establish between IOP verifiers and certain games, which we call *tree exploration games*, pitting one player against Nature. We go in more detail about this result in later sections (see Section 1.4 and Section 8).

[2] We note that [BGGL01] prove an analogous upper bound on the more restrictive notion of resettable soundness (see Remark 1.3), and does not imply our bound. Also, [BD16] prove an analogous, weaker upper bound on the related notion of backtracking soundness (see Remark 1.3). Neither of the two studies lower bounds, or tightness of bounds.

1.4 Techniques

We summarize the techniques that we use to prove our technical contributions.

The transformation. Our transformation maps any public-coin IOP to a corresponding NIROP, and it generalizes two transformations that we now recall.

The first transformation is the Fiat–Shamir paradigm [FS86, PS96], which maps any public-coin IP to a corresponding NIROP, and it works as follows. The NIROP prover runs the interaction between the IP prover and the IP verifier "in his head", by setting the IP verifier's next message to be the output of the random oracle on the query that equals the transcript of previously exchanged messages. The NIROP prover sends a non-interactive proof that contains the final transcript of interaction; the NIROP verifier checks the proof's validity by checking that all of the IP verifier's messages are computed correctly through the random oracle.

The second transformation is the "CS proof" construction of Micali [Mic00] and Valiant [Val08], which maps any PCP to a corresponding NIROP, and it works as follows. The NIROP prover first commits to the PCP via a Merkle tree [Mer89a] based on the random oracle, then queries the random oracle with the root of this tree to obtain randomness for the PCP verifier, and finally sends a non-interactive proof that contains the root as well as authentication paths for each query by the PCP verifier to the PCP; the NIROP verifier checks the proof's validity by checking that the PCP verifier's randomness is computed correctly through the random oracle, and that all authentication paths are valid. (The transformation can be viewed as a non-interactive variant of Kilian's protocol [Kil92, BG08] that uses ideas from the aforementioned Fiat–Shamir paradigm.)

Our transformation takes as input IOPs, for which both IPs and PCPs are special cases, and hence must support both (i) multiple rounds of interaction between the IOP prover and IOP verifier, as well as (ii) oracle access by the IOP verifier to the IOP prover messages. Given an instance $\mathbb{x}$, the NIROP prover thus uses the random oracle ρ to run the interaction between the IOP prover and the IOP verifier "in his head" in a way that combines the aforementioned two approaches, as follows. First, the NIROP prover computes an initial value $\sigma_0 := \rho(\mathbb{x})$. Then, for $i = 1, 2, \ldots$, it simulates the i-th round by deriving the IOP verifier's i-th message m_i as $\rho(\mathbb{x} \| \sigma_{i-1})$, compressing the IOP prover's i-th message f_i via a Merkle tree to obtain the root rt_i, and computing the new value $\sigma_i := \rho(\mathsf{rt}_i \| \sigma_{i-1})$. The values $\sigma_0, \sigma_1, \ldots$ are related by the Merkle–Damgård transform [Dam89, Mer89b] that, intuitively, enforces ordering between rounds. If there are $k(\mathbb{x})$ rounds of interaction, then $\rho(\mathbb{x} \| \sigma_{k(\mathbb{x})})$ is used as randomness for the queries to $f_1, \ldots, f_{k(\mathbb{x})}$. The NIROP prover provides in the non-interactive proof all the roots rt_i, the final value $\sigma_{k(\mathbb{x})}$, the answers to the queries, and an authentication path for each query. This sketch omits several details; see Section 6 and Figure 6.

Soundness analysis of the transformation. We prove that the soundness of the NIROP produced by the above transformation is tightly characterized by the state restoration soundness of the underlying IOP. This characterization comprises two arguments: an upper bound and a lower bound on the NIROP's soundness. We only discuss the upper bound here: proving that the soundness (error) of the NIROP is at most the soundness (error) of the IOP against state restoration attacks, up to small additive factors.

The upper bound essentially implies that all that a malicious prover $\tilde{\mathbb{P}}$ can do to attack the NIROP verifier is to conduct a state restoration attack against the underlying IOP verifier "in his own head": roughly, $\tilde{\mathbb{P}}$ can provide multiple inputs to the random oracle in order to induce multiple fresh samples of verifier messages for a given round so to find a lucky one, or instead go back to previous rounds and do the same there.

In more detail, the proof itself relies on a reduction: given a malicious prover $\tilde{\mathbb{P}}$ against the NIROP verifier, we show how to construct a corresponding malicious prover $\tilde{P}$ that conducts a state restoration attack against the underlying IOP verifier. We prove that the winning probability of $\tilde{P}$ is essentially the same as that of $\tilde{\mathbb{P}}$; moreover, we also prove that the reduction preserves the resources needed for the attack in the sense that if $\tilde{\mathbb{P}}$ asks at most m queries to the random oracle, then $\tilde{P}$ plays at most m rounds during the attack.

Intuitively, the construction of $\tilde{P}$ in terms of $\tilde{\mathbb{P}}$ must use some form of extraction: $\tilde{\mathbb{P}}$ outputs a non-interactive proof that contains only (i) the roots that (allegedly) are commitments to underlying IOP prover's messages, and (ii) answers to the IOP verifier's queries and corresponding authentication paths; in contrast, $\tilde{P}$ needs to actually output these IOP prover's messages. In principle, the malicious prover $\tilde{\mathbb{P}}$ may not have "in mind" any underlying IOP prover, and we must prove that, nevertheless, there is a way for $\tilde{P}$ to extract some IOP prover message for each round that convince the verifier with the claimed probability.

Our starting point is the extractor algorithm of Valiant [Val08] for the "CS proof" construction of Micali [Mic00]: Valiant proves that Micali's NIROP construction is a proof of knowledge by exhibiting an algorithm, let us call it *Valiant's extractor*, that recovers the underlying PCP whenever the NIROP prover convinces the NIROP verifier with sufficient probability. Our setting differs from Valiant's in that the IOP prover $\tilde{P}$ obtained from the NIROP prover $\tilde{\mathbb{P}}$ needs to be able to extract multiple times, "on the fly", while interacting with the IOP verifier; this more complex setting can potentially cause difficulties in terms of extractor size (e.g., if relying on rewinding the NIROP prover) or correlations (e.g., when extracting multiple times from the same NIROP prover). We tackle the more complex setting in two steps.

First, we prove an extractability property of Valiant's extractor and state it as a property of Merkle trees in the random oracle model (see Section 3.1). Informally, we prove that, except with negligible probability, whenever an algorithm with access to a random oracle outputs multiple Merkle tree roots each accompanied with some number of (valid) authentication paths, it holds that Valiant's extractor run separately on each of these roots outputs a decommitment that is consistent with each of the values revealed in authentication paths relative to that root. We believe that distilling and proving this extractability property of Valiant's extractor is of independent interest.

Second, we show how the IOP prover $\tilde{P}$ can interact with an IOP verifier, by successively extracting messages to send, throughout the interaction, by invoking Valiant's extractor multiple times on $\tilde{\mathbb{P}}$ relative to different roots. The IOP prover $\tilde{P}$ does not rely on rewinding $\tilde{\mathbb{P}}$, and its complexity is essentially that of a single run of $\tilde{\mathbb{P}}$ plus a small amount of work.

Preserving proof of knowledge. We prove that the above soundness analysis can be adapted so that, if the underlying IOP is a proof of knowledge, then we can construct an extractor to show that the resulting NIROP is also a proof of knowledge.

Preserving zero knowledge. We prove that, if the underlying IOP is *honest-verifier* statistical zero knowledge, then the resulting NIROP is statistical zero knowledge (i.e., is a non-interactive statistical zero knowledge proof in the explicitly-programmable random oracle model). This is because the transformation uses a Merkle tree with suitable privacy guarantees (see Section 3.2) to construct the NIROP. Indeed, the authentication path for a leaf in the Merkle tree reveals the sibling leaf, so one must ensure that the sibling leaf does not leak information about other values; this follows by letting leaves be commitments to the underlying values. A Merkle tree with privacy is similarly used by [IMS12, IMSX15], along with honest-verifier PCPs, to achieve zero knowledge in modifications of Kilian's [Kil92, BG08] and Micali's [Mic00] constructions.

Understanding state restoration attacks. We prove tight upper and lower bounds to state restoration soundness in terms of the IOP's (standard) soundness and round complexity k. The upper bound takes the form of a reduction: given a b-round state-restoring malicious prover $\tilde{P}_{\mathrm{sr}}$ that makes the IOP verifier accept with probability s_{sr}, we construct a (non state-restoring) malicious prover $\tilde{P}$ that makes the IOP verifier accept with probability at least $\binom{b}{k+1}^{-1} s_{\mathrm{sr}}$. Informally, $\tilde{P}$ internally simulates $\tilde{P}_{\mathrm{sr}}$, while interacting with the "real" IOP verifier, as follows: $\tilde{P}$ first selects a random subset S of $\{1, \ldots, b\}$ with cardinality $k+1$, and lets $S[i]$ be the i-th smallest value in S; then, $\tilde{P}$ runs $\tilde{P}_{\mathrm{sr}}$ and simulates its state restoration attack on a "virtual" IOP verifier, executing round j (a) by interacting with the real verifier if $j = S[i]$ for some i; (b) by sampling fresh randomness otherwise. While this reduction appears wasteful (since it relies on S being a good guess),

we show that there are IOPs for which the upper bound is tight. In other words, the sharp degradation as a function of round complexity (for large b, $\binom{b}{k+1} \approx b^{k+1}/(k+1)!$) is inherent for some choices of IOPs; this also gives a concrete answer to the intuition that compiling IOPs with large round complexity to NIROPs is "harder" (i.e., incurs in a greater soundness loss) than for IOPs with small round complexity. As for the lower bound on state restoration soundness, it takes the form of a universal state restoration attack that always achieves the lower bound; this bound is also tight.

While state restoration soundness may be far, in the worst case, from (standard) soundness for IOPs with large round complexity, it need not always be far. We thus investigate state restoration soundness for any particular IOP, and derive a simple attack strategy (which depends on the IOP) that we prove has optimal expected cost, where cost is the number of rounds until the prover wins. To do so, we "abstract away" various details of the proof system to obtain a simple game-theoretic notion, which we call *tree exploration games*, that pits a single player against Nature in reaching a node of a tree with label 1. Informally, such a game is specified by a rooted tree T and a predicate function ϕ that maps T's vertices to $\{0, 1\}$. The game proceeds in rounds: in the i-th round, a subtree $S_{i-1} \subseteq T$ is *accessible* to the player; the player picks a node $v \in S_{i-1}$, and Nature randomly samples a child u of v; the next accessible subtree is $S_i := S_{i-1} \cup \{u\}$. The initial S_0 is the set consisting of T's root vertex. The player wins in round r if there is $v \in S_r$ with $\phi(v) = 1$.

We establish a correspondence between state restoration attacks and strategies for tree exploration games, and then show a simple greedy strategy for such games with optimal expected cost. Via the correspondence, a strategy's cost determines whether the underlying IOP is strong or weak against sate restoration attacks.

1.5 Roadmap

The rest of this paper is organized as follows. In Section 2, we provide basic notations and definitions, including for Merkle trees and non-interactive random-oracle proofs (NIROPs). In Section 3, we state and prove the extractability and privacy properties of Merkle trees that we use in this work. In Section 4, we introduce interactive oracle proofs (IOPs), the new proof system that we study. In Section 5, we define state restoration attacks against IOPs and present our results about them. In Section 6, we describe a transformation for compiling IOPs into NIROPs; after that, in Section 7, we present our tight analysis of this transformation. In Section 8, we define tree exploration games and present our results about them.

2 Preliminaries

2.1 Basic notations

We denote the security parameter by λ. We let $f = O_\lambda(g)$ mean there exists $c > 0$ such that $f = O(\lambda^c g)$. For $f\colon \{0,1\}^* \to \mathbb{R}$, we define $\hat{f}\colon \mathbb{N} \to \mathbb{R}$ as $\hat{f}(n) := \max_{x \in \{0,1\}^n} f(x)$.

Languages and relations. We denote by $\mathscr{R}$ a relation consisting of pairs $(\mathtt{x}, \mathtt{w})$, where $\mathtt{x}$ is the *instance* and $\mathtt{w}$ is the *witness*, and by $\mathscr{R}_n$ the restriction of $\mathscr{R}$ to instances of size exactly n. We denote by $\mathscr{L}(\mathscr{R})$ the language corresponding to $\mathscr{R}$. For notational convenience, we define $\bar{\mathscr{L}}(\mathscr{R}_n) := \{\mathtt{x} \in \{0,1\}^n \mid \mathtt{x} \notin \mathscr{L}(\mathscr{R})\}$.

Random oracles. We denote by $\mathcal{U}(\lambda)$ the uniform distribution over all functions $\rho\colon \{0,1\}^* \to \{0,1\}^\lambda$ (implicitly defined by the probabilistic algorithm that assigns, uniformly and independently at random, a λ-bit string to each new input). If ρ is sampled from $\mathcal{U}(\lambda)$, then we say that ρ is a *random oracle*. Given an oracle algorithm A, $\mathsf{NumQueries}(A, \rho)$ is the number of oracle queries that A^ρ makes. We say that A is *m-query* if $\mathsf{NumQueries}(A, \rho) \le m$ for any $\rho \in \mathcal{U}(\lambda)$.

Statistical distance. The statistical distance between two discrete random variables X and Y with support V is $\Delta(X;Y) := \frac{1}{2}\sum_{v \in V} |\Pr[X = v] - \Pr[Y = v]|$. We say that X and Y are *δ-close* if $\Delta(X;Y) \le \delta$.

Remark 2.1. An oracle $\rho \in \mathcal{U}(\lambda)$ outputs λ bits. Occasionally we need ρ to output more than λ bits; in such cases (we point out where), we implicitly extend ρ's output via a simple strategy, e.g., we set $y := y_1 \| y_2 \| \cdots$ where $y_i := \rho(i \| x)$ and prefix 0 to all inputs that do not require an output extension.

2.2 Merkle trees

We use Merkle trees [Mer89a] based on random oracles as succinct commitments to long lists of values for which one can cheaply decommit to particular values in the list. Concretely, a *Merkle-tree scheme* is a tuple $\mathsf{MERKLE} = (\mathsf{MERKLE.GetRoot}, \mathsf{MERKLE.GetPath}, \mathsf{MERKLE.CheckPath})$ that uses a random oracle ρ sampled from $\mathcal{U}(\lambda)$ and works as follows.

- $\mathsf{MERKLE.GetRoot}^\rho(\mathbf{v}) \to \mathsf{rt}$. Given input list $\mathbf{v} = (v_i)_{i=1}^n$, the *root generator* $\mathsf{MERKLE.GetRoot}$ computes, in time $O_\lambda(n)$, a root rt of the Merkle tree over the list $\mathbf{v}$.
- $\mathsf{MERKLE.GetPath}^\rho(\mathbf{v}, i) \to \mathsf{ap}$. Given input list $\mathbf{v}$ and index i, the *authentication path generator* $\mathsf{MERKLE.GetPath}$ computes the authentication path ap for the i-th value in $\mathbf{v}$.
- $\mathsf{MERKLE.CheckPath}^\rho(\mathsf{rt}, i, v, \mathsf{ap}) \to b$. Given root rt, index i, input value v, and authentication path ap, the *path checker* $\mathsf{MERKLE.CheckPath}$ outputs $b = 1$ if ap is a valid path for v as the i-th value in a Merkle tree with root rt; the check can be carried out in time $O_\lambda(\log_2 n)$.

We assume that an authentication path ap contains the root rt, position i, and value v; accordingly, we define $\mathsf{Root}(\mathsf{ap}) := \mathsf{rt}$, $\mathsf{Position}(\mathsf{ap}) := i$, and $\mathsf{Value}(\mathsf{ap}) := v$.

Merkle trees are well known, so we do not review their construction. Less known, however, are the hiding and extractability properties of Merkle trees that we rely on in this work; we describe these in Section 3.

2.3 Non-interactive random-oracle proofs

A *non-interactive random-oracle proof system* for a relation $\mathscr{R}$ with soundness $s\colon \{0,1\}^* \to [0,1]$ is a tuple $(\mathbb{P}, \mathbb{V})$, where $\mathbb{P}, \mathbb{V}$ are (oracle) probabilistic algorithms, that satisfies the following properties.

1. COMPLETENESS. For every $(\mathtt{x}, \mathtt{w}) \in \mathscr{R}$ and $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathbb{V}^{\rho}(\mathtt{x}, \pi) = 1 \;\middle|\; \begin{array}{c} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \mathbb{P}^{\rho}(\mathtt{x}, \mathtt{w}) \end{array}\right] = 1 \ .$$

2. SOUNDNESS. For every $\mathtt{x} \notin \mathscr{L}(\mathscr{R})$, m-query $\tilde{\mathbb{P}}$, and $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathbb{V}^{\rho}(\mathtt{x}, \pi) = 1 \;\middle|\; \begin{array}{c} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \tilde{\mathbb{P}}^{\rho} \end{array}\right] \leq s(\mathtt{x}, m, \lambda) \ .$$

Complexity measures. Beyond soundness, we consider other complexity measures. Given $p\colon \{0,1\}^* \to \mathbb{N}$, we say that $(\mathbb{P}, \mathbb{V})$ has proof length p if π has length $p(\mathtt{x}, \lambda)$. Given $t_{\mathrm{prv}}, t_{\mathrm{ver}}\colon \{0,1\}^* \to \mathbb{N}$, we say that $(\mathbb{P}, \mathbb{V})$ has prover time complexity t_{prv} and verifier time complexity t_{ver} if $\mathbb{P}^{\rho}(\mathtt{x}, \mathtt{w})$ runs in time $t_{\mathrm{prv}}(\mathtt{x}, \lambda)$ and $\mathbb{V}^{\rho}(\mathtt{x}, \pi)$ runs in time $t_{\mathrm{ver}}(\mathtt{x}, \lambda)$. In sum, we say that $(\mathbb{P}, \mathbb{V})$ has complexity $(s, p, t_{\mathrm{prv}}, t_{\mathrm{ver}})$ if $(\mathbb{P}, \mathbb{V})$ has soundness s, proof length p, prover time complexity t_{prv}, and verifier time complexity t_{ver}.

Proof of knowledge. Given $e\colon \{0,1\}^* \to [0,1]$, we say that $(\mathbb{P}, \mathbb{V})$ has proof of knowledge e if there exists a probabilistic polynomial-time algorithm $\mathbb{E}$ (the *extractor*) such that, for every $\mathtt{x}$, m-query $\tilde{\mathbb{P}}$, and $\lambda \in \mathbb{N}$,

$$\Pr\left[(\mathtt{x}, \mathtt{w}) \in \mathscr{R} \;\middle|\; \mathtt{w} \leftarrow \mathbb{E}^{\tilde{\mathbb{P}}}(\mathtt{x}, 1^m, 1^\lambda)\right] \geq \Pr\left[\mathbb{V}^{\rho}(\mathtt{x}, \pi) = 1 \;\middle|\; \begin{array}{c} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \tilde{\mathbb{P}}^{\rho} \end{array}\right] - e(\mathtt{x}, m, \lambda) \ .$$

The notation $\mathbb{E}^{\tilde{\mathbb{P}}}(\mathtt{x}, 1^m, 1^\lambda)$ means that $\mathbb{E}$ receives as input $(\mathtt{x}, 1^m, 1^\lambda)$ and may obtain an output of $\tilde{\mathbb{P}}^{\rho}$ for choices of oracles ρ, as we now describe. At any time, $\mathbb{E}$ may send a λ-bit string z to $\tilde{\mathbb{P}}$; then $\tilde{\mathbb{P}}$ interprets z as the answer to its last query to ρ (if any) and then continues computing until it reaches either its next query θ or its output π; then this query or output is sent to $\mathbb{E}$ (distinguishing the two cases in some way); in the latter case, $\tilde{\mathbb{P}}$ goes back to the start of its computation (with the same randomness and any auxiliary inputs). Throughout, the code, randomness, and any auxiliary inputs of $\tilde{\mathbb{P}}$ are not available to $\mathbb{E}$.

Zero knowledge. Given $z\colon \{0,1\}^* \to [0,1]$, we say that $(\mathbb{P}, \mathbb{V})$ has z-statistical zero knowledge (in the explicitly-programmable random oracle model) if there exists a probabilistic polynomial-time algorithm $\mathbb{S}$ (the *simulator*) such that, for every $(\mathtt{x}, \mathtt{w}) \in \mathscr{R}$ and unbounded distinguisher D, the following two probabilities are $z(\mathtt{x}, \lambda)$-close:

$$\Pr\left[D^{\rho[\mu]}(\pi) = 1 \;\middle|\; \begin{array}{c} \rho \leftarrow \mathcal{U}(\lambda) \\ (\pi, \mu) \leftarrow \mathbb{S}^{\rho}(\mathtt{x}) \end{array}\right] \text{ and } \Pr\left[D^{\rho}(\pi) = 1 \;\middle|\; \begin{array}{c} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \mathbb{P}^{\rho}(\mathtt{x}, \mathtt{w}) \end{array}\right] \ .$$

Above, $\rho[\mu]$ is the function such that, given an input x, equals $\mu(x)$ if μ is defined on x, or $\rho(x)$ otherwise.

