Sometimes You Can't Distribute Random-Oracle-Based Proofs

Jack Doerner
j@ckdoerner.net
Technion

Yashvanth Kondi
yash@ykondi.net
Aarhus University

Leah Namisa Rosenbloom
leah_rosenbloom@brown.edu
Brown University

September 14, 2023

Abstract

We investigate the conditions under which straight-line extractable NIZKs in the random oracle model (i.e. without a CRS) permit multiparty realizations that are black-box in the same random oracle. We show that even in the semi-honest setting, any MPC protocol to compute such a NIZK cannot make black-box use of the random oracle or a hash function instantiating it if security against all-but-one corruptions is desired, unless the size of the NIZK grows with the number of parties. This presents a fundamental barrier to constructing efficient protocols to securely distribute the computation of NIZKs (and signatures) based on MPC-in-the-head, PCPs/IOPs, and sigma protocols compiled with transformations due to Fischlin, Pass, or Unruh.

When the adversary is restricted to corrupt only a constant fraction of parties, we give a positive result by means of a tailored construction, which demonstrates that our impossibility does not extend to weaker corruptions models in general.

Contents

1 Introduction

Zero-knowledge proofs of knowledge (ZKPoKs) are widely used in cryptographic protocols as a mechanism to enforce honest behavior. Non-interactive proofs of knowledge (NIZKPoKs, or simply NIZKs) in particular enable a multitude of interesting applications, and underlie the design of several signature schemes [BG90]. The *proof of knowledge* property informally guarantees that a prover P must *actually know* a witness to the statement if it succeeds in generating an accepting proof. This is a far more powerful notion than *soundness*, which only guarantees that a witness *exists*. Formally, proof of knowledge is captured by the existence of an extractor algorithm E, which is able to produce a witness to the statement when given access to P. However, this access cannot always be unrestricted. For example, in order to achieve provably secure composition in a larger protocol context, it vastly simplifies matters if E makes *black box* use of P in a *straight-line* fashion, i.e. without rewinding it [Can01]. This example is critical, since NIZKs are common building blocks for other protocols.

One common method by which such black-box straight-line extraction can be achieved is to allow the extractor to access (or even sample) the trapdoor for some trusted setup that is used by the protocol. This also known as the Common/Structured Reference String (CRS/SRS) model. Apart from the immediate disadvantage of requiring a source for trusted reference strings in practice, NIZK techniques in the SRS model typically make use of expensive structured public-key cryptography, and are not known to permit efficient proofs for simple statements, such as proof of knowledge of discrete logarithm.

The alternative is to leverage an ideal object. Typically the ideal object is a random oracle. Roughly, a straight-line extractor in the random oracle model (ROM) reads the random oracle queries made by a prover during the production of a proof and deduces a witness for the statement. This notion of straight-line extraction in the ROM was formalized by Pass [Pas03], and has proven to be a popular method to construct efficient NIZKs that remain secure under concurrent composition [CJS14, CDG$^+$18]. The security of NIZKs for general statements based on PCPs/IOPs [BCS16, BCR$^+$19, BFH$^+$20] and MPC-in-the-head [IKOS07, GMO16, AHIV17, KKW18] is also proven via straight-line extraction in the ROM, and NIZKs of simple algebraic statements are efficient enough to use in practice [HMPs14, LN18, Lin22].

Why the Random Oracle Model? The random oracle model has been in use for three decades as an analytical tool [BR93] and, though idealized, it is arguably a relatively conservative assumption. The instantiation of random oracles in practice has been the subject of much scrutiny, and modern implementations typically employ complex and carefully salted hash functions such as the SHA family. Such hash functions already offer a lower Boolean circuit complexity than structured public key cryptography, and they are often further optimized through hardware implementations. Evaluating SHA-2 on commodity hardware today costs less than one microsecond. More recently, the plausible post-quantum resilience of hash functions has translated to plausible post-quantum security of NIZKs in the (quantum) ROM (and signatures built upon them [CDG$^+$17]), with straight-line extraction forming the basis for such analyses [Unr15]. In a nutshell, the random oracle serves as a versatile analytical tool, and is quite cheap to invoke in practice.

While the construction of efficient NIZKs in the ROM and their use in distributed protocols are well studied, the topic of designing protocols to distribute the computation of such NIZKs themselves is not.

Threshold Cryptographic Proofs. Many modern cryptographic applications employ decentralization as a design principle, so that the security of sensitive data is not vulnerable to single points of failure. One example of this methodology is the increasing adoption of threshold signatures to decentralize the management of cryptographic keys (see, for example, [MPs19, DOK$^+$20]). Given that signature schemes are commonly derived from NIZKs, threshold signing in many cases reduces to the task of designing Multiparty Computation (MPC) protocols to securely compute a NIZK where the parties hold secret shares of a witness. Distributed Schnorr signing [Lin22] is a classic example of this principle—Schnorr signatures are essentially NIZKs in the ROM that prove knowledge of the discrete logarithm of the public key, and so distributed signing for Schnorr corresponds to an MPC to compute Schnorr's NIZK (i.e. the Fiat-Shamir transform applied to the identification protocol).