Remark 2.2. Zero knowledge in the random oracle model can be defined in several ways. Wee [Wee09] details the following hierarchy. (i) *Fully-programmable random oracle (FPRO):* the simulator has access to the random oracle but the distinguisher does not; (ii) *Explicitly-programmable random oracle (EPRO) [BR93]:* the simulator has access to the random oracle and also outputs a function μ, but the distinguisher has access to the same oracle modified so that a query to an input x in the domain of μ is answered with $\mu(x)$; (iii) *Non-programmable random oracle (NPRO) [Nie02, Pas03]:* the simulator and the distinguisher have access to the same random oracle. ZK in the NPRO implies ZK in the EPRO, and the latter implies ZK in the FPRO; there exist protocols that are ZK in the EPRO but not in the NPRO, and ones that are ZK in the FPRO but (assuming one-way permutations exist) not in the EPRO.

The above definition that we use for NIROPs follows ZK in the EPRO. We do not use the stronger notion of ZK in the NPRO, since Pass shows that it cannot be achieved for non-trivial relations (see Appendix C). In particular, zero-knowledge NIROPs are not *deniable* in the sense of [Pas03].

3 Extractability and privacy of Merkle trees

We describe the specific extractability and privacy properties of Merkle trees that we rely on in this work.

3.1 Extractability

We rely on a certain extractability property of Merkle trees: there is an efficient procedure for extracting the committed list in a Merkle-tree scheme. We call the procedure *Valiant's extractor*, and denote it by VE, because it is described in [Val08]. Our presentation of the extractor and its guarantee differs from [Val08] because our use of it in this work requires "distilling" a more general property; see Lemma 3.2 below.

The extractor. For any oracle algorithm A, integers $\ell, i^\star, i_{\mathsf{max}} > 0$ with $i^\star \in \{1, \ldots, i_{\mathsf{max}}\}$, and ρ sampled from $\mathcal{U}(\lambda)$, the procedure VE, given input $(A, \ell, i^\star, i_{\mathsf{max}})$ and with oracle access to ρ, works as follows.

1. Run A^ρ until it has asked i_{max} unique queries to ρ (and abort if A^ρ asks fewer than i_{max}). Along the way, record the queries $\theta_1, \ldots, \theta_{i_{\mathsf{max}}}$ and answers $\rho(\theta_1), \ldots, \rho(\theta_{i_{\mathsf{max}}})$, in order and omitting duplicates.
2. Parse each query θ_i as $\theta_i^0 \| \theta_i^1$ where θ_i^0 are the first λ bits of θ_i and θ_i^1 the second λ bits. For brevity, we write $z \in \theta_i$ if $z = \theta_i^0$ or $z = \theta_i^1$. (If a query has length not equal to 2λ, then $z \notin \theta_i$ for all z.)
3. If there exist indices i, j such that $i \neq j$ and $\rho(\theta_i) = \rho(\theta_j)$, abort.
4. If there exist indices i, j such that $i \leq j$ and $\rho(\theta_j) \in \theta_i$, abort.
5. Construct a directed graph G with nodes $V = \{\theta_1, \ldots, \theta_{i_{\mathsf{max}}}\}$ and edges $E = \{(\theta_i, \theta_j) : \rho(\theta_j) \in \theta_i\}$. Note that G is acyclic, every node has out-degree ≤ 2, and $\theta_1, \ldots, \theta_{i_{\mathsf{max}}}$ is a (reverse) topological ordering.
6. Output $\mathbf{v}$, the string obtained by traversing in order the first ℓ leaf nodes of the depth-$\lceil \log_2 \ell \rceil$ binary tree rooted at $\theta_{i^\star}$ and recording the first bit of each node. If any such node does not exist, set this entry to 0.

A sample execution of the extractor is depicted in Figure 4.

Remark 3.1. The queries to ρ asked by $\mathsf{VE}^\rho(A, \ell, i^\star, i_{\mathsf{max}})$ equals the first i_{max} queries to ρ asked by A^ρ (provided that A does not ask fewer than i_{max} queries). Later on we use this fact.

The extractor's guarantee. We interpret A's output as containing a (possibly empty) list of tuples of the form $(\mathsf{rt}, i, v, \mathsf{ap})$, where rt is a root, i an index, v a value, and ap an authentication path.[3] We define the following events:

(i) E_1 is the event that, for each tuple $(\mathsf{rt}, i, v, \mathsf{ap})$ output by A^ρ, $\mathsf{MERKLE.CheckPath}(\mathsf{rt}, i, v, \mathsf{ap}) = 1$;

(ii) E_2 is the event that, for each $\mathsf{rt} \in \{0,1\}^\lambda$, there exists an integer ℓ_{rt} such that if A^ρ outputs a tuple of the form $(\mathsf{rt}, \cdot, \cdot, \mathsf{ap})$ then ap is an authentication path for a ℓ_{rt}-leaf Merkle tree; and

(iii) E_3 is the event that, for every $\mathsf{rt} \in \{0,1\}^\lambda$ such that A^ρ outputs some tuple of the form $(\mathsf{rt}, \cdot, \cdot, \cdot)$, there is a unique $j_{\mathsf{rt}} \in \{0, \ldots, \mathsf{NumQueries}(A, \rho)\}$ such that $\rho(\theta_{j_{\mathsf{rt}}}) = \mathsf{rt}$ and, for every $i_{\mathsf{max}} \in \{j_{\mathsf{rt}}, \ldots, \mathsf{NumQueries}(A, \rho)\}$, $\mathbf{v} := \mathsf{VE}^\rho(A, \ell_{\mathsf{rt}}, j_{\mathsf{rt}}, i_{\mathsf{max}})$ is such that $\mathbf{v}$'s i-th entry equals v_i for any tuple of the form $(\mathsf{rt}, i, v, \mathsf{ap})$ output by A^ρ.

The extractability property that we rely on is the following.

Lemma 3.2. *Let A^ρ be a m-query algorithm. Then*

$$\Pr\left[(\neg(E_1 \wedge E_2)) \vee E_3 \mid \rho \leftarrow \mathcal{U}(\lambda)\right] \geq 1 - (m^2 + 1)2^{-\lambda} \ .$$

Proof. Observe the following.

- By the union bound, the probability that there exist indices i, j such that $(i \neq j) \wedge (\rho(\theta_i) = \rho(\theta_j))$ or $(i \leq j) \wedge (\rho(\theta_j) \in \theta_i)$ is at most $m^2 2^{-\lambda}$. If this occurs, we say that A^ρ has found a *collision*.

[3] Note that A's output may contain additional information not of the above form; if so, we simply ignore it for now.

- The probability that, for a tuple $(\mathsf{rt}, i, v, \mathsf{ap})$ output by A^ρ such that $\mathsf{MERKLE.CheckPath}(\mathsf{rt}, i, v, \mathsf{ap}) = 1$, the authentication path ap contains a node with no corresponding query is at most $2^{-\lambda}$, since this would mean that A^ρ has 'guessed' the answer to the query. In other words, no matter what strategy A uses to generate the result, if it does not query the oracle on this input then it can perform no better than chance.

Now suppose that $E_1 \wedge E_2$ occurs with probability δ. Then, with probability at least $\delta - (m^2+1)2^{-\lambda}$: (a) for each root rt output by A^ρ there is a unique query $\theta_{i^\star}$ such that $\rho(\theta_{i^\star}) = \mathsf{rt}$; (b) for each root rt output by A^ρ, if an authentication path ap claims to have root rt then ap appears in the tree rooted at $\theta_{i^\star}$ in G; and (c) the condition in the VE's Step 3 or Step 4 does not hold. In such a case we may take $j_{\mathsf{rt}} := i^\star$, and then $\mathsf{VE}^\rho(A, \ell_{\mathsf{rt}}, j_{\mathsf{rt}}, i_{\mathsf{max}})$ outputs a list $\mathbf{v}$ with the desired property. Hence, $\Pr[E_1 \wedge E_2 \wedge E_3] \geq \delta - (m^2+1)2^{-\lambda}$. The predicate is also satisfied if $\neg(E_1 \wedge E_2)$ occurs, which is the case with probability $1-\delta$ and is disjoint from $E_1 \wedge E_2 \wedge E_3$. The lemma follows. □

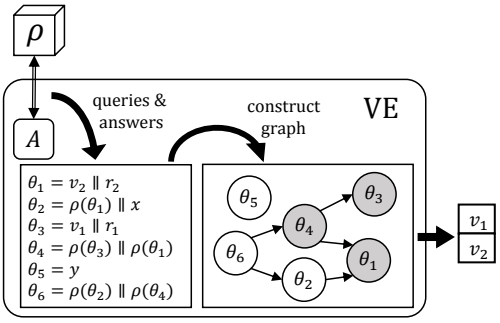

Figure 4: A diagram of an execution of Valiant's extractor VE, with input parameters $\ell = 2$, $i^\star = 4$, and $i_{\mathsf{max}} = 6$.

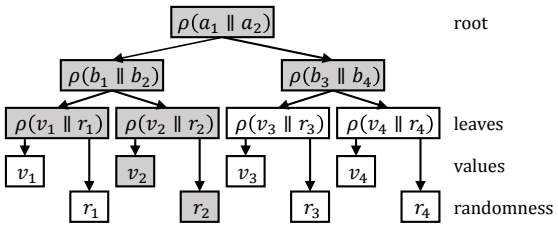

Figure 5: A diagram of the data structure of a Merkle tree with privacy. An authentication path for v_2 is shaded; the corresponding truncated authentication path is the same minus r_2 and v_2.

3.2 Privacy

We rely not only on the fact that the root rt of a Merkle tree is hiding, but also on the fact that an authentication path ap reveals no information about values other than the decommitted one. The latter property can be ensured via a slight tweak of the standard construction of Merkle trees: when committing to a list $\mathbf{v} = (v_i)_{i=1}^n$, the i-th leaf is not v_i but, instead, is a hiding commitment to v_i. In our case, we will store the value $\rho(v_i \| r_i)$ in the i-th leaf, where $r_i \in \{0,1\}^{2\lambda}$ is drawn uniformly at random; see Figure 5. (An authentication path for v_i then additionally includes r_i, and path verification is modified accordingly.) In what follows, we regard $\rho(v_i \| r_i)$ as a *leaf*, rather than v_i; moreover, a *truncated authentication path* ap'_i is identical to ap_i except that it does not contain r_i or v_i, and the *truncated Merkle tree* for $\mathbf{v}$ is $T'_{\mathbf{v}} := (\mathsf{ap}'_i)_{1 \leq i \leq n}$. Note that the *same* randomness $\mathbf{r} \in \{0,1\}^{2\lambda n}$ is used by $\mathsf{MERKLE.GetRoot}$ and $\mathsf{MERKLE.GetPath}$ (to be "in sync").

We summarize the privacy property of Merkle trees as above via the following definition and lemma.

Definition 3.3. *A Merkle-tree scheme has $z(n,\lambda)$-statistical privacy if there exists a probabilistic polynomial-time simulator S such that, for every list $\mathbf{v} = (v_i)_{i=1}^n$ and unbounded distinguisher D, the following two*

probabilities are $z(n,\lambda)$-close:

$$\Pr_{\mathbf{r}}\left[\begin{array}{c|r} I \subseteq \{1,\ldots,n\} & \rho \leftarrow \mathcal{U}(\lambda) \\ D^{\rho}(\mathsf{rt},(\mathsf{ap}_i)_{i\in I})=1 & \mathsf{rt} \leftarrow \mathsf{MERKLE.GetRoot}^{\rho}(\mathbf{v};\mathbf{r}) \\ & I \leftarrow D^{\rho} \\ & \forall\, i \in I\,,\ \mathsf{ap}_i \leftarrow \mathsf{MERKLE.GetPath}^{\rho}(\mathbf{v},i;\mathbf{r}) \end{array}\right]$$

and

$$\Pr\left[\begin{array}{c|r} I \subseteq \{1,\ldots,n\} & \rho \leftarrow \mathcal{U}(\lambda) \\ D^{\rho}(\mathsf{rt},(\mathsf{ap}_i)_{i\in I})=1 & I \leftarrow D^{\rho} \\ & (\mathsf{rt},(\mathsf{ap}_i)_{i\in I}) \leftarrow S^{\rho}(n,(i,v_i)_{i\in I}) \end{array}\right] .$$

We make no assumption on the power of the distinguisher D in the definition above. In particular, D may query the random oracle ρ at every input, and use the information to attempt to learn v_i for some $i \notin I$. For example, for some ρ, it is the case that $\Pr_r[v = 1 \mid \rho(v\|r) = x] \gg \Pr_r[v = 0 \mid \rho(v\|r) = x]$ for $x = \rho(v_2\|r_2)$, in which case D can determine v_2 from ap_1 with good accuracy. The next lemma shows that the probability that D gains a significant statistical advantage in this way (or otherwise) is negligible in λ.

Lemma 3.4. *There exists a Merkle-tree scheme having $z(n,\lambda)$-statistical privacy with $z(n,\lambda) := n2^{-\lambda/4+2}$.*

Proof. Given $\mathbf{v} = (v_i)_{i=1}^n$ and $I \subseteq \{1,\ldots,n\}$, we define a sequence of random variables $W_0,\ldots,W_\ell$ for which $W_0 = T'_{\mathbf{v}}$ and W_ℓ contains the output of a probabilistic polynomial time simulator S (see below) on input $(n,(i,v_i)_{i\in I})$ and with access to a random oracle ρ. The random variables are defined as follows:

- We obtain W_1 from W_0 by (i) replacing each leaf node $j \in [n] - I$ with an independently uniformly random string in $\{0,1\}^\lambda$; (ii) 'marking' each such leaf; and (iii) recomputing the rest of the tree.
- For $i \in \{2,\ldots,\ell\}$, we obtain W_i from W_{i-1} by choosing a node whose children are both marked and replacing its value with a string drawn uniformly at random from $\{0,1\}^\lambda$, marking that node, and then recomputing the Merkle tree above it. (Note that the authentication paths for the leaves $i \in I$ are still valid with respect to the root and each other.) The last W_ℓ is when there are no such nodes remaining.

We now argue that the statistical distance between W_0 and W_ℓ is at most $n2^{-\lambda/4+1}$. For this, we require the following two facts:

- FACT 1. For all but a $2^{-\lambda}$ fraction of $\rho \in \mathcal{U}(\lambda)$, $\rho(r)$ is $2^{-\lambda/4}$-close to r' where r is random in $\{0,1\}^{2\lambda}$ and r' is random in $\{0,1\}^\lambda$. (*Proof:* For $z \in \{0,1\}^\lambda$ and $\rho \in \mathcal{U}(\lambda)$ fixed, define $\delta_\rho(z) := |\Pr[\rho(r) = z] - \Pr[X = z]|$ where $r \in \{0,1\}^{2\lambda}$ and $X \in \{0,1\}^\lambda$ are random. Clearly $\Pr[X = z] = 2^{-\lambda}$. By a Chernoff bound, for every $z \in \{0,1\}^\lambda$: $\Pr_\rho[\delta_\rho(z) \geq 2^{-5\lambda/4} \,|\, \rho \leftarrow \mathcal{U}(\lambda)] \leq 2^{-2\lambda}$. By a union bound, $\Pr_\rho[\exists\, z \in \{0,1\}^\lambda \text{ s.t. } \delta_\rho(z) \geq 2^{-5\lambda/4} \,|\, \rho \leftarrow \mathcal{U}(\lambda)] \leq 2^{-\lambda}$. Hence, with probability at least $1 - 2^{-\lambda}$ over ρ, the statistical distance between $\rho(r)$ and X is at most $\frac{1}{2}\sum_{z\in\{0,1\}^\lambda} \delta_{z,\rho} \leq 2^\lambda \cdot 2^{-5\lambda/4} = 2^{-\lambda/4}$.)
- FACT 2. If $X_1,\ldots,X_n$ and $Y_1,\ldots,Y_n$ are independent random variables such that, for each i, X_i and Y_i are δ-close, then $(X_1,\ldots,X_n)$ and $(Y_1,\ldots,Y_n)$ are $n\delta$-close.

First we examine the statistical distance between W_0 and W_1: every node we replace has the form $\rho(a\|r)$ for $a \in \{0,1\}$; since $\rho(r)$ and $\rho(a\|r)$ are identically distributed, Fact 1 holds for $\rho(a\|r)$ also; thus, using Fact 2 and the fact that we place at most n leaves, we deduce that W_0 and W_1 are $n2^{-\lambda/4}$-close for almost all ρ. Next we examine the statistical distance between W_i and W_{i+1} for $i \in \{1,\ldots,k-1\}$: whenever we replace a node with a random string, it is of the form $\rho(r)$ where $r \in \{0,1\}^{2\lambda}$ is random, since we only replace nodes

whose children we have already replaced; so, using Fact 1, W_i and W_{i+1} are $n2^{-\lambda/4}$-close for almost all ρ. Thus each step in the sequence marks a non-leaf node and ℓ is bounded by the number of non-leaf nodes in $T'_{\mathbf{v}}$; so $\ell \leq n$. By the triangle inequality, the statistical distance between W_0 and W_ℓ is at most $n2^{-\lambda/4+1}$.

We now construct the simulator S. On input $(n, (i, v_i)_{i \in I})$ and with oracle access to ρ, S works as follows: (i) set $\tilde{\mathbf{v}} := (\tilde{v}_i)_{i=1}^n$ where $\tilde{v}_i := v_i$ for $i \in I$ and $\tilde{v}_i := 0$ for $i \notin I$; (ii) for every $i \in I$, compute $\mathsf{ap}_i \leftarrow \mathsf{MERKLE.GetPath}^\rho(\tilde{\mathbf{v}}, i; \mathbf{r})$, except whenever we require the value of a node which is marked in W_ℓ, use a random string in $\{0,1\}^\lambda$; (iii) compute $\mathsf{rt} \leftarrow \mathsf{MERKLE.GetRoot}^\rho(\tilde{\mathbf{v}}; \mathbf{r})$ in the same way; (iv) output $(\mathsf{rt}, (\mathsf{ap}_i)_{i \in I})$. Note that S runs in probabilistic polynomial time since the number of unmarked nodes is at most $|I| \log n$, and the only marked nodes we consider are those that are siblings of unmarked nodes.

Finally, in one case, all the information given to D is contained in W_0 and $(v_i, r_i)_{i \in I}$; in the other case, all the information given to D is contained in W_ℓ and $(v_i, r_i)_{i \in I}$. Moreover, W_ℓ was constructed from W_0 in a way that is independent of $(v_i, r_i)_{i \in I}$. Thus, since these two sets of random variables are $n2^{\lambda/4+1}$-close, the corresponding outputs of D must also be $n2^{\lambda/4+1}$-close.

Choosing ρ uniformly at random, the result follows by the union bound. □

4 Interactive oracle proofs

We first define *interactive oracle protocols*, and then use these to define *interactive oracle proof systems*. Afterwards, we make some remarks regarding basic complexity-theoretic properties of such proof systems.

4.1 Interactive oracle protocols

A k-round interactive oracle protocol between two parties, call them Alice and Bob, comprises k rounds of interaction. In the i-th round of interaction: Alice sends a message m_i to Bob, which he reads in full; then Bob replies with a message f_i to Alice, which she can query (via random access) in this and all later rounds. After the k rounds of interaction, Alice either accepts or rejects.

More precisely, let k be in $\mathbb{N}$ and A, B be two interactive probabilistic algorithms. A k-round interactive oracle protocol between A and B, denoted $\langle B, A\rangle$, works as follows. Let r_A, r_B denote the randomness for A, B; set $f_0 := \perp$ and $\mathsf{state}_0 := \perp$. For $i = 1, \ldots, k$, in the i-th round:

(i) Alice sends a message $m_i \in \{0,1\}^{u_i}$, where $(m_i, \mathsf{state}_i) := A^{f_0,\ldots,f_{i-1}}(\mathsf{state}_{i-1}; r_A)$ and $u_i \in \mathbb{N}$;

(ii) Bob sends a message $f_i \in \{0,1\}^{\ell_i}$, where $f_i := B(m_1, \ldots, m_i; r_B)$ and $\ell_i \in \mathbb{N}$.

The output of the protocol is $m_{\mathsf{fin}} := A^{f_0,\ldots,f_k}(\mathsf{state}_k; r_A)$, and belongs to $\{0,1\}$.

The accepting probability of $\langle B, A\rangle$ is the probability that $m_{\mathsf{fin}} = 1$ for a random choice of r_A, r_B; this probability is denoted $\Pr[\langle B, A\rangle = 1]$ (leaving r_A, r_B implicit). The query complexity of $\langle B, A\rangle$ is the number of queries asked by A to any of the oracles during the k rounds. The proof complexity of $\langle B, A\rangle$ is the number of bits communicated by Bob to Alice (i.e., $\sum_{i=1}^{k} \ell_i$). The view of A in $\langle B, A\rangle$, denoted $\mathrm{View}_{\langle B,A\rangle}(A)$, is the random variable $(a_1, \ldots, a_q, r_A)$ where a_j denotes the answer to the j-th query.