Distributing Schnorr—Why is it efficient? It is widely regarded that Schnorr's NIZK is "MPC-friendly", as it permits an elegant MPC protocol to distribute its computation. Roughly, a Schnorr signature is an affine function of the secret key x and a random nonce r, i.e. $\mathsf{Sign}(x, r) \mapsto x \cdot H(r \cdot G) + r$. The equation Sign is regarded as easy to compute with an MPC as most secret sharing schemes permit linear operations for free, meaning that if secret shares of x and r are available, given that $r \cdot G$ is public, Sign itself can be computed for free. Notice that because $r \cdot G$ is public, $H(r \cdot G)$ can be evaluated *locally* by the parties, i.e. the MPC protocol makes only oracle use of H. Other practical threshold signing schemes, such as those for ECDSA [DKLs19, CGG$^+$20, LN18] and BLS [BLS01] also have this property—that the signature algorithm makes use of a hash function H, and the MPC protocol also only makes oracle use of H. We formalize this notion as as that of an *oracle-respecting* distributed protocol.

Oracle-Respecting Protocols. Informally, an oracle-respecting distributed protocol for a NIZK in the ROM is an MPC protocol that distributes the computation of the NIZK *without* replacing the oracle with a concrete function, by allowing the protocol's participants ideal access to the same oracle.

MPC protocols that are not oracle-respecting are unsatisfying theoretically, because the security proof of the NIZK relies upon H being a random oracle, and it is unclear whether the NIZK retains a meaningful security proof the parties (and therefore the adversary) use a concrete function instead. They are also unsatisfying in practice, because the only general technique for using the code of a hash function (like SHA, for example) is to evaluate it using an MPC protocol. This is expensive, complicated, and several orders of magnitude slower than a local evaluation of H. Consequently, threshold signature schemes that employ MPC protocols that are not oracle-respecting are bound to be prohibitively expensive for many common use cases.

Straight-line extraction and oracle-respecting distribution. Interestingly, no existing straight-line extractable NIZK in the ROM is known to permit an oracle-respecting distributed protocol. The only practical technique to distribute the computation of NIZKs for algebraic languages is to simply concatenate proofs produced by each prover (see Cohen et al. [CDKs22] for instance), the only IOP that supports distributed provers incurs an overhead proportionate to the number of provers [OB22], and ironically there is no oracle-respecting MPC to securely compute MPC-in-the-head proofs.

Of course it is possible to design a simple such NIZK from scratch, for e.g. a proof of the format $\boldsymbol{\rho} = \{\rho_1, \rho_2\}$ where ρ_i is generated with witness share w_i such that $w_0 + w_1 = w$ forms the original witness. This way, parties $\mathcal{P}_1$ and $\mathcal{P}_2$ may be given w_1 and w_2 respectively, and jointly compute $\boldsymbol{\rho}$ with a simple oracle-respecting distributed protocol.

Besides appearing artificial, the above scheme embodies an interesting principle: the NIZK itself is designed with the distributed protocol in mind. The focus of this paper is to ask whether this is inherent, informally:

> Under what conditions do oracle-respecting distributed protocols exist for computing straight-line extractable NIZKs in the ROM?

In particular, we wish to know whether straight-line extractable NIZKs in the ROM can be agnostic to their oracle-respecting distributed protocols. In order to capture this intuition formally, we require the output of an oracle-respecting distributed protocol to be verifiable by the same verifier as the original single party NIZK. This requirement is already implicit in standard definitions of threshold signatures, and captures a number of useful properties:

- The output of the distributed protocol serves as a drop-in replacement for any usage of the original NIZK. In other words, the original verifier is convinced.
- The size of the NIZK does not grow with the number of parties involved in its distributed computation.
- Access structures and the identities of parties involved in the protocol are concealed from entities external to the system.

In this work, we restrict ourselves to semi-honest MPC, and consider two corruption models: all-but-one corruption, and constant-fraction corruption. In the all-but-one case we uncover an inherent barrier that explains the lack of constructions, and precludes oracle-based distributed protocols for a large class of NIZKs outright. Our barrier has the following intuitive form: for NIZKs meeting certain common criteria, there exists some number n such that if the computation of the NIZK is distributed among n or more parties in an oracle-respecting way, then security against $n-1$ corruptions cannot be achieved. The number n is determined by the number of queries that the NIZK verifier must make, and thus to the verification complexity and the size of the proof string. Since this barrier exists even when the corruptions are semi-honest, it also holds for any *stronger* adversary. In the constant-fraction case, we show that the techniques we use to prove impossibility do not extend in general by constructing a NIZKPoK for the discrete logarithm relation that can be distributed in an oracle-respecting fashion amongst arbitrarily many parties, without increasing the proof size, assuming that the discrete logarithm assumption holds.

1.1 Our Results

We formalize the notion of an oracle-respecting distributed (ORD) protocol in Definition 2.4. Intuitively, a protocol π^H *ORD-computes* a prover P^H if it convinces the same verifier as P^H. This is a much weaker notion than, for example, producing indistinguishable output.

All-but-One ($n-1$ Corruptions) Setting. We first show that any n-party protocol that ORD-computes a prover P from a certain (common) class of provers can not hide the witness from an adversary that passively corrupt $n-1$ parties, where $n-1$ is an upper bound on the number of queries that the verifier V makes in checking a proof.