Public coins. An interactive oracle protocol is *public-coin* if for every $i \in \{1, \ldots, k\}$: (i) all queries to $f_0, \ldots, f_{i-1}$ before the i-th round depend only on $m_1, \ldots, m_{i-1}$ (in particular, the prover knows these queries); (ii) m_i is random in $\{0,1\}^{u_i}$. We assume, without loss of generality, that in a public-coin protocol Alice does not maintain state (i.e., $\mathsf{state}_i = \emptyset$) and postpones any query to after the k-th round (i.e., all queries are asked when running $A^{f_0,\ldots,f_k}(\mathsf{state}_k; r_A)$). We can thus take the randomness r_A to be of the form $(m_1, \ldots, m_k, r)$, where r is additional randomness that A may use of to compute m_{fin} after the last round.

4.2 Interactive oracle proof systems

An *interactive oracle proof system* for a relation $\mathscr{R}$ with round complexity $k\colon \{0,1\}^* \to \mathbb{N}$ and soundness $s\colon \{0,1\}^* \to [0,1]$ is a tuple (P, V), where P, V are probabilistic algorithms, that satisfies the following properties.

1. COMPLETENESS. For every $(\mathtt{x}, \mathtt{w}) \in \mathscr{R}$, $\langle P(\mathtt{x}, \mathtt{w}), V(\mathtt{x})\rangle$ is a $k(\mathtt{x})$-round interactive oracle protocol with accepting probability 1.

2. SOUNDNESS. For every $\mathtt{x} \notin \mathscr{L}(\mathscr{R})$ and $\tilde{P}$, $\langle \tilde{P}, V(\mathtt{x})\rangle$ is a $k(\mathtt{x})$-round interactive oracle protocol with accepting probability at most $s(\mathtt{x})$.

Message lengths. We assume the existence of polynomial-time functions that determine the message lengths. Namely, for any instance $\mathtt{x}$ and malicious prover $\tilde{P}$, when considering the interactive oracle protocol $\langle \tilde{P}, V(\mathtt{x})\rangle$, the i-th messages m_i (from $V(\mathtt{x})$) and f_i (to $V(\mathtt{x})$) lie in $\{0,1\}^{u_i(\mathtt{x})}$ and $\{0,1\}^{\ell_i(\mathtt{x})}$ respectively.

Complexity measures. Beyond round complexity and soundness, we consider other complexity measures. Given $p, q\colon \{0,1\}^* \to \mathbb{N}$, we say that (P, V) has proof length p and query complexity q if the proof length and query complexity of $\langle \tilde{P}, V(\mathtt{x})\rangle$ are $p(\mathtt{x})$ and $q(\mathtt{x})$ respectively. (Note that $q(\mathtt{x}) \leq p(\mathtt{x})$ and

$p(\mathtt{x}) = \sum_{i=1}^{k(\mathtt{x})} \ell_i(\mathtt{x})$.) Given $t_{\mathrm{prv}}, t_{\mathrm{ver}} \colon \{0,1\}^* \to \mathbb{N}$, we say that (P, V) has prover time complexity t_{prv} and verifier time complexity t_{ver} if $P(\mathtt{x}, \mathtt{w})$ runs in time $t_{\mathrm{prv}}(\mathtt{x})$ and $V(\mathtt{x})$ runs in time $t_{\mathrm{ver}}(\mathtt{x})$. In sum, we say that (P, V) has complexity $(k, s, p, q, t_{\mathrm{prv}}, t_{\mathrm{ver}})$ if (P, V) has round complexity k, soundness s, proof length p, query complexity q, prover time complexity t_{prv}, and verifier time complexity t_{ver}.

Proof of knowledge. Given $e \colon \{0,1\}^* \to [0,1]$, we say that (P, V) has proof of knowledge e if there exists a probabilistic polynomial-time oracle algorithm E (the *extractor*) such that, for every $\mathtt{x}$ and $\tilde{P}$, $\Pr[(\mathtt{x}, E^{\tilde{P}}(\mathtt{x})) \in \mathscr{R}] \geq \Pr[\langle \tilde{P}, V(\mathtt{x}) \rangle = 1] - e(\mathtt{x})$.[4] The notation $E^{\tilde{P}}(\mathtt{x})$ means that E receives as input $\mathtt{x}$ and may interact with $\tilde{P}$ via rewinding, as we now describe. At any time, E may send a partial prover-verifier transcript to $\tilde{P}$ and then receive $\tilde{P}$'s next message (which is empty for invalid transcripts) in the subsequent computation step; the code, randomness, and any auxiliary inputs of $\tilde{P}$ are not available to E.

Honest-verifier zero knowledge. Given $z \colon \{0,1\}^* \to [0,1]$, we say that (P, V) has z-statistical honest-verifier zero knowledge if there exists a probabilistic polynomial-time algorithm S (the *simulator*) such that, for every $(\mathtt{x}, \mathtt{w}) \in \mathscr{R}$, $S(\mathtt{x})$ is $z(\mathtt{x})$-close to $\mathrm{View}_{\langle P(\mathtt{x}, \mathtt{w}), V(\mathtt{x}) \rangle}(V(\mathtt{x}))$.

Public coins. We say that (P, V) is *public-coin* if the underlying interactive oracle protocol is public-coin.

4.3 Basic complexity-theoretic properties

Several basic complexity-theoretic properties are straightforward to show and provide different points of comparison with other proof system models.

- *Expressivity.* Interactive oracle proofs characterize NEXP (see Appendix A). This is like probabilistically-checkable proofs and multi-prover interactive proofs (which also characterize NEXP), and unlike interactive proofs (which characterize PSPACE).

- *Repetition.* Both sequential and parallel repetition of interactive oracle proofs yield (perfect) exponential soundness error reduction (see Appendix B.1). This is like interactive proofs and probabilistically-checkable proofs (where the same holds), and unlike multi-prover interactive proofs (where parallel repetition decays exponentially but not perfectly [Raz95, Hol07]).

- *Private vs. public coins.* Any interactive oracle proof system can be converted into a public-coin one with at most two additional rounds. This is like interactive proofs (where the same holds [GS86]), and unlike probabilistically-checkable proofs and multi-prover interactive proofs (where correlations among queries are crucial for soundness). This is because the Goldwasser–Sipser set lower-bound approach [GS86] continues to apply in our setting (since the approach only relies on the verifier's possible randomness and corresponding messages, which do not differ from the standard setting of interactive proofs).

[4] Proof of knowledge e implies soundness $s := e$. The definition that we use is equivalent to the one in [BG93, Section 6] except that: (a) we use extractors that run in strict, rather than expected, probabilistic polynomial time; and (b) we extend the condition to hold for all $\mathtt{x}$, rather than for only those in $\mathscr{L}(\mathscr{R})$, so that proof of knowledge implies soundness.

5 State restoration attacks on interactive oracle proofs

We introduce state restoration attacks on interactive oracle proofs and discuss our results for these. First, in Section 5.1, we define the notion of state restoration attacks for interactive oracle proof systems; these induce corresponding notions of soundness (and proof of knowledge). Afterwards, in Section 5.2, we provide upper and lower bounds on state restoration soundness as a function of (standard) soundness and round complexity; in Section 5.3, we prove the tightness of these bounds. In Section 5.4, we define restricted state restoration attacks, a sub-class of state restoration attacks that enables us to state our results of Section 7 in a tighter way.

5.1 State restoration attacks, and soundness and proof of knowledge against these

In an interactive oracle proof, a malicious prover $\tilde{P}$ works as follows: for each round i, $\tilde{P}$ receives the i-th verifier message m_i and then sends to the verifier a message f_i computed as a function of his own randomness and all the verifier messages received so far, i.e., $m_1, \ldots, m_i$.

For the case of public-coin interactive oracle proof systems, we also consider a larger class of malicious provers, called *state-restoring provers*. Informally, a state-restoring prover receives in each round a verifier message as well as a *complete verifier state*, and then sends to the verifier a message and a previously-seen complete verifier state, which sets the verifier to that state; this forms a state restoration attack on the verifier.

More precisely, let (P, V) be a k-round public-coin interactive proof system (see Section 4.2) and $\mathtt{x}$ an instance. A complete verifier state cvs of $V(\mathtt{x})$ takes one of three forms: (1) the symbol null, which denotes the "empty" complete verifier state; (2) a tuple of the form $(m_1, f_1, \ldots, m_i)$, with $i \in \{1, \ldots, k(\mathtt{x})\}$, where each m_j is in $\{0,1\}^{u_j(\mathtt{x})}$ and each f_j is in $\{0,1\}^{\ell_j(\mathtt{x})}$; (3) a tuple of the form $(m_1, f_1, \ldots, m_{k(\mathtt{x})}, f_{k(\mathtt{x})}, r)$ where each m_j and f_j is as in the previous case and r is the additional randomness of the verifier $V(\mathtt{x})$.

The interaction between a state-restoring prover $\tilde{P}$ and the verifier $V(\mathtt{x})$ is mediated through a game:

1. The game initializes the list $\mathsf{SeenStates}$ to be (null).
2. Repeat the following until the game halts and outputs:
   (a) The prover chooses a complete verifier state cvs in the list $\mathsf{SeenStates}$.
   (b) The game sets the verifier to cvs.
   (c) If $\mathsf{cvs} = \mathsf{null}$: the verifier samples a message m_1 in $\{0,1\}^{u_1(\mathtt{x})}$ and sends it to the prover; the game appends $\mathsf{cvs}' := (m_1)$ to the list $\mathsf{SeenStates}$.
   (d) If $\mathsf{cvs} = (m_1, f_1, \ldots, m_{i-1})$ with $i \in \{2, \ldots, k(\mathtt{x})\}$: the prover outputs a message f_{i-1} in $\{0,1\}^{\ell_{i-1}(\mathtt{x})}$; the verifier samples a message m_i in $\{0,1\}^{u_i(\mathtt{x})}$ and sends it to the prover; the game appends $\mathsf{cvs}' := \mathsf{cvs} \| f_{i-1} \| m_i$ to the list $\mathsf{SeenStates}$.
   (e) If $\mathsf{cvs} = (m_1, f_1, \ldots, m_{k(\mathtt{x})})$: the prover outputs a message $f_{k(\mathtt{x})}$ in $\{0,1\}^{\ell_{k(\mathtt{x})}(\mathtt{x})}$; the verifier samples additional randomness r; the game appends $\mathsf{cvs}' := \mathsf{cvs} \| f_{k(\mathtt{x})} \| r$ to the list $\mathsf{SeenStates}$.
   (f) If $\mathsf{cvs} = (m_1, f_1, \ldots, m_{k(\mathtt{x})}, f_{k(\mathtt{x})}, r)$: the verifier computes his decision $b := V^{f_0, \ldots, f_{k(\mathtt{x})}}(\mathtt{x}, \mathsf{state}_{k(\mathtt{x})}; r_V)$ where $\mathsf{state}_{k(\mathtt{x})} := \emptyset$ and $r_V := (m_1, \ldots, m_k, r)$; then the game halts and outputs b.

Note that there are two distinct notions of a round. *Verifier rounds* are the rounds played by the verifier within a single execution, as tracked by a complete verifier state cvs; the number of such rounds lies in the set $\{0, \ldots, k(\mathtt{x}) + 1\}$ (the extra $(k(\mathtt{x}) + 1)$-th round represents the verifier V sampling r after receiving the last prover message). *Prover rounds* are all verifier rounds played by the prover across different verifier executions; the number of such rounds is the number of states in $\mathsf{SeenStates}$ above. Accordingly, for $b \in \mathbb{N}$, we say a prover is *b-round* if it plays at most b prover rounds during any interaction with any verifier.

Also note that the prover is not able to set the verifier to arbitrary states but only to previously-seen ones (starting with the empty state null); naturally, setting the verifier multiple times to the same state may yield distinct new states, because the verifier samples his message afresh each time. After being set to a state cvs,

the verifier does one of three things: (i) if the number of verifier rounds in cvs is less than $k(\mathtt{x})$ (see Step 2c and Step 2d), the verifier samples a fresh next message; (ii) if the number of verifier rounds in cvs is $k(\mathtt{x})$ (see Step 2e), the verifier samples his additional randomness r; (iii) if cvs contains a full protocol execution (see Step 2f), the verifier outputs the decision corresponding to this execution. The second case means that the prover can set the verifier even *after* the conclusion of the execution (after r is sampled and known to the prover). The game halts only in the third case.

The above game between a state-restoring prover and a verifier yields corresponding notions of soundness and proof of knowledge. Below, we denote by $\Pr[\langle \tilde{P}, V(\mathtt{x})\rangle_{\mathrm{sr}} = 1]$ the probability that the state-restoring prover $\tilde{P}$ makes the verifier V accept $\mathtt{x}$ in this game.

Definition 5.1. *Given $s_{\mathrm{sr}}, e_{\mathrm{sr}} \colon \{0,1\}^* \to [0,1]$, a public-coin interactive oracle proof system (P,V) has*

- STATE RESTORATION SOUNDNESS s_{sr} *if, for every $\mathtt{x} \notin \mathscr{L}(\mathscr{R})$ and b-round state-restoring prover $\tilde{P}$, $\Pr[\langle \tilde{P}, V(\mathtt{x})\rangle_{\mathrm{sr}} = 1] \leq s_{\mathrm{sr}}(\mathtt{x}, b)$.*
- STATE RESTORATION PROOF OF KNOWLEDGE e_{sr} *if there exists a probabilistic polynomial-time algorithm E_{sr} (the* extractor*) such that, for every $\mathtt{x}$ and b-round state-restoring prover $\tilde{P}$, $\Pr[(\mathtt{x}, E_{\mathrm{sr}}^{\tilde{P}}(\mathtt{x})) \in \mathscr{R}] \geq \Pr[\langle \tilde{P}, V(\mathtt{x})\rangle_{\mathrm{sr}} = 1] - e_{\mathrm{sr}}(\mathtt{x}, b)$.*

Remark 5.2 (results about proof of knowledge)**.** The discussions in Section 5.2, Section 5.3, and Section 5.4 below mention only state restoration soundness s_{sr}. All of these discussions (as well as theorems) also apply, without change, to state restoration proof of knowledge. (Just substitute e_{sr} for s_{sr}, and e for s.)

5.2 General bounds

Let (P,V) be a k-round public-coin interactive proof system with soundness s and state restoration soundness s_{sr}. We give upper and lower bounds on s_{sr} as a function of k and s. First, for every instance $\mathtt{x} \in \{0,1\}^*$ and small enough budget $b \in \mathbb{N}$ there are two "degenerate" cases: if $b \leq k(\mathtt{x})$ then $s_{\mathrm{sr}}(\mathtt{x}, b) = 0$; and if $b = k(\mathtt{x}) + 1$ then $s_{\mathrm{sr}}(\mathtt{x}, b) = s(\mathtt{x})$. For larger values of b, we prove the following bounds:

Lemma 5.3. *Let (P,V) be a k-round public-coin interactive proof system with soundness s and state restoration soundness s_{sr}. For every instance $\mathtt{x} \in \{0,1\}^*$ and budget $b \in \mathbb{N}$ with $b \geq k(\mathtt{x}) + 1$,*

$$\left\lfloor \frac{b}{k(\mathtt{x})+1} \right\rfloor s(\mathtt{x})(1 - o(1)) \leq s_{\mathrm{sr}}(\mathtt{x}, b) \leq \binom{b}{k(\mathtt{x})+1} s(\mathtt{x}) \ ,$$

where the lower bound holds if $\lfloor \frac{b}{k(\mathtt{x})+1} \rfloor s(\mathtt{x}) = o(1)$.

Proof. We first prove the lower bound by exhibiting a universal "resetting" attack. Let $\tilde{P}$ be a prover that attains $s(\mathtt{x})$, i.e., makes $V(\mathtt{x})$ accept with probability $s(\mathtt{x})$. Consider the state-restoring prover $\tilde{P}_{\mathrm{sr}}$ that runs $\tilde{P}$ until it halts, then sets $V(\mathtt{x})$ to the empty state null, and repeats this until either $V(\mathtt{x})$ accepts or b prover rounds have elapsed. Note that $\tilde{P}_{\mathrm{sr}}$ succeeds with probability at least $\lfloor \frac{b}{k(\mathtt{x})+1} \rfloor s(\mathtt{x})(1 - o(1))$ provided $\lfloor \frac{b}{k(\mathtt{x})+1} \rfloor s(\mathtt{x}) = o(1)$. This establishes the lower bound.

We now prove the upper bound. Let $\tilde{P}_{\mathrm{sr}}$ be a b-round state-restoring prover that makes V accept $\mathtt{x}$ with probability $s_{\mathrm{sr}}(\mathtt{x}, b)$. We construct a prover $\tilde{P}$ that, using $\tilde{P}_{\mathrm{sr}}$ as a subroutine, makes V accept $\mathtt{x}$ with probability $\binom{b}{k(\mathtt{x})+1}^{-1} s_{\mathrm{sr}}(\mathtt{x}, b)$, without any state restoration. The prover $\tilde{P}$ works as follows. First, $\tilde{P}$ chooses a uniformly random subset $S \subseteq \{1, \ldots, b\}$ of cardinality $k(\mathtt{x}) + 1$; we denote by $S[i]$ the i-th

smallest value in S. Then $\tilde{P}$ runs $\tilde{P}_{\mathrm{sr}}$, simulating an interaction between $\tilde{P}_{\mathrm{sr}}$ and a virtual verifier using internal randomness, except that, in the simulated prover round $S[i]$ with $i \in \{1, \ldots, k(\mathtt{x})\}$, $\tilde{P}$ gives to $\tilde{P}_{\mathrm{sr}}$ the message m_i from (the real verifier) V, and sends $\tilde{P}_{\mathrm{sr}}$'s response message f_i back to V. Finally, in the simulated prover round $S[k(\mathtt{x}) + 1]$, we set the virtual verifier's randomness to that of V. (More precisely, since this randomness is not visible to $\tilde{P}$, we *couple* these two random variables for the purpose of this analysis; this coupling does not affect the acceptance probability of the real verifier V.)

The view of $\tilde{P}_{\mathrm{sr}}$ is as in a real state restoration attack, so the virtual verifier accepts with probability $s_{\mathrm{sr}}(\mathtt{x}, b)$. If the virtual verifier accepts, there is $S' \in \binom{[b]}{k(\mathtt{x})+1}$ such that $V^{f'_1, \ldots, f'_{k(\mathtt{x})}}(\mathtt{x}, \mathsf{state}_{k(\mathtt{x})}; r_V) = 1$ where $\mathsf{state}_{k(\mathtt{x})} := \emptyset$, $r_V := (m'_1, \ldots, m'_{k(\mathtt{x})}, r')$, the m'_i and f'_i are messages received and sent by $\tilde{P}_{\mathrm{sr}}$ in prover round $S'[i]$ for $i \in \{1, \ldots, k(\mathtt{x})\}$, and r' is the additional randomness of the virtual verifier for prover round $S'[k(\mathtt{x}) + 1]$. So if $S = S'$ then (the real verifier) V accepts $\mathtt{x}$; the equality occurs with probability $\binom{b}{k(\mathtt{x})+1}^{-1}$. So $\tilde{P}$ makes V accept $\mathtt{x}$ with probability $\binom{b}{k(\mathtt{x})+1}^{-1} s_{\mathrm{sr}}(\mathtt{x}, b)$; rearranging yields the claim. □

5.3 Tightness of general bounds

We prove that the bounds of Section 5.2 are essentially tight. First, we show that the lower bound of Lemma 5.3 is tight up to a factor $k(\mathtt{x})$.

Theorem 5.4. *For every $k \in \mathbb{N}$ there exists a k-round protocol (P, V) with soundness $s(\mathtt{x})$ and state restoration soundness $s_{\mathrm{sr}}(\mathtt{x}, b)$ such that $s_{\mathrm{sr}}(\mathtt{x}, b) \leq bs(\mathtt{x})$.*

Proof. Take any zero-round protocol (P', V') with soundness $s(\mathtt{x})$, and add k dummy rounds to obtain a k-round protocol (P, V). This protocol also has soundness $s(\mathtt{x})$. Moreover, any b-round state-restoring prover $\tilde{P}$ for (P, V) can be converted into a b-round state-restoring prover $\tilde{P}'$ for (P', V') by restoring (null) whenever $\tilde{P}$ restores $(\ldots, m_k)$, and simulating dummy rounds for all other messages. One can verify that $\tilde{P}'$ convinces V' with the same probability as $\tilde{P}$ convinces V. By Lemma 5.3, (P', V') has state restoration soundness at most $bs(\mathtt{x})$, and hence so does (P, V). □

Next, we show that the upper bound of Lemma 5.3 is tight up to a polynomial factor in $|\mathtt{x}|$, independent of b, when $k(\mathtt{x}) = O(\log |\mathtt{x}|)$.

Theorem 5.5. *For every $\mathscr{R} \in$ NEXP and $k \in \mathbb{N}$ there is a k-round public-coin interactive oracle proof system (P, V) for $\mathscr{R}$ such that, for every sufficiently large $b \in \mathbb{N}$, (P, V) has state restoration soundness $s_{\mathrm{sr}}(\mathtt{x}, b) \geq \frac{1}{4} \cdot 4^{-(k(\mathtt{x})+1)} \binom{b}{k(\mathtt{x})+1} s(\mathtt{x})$, where $s(\mathtt{x})$ is the soundness of (P, V).*

We first prove the following lemma.

Lemma 5.6. *For every $k \in \mathbb{N}$ there exists $u \in \mathbb{N}$ for which there is a k-round public-coin interactive oracle proof system (P, V) such that, for every sufficiently large $b \in \mathbb{N}$, (P, V) has state restoration soundness $s_{\mathrm{sr}}(\mathtt{x}, b) \geq \frac{1}{2} \geq \frac{1}{2} \cdot 4^{-(k(\mathtt{x})+1)} \binom{b}{k(\mathtt{x})+1} s(\mathtt{x})$, where $s(\mathtt{x})$ is the soundness of (P, V). Moreover, each of V's messages has length $u \leq \left\lfloor \log \left(\frac{b}{k+1} \right) \right\rfloor$.*

Proof. Let $u \in \mathbb{N}$ be a parameter to be fixed later.

Consider the k-round public-coin interactive oracle proof system (P, V) defined as follows. Each message m_i of V is sampled uniformly at random from $\{0, 1\}^u$; after k interactions with the prover, V samples r uniformly at random from $\{0, 1\}^u$ and accepts if and only if $m_1 = \cdots = m_k = r = 1^u$. Note that no malicious prover can make V accept with probability more than $2^{-(k+1)u}$ without state restoration.

Allowing state restoration, we construct a state-restoring prover $\tilde{P}$ as follows.

(i) Let $\mathsf{cvs}_0 := \mathsf{null}$.
(ii) For $i = 1, \ldots, k$, restore cvs_{i-1} repeatedly (and sending the verifier empty messages) until $m_i = 1^u$. Let the state received when this happens be cvs_i.
(iii) Restore cvs_k (with an empty proof) repeatedly until V accepts.

We make $\tilde{P}$ be a b-round state-restoring prover by having it abort after playing b rounds.

The prover $\tilde{P}$ makes V accept if and only if drawing b strings uniformly at random from $\{0,1\}^u$ yields at least $k+1$ copies of 1^u. The expected number of strings we need to draw to obtain $k+1$ copies is $(k+1)2^u$, by linearity of expectation. Thus by Markov's inequality, the probability that more than $2(k+1)2^u$ strings need to be drawn is at most $1/2$, that is, with probability at least $1/2$, $k+1$ copies of 1^u are obtained within $2(k+1)2^u$ draws.

We choose u to be the smallest integer such that $2^{-(k+1)u} \leq 2^{2(k+1)}\binom{b}{k+1}^{-1}$. This condition holds for all $u \geq \log\left(\frac{b}{k+1}\right) - 2$ (using the inequality $\binom{x}{y} \geq (\frac{x}{y})^y$ for $x \geq y \geq 1$). By minimality of u, $u \leq \log\left(\frac{b}{k+1}\right) - 1$, and thus $2(k+1)2^u \leq b$. With this choice of u, (P, V) has soundness $s(\mathtt{x}) = 2^{-(k+1)u} \leq 4^{k+1}\binom{b}{k+1}^{-1}$, but state restoration soundness $s_{\mathrm{sr}}(\mathtt{x}, b) \geq \frac{1}{2} \geq \frac{1}{2} \cdot 4^{-(k+1)}\binom{b}{k+1} s(\mathtt{x})$. □

We now return to the proof of the theorem.

Proof of Theorem 5.5. The relation $\mathscr{R}$ has a public-coin k-round interactive oracle proof system $(\hat{P}, \hat{V})$, because $\mathscr{R}$ is in NEXP (see the lemma in Appendix A). Moreover, we may assume that $(\hat{P}, \hat{V})$ has soundness $4^{k+1}\binom{b}{k+1}^{-1} = 2^{-O(k \log b)}$ (see parallel repetition in Appendix B.1).

Construct a k-round interactive oracle proof system (P, V). Let (P', V') be the k-round IOP implied by Lemma 5.6, and $u \in \mathbb{N}$ the corresponding message length. The verifier V works as follows.

(i) Let $\hat{m}_1 \| \ldots \| \hat{m}_k \| \hat{r}$ denote the randomness for $\hat{V}$.
(ii) Simulate $\hat{V}$ until round k (i.e., until before $\hat{V}$ makes its first query), forwarding $\hat{V}$'s messages to the prover with the following adjustment. For each i, the i-th message m_i equals $\hat{m}_i$ padded with $\min\{0, u - |\hat{m}_i|\}$ random bits. The randomness r equals $\hat{r}$ padded with $\min\{0, u - |\hat{r}|\}$ random bits.
(iii) If $V'^{f_1,\ldots,f_k}(\mathtt{x}, m'_1, \ldots, m'_k; r') = 1$, where m'_i and r' are the first u bits of m_i for $i = 1, \ldots, k$ and r respectively, then halt and accept.
(iv) Continue to simulate $\hat{V}$ until it halts, and output its decision.

The prover P equals $\hat{P}$ up to the fact that P ignores any padding in the messages received from V.

Clearly (P, V) has completeness 1. Also, by the union bound, we see that for all $\mathtt{x} \notin \mathscr{L}(\mathscr{R})$ and $\tilde{P}$ without state restoration, $\Pr[\langle \tilde{P}, V(\mathtt{x}) \rangle = 1] \leq 2 \cdot 4^{k+1}\binom{b}{k+1}^{-1}$. (Any additional random bits generated by V do not help a malicious prover make $\hat{V}$ accept because they do not affect $\hat{V}$'s decision.)

The malicious prover $\tilde{P}$ implied by Lemma 5.6 achieves the same soundness (using state restoration) against V as it does against V'. The theorem follows analogously. □

5.4 Restricted state restoration attacks

We briefly discuss a technical restriction of state restoration attacks that we use for stating bounds in Section 7; these bounds can also be stated relative to (unrestricted) state restoration attacks, but would not be as tight.

A *restricted* state restoration attack is one where the prover cannot restore the verifier to the empty state $\mathsf{cvs} = \mathsf{null}$ (except for in the first iteration of the game in Section 5.1), but can still restore the verifier to any other previously-seen verifier state. In other words, a restricted state restoration attack on a k-round protocol is equivalent to playing the first round as normal, and then performing an (unrestricted) state restoration attack on the remaining $k-1$ rounds. The notion of restricted state restoration yields corresponding notions

of *restricted state restoration soundness*, which we denote $\bar{s}_{\mathrm{sr}}$, and *restricted state restoration proof of knowledge*, which we denote $\bar{e}_{\mathrm{sr}}$.

The above equivalence implies the following lemma, analogous to Lemma 5.3.

Lemma 5.7. *Let (P, V) be a k-round public-coin interactive oracle proof system with soundness s and restricted state restoration soundness $\bar{s}_{\mathrm{sr}}$. For every instance $\mathtt{x} \in \{0,1\}^*$ and budget $b \in \mathbb{N}$ with $b \geq k(\mathtt{x})$,*

$$s(\mathtt{x}) \leq \bar{s}_{\mathrm{sr}}(\mathtt{x}, b) \leq \binom{b}{k(\mathtt{x})} s(\mathtt{x}) \ .$$

Note that the lower bound of Lemma 5.3 no longer holds, because there are interactive oracle proof systems for which restricted state restoration soundness is equal to standard soundness. (E.g., consider a k-round proof system where every round but the first does not affect the verifier's acceptance probability.) This shows that the lower bound of Lemma 5.7 is tight; also, the upper bound of Lemma 5.7 is tight in a similar way to the upper bound of Lemma 5.3, by the following theorem (analogous to Theorem 5.5).

Theorem 5.8. *For every $\mathscr{R} \in$ NEXP and $k \in \mathbb{N}$ there is a k-round public-coin interactive oracle proof system (P, V) for $\mathscr{R}$ such that, for every sufficiently large $b \in \mathbb{N}$, (P, V) has restricted state restoration soundness $\bar{s}_{\mathrm{sr}}(\mathtt{x}, b) \geq \frac{1}{4} \cdot 4^{-k(\mathtt{x})} \binom{b-1}{k(\mathtt{x})} s(\mathtt{x})$, where $s(\mathtt{x})$ is the soundness of (P, V).*

Finally, using a similar technique to the proof of the upper bound in Lemma 5.3, we also obtain the following bounds for $\bar{s}_{\mathrm{sr}}$ in terms of s_{sr}:

Lemma 5.9. *For every instance $\mathtt{x} \in \{0,1\}^*$ and budget $b \in \mathbb{N}$ it holds that*

$$\frac{1}{b} s_{\mathrm{sr}}(\mathtt{x}, b) \leq \bar{s}_{\mathrm{sr}}(\mathtt{x}, b) \leq s_{\mathrm{sr}}(\mathtt{x}, b) \ .$$

6 From interactive oracle proofs to non-interactive random-oracle proofs

We describe a transformation T such that if (P, V) is a public-coin interactive oracle proof system for a relation $\mathscr{R}$ then $(\mathbb{P}, \mathbb{V}) := T(P, V)$ is a non-interactive random-oracle proof system for $\mathscr{R}$. The transformation T runs in polynomial time: given as input code for P and V, it runs in time polynomial in the size of this code and then outputs code for $\mathbb{P}$ and $\mathbb{V}$. Concretely, we proceed as follows:

- Below, we describe the polynomial-time transformation T by giving the construction of $\mathbb{P}$ and $\mathbb{V}$.
- In Section 7, we state and prove T's main properties, relating features of (P, V) to those of $(\mathbb{P}, \mathbb{V})$.

Notation. For convenience, we split the random oracle ρ into two random oracles, denoted ρ_1 and ρ_2, as follows: $\rho_1(x) := \rho(0\|x)$ and $\rho_2(x) := \rho(1\|x)$. At a high level, we use ρ_1 for the verifier's randomness, and ρ_2 for Merkle trees and other hashing purposes. When counting queries, we count queries to both ρ_1 and ρ_2.

Construction of $\mathbb{P}$. The algorithm $\mathbb{P}$, given input $(\mathtt{x}, \mathtt{w})$ and oracle access to ρ, works as follows.

1. Set $k := k(\mathtt{x})$, $q := q(\mathtt{x})$, $f_0 := \perp$, and $\sigma_0 := \rho_2(\mathtt{x})$.
2. Start running $P(\mathtt{x}, \mathtt{w})$ and, for $i = 1, \ldots, k$:
   (a) Compute the verifier message $m_i := \rho_1(\mathtt{x}\|\sigma_{i-1})$ (and Remark 2.1 may apply here).
   (b) Give m_i to $P(\mathtt{x}, \mathtt{w})$ to obtain f_i.
   (c) Compute the Merkle-tree root $\mathsf{rt}_i := \mathsf{MERKLE.GetRoot}^{\rho_2}(f_i)$.
   (d) Compute the "root hash" $\sigma_i := \rho_2(\mathsf{rt}_i\|\sigma_{i-1})$.
3. Set $\mathsf{state}_k := \emptyset$ and $r_V := (m_1, \ldots, m_k, r)$, where $r := \rho_1(\mathtt{x}\|\sigma_k)$ (and Remark 2.1 may apply here).
4. Run $V^{f_0,\ldots,f_k}(\mathtt{x}, \mathsf{state}_k; r_V)$ and compute an authentication path for each query. Namely, for $j = 1, \ldots, q$: if the j-th query is to the x_j-th bit of the y_j-th oracle, then compute $\mathsf{ap}_j := \mathsf{MERKLE.GetPath}^{\rho_2}(f_{y_j}, x_j)$. (If $\mathsf{MERKLE.GetRoot}$ is probabilistic, then give the same randomness to $\mathsf{MERKLE.GetPath}$ as well.)
5. Set $\pi := \big((\mathsf{rt}_1, \ldots, \mathsf{rt}_k), (\mathsf{ap}_1, \ldots, \mathsf{ap}_q), \sigma_k\big)$. That is, π comprises the Merkle-tree roots, an authentication path for each query, and the final root hash.
6. Output π.

See Figure 6 for a diagram of the construction above.

Construction of $\mathbb{V}$. The algorithm $\mathbb{V}$, given input $(\mathtt{x}, \tilde{\pi})$ and oracle access to ρ, works as follows.

1. Set $k := k(\mathtt{x})$, $q := q(\mathtt{x})$, $f_0 := \perp$, and $\sigma_0 := \rho_2(\mathtt{x})$.
2. Parse $\tilde{\pi}$ as a tuple $\big((\tilde{\mathsf{rt}}_1, \ldots, \tilde{\mathsf{rt}}_k), (\tilde{\mathsf{ap}}_1, \ldots, \tilde{\mathsf{ap}}_q), \tilde{\sigma}_k\big)$.
3. For $i = 1, \ldots, k$:
   (a) Compute $m_i := \rho_1(\mathtt{x}\|\sigma_{i-1})$.
   (b) Compute $\sigma_i := \rho_2(\tilde{\mathsf{rt}}_i\|\sigma_{i-1})$.
4. Set $\mathsf{state}_k := \emptyset$ and $r_V := (m_1, \ldots, m_k, r)$, where $r := \rho_1(\mathtt{x}\|\sigma_k)$.
5. Compute $m_{\mathsf{fin}} := V^{f_0,\ldots,f_k}(\mathtt{x}, \mathsf{state}_k; r_V)$, answering the j-th query with the answer a_j in the path $\tilde{\mathsf{ap}}_j$.
6. If $\sigma_k \neq \tilde{\sigma}_k$, halt and output 0.
7. For $j = 1, \ldots, q$: if the j-th query is to the x_j-th bit of the y_j-th oracle and $\mathsf{MERKLE.CheckPath}^{\rho_2}(\mathsf{rt}_{y_j}, x_j, a_j, \tilde{\mathsf{ap}}_j) \neq 1$, halt and output 0.
8. Output m_{fin}.

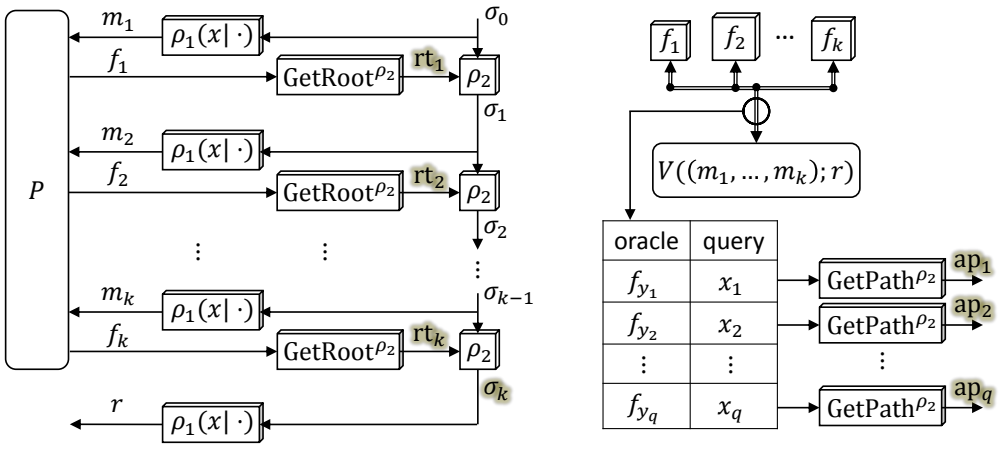

Figure 6: Diagram of how the NIROP prover $\mathbb{P}$ works. On the left, the diagram shows the part of $\mathbb{P}$'s computation that involves running the IOP prover P; on the right, the diagram shows the part of $\mathbb{P}$'s computation that involves running the IOP verifier V. The highlighted values are those that are eventually included in the output proof π.

7 Analysis of the transformation T

The theorem below specifies guarantees of the transformation T, described in Section 6. The statement of the theorem relies on the notions of soundness and proof of knowledge against state restoration attacks, described in Section 5 (more precisely, the statement relies on their restricted variants, described in Section 5.4).

Theorem 7.1 (IOP $\to$ NIROP)**.** *For every relation $\mathscr{R}$, if (P, V) is a public-coin interactive oracle proof system for $\mathscr{R}$ with*

$$\begin{aligned} \text{round complexity} &\quad k(\mathtt{x}) \\ \text{restricted state restoration soundness} &\quad \bar{s}_{\mathrm{sr}}(\mathtt{x}, b) \\ \text{proof length} &\quad p(\mathtt{x}) \\ \text{prover time} &\quad t_{\mathrm{ver}}(\mathtt{x}) \\ \text{verifier time} &\quad t_{\mathrm{prv}}(\mathtt{x}) \end{aligned}$$

then $(\mathbb{P}, \mathbb{V}) := T(P, V)$ is a non-interactive random-oracle proof system for $\mathscr{R}$ with

$$\begin{aligned} \text{soundness} &\quad s'(\mathtt{x}, m, \lambda) := \bar{s}_{\mathrm{sr}}(\mathtt{x}, m) + 3(m^2+1)2^{-\lambda} \\ \text{proof length} &\quad p'(\mathtt{x}, \lambda) := \big(k(\mathtt{x}) + q(\mathtt{x}) \cdot (\lceil \log_2 p(\mathtt{x}) \rceil + 2) + 1\big) \cdot \lambda \quad [5] \\ \text{prover time} &\quad t'_{\mathrm{prv}}(\mathtt{x}, \lambda) := O_\lambda(k(\mathtt{x}) + p(\mathtt{x})) + t_{\mathrm{prv}}(\mathtt{x}) + t_{\mathrm{ver}}(\mathtt{x}) \\ \text{verifier time} &\quad t'_{\mathrm{ver}}(\mathtt{x}, \lambda) := O_\lambda(k(\mathtt{x}) + q(\mathtt{x})) + t_{\mathrm{ver}}(\mathtt{x}) \end{aligned}$$

Moreover, the transformation T preserves both proof of knowledge and zero knowledge (if present):

- *If (P, V) has restricted state restoration proof of knowledge $\bar{e}_{\mathrm{sr}}$ then $(\mathbb{P}, \mathbb{V})$ has proof of knowledge e' with*

$$e'(\mathtt{x}, m, \lambda) := \bar{e}_{\mathrm{sr}}(\mathtt{x}, m) + 3(m^2+1)2^{-\lambda} \ .$$

- *If (P, V) has z-statistical honest-verifier zero knowledge then $(\mathbb{P}, \mathbb{V})$ has z'-statistical zero knowledge with*

$$z'(\mathtt{x}, \lambda) := z(\mathtt{x}) + p(\mathtt{x})2^{-\lambda/4+2} \ .$$

Finally, the above soundness $s'(\mathtt{x}, m, \lambda)$ and proof of knowledge $e'(\mathtt{x}, m, \lambda)$ are tight up to a multiplicative factor $O(\lambda + p(\mathtt{x}) + u(\mathtt{x}))$ in m, where $u(\mathtt{x})$ is the total number of bits sent by $V(\mathtt{x})$ (for any choice of its randomness).

By construction, if $\langle P(\mathtt{x}, \mathtt{w}), V(\mathtt{x}) \rangle$ has accepting probability δ, then the probability that $\mathbb{V}^\rho(\mathtt{x}, \mathbb{P}^\rho(\mathtt{x}, \mathtt{w}))$ accepts is δ. The complexities $p', t'_{\mathrm{prv}}, t'_{\mathrm{ver}}$ above also directly follow from the construction. Therefore, we are left to discuss soundness (upper bound in Section 7.1 and lower bound in Section 7.2), proof of knowledge (Section 7.3), and zero knowledge (Section 7.4).

[5] This bound can be tightened by using additional notation: the term $q(\mathtt{x}) \cdot \lceil \log_2 p(\mathtt{x}) \rceil$ can be replaced by $\sum_{j=1}^{q(\mathtt{x})} \lceil \log_2 |f_{i_j}| \rceil$ where f_{i_j} is the oracle queried by the j-th query.

7.1 Upper bound for soundness

Let $\mathtt{x} \notin \mathscr{L}(\mathscr{R})$ and let $\tilde{\mathbb{P}}$ be an m-query prover for the non-interactive random-oracle proof system $(\mathbb{P}, \mathbb{V})$. We construct a prover $\tilde{P}$ (depending on $\mathtt{x}$ and $\tilde{\mathbb{P}}$) for the interactive oracle proof system (P, V), and show that $\tilde{P}$'s ability to cheat in a (restricted) state restoration attack is closely related to $\tilde{\mathbb{P}}$'s ability to cheat.

Construction of $\tilde{P}$. Given no inputs or oracles, the prover $\tilde{P}$ works as follows.