Intuitively, since the verifier checks only $n-1$ queries and there are n parties, there must be at least one party whose query "does not matter" for the proof. In particular, there is a noticeable probability p that if one party's queries to H were simply omitted, the outcome would still be an accepting proof. By straight-line extraction, this means that the queries made to H by the remaining $n-1$ parties (i.e. the ones that were actually used in the production of the proof) must yield a witness when given to the extractor E. This suggests a simple attack on π: corrupt $n-1$ parties at random, collect the queries to H that those parties make, and feed the queries to E.

This is essentially the template followed by our proof, however we encountered a very subtle technical hurdle that necessitated prefacing our theorem statement with a caveat, which we will discuss below.

Theorem 1.1. *(Informal) If P is the prover of a NIZK in the ROM, and there is an extractor E that observes P's queries to H and outputs a witness with almost the same probability that P outputs an accepting proof, and the NIZK verifier V makes at most $n-1$ queries to H, then any n party protocol π that* ORD*-computes P will leak the witness to an adversary who corrupts $n-1$ parties with noticeable probability.*

The one caveat for the above theorem is that E must not depend on the *responses* to P's queries to H. This is not an issue for schemes where a constant number of well-chosen queries made by P suffice for extraction—such as NIZKs based on Sigma protocols, including MPC-in-the-head and algebraic languages—as E can simply guess which queries to use without checking their outputs. For succinct proof systems like IOPs where the extractor must necessarily read a large number of queries, this caveat does make a difference in terms of the applicability of our theorem (though whether our attack strategy also fails is another question). Regardless, schemes excluded from our first theorem are covered by an alternate theorem:

Theorem 1.2. *(Informal) If P is the prover of a straight-line extractable NIZK in the ROM, for which the NIZK verifier V makes at most $n-1$ queries to H, and π is an n-party protocol that ORD-computes P in which any given query to H can be traced back to the first party that made it (see Def. 2.5), then there is an adversary who corrupts $n-1$ parties and learns the witness with noticeable probability.*

The above theorem essentially rules out any protocol strategy in which it is possible for the adversary to determine whether a given query was made by the one honest party *before* a corrupt party was able to make it. Intuitively, this is because the adversary must be able to filter out queries that were first made by an honest party (and therefore "omitted") from the set that it feeds to the extractor. To our knowledge, there are no MPC protocol strategies that do not have this property, and therefore taking advantage of this caveat (if it is possible) would require a fundamentally new protocol design strategy.

In prior works, the distinction between using the list of queried values for extraction and giving the extractor access to H has not been treated as a matter of consequence. As we will show, several prior works give the extractor access to H unnecessarily. Our work shows that there may be a meaningful difference between the two notions of extraction.

The formal versions of these theorems are given as theorems 3.16 and 3.20.

Constant Fraction ($c \cdot n$ Corruptions) Setting. We show that the impossibility from the $n-1$ case cannot be extended in general to the next strongest corruption setting, wherein a constant fraction c of parties are corrupted, for a positive constant $c < 1$. We prove this by construction: first, we propose a NIZK with proof strings of the form $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_\lambda)$, where λ is the security parameter, and then we introduce a committee-based protocol that ORD-computes our NIZK. The protocol elects a randomly committee of size λ and provides each member of the committee with an additive share of the witness. The j^{th} committee member computes and broadcasts a sub-proof ρ_j of knowledge of its share of the witness, and the final proof string is simply a concatenation of the sub-proofs. As the witness is additively shared amongst the committee and the sub-proofs all have zero-knowledge individually, the adversary can only recover information about the witness if the entire committee is corrupt, which happens with probability $1/\mathsf{exp}(\lambda)$. We formalize the security notion achieved by our protocol as a multi-prover variant of zero-knowledge, and show that if the protocol is *also* witness-revealing, then an adversary who learns the witness can be used in a black-box fashion to break the discrete logarithm assumption. Furthermore, our simple scheme is more efficient than a trivial concatenation of proofs so long as the total number of parties n is slightly larger than λ.

Theorem 1.3. *(Informal) If there exists a group $\mathbb{G}$ in which the discrete logarithm assumption holds, then there exists a NIZKPoK of discrete logarithm in $\mathbb{G}$ and an n-party protocol π that ORD-computes the NIZK prover P, such that for any positive rational $c < 1$ and any n such that $n > \lambda/(1-c)$, the NIZK verifier V makes at most $n-1$ queries to H, and the probability that π leaks the witness in the presence of $c \cdot n$ corrupt parties is negligible in λ.*

The formal version of this theorem is given as theorem 4.9.

1.2 Implications of Our Results

To our knowledge, no prior work gives a formal treatment of bounds and limitations on the distribution of straight-line extractable proofs in the ROM. These limitations have wide-ranging implications in practice as well as in theory. We give a few examples below of tasks that must inherently pay the price of making non-black-box use of hash functions, or else tolerate proof or signature sizes that grow with the number of provers or signers:

- Threshold versions of MPC-in-the-head-based signature schemes and multi-prover versions of MPC-in-the-head NIZKs, even if they use the Fiat-Shamir transform, so long as random-oracle-based commitments are used. Such schemes typically require the prover to commit to the views of many virtual parties, then query the random oracle on the commitment strings (i.e. they apply the Fiat-Shamir transform) to generate a challenge that determines which commitments to open. The witness is extracted via the commitments, not by rewinding and reprogramming the challenge, and because the commitment strings are included in the proof, there usually exists an efficient straight-line extractor requires only the query values (and not access to the random oracle itself), which means that theorem 3.16 applies.