1. Let ρ_1, ρ_2 be tables mapping $\{0,1\}^*$ to $\{0,1\}^\lambda$, and let α be a table mapping λ-bit strings to verifier states. The tables are initially empty and are later populated with suitable values, during the simulation of $\tilde{\mathbb{P}}$. Intuitively, ρ_1, ρ_2 are used to simulate $\tilde{\mathbb{P}}$'s access to a random oracle, while α is used to keep track of which verifier states $\tilde{\mathbb{P}}$ has "seen in his mind".
2. Draw $\sigma_0 \in \{0,1\}^\lambda$ at random, and define $\rho_2(\mathtt{x}) := \sigma_0$ (i.e., the oracle ρ_2 replies the query $\mathtt{x}$ with the answer σ_0). After receiving V's first message m_1, also define $\rho_1(\mathtt{x} \| \sigma_0) := m_1$ and $\alpha(\sigma_0) := (m_1)$.
3. Begin simulating $\tilde{\mathbb{P}}^\rho$ and, for $i = 1, \ldots, m$:
   (a) Let θ_i be the i-th query made by $\tilde{\mathbb{P}}^\rho$.
   (b) If θ_i is a query to a location of ρ_1 that is defined, respond with $\rho_1(\theta_i)$. Otherwise (if θ_i to an undefined location of ρ_1), draw a string in $\{0,1\}^\lambda$ at random and respond with it. Then go to the next iteration of Step 3.
   (c) If θ_i is a query to a location of ρ_2 that is defined, respond with $\rho_2(\theta_i)$; then go to the next iteration of Step 3. Otherwise (if θ_i is to an undefined location of ρ_2), draw a string $\sigma' \in \{0,1\}^\lambda$ at random and respond with it; then continue as follows.
   (d) Let rt be the first λ bits of θ_i, and σ be the second λ bits. (If the length of θ_i is not 2λ bits, go to the next iteration of Step 3.) If $\alpha(\sigma)$ is defined, let $\mathsf{cvs} := \alpha(\sigma)$ and let j be the number of verifier rounds in the state cvs. If $\alpha(\sigma)$ is not defined, go to the next iteration of Step 3.
   (e) Find the query $\theta_{i^\star}$ whose result is rt. If this query is not unique, or there is no such query, then answer the verifier V with some dummy message (e.g., an all zero message of the correct length) and skip to Step 3g. Otherwise, note the index $i^\star$ and continue.
   (f) Compute $f := \mathsf{VE}^{\rho_2}(\tilde{\mathbb{P}}, \ell_j(\mathtt{x}), i^\star, i)$; if VE aborts, set $f := 0^{\ell_j(\mathtt{x})}$. Recall that $\ell_j(\mathtt{x})$ is the length of the prover message in the j-th verifier round, and VE is Valiant's extractor (see Section 3.1). Also note that VE does not query ρ_2 on any value outside the table, because we have already simulated the first i queries of $\tilde{\mathbb{P}}$ (see Remark 3.1).
   (g) Send the message f to the verifier and tell the game to set the verifier to the state cvs. (Whether cvs lies in the set $\mathsf{SeenStates}$ is a matter of analysis further below.) If the game is not over, the verifier replies with a new message m'. (If $j = k(\mathtt{x}) + 1$, for the purposes of the proof, we interpret m' as the additional randomness r.) The game adds $\mathsf{cvs}' := \mathsf{cvs} \| f \| m'$ to $\mathsf{SeenStates}$. The prover defines $\rho_1(\mathtt{x} \| \sigma') := m'$ and $\alpha(\sigma') := \mathsf{cvs}'$.

Analysis of $\tilde{P}$. We now analyze $\tilde{P}$. We first prove a simple lemma, and then discuss $\tilde{P}$'s ability to cheat.

Lemma 7.2. *Let A be an m-query algorithm. Define:*

1. *E_1 to be the event that A^{ρ_2} outputs $\mathtt{x} \in \{0,1\}^n$, $\mathsf{rt}_1, \ldots, \mathsf{rt}_{k(\mathtt{x})} \in \{0,1\}^\lambda$, and $\sigma_{k(\mathtt{x})} \in \{0,1\}^\lambda$ that satisfy the recurrence $\sigma_0 = \rho_2(\mathtt{x})$ and $\sigma_i = \rho_2(\mathsf{rt}_i \| \sigma_{i-1})$ for all $i \in \{1, \ldots, k(\mathtt{x})\}$;*
2. *E_2 to be the event that A^{ρ_2} queries ρ_2 at $\mathtt{x}, \mathsf{rt}_1 \| \sigma_0, \ldots, \mathsf{rt}_{k(\mathtt{x})} \| \sigma_{k(\mathtt{x})-1}$ (in order) and, if any rt_i is the result of a query, this query first occurs before $\mathsf{rt}_i \| \sigma_{i-1}$.*

Then

$$\Pr\left[(\neg E_1) \vee E_2 \mid \rho_2 \leftarrow \mathcal{U}(\lambda)\right] \geq 1 - (m^2+1)2^{-\lambda} \ .$$

Proof. Let rt_0 be $\mathtt{x}$ and σ_{-1} be the empty string. Suppose, by contradiction, that E_1 occurs and E_2 does not. Then there exists $i \in \{0, \ldots, k(\mathtt{x})\}$ for which at least one of the following holds: (i) A^{ρ_2} does not query

$\mathsf{rt}_i \| \sigma_{i-1}$; (ii) A^{ρ_2} queries $\mathsf{rt}_{i+1} \| \sigma_i$ before it queries $\mathsf{rt}_i \| \sigma_{i-1}$; (iii) rt_i is the result of a query but this query first occurs after $\mathsf{rt}_i \| \sigma_{i-1}$. Consider the largest index i for which one of the above holds.

In case (i), the behavior of A^{ρ_2} is independent of $\rho_2(\mathsf{rt}_i \| \tilde{\sigma}_{i-1})$. If $i = k(\mathbb{x})$, then the output $\sigma_{k(\mathbb{x})}$ of A^{ρ_2} equals $\rho_2(\mathsf{rt}_{k(\mathbb{x})} \| \sigma_{k(\mathbb{x})-1})$ with probability $2^{-\lambda}$. If $i < k(\mathbb{x})$, then there is a sequence of queries $\mathsf{rt}_{i+1} \| \tilde{\sigma}_i, \ldots, \mathsf{rt}_{k(\mathbb{x})} \| \tilde{\sigma}_{k(\mathbb{x})-1}$ for which $\tilde{\sigma}_i = \rho_2(\mathsf{rt}_i \| \tilde{\sigma}_{i-1})$ for $i = 1, \ldots, k(\mathbb{x}) - 1$ and $\rho_2(\mathsf{rt}_{k(\mathbb{x})} \| \tilde{\sigma}_{k(\mathbb{x})-1}) = \sigma_{k(\mathbb{x})}$. If this sequence is not unique, then A^{ρ_2} has found a collision. Otherwise, the unique sequence has $\tilde{\sigma}_i = \sigma_i$ for each i, which occurs with probability at most $2^{-\lambda}$.

In cases (ii) and (iii), A^{ρ_2} has found a collision, since $\sigma_i = \rho_2(\mathsf{rt}_i \| \sigma_{i-1})$. The fraction of oracles ρ_2 for which A^{ρ_2} finds a collision is at most $m^2 2^{-\lambda}$. Overall, the probability that E_2 does not occur and E_1 does is, by the union bound, at most $(m^2 + 1)2^{-\lambda}$. □

We now state and prove the lemma that establishes the soundness s' as stated in Theorem 7.1.

Lemma 7.3. *Define*

$$\epsilon := \Pr\left[\mathbb{V}^{\rho}(\mathbb{x}, \pi) = 1 \;\middle|\; \begin{array}{c} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \tilde{\mathbb{P}}^{\rho} \end{array} \right] .$$

Then there exists $b \in \mathbb{N}$ with $b \leq m$ such that $\tilde{P}$ is a b-round state-restoring prover that makes V accept with probability at least $\epsilon - 3(m^2 + 1)2^{-\lambda}$.

Proof. We first note that $\tilde{P}$ described plays no more than m rounds, because $\tilde{P}$ sends a message to the verifier V only in response to $\tilde{\mathbb{P}}$ making a query. Next, we define some useful notions, and use them to prove three claims which together imply the lemma.

DEFINITION 1. We say $\rho \in \mathcal{U}(\lambda)$ is *good* if

1. The verifier accepts relative to ρ, i.e., $\mathbb{V}^{\rho}(\mathbb{x}, \pi) = 1$ where $\pi \leftarrow \tilde{\mathbb{P}}^{\rho}$.
2. Parsing π as $\big((\tilde{\mathsf{rt}}_1, \ldots, \tilde{\mathsf{rt}}_{k(\mathbb{x})}), (\tilde{\mathsf{ap}}_1, \ldots, \tilde{\mathsf{ap}}_q), \tilde{\sigma}_{k(\mathbb{x})}\big)$ and setting $\sigma_0 := \mathbb{x}$, for each $i \in \{1, \ldots, k(\mathbb{x})\}$, where $\sigma_i := \rho_2(\tilde{\mathsf{rt}}_i \| \sigma_{i-1})$, there exist indices $1 \leq j_1 < \cdots < j_k \leq m$ such that:
   (a) $\tilde{\mathbb{P}}^{\rho}$'s j_i-th query is to ρ_2 at $\tilde{\mathsf{rt}}_i \| \sigma_{i-1}$;
   (b) if rt_i is the result of a query, this query first occurs before j_i;
   (c) if $\tilde{\mathbb{P}}^{\rho}$ queries ρ_1 at $\mathbb{x} \| \sigma_i$, then this query occurs *after* query j_i;
   (d) if there exists l such that $\mathsf{Root}(\tilde{\mathsf{ap}}_l) = \tilde{\mathsf{rt}}_i$, there is a unique (up to duplicate queries) $a_i \in \{0, \ldots, j_i\}$ such that $\rho_2(\theta_{a_i}) = \tilde{\mathsf{rt}}_i$ and, for every $i_{\mathsf{max}} \in \{a_i, \ldots, j_i\}$, $\mathbf{v} := \mathsf{VE}^{\rho_2}(A, \ell_i, a_i, i_{\mathsf{max}})$ is such that, for all l with $\mathsf{Root}(\tilde{\mathsf{ap}}_l) = \tilde{\mathsf{rt}}_i$, $\mathsf{Value}(\tilde{\mathsf{ap}}_l)$ equals the $\mathsf{Position}(\tilde{\mathsf{ap}}_l)$-th value in $\mathbf{v}$; we say $\mathbf{v}$ is *extracted at i* if this holds.
3. $\tilde{\sigma}_{k(\mathbb{x})} = \sigma_{k(\mathbb{x})}$.

DEFINITION 2. We say that $\tilde{\mathbb{P}}$ *chooses* $\rho \in \mathcal{U}(\lambda)$ if for every query θ made by $\tilde{\mathbb{P}}^{\rho}$ to its oracle, $\tilde{P}$ supplies it with $\rho(\theta)$ (ignoring whether this response comes from $\tilde{P}$ itself or the messages sent by V; this choice is fixed for a given ρ).

CLAIM 1. $(\tilde{P}, V)$ chooses $\rho \in \mathcal{U}(\lambda)$ uniformly at random.

Whenever the simulation of $\tilde{\mathbb{P}}$ makes a query, $\tilde{P}$ responds consistently, either with a uniformly randomly drawn string of its own, or the uniform randomness provided by V. This is equivalent in distribution to drawing ρ uniformly at random at the beginning of the protocol. ■

CLAIM 2. For any choice of randomness such that $\tilde{P}$ chooses a good ρ, $\tilde{P}$ makes $V(\mathbb{x})$ accept with a state restoration attack.

We begin by defining a property of the map α.

DEFINITION 3. For $i = 0, \ldots, k$, we say that α is *correct at* i if, immediately before $\tilde{\mathbb{P}}$'s j_{i+1}-th query is simulated (for $i = k$, at the end of the simulation), it holds that

$$\alpha(\sigma_i) = (\rho_1(\mathtt{x}\|\sigma_0), f_1, \ldots, \rho_1(\mathtt{x}\|\sigma_i)) \ ,$$

where for each $l \in \{1, \ldots, i\}$, f_l is extracted at l (see Condition 2d above), and $\alpha(\sigma_i) \in \mathsf{SeenStates}$.

We show by induction that α is correct at i for every $i \in \{0, \ldots, k\}$. First, α is correct at 0 since $\alpha(\sigma_0) = (\rho_1(\mathtt{x}\|\sigma_0))$ by construction. Suppose that α is correct at $i-1$. When $\tilde{\mathbb{P}}^{\rho}$ queries $\tilde{\mathsf{rt}}_i\|\sigma_{i-1}$ (i.e., query θ_{j_i}), $\tilde{P}$ restores $\alpha(\sigma_{i-1}) \in \mathsf{SeenStates}$. By Condition 2d, f_i is extracted at i. In Step 3g, $\rho_1(\mathtt{x}\|\sigma_i)$ is set to the message (or, similarly, internal randomness) sent by V in this round, which is possible by Condition 2c. The newly stored state is then $\alpha(\sigma_i) = (\rho_1(\mathtt{x}\|\sigma_0), f_1, \ldots, \rho_1(\mathtt{x}\|\sigma_{i-1}), f_i, \rho_1(\mathtt{x}\|\sigma_i)) \in \mathsf{SeenStates}$. This state is stored before query j_{i+1} by Condition 2a, and so α is correct at i.

Hence $\tilde{\mathbb{P}}$ sends a state $\alpha(\sigma_k) = (\rho_1(\mathtt{x}\|\sigma_1), f_1, \ldots, \rho_1(\mathtt{x}\|\sigma_k)) \in \mathsf{SeenStates}$. Since $\mathbb{V}$'s simulation of V accepts with this state, so does the real V when interacting with $\tilde{\mathbb{P}}$. ■

CLAIM 3. The probability that $\rho \in \mathcal{U}(\lambda)$ is good is at least $\epsilon - 3(m^2+1)2^{-\lambda}$.

By assumption, the density of oracles satisfying Condition 1 is ϵ. Lemma 7.2 implies that the density of oracles satisfying Condition 1 but not satisfying Condition 2a, Condition 2b, and Condition 3 is at most $(m^2+1)2^{-\lambda}$.[6] The density of oracles failing to satisfy Condition 2c is at most $m^2 2^{-\lambda}$, since this implies a 'collision' (in the sense of Lemma 3.2) between ρ_1 and ρ_2. Finally, the density of oracles satisfying Condition 1, Condition 2a, and Condition 2b, but not Condition 2d is at most $(m^2+1)2^{-\lambda}$, by Lemma 3.2 and Condition 2b (where Condition 2b allows us to restrict the possible values for a_i to $0 \le a_i < j_i$).

By the union bound, the density of good oracles ρ is at least $\epsilon - 3(m^2+1)2^{-\lambda}$. ■

Combining the claims, we deduce that $\tilde{P}$ makes V accept with probability at least $\epsilon - 3(m^2+1)2^{-\lambda}$ with a state restoration attack. Finally, note that this state restoration attack is restricted because $\tilde{P}$ never requests to set V to the empty verifier state null. □

7.2 Lower bound for soundness

We show that the soundness analysis of Section 7.2 is essentially tight. Let $\mathtt{x} \notin \mathscr{L}(\mathscr{R})$ and let $\tilde{P}$ be a b-round state-restoring prover for the interactive oracle proof system (P, V). We construct an m-query prover $\tilde{\mathbb{P}}$ for the non-interactive random-oracle proof system $(\mathbb{P}, \mathbb{V})$, and show that $\tilde{\mathbb{P}}$'s ability to cheat is closely related to $\tilde{P}$'s ability to cheat in a state restoration attack.

Construction of $\tilde{\mathbb{P}}$. Given oracle access to ρ, the prover $\tilde{\mathbb{P}}$ works as follows.

1. Initialize $\mathsf{SeenStates}$ to be an empty set and α to be an empty mapping from verifier states to $\{0,1\}^{\lambda}$.
2. Compute $k := k(\mathtt{x})$, $\sigma_0 := \rho_2(\mathtt{x})$, and $m_1 := \rho_1(\mathtt{x}\|\sigma_0)$.
3. Send the verifier message m_1 to the simulation of $\tilde{P}$.
4. Set the verifier state $\mathsf{cvs} := (m_1)$, add cvs to $\mathsf{SeenStates}$, and set $\alpha(\mathsf{cvs}) := \sigma_0$.
5. For $j = 1, \ldots, b$:
   (a) Simulate $\tilde{P}$ until it selects $\mathsf{cvs}^{(j)} = (m_1, f_1, \ldots, m_{i^{(j)}}) \in \mathsf{SeenStates}$ and outputs a message $f^{(j)}$.
   (b) Compute the root $\mathsf{rt} := \mathsf{MERKLE.GetRoot}^{\rho_2}(f^{(j)}\|j)$, where j is encoded in $\{0,1\}^{\lambda}$.

[6] More precisely, we apply Lemma 7.2 to an algorithm $\tilde{\mathbb{P}}$ that does not itself output $\mathtt{x}$ but this does not affect the lemma's validity because we can substitute into the definition of the event E_1 the fixed instance $\mathtt{x}$.

(c) Set $\sigma := \alpha(\mathsf{cvs}_j)$, $\sigma' := \rho_2(\mathsf{rt}\|\sigma)$, $m^{(j)} := \rho_1(\mathtt{x}\|\sigma')$, and $j_{f^{(j)}} := j$.

(d) If $i^{(j)} < k$, send the verifier message $m^{(j)}$ to the simulation of $\tilde{P}$, add $\mathsf{cvs} := \mathsf{cvs}^{(j)}\|(f^{(j)}, m^{(j)})$ to $\mathsf{SeenStates}$, set $\alpha(\mathsf{cvs}) := \sigma'$, and continue to the next iteration.

(e) If $i^{(j)} = k$, compute the bit $d := V^{f_1,\ldots,f_k}(\mathtt{x}, \emptyset; (m_1, \ldots, m_k, m^{(j)}))$. If $d = 0$, continue to the next iteration; else if $d = 1$, compute a proof π as in Step 4 and 5 of the construction of $\mathbb{P}$ in Section 6, using $\sigma_k := \sigma$ and replacing f_i with $f_i\|i_{f_i}$ for $i = 1, \ldots, k$, then output π.

Analysis of $\tilde{\mathbb{P}}$. We now analyze $\tilde{\mathbb{P}}$.

Lemma 7.4. *There exists a constant c such that if $\tilde{P}$ is a b-round state-restoring prover that makes V accept $\mathtt{x}$ with probability ϵ, then $\tilde{\mathbb{P}}$ is an m-query prover that makes $\mathbb{V}$ accept $\mathtt{x}$ with probability at least $\epsilon - m^2 2^{-\lambda}$, where $m := c \cdot (\lambda + p(\mathtt{x}) + u(\mathtt{x})) \cdot b$ and $u(\mathtt{x})$ is the sum of the lengths of all the verifier's messages.*

Proof. First, we argue that $\tilde{\mathbb{P}}$ makes no more than $c \cdot (\lambda + p(\mathtt{x}) + u(\mathtt{x})) \cdot b$ oracle queries for some universal constant c. Whenever $\tilde{P}$ plays a round, $\tilde{\mathbb{P}}$ makes:

- one call to ρ_1 (chained $\lceil u_i(\mathtt{x})/\lambda \rceil \leq u(\mathtt{x})$ times where u_i is the length of the i-th verifier message);
- one call to ρ_2; and
- one call to $\mathsf{MERKLE.GetRoot}^{\rho_2}$ on a string of length at most $p(\mathtt{x})+\lambda$, which requires at most $c' \cdot (p(\mathtt{x})+\lambda)$ queries for some universal constant c'.

In Step 5e, $\tilde{\mathbb{P}}$ does not need to make any new queries. Thus, the number of queries is at most m.

Next, we argue if (the simulation of) $\tilde{P}$ convinces (the simulation of) V, then $\tilde{\mathbb{P}}$ convinces $\mathbb{V}$. This is clear because $\tilde{\mathbb{P}}$ constructs the proof π from messages and randomness that cause (the simulation of) V to accept. It is also clear that (the simulation of) $\tilde{P}$ sees a legal restricted state restoration transcript.

Finally, we argue that (the simulation of) $\tilde{P}$ convinces (the simulation of) V with the claimed probability. If all the root hashes σ' are distinct, then all the verifier messages are uniformly randomly drawn, because each message is computed as $\rho_1(\mathtt{x}\|\sigma')$; hence, by assumption, $\tilde{P}$ convinces V with probability ϵ. Every Merkle tree root is computed relative to a distinct string (because we include the prover round index in the string), so that if two root hashes collide then this implies a collision in ρ_2, which can occur with probability at most $m^2 2^{-\lambda}$. By the union bound, $\tilde{\mathbb{P}}$ causes $\mathbb{V}$ to accept with probability at least $\epsilon - m^2 2^{-\lambda}$. □

7.3 Proof of knowledge

We describe how the transformation T preserves proof of knowledge. Suppose that the interactive oracle proof system (P, V) has restricted state restoration proof of knowledge $\bar{e}_{\mathrm{sr}}(\mathtt{x}, m)$, and denote by E_{sr} the corresponding extractor. The security reduction in Section 7.1, in which we construct $\tilde{P}$ from $\mathtt{x}$ and $\tilde{\mathbb{P}}$, suggests a natural strategy for extracting a witness from $\tilde{\mathbb{P}}$. Indeed, note that $\tilde{P}$ does not need to know the code, randomness, or auxiliary inputs of $\tilde{\mathbb{P}}$, but only needs to choose answers for its queries to the random oracle; this means that one can compute $\tilde{P}$ when only given $\mathtt{x}$ and oracle access to $\tilde{\mathbb{P}}$.

So consider the extractor $\mathbb{E}$ that, given input $(\mathtt{x}, 1^m, 1^\lambda)$ and oracle access to $\tilde{\mathbb{P}}$, works as follows: (1) construct the oracle $\tilde{P}$ from the instance $\mathtt{x}$ and oracle $\tilde{\mathbb{P}}$; (2) compute $\mathtt{w} := E_{\mathrm{sr}}^{\tilde{P}}(\mathtt{x})$; (3) output $\mathtt{w}$. Note that $\mathbb{E}$ runs in probabilistic polynomial time because the oracle $\tilde{P}$ can be simulated in probabilistic $\mathrm{poly}(|\mathtt{x}|+m+\lambda)$ time when given input $(\mathtt{x}, 1^m, 1^\lambda)$ and the oracle $\tilde{\mathbb{P}}$ and, moreover, E_{sr} runs in probabilistic $\mathrm{poly}(|\mathtt{x}|)$ time.

The extractor $\mathbb{E}$ shows that $(\mathbb{P}, \mathbb{V}) := T(P, V)$ has proof of knowledge $e'(\mathtt{x}, m, \lambda) := \bar{e}_{\mathrm{sr}}(\mathtt{x}, m) + 3(m^2 + 1)2^{-\lambda}$ (as stated in Theorem 7.1), as we now argue. By Lemma 7.3, the state-restoring prover $\tilde{P}$ makes V accept $\mathtt{x}$ with probability that is at least the probability that $\tilde{\mathbb{P}}$ makes $\mathbb{V}$ accept $\mathtt{x}$ minus $3(m^2 + 1)2^{-\lambda}$.

Moreover, (P, V) has restricted state restoration proof of knowledge $\bar{e}_{\mathrm{sr}}$, so that $\Pr[(\mathtt{x}, E^{\tilde{P}}_{\mathrm{sr}}(\mathtt{x})) \in \mathscr{R}] \geq \Pr[\langle \tilde{P}, V(\mathtt{x})\rangle_{\mathrm{sr}} = 1] - \bar{e}_{\mathrm{sr}}(\mathtt{x}, m)$. We conclude that

$$\begin{aligned}
\Pr\left[(\mathtt{x},\mathtt{w}) \in \mathscr{R} \;\middle|\; \mathtt{w} \leftarrow \mathbb{E}^{\tilde{\mathbb{P}}}(\mathtt{x}, 1^m, 1^\lambda)\right] &= \Pr[(\mathtt{x}, E^{\tilde{P}}_{\mathrm{sr}}(\mathtt{x})) \in \mathscr{R}] \\
&\geq \Pr[\langle \tilde{P}, V(\mathtt{x})\rangle_{\mathrm{sr}} = 1] - \bar{e}_{\mathrm{sr}}(\mathtt{x}, m) \\
&\geq \Pr\left[\mathbb{V}^\rho(\mathtt{x}, \pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \tilde{\mathbb{P}}^\rho \end{array}\right] - 3(m^2+1)2^{-\lambda} - \bar{e}_{\mathrm{sr}}(\mathtt{x}, m) \\
&= \Pr\left[\mathbb{V}^\rho(\mathtt{x}, \pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \tilde{\mathbb{P}}^\rho \end{array}\right] - e'(\mathtt{x}, m, \lambda) \ ,
\end{aligned}$$

as desired.

Finally, the tightness argument of Section 7.2 can be extended from applying to (restricted state restoration) soundness to also applying to (restricted state restoration) proof of knowledge.

7.4 Zero knowledge

We describe how the transformation T preserves zero knowledge, by proving the following lemma.

Lemma 7.5. *If (P, V) has z-statistical honest-verifier zero knowledge, then $(\mathbb{P}, \mathbb{V})$ has z'-statistical zero knowledge (in the explicitly-programmable random oracle model) with $z'(\mathtt{x}, \lambda) := z(\mathtt{x}) + p(\mathtt{x})2^{-\lambda/4+2}$.*

Proof. Let S be the simulator for (P, V), and S_{MERKLE} the Merkle-tree simulator (see Section 3.2). We construct a (candidate) simulator $\mathbb{S}$ for $(\mathbb{P}, \mathbb{V}) := T(P, V)$ that uses S and S_{MERKLE} as subroutines. To account for the split of ρ into the two sub-oracles ρ_1 and ρ_2, we view the function μ from the definition of zero knowledge (see Section 2.3) as a pair (μ_1, μ_2) such that $\rho[\mu]$ splits into the function pair $(\rho_1[\mu_1], \rho_2[\mu_2])$. The simulator $\mathbb{S}$, given instance $\mathtt{x}$ and oracle access to ρ, works as follows:

1. Set $k := k(\mathtt{x}), q := q(\mathtt{x}), \ell_1 := \ell_1(\mathtt{x}), \ldots, \ell_k := \ell_k(\mathtt{x}), \sigma_0 := \rho_2(\mathtt{x})$.
2. Compute the view $(a_1, \ldots, a_q, r_V) := S(\mathtt{x})$.
3. Run V with randomness r_V, recording its messages $m_1, \ldots, m_k$ and answering its oracle queries with the answers $a_1, \ldots, a_q$, to determine the oracle index y_j and bit index x_j of each query θ_j.
4. For $i = 1, \ldots, k$:
   (a) Run S_{MERKLE} to obtain the requested authentication paths corresponding to the i-th oracle by setting $(\mathsf{rt}_i, (\mathsf{ap}_j)_{j \in J_i}) := S_{\mathsf{MERKLE}}(\ell_i, (x_j, a_j)_{j \in J_i})$ where $J_i := \{j \in \{1, \ldots, q\} \text{ s.t. } y_j = i\}$.
   (b) Compute $\sigma_i := \rho_2(\mathsf{rt}_i \| \sigma_{i-1})$.
   (c) Set $\mu_1(\mathtt{x} \| \sigma_{i-1}) := m_i$.
5. Set $\pi := \big((\mathsf{rt}_1, \ldots, \mathsf{rt}_k), (\mathsf{ap}_1, \ldots, \mathsf{ap}_q), (\sigma_1, \ldots, \sigma_k)\big)$, and output (π, μ).

Note that $\mathbb{S}$ runs in probabilistic polynomial time (as its complexity is dominated by that of S_{MERKLE} and S).

Next, we construct a "hybrid simulator" $\mathbb{S}'$ that is identical to $\mathbb{S}$ except that: (a) $\mathbb{S}'$ is given not only an instance $\mathtt{x}$ but also the corresponding witness $\mathtt{w}$; (b) in Step 2, instead of using S's output view, $\mathbb{S}'$ generates a uniformly random string r_V and uses it to simulate $(P(\mathtt{x}, \mathtt{w}), V(\mathtt{x}; r_V))$ and obtain the answers $a_1, \ldots, a_q$ and corresponding oracle and bit indices. We claim that, for any distinguisher D, the following two distributions are $p(\mathtt{x})2^{-\lambda/4+2}$-close:

$$\Pr\left[D^\rho(\pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \mathbb{P}^\rho(\mathtt{x}, \mathtt{w}) \end{array}\right] \text{ and } \Pr\left[D^{\rho[\mu']}(\pi') = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ (\pi', \mu') \leftarrow \mathbb{S}'^\rho(\mathtt{x}, \mathtt{w}) \end{array}\right] \ .$$

Indeed, note that ρ and $\rho[\mu']$ are identically distributed (since the values in μ' are uniformly random) and $\pi \leftarrow \mathbb{P}^{\rho[\mu']}(\mathtt{x}, \mathtt{w})$ is $p(\mathtt{x})2^{-\lambda/4+2}$-close to π' (by Lemma 3.4).

Finally, we know by hypothesis that, for any distinguisher D, the following two distributions are z-close:

$$\Pr\left[D^{\rho[\mu']}(\pi') = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ (\pi', \mu') \leftarrow \mathbb{S}'^{\rho}(\mathtt{x}, \mathtt{w}) \end{array}\right] \text{ and } \Pr\left[D^{\rho[\mu]}(\pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ (\pi, \mu) \leftarrow \mathbb{S}^{\rho}(\mathtt{x}) \end{array}\right] .$$

The lemma then follows by the triangle inequality. □

8 Tree exploration games

We introduce tree exploration games, which are a class of games pitting a single player against Nature. We prove a connection between tree exploration games and state restoration attacks on IOPs: we establish a correspondence between public-coin IOP verifiers and tree exploration games and, similarly, between state-restoring provers and strategies on these games. Finally, we describe a simple universal strategy, and prove that it has optimal expected cost (where cost is the number of rounds until the player wins).

8.1 Definition of tree exploration games

A *tree exploration game* is a complete-information game, between a single player and Nature, that is specified by a pair $\mathcal{G} = (T, \phi)$, where T is a rooted tree and ϕ is a predicate function that maps T's vertices to $\{0, 1\}$; we denote by $V(T)$ the vertices of T, and by $\mathsf{rt}(T)$ the root vertex of T.

The game proceeds in rounds: for $i = 1, 2, \ldots$, in the i-th round, a subtree $S_{i-1} \subseteq T$ is *accessible* to the player; the player chooses a node $v \in S_{i-1}$, and Nature samples a child u of v at random; the next accessible subtree is $S_i := S_{i-1} \cup \{u\}$. The initial accessible set S_0 is $\{\mathsf{rt}(T)\}$, i.e., the set consisting of T's root vertex. The player wins in round r if there is $v \in S_r$ such that $\phi(v) = 1$.

A *strategy* for $\mathcal{G}$ is a probabilistic function $s \colon 2^{V(T)} \to V(T)$ such that $s(W) \in W$ for all $W \subseteq V(T)$. Note that $s(S_0)$ always equals $\mathsf{rt}(T)$, because the root vertex is the only vertex in the initial accessible set; if Nature samples the child u of $\mathsf{rt}(T)$, then $S_1 := \{\mathsf{rt}(T), u\}$, and so $s(S_1)$ may output $\mathsf{rt}(T)$ or u (or each with some probability). Thus the accessible sets form a sequence of random variables each of which depends on the strategy s; we emphasize this by writing $S_i(s)$. Note that we have defined a strategy to be a function only of the current accessible set (and not also of previous choices); we justify this choice later (see Lemma 8.3).

The *cost* of a strategy s is the random variable τ_s that denotes the smallest index r such that there is $v \in S_r(s)$ for which $\phi(v) = 1$. More generally, given a set $W \subseteq T$ containing $\mathsf{rt}(T)$, $\tau_s(W)$ denotes the cost of s when the initial accessible set equals W; thus, in particular, $\tau_s = \tau_s(\{\mathsf{rt}(T)\})$. Note that $\tau_s(W) = 0$ if $\phi(v) = 1$ for some $v \in W$ and $\tau_s(W) = \infty$ if $\phi(v) = 0$ for all $v \in W$. For other cases, $\tau_s(W)$ satisfies the following recursion:

$$\mathbb{E}[\tau_s(W)] = 1 + \frac{1}{|C_{s(W)}|} \sum_{v \in C_{s(W)}} \mathbb{E}[\tau_s(W \cup \{v\})] \tag{1}$$

where C_v for $v \in T$ is the set of children of v.

Remark 8.1. We do not discuss probabilistic strategies in this paper. One can verify, however, that for every probabilistic strategy s there is a deterministic strategy s' whose expected cost is no higher; the strategy s' can be obtained by iteratively fixing the random choices of s such that $\mathbb{E}[\tau_s]$ is minimized.

8.2 A connection between tree exploration and state restoration

We establish a connection between tree exploration and state restoration (discussed in Section 5).

From IOPs to tree exploration games. First, we describe how to map an interactive oracle proof system to a tree exploration game. Let (P, V) be a k-round public-coin interactive oracle proof system, and let $\mathtt{x}$ be an instance; we use V and $\mathtt{x}$ to construct a tree exploration game $\mathcal{G}_{V,\mathtt{x}} = (T, \phi)$ as follows. The vertices of the rooted tree T consist of sequences of at most $k(\mathtt{x}) + 1$ verifier messages: a vertex v looks like $(m_1, \ldots, m_i)$ with $i \in \{1, \ldots, k(\mathtt{x})\}$ and each m_j in $\{0,1\}^{u_j(\mathtt{x})}$, or looks like $(m_1, \ldots, m_{k(\mathtt{x})}, r)$ with each m_j as before and r being additional randomness for the verifier; we denote by $v(j)$ the j-prefix of v. The edges of T correspond to sequences for which one sequence extends the previous one by an additional

message: an edge (u, v) is in the edge set if $v = u \| m$ for some message m. The predicate function $\phi\colon V(T) \to \{0,1\}$ is such that for every vertex v: (a) if v is not a leaf node (i.e., contains less than $k(\mathtt{x}) + 1$ messages), ϕ maps v to 0; (b) if v is a leaf node (i.e., contains exactly $k(\mathtt{x}) + 1$ messages), ϕ maps v to the bit $V^{f_{v(1)},\ldots,f_{v(k(\mathtt{x}))}}(\mathtt{x}; \mathsf{state}_{k(\mathtt{x})}; r_V)$ where $\mathsf{state}_{k(\mathtt{x})} := \emptyset$, $r_V := v$, and the $f_{v(j)}$ are defined as follows. Given a vertex $u = (m_1, \ldots, m_i)$, let $X_u = (m_1, \ldots, m_i, m_{i+1}, \ldots, m_{k(\mathtt{x})}, r)$ be the random variable obtained by setting the first i coordinates according to u and choosing the rest uniformly at random. Then

$$f_u := \arg\max_{f_i} \left(\max_{f'_{i+1},\ldots,f'_{k(\mathtt{x})}} \Pr_{X_u} \left[V^{f_{u(1)},\ldots,f_{u(i-1)},f_i,f'_{i+1}\cdots,f'_{k(\mathtt{x})}}(\mathtt{x}, \emptyset; X_u) = 1 \right] \right) \ ,$$

where, for $j = i+1, \ldots, k(\mathtt{x})$, f'_j ranges over probabilistic functions of the random variables $m_{i+1}, \ldots, m_j$. The following lemma is an easy consequence of this definition.

Lemma 8.2. *Let $X := (m_1, \ldots, m_{k(\mathtt{x})}, r)$ be drawn uniformly at random, with $m_i \in \{0,1\}^{u_i}$ and $r \in \{0,1\}^*$ sufficiently long. Let $f_1, \ldots, f_{k(\mathtt{x})}$ be such that, for $j = 1, \ldots, k(\mathtt{x})$, f_j is a probabilistic function of the random variables $m_1, \ldots, m_j$. Then*

$$\Pr_X \left[V^{f_1,\ldots,f_{k(\mathtt{x})}}(\mathtt{x}, \emptyset; X) = 1 \right] \leq \Pr_X \left[V^{f_{X(1)},\ldots,f_{X(k(\mathtt{x}))}}(\mathtt{x}, \emptyset; X) = 1 \right] \ .$$

Proof. Follows easily by induction and the definition of f_u. □

Augmented strategies vs. standard strategies. Next, we extend the notion of a strategy to allow tracking, and depending on, previous choices; we use this notion in establishing a correspondence later on. We say a sequence $H = (v_1, \ldots, v_h) \in V(T)^h$ is a *valid history* if $v_1 = \mathsf{rt}(T)$ and, for all $j \in \{2, \ldots, h\}$, v_j is adjacent to some vertex $u \in \{v_1, \ldots, v_{j-1}\}$. Every nonempty prefix of a valid history is also a valid history, and that vertices in the sequence need not be distinct.

Given a tree exploration game $\mathcal{G} = (T, \phi)$, a function $s\colon (V(T))^{\leq b} \to V(T) \cup \{\perp\}$ is a *b-augmented strategy* for $\mathcal{G}$ if, for all valid histories $(v_1, \ldots, v_b)$, it holds that $s((v_1, \ldots, v_j)) \in \{v_1, \ldots, v_j\} \cup \{\perp\}$ for all $j \in \{1, \ldots, b\}$. (The value of $s(H)$ when H is not a valid history is irrelevant.) To play a b-augmented strategy on a tree, we provide it with the history H of vertices drawn so far. We prove that, in terms of expected cost, b-augmented strategies are no more powerful than (standard) strategies.

Lemma 8.3. *Let $\mathcal{G} = (T, \phi)$ be a tree exploration game. For every b-augmented strategy s for $\mathcal{G}$ there is a (standard) strategy s' for $\mathcal{G}$ such that $\mathbb{E}[\tau_{s'}] \leq \mathbb{E}[\tau_s]$.*

Proof. For $W = \{v_1, \ldots, v_i\} \subseteq T$, let

$$s'(W) := \begin{cases} s((v_{\pi(1)}, \ldots, v_{\pi(i)})) & \text{if } s((v_{\pi(1)}, \ldots, v_{\pi(i)})) \neq \perp \\ \mathsf{rt}(T) & \text{otherwise} \end{cases}$$

where π minimizes $\mathbb{E}[\tau_s((v_{\pi(1)}, \ldots, v_{\pi(i)}))]$ across all permutations over $[i]$ for which $(v_{\pi(1)}, \ldots, v_{\pi(i)})$ is a valid history. Note that $s'(W) \in W$ since if $s((v_{\pi(1)}, \ldots, v_{\pi(i)})) \neq \perp$ then $s((v_{\pi(1)}, \ldots, v_{\pi(i)})) \in W$. The expected cost of s for any valid history $(v_1, \ldots, v_i)$ is thus bounded from below as follows

$$\mathbb{E}[\tau_s((v_1, \ldots, v_i))] \geq \mathbb{E}[\tau_{s'}(\{v_1, \ldots, v_i\})] \ .$$

Hence in particular, $\mathbb{E}[\tau_s] = \mathbb{E}[\tau_s((\mathsf{rt}(T)))] \geq \mathbb{E}[\tau_{s'}(\{\mathsf{rt}(T)\})] = \mathbb{E}[\tau_{s'}]$. □

The correspondence. Let (P, V) be a k-round public-coin interactive oracle proof system, and let $\mathtt{x}$ be an instance; let $\mathcal{G}_{V,\mathtt{x}} = (T, \phi)$ be the tree exploration game constructed as above. We now describe, for any budget $b \in \mathbb{N}$, a correspondence between (deterministic) b-round state-restoring provers for $(P, V(\mathtt{x}))$ and b-augmented strategies for $\mathcal{G}_{V,\mathtt{x}}$.

Given $m_1, \ldots, m_i \in \{0,1\}^*$, the state corresponding to a vertex $v = (m_1, \ldots, m_i)$ is $\mathsf{cvs}(v) := (m_1, f_{v(1)}, \ldots, m_{i-1}, f_{v(i-1)}, m_i)$. The vertex corresponding to a state $\mathsf{cvs} := (m_1, f_1, \ldots, m_i)$ is $v(\mathsf{cvs}) := (m_1, \ldots, m_i)$. A history $(v_1, \ldots, v_j)$ corresponds to the list of states $(\mathsf{cvs}(v_1), \ldots, \mathsf{cvs}(v_j))$ and vice versa.

- *From provers to strategies.* Let P be a b-round state-restoring prover; construct a b-augmented strategy s_P as follows. Given a valid history $(v_1, \ldots, v_h)$ as input, s_P works by simulating a state-restoration attack by P on $V(\mathtt{x})$, as follows. For $i \in \{1, \ldots, h-1\}$, in the i-th prover round, when P chooses a complete verifier state $\mathsf{cvs} = (m_1, f_1, \ldots, m_j)$, we check whether the vertex $v_{i+1} =: (m'_1, \ldots, m'_{j'})$ in the history is a child of $v(\mathsf{cvs})$. If it is not, or if P halts, then s_P outputs $\perp$. Otherwise, we respond to P with $m'_{j'}$ and continue. In the h-th prover round, s_P outputs $v(\mathsf{cvs})$.

- *From strategies to provers.* Let s be a b-augmented strategy; construct a b-round state-restoring prover P_s as follows. In prover round j, P_s computes $v_s := s((v_1, \ldots, v_j))$, then restores the state $\mathsf{cvs}(v_s)$, and sends the prover message f_{v_s}. Afterwards, P_s sets $v_{j+1} := (m_1, \ldots, m_i, m_{i+1})$, where $(m_1, \ldots, m_i)$ are the messages in v_s and m_{i+1} is the verifier's response; if $s((v_1, \ldots, v_{j+1})) = \perp$, P_s halts.

The following lemma justifies the study of tree exploration games in relation to state restoration soundness of interactive oracle proof systems; namely, we prove that b-augmented strategies and b-round state-restoring provers are equivalent in the following sense.

Lemma 8.4. *Let $b \in \mathbb{N}$.*

- *For every b-round state-restoring prover $\tilde{P}$ for $(P, V(\mathtt{x}))$, $\Pr[s_{\tilde{P}}$ wins in $\mathcal{G}_{V,\mathtt{x}}] \geq \Pr[\tilde{P}$ makes V accept $\mathtt{x}]$.*
- *For every b-augmented strategy s for $\mathcal{G}_{V,\mathtt{x}}$, $\Pr[\tilde{P}_s$ makes V accept $\mathtt{x}] = \Pr[s$ wins in $\mathcal{G}_{V,\mathtt{x}}]$.*

Proof. First we prove the first bullet. Note that, by the definition of $\mathcal{G}_{V,\mathtt{x}}$, a b-augmented strategy s wins in $\mathcal{G}_{V,\mathtt{x}}$ if and only if there exists $v = (m_1, \ldots, m_{k(\mathtt{x})}, r) \in S_b$ such that $V^{f_{v(1)}, \ldots, f_{v(k(\mathtt{x}))}}(\mathtt{x}, \emptyset; v) = 1$.