  As an example, consider the Picnic signature scheme [CDG+17], which was considered (but ultimately not chosen) for standardization by NIST. Picnic makes extensive use of the SHA-3 hash function, which has a boolean circuit complexity of several tens of thousands of gates, making it expensive to evaluate in a non-black-box fashion within generic MPC.

- Threshold versions of any post-quantum signature scheme based on Unruh's transformation [Unr15], to which theorem 3.16 applies. The post-quantum variant of the Picnic scheme [CDG+17] is an example.

- Multi-prover versions of NIZKs that are based on compiling sigma protocols for algebraic statements (such as the discrete logarithm relation) to non-interactive proofs via the transformations due to Pass [Pas03], Fischlin [Fis05], Unruh [Unr15], or Kondi and shelat [Ks22], to which theorem 3.16 applies. These are common building blocks for larger MPC protocols (particularly in the UC model [Can01]), and our work suggests that their costs cannot be significantly reduced under current well-known techniques.

- Protocols that securely compute other random-oracle-based straight-line extractable NIZKs amongst groups parties that hold secret shares of the witness. This class notably includes many PCP/IOP based NIZKs [BCS16, BCR+19, BFH+20]. In this case, theorem 3.16 may not apply, but theorem 3.20 does, which implies that any MPC protocol that securely distributes such NIZKs in the presence of $n-1$ corruptions must have the property that the adversary cannot guess with noticeable probability which of its own queries were previously made by an honest party. This rather unnatural property is not satisfied by any existing technique for secure computation, to our knowledge.

We wish to draw attention to one hazard in particular: the standardization of impacted signature schemes without accommodations for our result. During standardization, it is likely that both the hash function used to realize a random oracle and the parameterization of the scheme (in particular, the number of verifier queries) would become fixed. This could effectively preclude efficient threshold signing by forcing non-black-box use of a hash with a large circuit.

As our positive result indicates, it may be possible to design custom schemes that support oracle respecting distributed computation for lower corruption thresholds. At present, the authors do not know of any pre-existing NIZK that supports oracle respecting distributed computation in such settings.

1.3 Actually, It's Much Worse than That

The results presented in this work make minimal assumptions about the internal structure of the NIZK to be distributed, but we observe that many common NIZKs have a structure that is pathological in the context of ORD-computation. In particular, consider the case of a sigma protocol to which one of the common straight-line extraction compilers [Pas03, Fis05, Unr15, Ks22] has been applied. This yields a NIZK with the following structure: the prover makes queries on a set of sigma protocol transcripts that share a statement and first-round message (i.e. "commitment"), but differ in their second and third-round messages (i.e. their "challenge" and "response"). Of the queries in this set, at most one is randomly chosen to be included in the proof string and checked by the verifier. Because sigma protocols are 2-special-sound, even *one* additional query from the set is enough to extract the witness. This is catastrophic in the context of ORD-computation: if queries of this format are allocated among multiple provers, and even *one* of the provers is corrupt, then the witness must leak with noticeable probability. Therefore, the only option to distribute the computation of such a NIZK (without modifying it) is to realize the oracle heuristically via some concrete function and evaluate the oracle queries in a non-black-box fashion using MPC. We suggest that a more viable approach is to modify the query format to avoid this problem; we leave this direction open for future work.

1.4 Organization

We begin by formally defining the notions relevant to the results in this work in section 2, following which we give our main negative results for the all-but-one setting in section 3, and our positive result for the constant-fraction setting in section 4. Finally, we review related works and discuss their relevance to our results in section 5.

2 Definitions

Notation. We adhere to standard notational conventions for proofs and multiparty computation. Throughout this paper, we use λ to denote the security parameter, x to denote a statement from the domain $\mathbb{X}$, and w to denote a witness from the domain $\mathbb{W}$. We say that a pair (x, w) is in the NP-language L if $\mathcal{R}_L(x, w) = 1$, where $\mathcal{R}_L$ is the polynomial-time relation that defines L. A prover is denoted P, a verifier V, an extractor E, and a random oracle H. Protocols are denoted π and the i^{th} party who participates in a protocol is denoted $\mathcal{P}_i$. Proof strings are denoted ρ and values on which the random oracle is queried are denoted by Q, whereas (ordered) vectors of queries are denoted by $\mathbf{Q}$. When a query appears with a subscript, it does *not* indicate a position in any particular vector, but instead the position in the original order in which the queries were made by a prover. We use $\leftarrow$ to indicate sampling from a distribution, $:=$ to indicate assignment, and $=$ to indicate equality testing.

Proofs and Extractability. We begin with a standard definition for non-interactive proofs in the random-oracle model. Our definition allows the prover and verifier access to a *non-programmable* random oracle, but syntactically forbids the use of a common reference string (CRS) or oracle programming.