Each prover message f_i is a probabilistic function of the messages $m_1, \ldots, m_i$ that precede it ('probabilistic' because $\tilde{P}$ can obtain independent randomness by restoring some state). Then by Lemma 8.2 we can replace $\tilde{P}$'s messages with f_u, where u is the tuple of verifier messages in the state cvs restored at the beginning of the prover round, without reducing its success probability. We call this prover $\tilde{P}'$; note that $s_{\tilde{P}'} = s_{\tilde{P}}$. We observe that $\tilde{P}'$ succeeds if and only if, after b prover rounds, there is some state $\mathsf{cvs} = (m_1, f_{(m_1)}, m_2, f_{(m_1,m_2)}, \ldots, m_{k(\mathtt{x})}, f_{(m_1,\ldots,m_{k(\mathtt{x})})}, r) \in \mathsf{SeenStates}$ such that for $u := (m_1, \ldots, m_{k(\mathtt{x})}, r)$ it holds that $V^{f_{u(1)}, \ldots, f_{u(k(\mathtt{x}))}}(\mathtt{x}, \emptyset; u) = 1$.

Suppose that we play the tree exploration game $\mathcal{G}_{V,\mathtt{x}}$ with strategy $s_{\tilde{P}'}$. We couple this game to an interaction between $\tilde{P}'$ and V, and show that, whenever we add a state cvs to $\mathsf{SeenStates}$, we also add the corresponding vertex $v(\mathsf{cvs})$ to H, the sequence of vertices drawn during the game. These correspond initially because $\mathsf{SeenStates} = (\mathsf{null})$, and $H = (\mathsf{rt}(T)) = (v(\mathsf{null}))$. Now suppose that we have played $\mathcal{G}_{V,\mathtt{x}}$ coupled to $(\tilde{P}', V)$ for some number of rounds, and the current history H corresponds to $\mathsf{SeenStates}$. Also, $s_{\tilde{P}'}(H) = \perp$ if and only if $\tilde{P}'$ halts because H must be consistent with the choices of $s_{\tilde{P}'}$, and moreover $\tilde{P}'$ is deterministic. Let $v' := s_{\tilde{P}'}(H)$; observe that $v' \in H$ since $\mathsf{cvs}(v')$ is restored by $\tilde{P}'$ and hence must be in $\mathsf{SeenStates}$. Then the new vertex v added to H is chosen uniformly at random from the children of v'. This is equivalent in distribution to V's choice of message after $\tilde{P}'$ restores the state $\mathsf{cvs}(v')$ and sends its message $f_{v'}$; hence we may couple these random choices so that the vertex $v := v_s \| m$ is added to H if and only if

$\mathsf{cvs}(v)$ is added to $\mathsf{SeenStates}$. Hence $s_{\tilde{P}'}$ (i.e., $s_{\tilde{P}}$) wins in $\mathcal{G}_{V,\mathtt{x}}$ if and only if $\tilde{P}'$ makes V accept $\mathtt{x}$ under this coupling, and since the probability that this happens is at least the probability that $\tilde{P}$ makes V accept $\mathtt{x}$, this concludes the proof of the first bullet.

Next we prove the second bullet. The proof is again via coupling argument: consider an interaction between $\tilde{P}_s$ and V. We couple this interaction to s played on $\mathcal{G}_{V,\mathtt{x}}$, and show that whenever we add a vertex v to the history H, we also add the corresponding state $\mathsf{cvs}(v)$ to $\mathsf{SeenStates}$. Again, $\mathsf{SeenStates}$ and H correspond initially. Suppose that $(\tilde{P}_s, V)$ have interacted for some number of rounds, coupled to $\mathcal{G}_{V,\mathtt{x}}$, and that $\mathsf{SeenStates}$ and H correspond. Then $\mathsf{cvs} := \mathsf{cvs}(s(H)) \in \mathsf{SeenStates}$ since $s(H) \in H$, and $\tilde{P}_s$ is able to restore cvs. Again we observe that the response of V and the choice of child of $s(H)$ are identically distributed, and we may couple them. Under this coupling, a state $\mathsf{cvs}' := \mathsf{cvs}\|(f_{s(H)}, m)$ is added to $\mathsf{SeenStates}$ if and only if $v(\mathsf{cvs}')$ is added to H. Hence $\tilde{P}_s$ makes V accept $\mathtt{x}$ if and only if s wins in $\mathcal{G}_{V,\mathtt{x}}$ under this coupling, which proves the equality. □

8.3 The expected value strategy

We show how to construct, for every tree exploration game $\mathcal{G} = (T, \phi)$, a simple strategy $s^\star$ that has optimal expected cost with respect to all strategies of that game; we call this strategy the *expected value strategy*, for reasons that will become clear below.

We associate to each vertex $v \in T$ a *potential* $\Delta(v)$. If v is a leaf and $\phi(v) = 0$, then we set $\Delta(v) := 1$; if v is a leaf and $\phi(v) = 0$, we set $\Delta(v) := \infty$. If v is not a leaf, we set

$$\Delta(v) := \min_{U \subseteq C_v} \frac{1}{|U|} \left(|C_v| + \sum_{u \in U} \Delta(u) \right) \ .$$

One can show that the smallest U minimizing the right hand side is $C_{<v} := \{u \in C_v : \Delta(u) < \Delta(v)\}$. The expected value strategy is defined in terms of potentials: for every $S \subseteq V(T)$, $s^\star(S) := \arg\min_{v \in S} \Delta(v)$.

We first prove that the expected cost of the expected value strategy equals the potential of the root vertex.

Claim 8.5. $\mathbb{E}[\tau_{s^\star}] = \Delta(\mathsf{rt}(T))$.

Proof. Since $\tau_{s^\star} = \tau_{s^\star}(\{\mathsf{rt}(T)\})$ and $s^\star(\{\mathsf{rt}(T)\}) = \mathsf{rt}(T)$, the statement is equivalent to $\mathbb{E}[\tau_{s^\star}(W)] = \Delta(s^\star(W))$ for $W = \{\mathsf{rt}(T)\}$. We prove this by reverse induction on the subtree W. So first suppose that $W = V(T)$. In this case, if there is $v \in V(T)$ such that $\phi(v) = 1$ then $\mathbb{E}[\tau_{s^\star}(V(T))] = 0 = \Delta(s^\star(V(T)))$; otherwise, $\mathbb{E}[\tau_{s^\star}(V(T))] = \infty = \Delta(s^\star(V(T)))$. This establishes the base case; next we establish the inductive step. Let W be a subtree of T, and suppose that $\mathbb{E}[\tau_{s^\star}(W')] = \Delta(s^\star(W'))$ for all $W' \subseteq T$ that contain, but do not equal, W; we now argue that $\mathbb{E}[\tau_{s^\star}(W)] = \Delta(s^\star(W))$ as well. Letting $u := s^\star(W)$ and invoking Equation 1, we obtain that

$$\begin{aligned}
\mathbb{E}[\tau_{s^\star}(W)] &= 1 + \frac{1}{|C_u|} \sum_{v \in C_u} \mathbb{E}[\tau_{s^\star}(W \cup \{v\})] \\
&= 1 + \frac{1}{|C_u|} \left[\sum_{v \in C_{<u}} \mathbb{E}[\tau_{s^\star}(W \cup \{v\})] + \sum_{v \in C_u - C_{<u}} \mathbb{E}[\tau_{s^\star}(W)] \right] \\
&= 1 + \frac{1}{|C_u|} \left[\sum_{v \in C_{<u}} \Delta(v) + |C_u - C_{<u}| \cdot \mathbb{E}[\tau_{s^\star}(W)] \right] \qquad \text{(by the inductive hypothesis)}
\end{aligned}$$

Then since $|C_u - C_{<u}| = |C_u| - |C_{<u}|$, we can rearrange to obtain

$$\frac{|C_{<u}|\,\mathbb{E}[\tau_{s^\star}(W)]}{|C_u|} = \frac{1}{|C_u|}\left[|C_u| + \sum_{v \in C_{<u}} \Delta(v)\right]$$

so that

$$\mathbb{E}[\tau_{s^\star}(W)] = \frac{1}{|C_{<u}|}\left[|C_u| + \sum_{v \in C_{<u}} \Delta(v)\right] = \Delta(u) \ ,$$

as claimed. □

Next we prove that the expected value strategy has optimal expected cost.

Claim 8.6. *For any strategy s, $\mathbb{E}[\tau_s] \geq \mathbb{E}[\tau_{s^\star}]$.*

Proof. Claim 8.5 states that $\mathbb{E}[\tau_{s^\star}] = \Delta(\mathsf{rt}(T))$, so it suffices to show that $\mathbb{E}[\tau_s] \geq \Delta(\mathsf{rt}(T))$. We do so by proving the following statement: for every subtree W of T, it holds that $\mathbb{E}[\tau_s(W)] \geq \min_{v \in W} \Delta(v)$. (We can then take $W = \{\mathsf{rt}(T)\}$ to finish the proof because $\tau_s = \tau_s(\{\mathsf{rt}(T)\})$ and $\min_{v \in \{\mathsf{rt}(T)\}} \Delta(v) = \Delta(\mathsf{rt}(T))$.)

We proceed by a reverse induction on the subtree W. So first suppose that $W = V(T)$. In this case, if there is $v \in V(T)$ such that $\phi(v) = 1$ then $\mathbb{E}[\tau_s(V(T))] = 0 = \min_{v \in V(T)} \Delta(v)$; otherwise, $\mathbb{E}[\tau_s(V(T))] = \infty = \min_{v \in V(T)} \Delta(v)$. This establishes the base case; next we establish the inductive step. Let W be a subtree of T, and suppose that $\mathbb{E}[\tau_s(W')] = \min_{v \in W'} \Delta(v)$ for all $W' \subseteq T$ that contain, but do not equal, W; we now argue that $\mathbb{E}[\tau_s(W)] = \min_{v \in W} \Delta(v)$ as well. By invoking Equation 1 (and rearranging), we obtain that

$$\mathbb{E}[\tau_s(W)] = \frac{1}{|C_u - W|}\left[|C_u| + \sum_{v \in C_u - W} \mathbb{E}[\tau_s(W \cup \{v\})]\right] \ .$$

By the inductive hypothesis, the above is equal to

$$\frac{1}{|C_u - W|}\left[|C_u| + \sum_{v \in C_u - W} \min_{x \in W \cup \{v\}} \Delta(x)\right] = \frac{1}{|C_u - W|}\left[|C_u| + \sum_{v \in C_{<u} - W} \Delta(v) + |C_u - C_{<u} - W|\Delta(w)\right] \ ,$$

where $w := \arg\min_{v \in W} \Delta(v)$. Rearranging, this is equal to

$$\begin{aligned}
&\frac{|C_{<u} - W|}{|C_u - W|} \cdot \frac{1}{|C_{<u} - W|}\left[|C_u| + \sum_{v \in C_{<u} - W} \Delta(v)\right] + \left(1 - \frac{|C_{<u} - W|}{|C_u - W|}\right)\Delta(w) \\
&\geq \frac{|C_{<u} - W|}{|C_u - W|} \cdot \Delta(u) + \left(1 - \frac{|C_{<u} - W|}{|C_u - W|}\right)\Delta(w) \geq \Delta(w) \ ,
\end{aligned}$$

where the first inequality follows from the construction of $\Delta(u)$ and that $(C_{<u} - W) \subseteq C_u$, and the second follows since $\Delta(u) \geq \Delta(w)$ by the choice of w; this concludes the proof the claim. □

Acknowledgments

The authors thank Allan Borodin and Tyrone Strangway for their input on tree exploration games, and Ran Canetti and Nir Bitansky for helpful discussions. This project was funded in part by the ETH Foundation.

A Interactive oracle proofs characterize NEXP

We prove that interactive oracle proofs characterize the complexity class NEXP.

Lemma A.1. *For every relation $\mathscr{R}$ and polynomially-bounded $k\colon \{0,1\}^* \to \mathbb{N}$, $\mathscr{R}$ is in* NEXP *if and only if $\mathscr{R}$ has an interactive oracle proof system with round complexity k and constant soundness. Moreover, the same is true if we restrict our attention to only interactive oracle proof systems that are public-coin.*

Proof. Suppose that $\mathscr{R}$ is in NEXP. Then $\mathscr{R}$ has a PCP with polynomially-bounded randomness and query complexity, since $\mathsf{NEXP} = \mathsf{PCP}(\mathrm{poly}, \mathrm{poly})$ (see, e.g., [AB09, p. 164]). The PCP immediately gives a public-coin interactive oracle proof with one round and the same soundness: the IOP verifier sends an empty first message, the IOP prover answers with the PCP proof, and the IOP verifier runs the PCP verifier.

Conversely, suppose that $\mathscr{R}$ has an interactive oracle proof system (P, V) with round complexity k and constant soundness. We argue that $\mathscr{R}$ has a $2k$-prover interactive proof system, which implies that $\mathscr{R}$ is in NEXP (since $\mathsf{MIP} = \mathsf{NEXP}$). We think of the $2k$ provers as k pairs of provers such that the i-th pair is responsible for the i-th IOP prover message f_i. Following a simulation argument in [BFL90], the MIP verifier simulates the IOP verifier V as follows. First, run V until it halts, forwarding its output messages and answering its oracle queries as follows: when V outputs the i-th message m_i, send the message history $m_1, \ldots, m_i$ to each prover in the i-th pair; when V makes a query θ to the i-th oracle, send θ to the first prover in the i-th pair, and answer θ by returning that prover's answer. Next, once has V halted, pick a query θ of V at random and send it to the second prover of the pair whose first prover had answered θ. Finally, reject if V had rejected or the answer to the randomly-picked query is inconsistent with the previous one.

Define f_i to be the string such that $f_i(\theta)$ is the value returned by the second prover of the i-th pair when receiving the query θ (and no other query). If the first prover in the i-th pair ever answers a query θ with a value different from $f_i(\theta)$, then the MIP verifier rejects with probability at least $1/q$; else, if the first prover in the i-th pair answers every query consistently with f_i, then it behaves as the oracle f_i.

Overall, if some pair's first prover answers inconsistently, then the MIP verifier rejects with probability at least $1/q$; else, if every pair's first prover answers consistently with its second prover, then convincing the MIP verifier is equivalent to convincing the IOP verifier V. The soundness error is thus $1 - \Omega(1/q)$. Repeating the protocol polynomially many times (sequential repetition suffices) can reduce the soundness error to, e.g., less than $1/3$. □

B Soundness error reduction

We discuss approaches for soundness error reduction for IOPs (Appendix B.1) and NIROPs (Appendix B.2), and then discuss how these interact with the transformation that compiles one into the other (Appendix B.3).

B.1 Soundness error reduction for IOPs

Similarly to the case of interactive proof systems, one can define sequential and parallel repetition of interactive oracle proof systems, and both yield exact exponential decay in the soundness error. Below we provide additional details about each, including how other parameters, in addition to soundness, change.

- *Sequential repetition.* Given $r \in \mathbb{N}$, the r-fold sequential repetition of an interactive oracle proof system (P, V) is the interactive oracle proof system (P_r, V_r) that is obtained by running r independent instances of (P, V) in sequence (and where the new verifier V_r accepts if and only if all r invocations of V accept).

  If (P, V) has complexity $(k, s, p, q, t_{\mathrm{prv}}, t_{\mathrm{ver}})$ then (P_r, V_r) has complexity $(rk, s^r, rp, rq, rt_{\mathrm{prv}}, rt_{\mathrm{ver}})$.

- *Parallel repetition.* Given $r \in \mathbb{N}$, the r-fold parallel repetition of an interactive oracle proof system (P, V) is the interactive oracle proof system (P_r, V_r) that is obtained by running r independent instances of (P, V) in parallel (and where the new verifier V_r accepts if and only if all r invocations of V accept).

  If (P, V) has complexity $(k, s, p, q, t_{\mathrm{prv}}, t_{\mathrm{ver}})$ then (P_r, V_r) has complexity $(k, s^r, rp, rq, rt_{\mathrm{prv}}, rt_{\mathrm{ver}})$.

In the case of parallel repetition, the proof of the exact exponential decay of the soundness error is similar to that for interactive proofs. Namely, Goldreich's game-tree approach [Gol99, pp. 145–149] carries over, because it does not depend on whether the prover's messages are read in their entirety by the verifier (as in interactive proofs) or are only queried at some locations (as in interactive oracle proofs); in particular, the prover has complete information about the game and may act to maximize V's acceptance probability.

B.2 Soundness error reduction for NIROPs

Because non-interactive random-oracle proof systems are not an interactive proof system, the intuitive notions of sequential and parallel repetition coincide; so we talk about *repetition* (without any qualifying adjectives). We now explain what this means, and show that it yields exact exponential decay in the soundness error. We also discuss how other parameters, in addition to soundness, change.

Repeating a NIROP. Given $r \in \mathbb{N}$, the r-fold repetition of a non-interactive random-oracle proof system $(\mathbb{P}, \mathbb{V})$ is the non-interactive random-oracle proof system $(\mathbb{P}_r, \mathbb{V}_r)$ that, letting $\rho_i(x)$ denote $\rho(i \| x)$ for each $\rho \in \mathcal{U}(\lambda)$ and $i \in \{1, \ldots, r\}$, works as follows.

- $\mathbb{P}_r^{\rho}(\mathtt{x}, \mathtt{w})$ independently computes $\pi_i := \mathbb{P}^{\rho_i}(\mathtt{x}, \mathtt{w})$ for each $i \in \{1, \ldots, r\}$ and outputs $\pi := (\pi_1, \ldots, \pi_r)$.
- $\mathbb{V}_r^{\rho}(\mathtt{x}, \pi)$ runs $\mathbb{V}^{\rho_1}(\mathtt{x}, \pi_1), \ldots, \mathbb{V}^{\rho_r}(\mathtt{x}, \pi_r)$ and accepts if and only if all of the verifiers accept.

Changes in parameters. Repetition of a NIROP affects its parameters as follows.

- If $(\mathbb{P}, \mathbb{V})$ has complexity $(s, p, t_{\mathrm{prv}}, t_{\mathrm{ver}})$ then $(\mathbb{P}_r, \mathbb{V}_r)$ has complexity $(s^r, rp, rt_{\mathrm{prv}}, rt_{\mathrm{ver}})$.

Lemma B.1. *If $(\mathbb{P}, \mathbb{V})$ has soundness $s(\mathtt{x}, m, \lambda)$, then $(\mathbb{P}_r, \mathbb{V}_r)$ has soundness s' satisfying*

$$s(\mathtt{x}, m/r, \lambda)^r \leq s'(\mathtt{x}, m, \lambda) \leq s(\mathtt{x}, m, \lambda)^r \ .$$

Proof. We prove the first inequality. Let $\tilde{\mathbb{P}}$ be an m/r-query prover that convinces $\mathbb{V}$ to accept on $\mathtt{x} \notin \mathscr{L}(\mathscr{R})$ with probability at least $s(\mathtt{x}, m/r, \lambda)$. Consider the m-query prover $\tilde{\mathbb{P}}^{(r)}$ that runs r copies of $\tilde{\mathbb{P}}$ independently, with the i-th copy querying ρ_i; the probability that $\tilde{\mathbb{P}}^{(r)}$ convinces $\mathbb{V}_r$ is at least $s(\mathtt{x}, m/r, \lambda)^r$, by independence.