Definition 2.1. Non-Interactive Proof in the ROM

Let L be a language in NP where membership of an instance $x \in \mathbb{X}$ can be verified by a witness $w \in \mathbb{W}$. A pair of algorithms $(\mathsf{P}^H, \mathsf{V}^H)$, both with access to a random oracle H, constitute a non-interactive proof system for L if they meet the following conditions:

1. Efficiency: P is an probabilistic expected-polynomial-time algorithm and V is a deterministic polynomial time algorithm.

2. Completeness: For $(x, w) \in L$, the prover can convince the verifier with overwhelming probability that $(x, w) \in L$ by means of a single string. Formally, for all $(x, w) \in L$,

$$\Pr\left[\mathsf{V}^H(1^\lambda, x, \rho) = 1 : \rho \leftarrow \mathsf{P}^H(1^\lambda, x, w)\right] \in \Omega(1 - \mathrm{negl}(\lambda))$$

where the probability is over the coins of P and H.

3. Computational Soundness: For all invalid statements $x \in \mathbb{X}$ such that $(x, w) \notin L$ for any $w \in \mathbb{W}$, no PPT algorithm $\mathcal{A}^H$ with access to the random oracle can convince a verifier to accept x with better than negligible probability. Formally, for all adversarial provers $\mathcal{A}$, all invalid statements $x \in \mathbb{X}$ such that $(x, w) \notin L$ for any $w \in \mathbb{W}$, and all non-uniform advice strings $z \in \{0,1\}^*$,

$$\Pr\left[\mathsf{V}^H(1^\lambda, x, \rho) = 1 : \rho \leftarrow \mathcal{A}^H(1^\lambda, x, z)\right] \in \mathrm{negl}(\lambda)$$

where the probability is taken over the coins of $\mathcal{A}$ and H.

The definition we have just given specifies deterministic verification, but we note that in the random oracle model, any randomized verifier can easily be transformed into a deterministic one simply by sampling the verifier's random coins as the output of the random oracle queried on the verifier's inputs (x, ρ). We do not explicitly define a zero-knowledge or witness-indistinguishability property for non-interactive proofs until section 4, but we stress that this work is mainly relevant to zero-knowledge proofs; any proof that does not hide the witness in some sense is trivially witness-revealing per definition 2.6.

Next, we give two variations on a straight-line extractability property for non-interactive proofs in the random oracle model. The first is the more general notion, in which the extractor is given access to the random oracle. This is the definition used by Pass [Pas03, Pas04] and Kondi and shelat [Ks22], who proposed mechanisms to achieve the definition by transforming sigma protocols (which ordinarily require rewinding for extraction).

Definition 2.2. Straight-Line Extractability in the ROM

Let L be a language in NP where membership of an instance $x \in \mathbb{X}$ can be verified by a witness $w \in \mathbb{W}$, and let $(\mathsf{P}^H, \mathsf{V}^H)$ be an efficient non-interactive proof scheme where both P and V have access to the random oracle H. We say that $(\mathsf{P}^H, \mathsf{V}^H)$ is *straight-line extractable in the random oracle model* if there exists a PPT extractor algorithm E^H with oracle access to H such that for any $(x, w) \in L$ and any PPT algorithm $\mathcal{A}^H$ with oracle access to H,

$$\Pr\left[\begin{array}{ccr} & & \rho \leftarrow \mathcal{A}^H(1^\lambda, x, w) \\ (x, w') \in L & : & \text{s.t. } \mathsf{V}^H(1^\lambda, x, \rho) = 1, \\ & & w' \leftarrow \mathsf{E}^H(1^\lambda, x, \rho, \mathbf{Q}) \end{array}\right] \in \Omega(1 - \mathrm{negl}(\lambda))$$

over the random coins of $\mathcal{A}$, E, and H, where $\mathbf{Q}$ is the ordered set of queries made by $\mathcal{A}$ to H.

The second, more restricted definition of straight-line extractability is the one used by Fischlin [Fis05], which allows the extractor access only to the values queried by the prover, but not to the oracle's responses. More precisely, the extractor E is allowed oracle acces to H in definition 2.2, but in Fischlin's definition, it is *denied* access to H. In section 3.3 we argue that Pass's and Kondi and shelat's transforms also satisfy this definition, and that any protocol that distributes a proof scheme with this notion of extraction is witness-revealing in the all-but-one security setting per definition 2.6.

Definition 2.3. Response-Independent SLE in the ROM

Let L be a language in NP where membership of an instance $x \in \mathbb{X}$ can be verified by a witness $w \in \mathbb{W}$, and let $(\mathsf{P}^H, \mathsf{V}^H)$ be an efficient non-interactive proof scheme where both P and V have oracle access to H. We say that $(\mathsf{P}^H, \mathsf{V}^H)$ is straight-line extractable in the random oracle model, *independently of the oracle's responses* if there exists a PPT extractor algorithm E *without access to* H such that for any $(x, w) \in L$ and any PPT algorithm $\mathcal{A}^H$ with oracle access to H,

$$\Pr\left[(x, w') \in L \quad : \quad \begin{array}{r} \rho \leftarrow \mathcal{A}^H(1^\lambda, x, w) \\ \text{s.t. } \mathsf{V}^H(1^\lambda, x, \rho) = 1, \\ w' \leftarrow \mathsf{E}(1^\lambda, x, \rho, \mathbf{Q}) \end{array} \right] \in \Omega(1 - \mathrm{negl}(\lambda))$$

over the random coins of $\mathcal{A}$, E, and H, where $\mathbf{Q}$ is the ordered set of queries made by $\mathcal{A}$ to H.

Distributed Provers. Next we give a minimal notion for the multiparty computation of a proof scheme. We specifically *do not* require that a protocol distributing a prover P compute a distribution similar to P; we require *only* that the protocol convince the same verifier that P does. This implies (for example) that the protocol can include auxilliary protocol-specific information in the proof or oracle queries, so long as the verifier tolerates it. Since the original verifier verifies with respect to an ideal random oracle, this also implies that the protocol must generate a proof with respect to the same ideal random oracle and *not* a heuristic realization of that oracle: this makes the protocol *oracle-respecting.*

Definition 2.4. Oracle-Respecting Distributed Computation

Let L be a language in NP where membership of an instance $x \in \mathbb{X}$ can be verified by a witness $w \in \mathbb{W}$, and let $(\mathsf{P}^H, \mathsf{V}^H)$ by a non-interactive proof scheme for L in the random oracle model, per definition 2.1. Let $\pi^H_{n\text{-}\mathsf{Dist}}(1^\lambda, x, \mathbf{w})$ be an expected PPT interactive n-party protocol with common input x and private inputs $\mathbf{w} \in \mathbb{S}^n$ in which all parties have oracle access to H and output the same value upon termination.[a] We say that $\pi^H_{n\text{-}\mathsf{Dist}}$ ORD-computes P^H among n parties if there exists a PPT secret-sharing function $\mathsf{Share}_n : \mathbb{W} \to \mathbb{S}^n$ such that for every $(x, w) \in L$,

$$\Pr\left[\mathsf{V}^H(1^\lambda, x, \rho) = 1 : \rho \leftarrow \pi^H_{n\text{-}\mathsf{Dist}}(1^\lambda, x, \mathsf{Share}_n(w))\right] \in \Omega(1 - \mathrm{negl}(\lambda))$$

over the random coins of Share, $\pi_{n\text{-}\mathsf{Dist}}$, and H.

[a]For convenience, we use this notation to refer to both the protocol itself, and to a function that runs the protocol and returns its output value.

It is possible that a distributed proof scheme has a slightly stronger form in which specific oracle queries are in some sense bound to particular parties. We show in section 3.3 that protocols of this form are witness-revealing in the all-but-one security setting per definition 2.6, even if the proof scheme they distribute satisfies only the weaker notion of straight-line extraction given in definition 2.2.

Definition 2.5. ORD with Traceable Query Allocation

Let $\pi^H_{n\text{-}\mathsf{Dist}}(1^\lambda, x, \mathbf{w})$ be a protocol that ORD-computes $\mathsf{P}^H(1^\lambda, x, w)$ per definition 2.4. We say that $\pi^H_{n\text{-}\mathsf{Dist}}(1^\lambda, x, \mathbf{w})$ has *traceable query allocation* if there exists an algorithm $\mathsf{M}^H : \mathbb{N} \times \mathbb{X} \times \{0,1\}^* \to [n]$ that outputs with overwhelming probability the index of the party who made the *first* query to H on a particular value Q, given only n, x, Q, and access to the random oracle H.

The above definitions for multiparty computation of a non-interactive proof *do not* specify any particular security property against an adversary that corrupts some of the participants. Instead, we introduce the notion that a protocol is *witness-revealing* under a certain number of corruptions, which implies that a protocol *cannot* achieve any security property that implies hiding the witness, even if the adversary is semi-honest. Witness-revealingness in the presence of t corruptions contradicts almost any intuitive notion of privacy in the presence of t corruptions, whether game-based, simulation-based, semi-honest, malicious, static, or adaptive.

Definition 2.6. Witness Revealingness with t-of-n Corruptions

Let $\pi^H_{n\text{-}\mathsf{Dist}}(1^\lambda, x, \mathbf{w})$ be a protocol that ORD-computes $\mathsf{P}^H(1^\lambda, x, w)$ per definition 2.4, and let $\mathsf{Views}^H_{\mathsf{Dist}}(1^\lambda, x, \mathbf{w})$ be an algorithm that runs $\pi^H_{n\text{-}\mathsf{Dist}}(1^\lambda, x, \mathbf{w})$ and outputs $\mathbf{v}$ such that $\mathbf{v}_i$ is the view of the i^{th} party. We say that $\pi^H_{n\text{-}\mathsf{Dist}}$ is *witness-revealing with t-of-n corruptions* if there exists a PPT algorithm $\mathcal{A}$ such for every $(x, w) \in L$,

$$\Pr\left[w \leftarrow \mathcal{A}(1^\lambda, x, \{\mathbf{v}_i\}_{i\in\mathbf{C}}) \;:\; \begin{array}{r} \mathbf{C} \leftarrow 2^{[n]} : |\mathbf{C}| = t, \\ \mathbf{w} \leftarrow \mathsf{Share}_n(w), \\ \mathbf{v} \leftarrow \mathsf{Views}^H_{\mathsf{Dist}}(1^\lambda, x, \mathbf{w}) \end{array} \right] \in \Omega(1/\operatorname{poly}(\lambda))$$

over the random coins of Share, $\pi_{n\text{-}\mathsf{Dist}}$, $\mathcal{A}$, and H.