We now prove the second inequality. Let $\tilde{\mathbb{P}}$ be an m-query prover and $\mathtt{x} \notin \mathscr{L}(\mathscr{R})$. We show that $\epsilon \leq s(\mathtt{x}, m, \lambda)^r$, where

$$\epsilon := \Pr\left[\mathbb{V}_r^{\rho}(\mathtt{x}, \pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \tilde{\mathbb{P}}^{\rho} \end{array}\right] .$$

Rewriting the right-hand side, we obtain that

$$\epsilon = \Pr\left[\begin{array}{c} \mathbb{V}^{\rho_1}(\mathtt{x}, \pi_1) = 1 \\ \vdots \\ \mathbb{V}^{\rho_r}(\mathtt{x}, \pi_r) = 1 \end{array} \;\middle|\; \begin{array}{r} \rho_1, \ldots, \rho_r \leftarrow \mathcal{U}(\lambda) \\ (\pi_1, \cdot, \ldots, \cdot) \leftarrow \tilde{\mathbb{P}}^{\rho_1, \ldots, \rho_r} \\ \vdots \\ (\cdot, \ldots, \cdot, \pi_r) \leftarrow \tilde{\mathbb{P}}^{\rho_1, \ldots, \rho_r} \end{array}\right] .$$

The above implies that there exist m-query provers $\tilde{\mathbb{P}}_1, \ldots, \tilde{\mathbb{P}}_r$ such that

$$\epsilon \leq \Pr\left[\begin{array}{c} \mathbb{V}^{\rho_1}(\mathtt{x}, \pi_1) = 1 \\ \vdots \\ \mathbb{V}^{\rho_r}(\mathtt{x}, \pi_r) = 1 \end{array} \;\middle|\; \begin{array}{r} \rho_1, \ldots, \rho_r \leftarrow \mathcal{U}(\lambda) \\ \pi_1 \leftarrow \tilde{\mathbb{P}}_1^{\rho_1} \\ \vdots \\ \pi_r \leftarrow \tilde{\mathbb{P}}_r^{\rho_r} \end{array}\right] ,$$

because, for each $i \in \{1, \ldots, r\}$, we can hardcode the answers to $\tilde{\mathbb{P}}_i$ from ρ_j with $j \neq i$ in order to maximize the probability that $\mathbb{V}^{\rho_i}(\mathtt{x}, \pi_i)$ accepts. Then, by independence, the above probability is bounded from above by $s(\mathtt{x}, m, \lambda)^r$, and the lemma follows. □

Lemma B.2. *If $(\mathbb{P}, \mathbb{V})$ has z-statistical zero knowledge, then $(\mathbb{P}_r, \mathbb{V}_r)$ has rz-statistical zero knowledge.*

Proof. The proof is by a hybrid argument. We start by defining some useful notation: for functions $\mu_1, \ldots, \mu_r$ with disjoint domains $X_1, \ldots, X_r$, let $\bigcup_{i=1}^r \mu_i$ denote the function f with domain $\bigcup_{i=1}^r X_i$ where $f(x) = \mu_i(x)$ for every $i \in \{1, \ldots, r\}$ and $x \in X_i$. Let $\mathbb{S}_r$ be the "repetition" of the simulator $\mathbb{S}$ for $(\mathbb{P}, \mathbb{V})$ that shows its z-statistical zero knowledge. Given any distinguisher D_r, define

$$\epsilon := \left|\Pr\left[D_r^{\rho[\mu]}(\pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ (\pi, \mu) \leftarrow \mathbb{S}_r^{\rho}(\mathtt{x}) \end{array}\right] - \Pr\left[D_r^{\rho}(\pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \mathbb{P}_r^{\rho}(\mathtt{x}, \mathtt{w}) \end{array}\right]\right| .$$

Let A_i be the oracle algorithm that, on input $(\mathtt{x}, \mathtt{w})$, outputs $((\pi_1, \ldots, \pi_r), \bigcup_{j=1}^i \mu_j)$ where: (i) for $j = 1, \ldots, i$, $(\pi_j, \mu_j) \leftarrow \mathbb{S}_r^{\rho_j}(\mathtt{x})$; and (ii) for $j = i+1, \ldots, r$, $\pi_j \leftarrow \mathbb{P}_r^{\rho_j}(\mathtt{x}, \mathtt{w})$. Note that A_0 is equivalent to $\mathbb{P}_r$ and A_r is equivalent to $\mathbb{S}_r$ (when discarding $\mathtt{w}$). Then

$$\epsilon = \left|\sum_{i=1}^{r} \left(\Pr\left[D_r^{\rho[\mu]}(\pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ (\pi, \mu) \leftarrow A_i^{\rho}(\mathtt{x}, \mathtt{w}) \end{array}\right] - \Pr\left[D_r^{\rho[\mu]}(\pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ (\pi, \mu) \leftarrow A_{i-1}^{\rho}(\mathtt{x}, \mathtt{w}) \end{array}\right]\right)\right| .$$

Consider the distinguisher D that, on input π and with oracle access to ρ, works as follows: (1) choose $i \in \{1, \ldots, r\}$ uniformly at random; (2) for $j = 1, \ldots, i-1$, compute $(\pi_j, \mu_j) \leftarrow \mathbb{S}_r^{\rho_j}(\mathtt{x})$; (3) for $j = i+1, \ldots, r$, compute $\pi_j \leftarrow \mathbb{P}_r^{\rho_j}(\mathtt{x}, \mathtt{w})$, where $\mathtt{w}$ may be hard-coded into D; (4) setting $\mu := \bigcup_{j=1}^{i-1} \mu_j$,

simulate $D_r^{\rho'[\mu]}((\pi_1, \ldots, \pi_{i-1}, \pi, \pi_{i+1}, \ldots, \pi_r))$, where $\rho'(i\|x) := \rho(x)$ and other responses are drawn uniformly at random, and return its output. Note that if π is produced by $\mathbb{P}$, then the distribution corresponds to A_{i-1}; if π is produced by $\mathbb{S}$, then it corresponds to A_i instead. Therefore,

$$\begin{aligned} z(\mathtt{x}, \lambda) &\geq \left| \Pr\left[D^{\rho[\mu]}(\pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ (\pi, \mu) \leftarrow \mathbb{S}^{\rho}(\mathtt{x}) \end{array} \right] - \Pr\left[D^{\rho}(\pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \mathbb{P}^{\rho}(\mathtt{x}, \mathtt{w}) \end{array} \right] \right| \\ &= \left| \frac{1}{r} \sum_{i=1}^{r} \left(\Pr\left[D_r^{\rho[\mu]}(\pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ (\pi, \mu) \leftarrow A_i^{\rho}(\mathtt{x}, \mathtt{w}) \end{array} \right] - \Pr\left[D_r^{\rho[\mu]}(\pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ (\pi, \mu) \leftarrow A_{i-1}^{\rho}(\mathtt{x}, \mathtt{w}) \end{array} \right] \right) \right| \\ &= \epsilon / r \ . \end{aligned}$$

Rearranging, we obtain that $\epsilon \leq r z(\mathtt{x}, \lambda)$ and the lemma follows. □

B.3 Strategies for soundness error reduction

Parallel repetition of IOPs is superior to sequential repetition of IOPs because the latter increases the number of rounds but not does not offer advantages in return. A more interesting question is whether it is better to: (1) first reduce the soundness error of an IOP and then convert it to a NIROP via the transformation T, or (2) first transform the IOP to a NIROP and then reduce the soundness error.

The first method yields a soundness error of $\binom{m}{k(\mathtt{x})} s(\mathtt{x})^r + O(m^2 2^{-\lambda})$, while the second yields one of $(\binom{m}{k(\mathtt{x})} s(\mathtt{x}) + O(m^2 2^{-\lambda}))^r$. The first method is the only one that is effective when $\binom{m}{k(\mathtt{x})} s(\mathtt{x}) \gg 1$; indeed, the second method also amplifies the soundness loss incurred by the transformation, and hence is less effective for reducing soundness error. With respect to other complexity measures, both methods perform similarly. In sum, the first method is generally the preferable strategy for soundness error reduction.

C Zero knowledge with non-programmable random oracles

We provide a lemma showing that, for non-interactive random-oracle proof systems with zero knowledge in the non-programmable random oracle model, "simulation is as hard as decision". The lemma is a simple adaptation of [Pas03, Theorem 7], and implies that non-trivial languages do not have efficient simulators.

Lemma C.1. *Let $(\mathbb{P}, \mathbb{V})$ be a non-interactive random oracle-proof system for a relation $\mathscr{R}$ with soundness $s(\mathtt{x}, m, \lambda) \leq 1/3$ (for all sufficiently large $|\mathtt{x}|, m, \lambda$). If $(\mathbb{P}, \mathbb{V})$ has z-statistical zero knowledge with $z(\mathtt{x}, \lambda) \leq 1/3$ (for all sufficiently large $|\mathtt{x}|, \lambda$) in the non-programmable random oracle model with a simulator $\mathbb{S}$ that runs in time $t(n)$, then $\mathscr{R} \in \mathsf{BPTIME}(\mathrm{poly}(t(n)))$.*

Proof. Consider the probabilistic decider algorithm D that, given an instance $\mathtt{x}$, seeks to decide if $\mathtt{x}$ belongs to $\mathscr{L}(\mathscr{R})$ as follows: (i) choose a sufficiently large security parameter λ; (ii) simulate a random oracle $\rho \in \mathcal{U}(\lambda)$ in the straightforward way; (iii) compute $(\pi, \mu) := \mathbb{S}^{\rho}(\mathtt{x})$; (iv) compute and output $\mathbb{V}^{\rho}(\mathtt{x}, \pi)$. Note that $\rho[\mu] = \rho$ (since the random oracle is non-programmable) and that D runs in time $\mathrm{poly}(t(n))$.

We claim that D shows that $\mathscr{R}$ lies in $\mathsf{BPTIME}(\mathrm{poly}(t(n)))$. Fixing a sufficiently large instance $\mathtt{x}$, consider the following two cases.

- Case 1: $\mathtt{x} \in \mathscr{L}(\mathscr{R})$. We claim that $\Pr[D(\mathtt{x}) = 1] \geq 2/3$. Suppose, by way of contradiction, that $\Pr[D(\mathtt{x}) = 1] < 2/3$. By construction of D, we deduce that

$$\epsilon := \Pr\left[\mathbb{V}^{\rho}(\mathtt{x}, \pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ (\pi, \mu) \leftarrow \mathbb{S}^{\rho}(\mathtt{x}) \end{array}\right] < 2/3 \ .$$

  On the other hand, by the completeness of $(\mathbb{P}, \mathbb{V})$, we know that

$$\Pr\left[\mathbb{V}^{\rho}(\mathtt{x}, \pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \mathbb{P}^{\rho}(\mathtt{x}, \mathtt{w}) \end{array}\right] = 1 \ .$$

  Now let X and Y be the random variables that denote the output of $\mathbb{V}$ in the probability distributions of the above two probability statements, respectively. Then

$$\begin{aligned} \Delta(X; Y) &= \frac{1}{2}\left(|\Pr[X = 0] - \Pr[Y = 0]| + |\Pr[X = 1] - \Pr[Y = 1]|\right) \\ &= \frac{1}{2}\left(|(1 - \epsilon) - 0| + |\epsilon - 1|\right) = 1 - \epsilon > 1/3 \geq z(\mathtt{x}, \lambda) \ , \end{aligned}$$

  which contradicts the zero knowledge property.

- Case 2: $\mathtt{x} \notin \mathscr{L}(\mathscr{R})$. We claim that $\Pr[D(\mathtt{x}) = 1] \leq 1/3$. By the soundness of $(\mathbb{P}, \mathbb{V})$, for every m-query $\tilde{\mathbb{P}}$, and $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathbb{V}^{\rho}(\mathtt{x}, \pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \tilde{\mathbb{P}}^{\rho} \end{array}\right] \leq s(\mathtt{x}, m, \lambda) \ .$$

  In particular, for sufficiently large m, this holds for the $t(|\mathtt{x}|)$-query prover $\tilde{\mathbb{P}}_*$ that computes $(\pi, \mu) := \mathbb{S}^{\rho}(\mathtt{x})$ and outputs π. Thus, by construction of D,

$$\begin{aligned} \Pr[D(\mathtt{x}) = 1] &= \Pr\left[\mathbb{V}^{\rho}(\mathtt{x}, \pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ (\pi, \mu) \leftarrow \mathbb{S}^{\rho}(\mathtt{x}) \end{array}\right] \\ &= \Pr\left[\mathbb{V}^{\rho}(\mathtt{x}, \pi) = 1 \;\middle|\; \begin{array}{r} \rho \leftarrow \mathcal{U}(\lambda) \\ \pi \leftarrow \tilde{\mathbb{P}}_*^{\rho} \end{array}\right] \leq s(\mathtt{x}, m, \lambda) \leq 1/3. \end{aligned}$$

□

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in FOCS '92.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in FOCS '92.

[Bab85] László Babai. Trading group theory for randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, STOC '85, pages 421–429, 1985.

[Bab90] Laszlo Babai. E-mail and the Unexpected Power of Interaction. Technical report, University of Chicago, Chicago, IL, USA, 1990.

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *Proceedings of the 23rd Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT '04, pages 171–188, 2004.

[BC86] G. Brassard and Claude Crepeau. Non-transitive transfer of confidence: A perfect zero-knowledge interactive protocol for SAT and beyond. In *27th Annual Symposium on Foundations of Computer Science, 1986*, pages 188–195, October 1986.

[BC12] Nir Bitansky and Alessandro Chiesa. Succinct arguments from multi-prover interactive proofs and their efficiency benefits. In *Proceedings of the 32nd Annual International Cryptology Conference*, CRYPTO '12, pages 255–272, 2012.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 326–349, 2012.

[BCG+16] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Short interactive oracle proofs with constant query complexity, via composition and sumcheck, 2016. Crypto ePrint 2016/324.

[BCGV16] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, and Madars Virza. Quasilinear-size zero knowledge from linear-algebraic PCPs. In *Proceedings of the 13th Theory of Cryptography Conference*, TCC '16, pages 33–64, 2016.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *Proceedings of the 10th Theory of Cryptography Conference*, TCC '13, pages 315–333, 2013.

[BD16] Allison Bishop and Yevgeniy Dodis. Interactive coding for interactive proofs. In *Proceedings of the 13th Theory of Cryptography Conference*, TCC '16, pages 352–366, 2016.

[BDG+13] Nir Bitansky, Dana Dachman-Soled, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, Adriana López-Alt, and Daniel Wichs. Why "Fiat-Shamir for proofs" lacks a proof. In *Proceedings of the 10th Theory of Cryptography Conference*, TCC '13, pages 182–201, 2013.

[BFL90] László Babai, Lance Fortnow, and Carsten Lund. Nondeterministic exponential time has two-prover interactive protocols. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, SFCS '90, pages 16–25, 1990.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, STOC '91, pages 21–32, 1991.

[BG93] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 390–420, 1993.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM Journal on Computing*, 38(5):1661–1694, 2008. Preliminary version appeared in CCC '02.

[BGGL01] Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resettably-sound zero-knowledge and its applications. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 116–125, 2001.

[BGH+04] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, STOC '04, pages 1–10, 2004.

[BGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: how to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, STOC '88, pages 113–131, 1988.

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In *Proceedings of the 47th ACM Symposium on the Theory of Computing*, STOC '15, pages 439–448, 2015.

[BHZ87] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):127–132, 1987.

[BKK+13] Eli Ben-Sasson, Yohay Kaplan, Swastik Kopparty, Or Meir, and Henning Stichtenoth. Constant rate PCPs for Circuit-SAT with sublinear query complexity. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '13, pages 320–329, 2013.

[BR93] Mihir Bellare and Phillip Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[BS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008. Preliminary version appeared in STOC '05.

[CCC+15] Yu-Chi Chen, Sherman S. M. Chow, Kai-Min Chung, Russell W. F. Lai, Wei-Kai Lin, and Hong-Sheng Zhou. Computation-trace indistinguishability obfuscation and its applications. Cryptology ePrint Archive, Report 2015/406, 2015.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004.

[CH15] Ran Canetti and Justin Holmgren. Succinct garbled RAM. Cryptology ePrint Archive, Report 2015/388, 2015.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In *Proceedings of the 47th ACM Symposium on the Theory of Computing*, STOC '15, pages 429–437, 2015.

[Dam89] Ivan Damgård. A design principle for hash functions. In *Proceedings of the 9th Annual International Cryptology Conference*, CRYPTO '89, pages 416–427, 1989.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *Proceedings of the 9th Theory of Cryptography Conference*, TCC '12, pages 54–74, 2012.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, FOCS '90, pages 308–317, 1990.

[FRS88] Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. In *Theoretical Computer Science*, pages 156–161, 1988.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings of the 6th Annual International Cryptology Conference*, CRYPTO '86, pages 186–194, 1986.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *Proceedings of the 32nd Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT '13, pages 626–645, 2013.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Information Processing Letters*, 67(4):205–214, 1998.

[GHRW14] Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Outsourcing private RAM computation. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '14, pages 404–413, 2014.

[GIMS10] Vipul Goyal, Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. Interactive locking, zero-knowledge PCPs, and unconditional cryptography. In *Proceedings of the 30th Annual Conference on Advances in Cryptology*, CRYPTO'10, pages 173–190, 2010.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, pages 102–113, 2003.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for Muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC '08, pages 113–122, 2008.

[GLR11] Shafi Goldwasser, Huijia Lin, and Aviad Rubinstein. Delegation of computation without rejection problem from designated verifier CS-proofs. Cryptology ePrint Archive, Report 2011/456, 2011.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Preliminary version appeared in STOC '85.

[Gol99] Oded Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, volume 17 of *Algorithms and Combinatorics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security*, ASIACRYPT '10, pages 321–340, 2010.

[GS86] S Goldwasser and M Sipser. Private Coins Versus Public Coins in Interactive Proof Systems. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 59–68, New York, NY, USA, 1986. ACM.

[GVW02] Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1/2):1–53, 2002.

[Hol07] Thomas Holenstein. Parallel repetition: simplifications and the no-signaling case. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, STOC '07, pages 411–419, 2007.

[HS00] Prahladh Harsha and Madhu Sudan. Small PCPs with low query complexity. *Computational Complexity*, 9(3–4):157–201, Dec 2000. Preliminary version in STACS '91.

[IKM09] Tsuyoshi Ito, Hirotada Kobayashi, and Keiji Matsumoto. Oracularization and two-prover one-round interactive proofs against nonlocal strategies. In *Proceedings of the 24th IEEE Annual Conference on Computational Complexity*, CCC '09, pages 217–228, 2009.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short PCPs. In *Proceedings of the Twenty-Second Annual IEEE Conference on Computational Complexity*, CCC '07, pages 278–291, 2007.

[IMS12] Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. On efficient zero-knowledge PCPs. In *Proceedings of the 9th International Conference on Theory of Cryptography*, TCC '12, pages 151–168, 2012.

[IMSX15] Yuval Ishai, Mohammad Mahmoody, Amit Sahai, and David Xiao. On zero-knowledge PCPs: Limitations, simplifications, and applications, 2015. Available at `http://www.cs.virginia.edu/~mohammad/files/papers/ZKPCPs-Full.pdf`.

[Ito10] Tsuyoshi Ito. Polynomial-space approximation of no-signaling provers. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming*, ICALP '10, pages 140–151, 2010.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, STOC '92, pages 723–732, 1992.

[KP15] Yael Tauman Kalai and Omer Paneth. Delegating RAM computations. Cryptology ePrint Archive, Report 2015/957, 2015.

[KR08] Yael Kalai and Ran Raz. Interactive PCP. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, ICALP '08, pages 536–547, 2008.

[KR09] Yael Tauman Kalai and Ran Raz. Probabilistically checkable arguments. In *Proceedings of the 29th Annual International Cryptology Conference*, CRYPTO '09, pages 143–159, 2009.

[KRR13] Yael Kalai, Ran Raz, and Ron Rothblum. Delegation for bounded space. In *Proceedings of the 45th ACM Symposium on the Theory of Computing*, STOC '13, pages 565–574, 2013.

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 485–494, 2014.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992.

[Lin15] Yehuda Lindell. An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle. In *Proceedings of the 12th Theory of Cryptography Conference*, TCC '15, pages 93–109, 2015.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *Proceedings of the 9th Theory of Cryptography Conference on Theory of Cryptography*, TCC '12, pages 169–189, 2012.

[Mer89a] Ralph C. Merkle. A certified digital signature. In *Proceedings of the 9th Annual International Cryptology Conference*, CRYPTO '89, pages 218–238, 1989.

[Mer89b] Ralph C. Merkle. One way hash functions and DES. In *Proceedings of the 9th Annual International Cryptology Conference*, CRYPTO '89, pages 428–446, 1989.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000. Preliminary version appeared in FOCS '94.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Proceedings of the 22nd Annual International Cryptology Conference*, CRYPTO '02, pages 111–126, 2002.

[Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In *Proceedings of the 23rd Annual International Cryptology Conference*, CRYPTO '03, pages 316–337, 2003.

[PGHR13] Brian Parno, Craig Gentry, Jon Howell, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *Proceedings of the 34th IEEE Symposium on Security and Privacy*, Oakland '13, pages 238–252, 2013.

[PR14] Omer Paneth and Guy N. Rothblum. Publicly verifiable non-interactive arguments for delegating computation. Cryptology ePrint Archive, Report 2014/981, 2014.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Proceedings of the 14th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT '96, pages 387–398, 1996.

[PS05] Rafael Pass and Abhi Shelat. Unconditional characterizations of non-interactive zero-knowledge. In *Proceedings of the 25th Annual International Cryptology Conference*, CRYPTO '05, pages 118–134, 2005.

[PSSV07] Aduri Pavan, Alan L. Selman, Samik Sengupta, and Vinodchandranm N. V. Polylogarithmic-round interactive proofs for coNP collapse the exponential hierarchy. *Theoretical Computer Science*, 385(1-3):167–178, 2007.

[Raz95] Ran Raz. A parallel repetition theorem. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, STOC '95, pages 447–456, 1995.

[RRR16] Omer Reingold, Ron Rothblum, and Guy Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th ACM Symposium on the Theory of Computing*, STOC '16, pages ???–???, 2016.

[SBV+13] Srinath Setty, Benjamin Braun, Victor Vu, Andrew J. Blumberg, Bryan Parno, and Michael Walfish. Resolving the conflict between generality and plausibility in verified computation. In *Proceedings of the 8th EuoroSys Conference*, EuroSys '13, pages 71–84, 2013.

[SBW11] Srinath Setty, Andrew J. Blumberg, and Michael Walfish. Toward practical and unconditional verification of remote computations. In *Proceedings of the 13th USENIX Conference on Hot Topics in Operating Systems*, HotOS '11, pages 29–29, 2011.

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[SMBW12] Srinath Setty, Michael McPherson, Andrew J. Blumberg, and Michael Walfish. Making argument systems for outsourced computation practical (sometimes). In *Proceedings of the 2012 Network and Distributed System Security Symposium*, NDSS '12, 2012.

[SVP+12] Srinath Setty, Victor Vu, Nikhil Panpalia, Benjamin Braun, Andrew J. Blumberg, and Michael Walfish. Taking proof-based verified computation a few steps closer to practicality. In *Proceedings of the 21st USENIX Security Symposium*, Security '12, pages 253–268, 2012.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *Proceedings of the 5th Theory of Cryptography Conference*, TCC '08, pages 1–18, 2008.

[Wee09] Hoeteck Wee. Zero Knowledge in the Random Oracle Model, Revisited. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, number 5912 in Lecture Notes in Computer Science, pages 417–434. Springer Berlin Heidelberg, 2009.