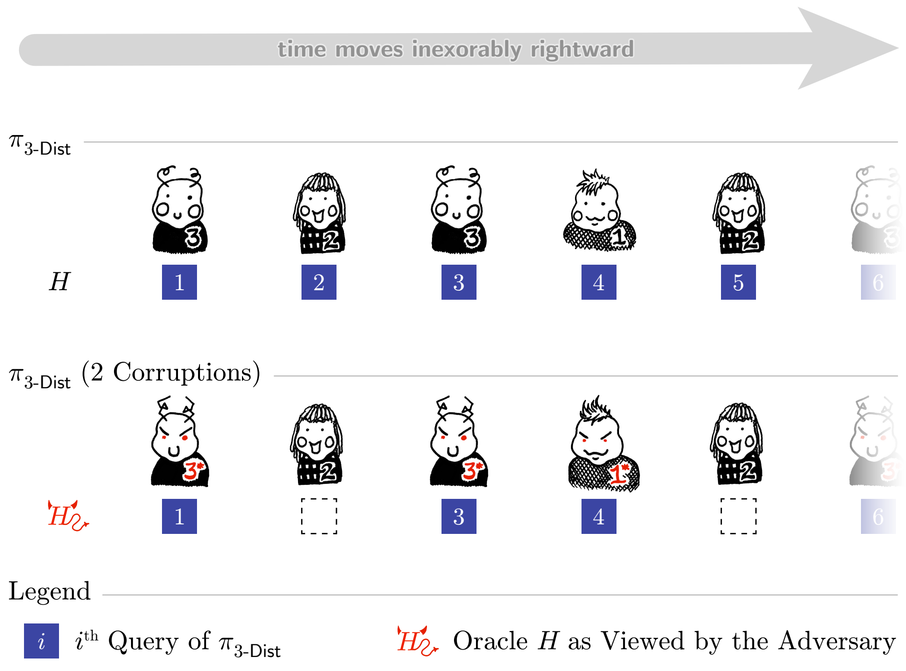

time moves inexorably rightward
$\pi_{3\text{-Dist}}$
H
1
2
3
4
5
6
$\pi_{3\text{-Dist}}$ (2 Corruptions)
1
3
4
6
Legend
i^{th} Query of $\pi_{3\text{-Dist}}$
Oracle H as Viewed by the Adversary

Figure 1: Suppose that the three-party interactive protocol $\pi_{3\text{-Dist}}$ ORD-computes some non-interactive straight-line-extractable proof that can be verified with two queries to H. We illustrate the sequence of queries made by the parties in the protocol over time: $\mathcal{P}_3$ makes the first query, then $\mathcal{P}_2$ makes the second, then $\mathcal{P}_3$ makes another query, and so on. We aim to prove that if the adversary corrupts any two parties randomly, then it can extract a witness with noticeable probability using only the output and the oracle queries of the corrupt parties.

3 All-but-One Security is Impossible

In this section we prove our main theorems. We will begin with an overview. In section 3.1, we show that given any non-interactive proof, if the prover's queries are partitioned into more subsets than the number of queries made by the verifier, and the queries in one partition omitted at random, then the prover retains a noticeable probability of convincing the original verifier. We give a transformation Trim for any prover that performs such a partitioning-and-omission operation. In section 3.2, we give a mechanism (comprising $\mathsf{P}_{n\text{-Dist}}$ and Shave) for applying our Trim transformation (which requires a single prover) to a protocol that *distributes* a prover, in such a way that each subset in the partitioning of Trim contains the queries of exactly one party in the protocol. We prove that under the combination of this mechanism and Trim, straight-line extraction is preserved with noticeable probability. Finally, in section 3.3, we use our collection of lemmas to prove that any protocol that distributes well-known straight-line extractable non-interactive proof schemes, such as those of Pass [Pas03, Pas04], Fischlin [Fis05], or Kondi and shelat [Ks22] is witness-revealing if the adversary corrupts as many parties as there are queries checked by the verifier. This is our main result, and we illustrate it (with a specific concrete parameterization) in figure 1. The implications of our main theorems have already been explored in section 1.

3.1 The Prover's Haircut

We begin with a lemma lower-bounding the probability that the verifier notices that a subset of queries have been omitted, if the prover partitions its queries into a sufficiently-large number of subsets, and then omits one.

Lemma 3.1. *Let $\mathbf{Q}_{\mathsf{P}}$ and $\mathbf{Q}_{\mathsf{V}}$ be sets such that $\mathbf{Q}_{\mathsf{V}} \subset \mathbf{Q}_{\mathsf{P}}$. Then, for any partitioning of $\mathbf{Q}_{\mathsf{P}}$ into $n > |\mathbf{Q}_{\mathsf{V}}|$ partitions $\mathbf{Q}_{\mathsf{P}1}, \mathbf{Q}_{\mathsf{P}2}, \cdots, \mathbf{Q}_{\mathsf{P}n}$,*

$$\Pr[\mathbf{Q}_{\mathsf{P}i} \cap \mathbf{Q}_{\mathsf{V}} = \emptyset : i \leftarrow [n]] \geq 1 - |\mathbf{Q}_{\mathsf{V}}|/n$$

Proof. There are n partitions (of which one is chosen uniformly), and there can be at most $|\mathbf{Q}_{\mathsf{V}}|$ partitions that have a non-empty intersection with $\mathbf{Q}_{\mathsf{V}}$. Therefore, the probability that a partition that has a non-empty intersection with $\mathbf{Q}_{\mathsf{V}}$ is chosen is at most $|\mathbf{Q}_{\mathsf{V}}|/n$. Taking the complement of this event yields the lemma. □

Now we give our partitioning-and-omission mechanism Trim, and prove that the verifier still accepts with noticeable probability when Trim is applied to a prover. We say that a verifier *checks* a query made by a prover if it queries the random oracle on the same value.

Algorithm 3.2. $\mathsf{Trim}^{H}_{\mathsf{P},n}(1^\lambda, x, w)$**:** n**-Trimming Algorithm**

The algorithm P expects to access a random oracle $H : \{0,1\}^* \mapsto \{0,1\}^\ell$. Its n-trimmed version $\mathsf{Trim}_{\mathsf{P},n}$ "deletes" every n^{th} query (starting from a random index $\delta \in [n]$), if it is fresh. To preserve the behaviour of P, the algorithm $\mathsf{Trim}^{H}_{\mathsf{P},n}$ answers "deleted" queries with random values. Effectively, $\mathsf{Trim}^{H}_{\mathsf{P},n}$ is simply P^{H^*} for a random oracle H^* that agrees with H except at every (fresh) n^{th} query made by P.

1. Initialize $i = 1$ and $H^* = \emptyset$, sample $\delta \leftarrow [n]$, and start running $\mathsf{P}(1^\lambda, x, w)$.
2. Upon receiving an oracle query from P,
   - Let Q_i be the value that was queried. If Q_i is fresh (i.e. if it has never before been queried),
     - If $i \equiv \delta \pmod{n}$, program $H^*(Q_i)$ to be a uniformly-sampled random value.

– Otherwise program $H^*(Q_i) \mapsto H(Q_i)$.

- Otherwise, if $i \not\equiv \delta \pmod{n}$, query H on the value Q_i.[a]

Regardless of the above conditions, respond to the query with $H^*(Q_i)$, and increment i.

3. Collect ρ as the output of P.

4. Output (ρ, H^*)

[a]The result of this query is not actually used by Trim. The purpose of this step is to maintain the distribution of *queries* made to H, in order to preserve the behaviour of an extractor that takes such queries as input. Notice that if the smallest j such that $Q_j = Q_i$ satisfies $j \equiv \delta \pmod{n}$, then the (unused) value $H(Q_i)$ will differ from the programmed value $H^*(Q_i)$ actually returned to P in this step. The set of such "unused" queries to H that is induced by Trim is later denoted $\mathbf{Q}_{\Delta \cap \neg\delta}$ in section 3.3.

Lemma 3.3. *If $(\mathsf{P}^H, \mathsf{V}^H)$ is a non-interactive proof scheme for a language L in the random oracle model, per definition 2.1, such that P makes at least n queries to H and V checks at most $n-1$, then ρ obtained by running $(\rho, H^*) \leftarrow \mathsf{Trim}_{\mathsf{P},n}(1^\lambda, x, w)$ is an accepting proof with noticeable probability when $(x, w) \in L$. That is, for every $(x, w) \in L$,*

$$\Pr\left[\mathsf{V}^H(1^\lambda, x, \rho) = 1 : (\rho, H^*) \leftarrow \mathsf{Trim}^H_{\mathsf{P},n}(1^\lambda, x, w)\right] \in \Omega(1/\operatorname{poly}(\lambda))$$

Proof. Our proof will begin with the observation that the ρ almost always verifies when checked using the oracle H^* that is programmed by Trim, and then we will bound from below the probability that ρ still verifies when H^* is replaced by H. We structure the proof as a series of claims that provide the components required to analyze the probability that $\mathsf{V}^H(1^\lambda, x, \rho) = 1$.

Claim 3.4. *There is a negligible function ϵ such that for any $(x, w) \in L$ and $n \in \mathbb{N}^+$,*

$$\Pr[\mathsf{V}^{H^*}(1^\lambda, x, \rho) = 1 : (\rho, H^*) \leftarrow \mathsf{Trim}^H_{\mathsf{P},n}(1^\lambda, x, w)] \geq 1 - \epsilon(\lambda)$$

This follows directly from the completeness property of $(\mathsf{P}^H, \mathsf{V}^H)$, as given in definition 2.1.

Without loss of generality, we assume that V's queries are a strict subset of P's and can therefore be answered by H^*.[1] Denote by $\mathbf{Q}_\mathsf{P}$ the vector of random oracle queries made by P to H^* within Trim. Throughout this proof, when we reference a query Q_i, the integer i will denote the *original* position of the query Q_i in $\mathbf{Q}_\mathsf{P}$.[2] Let $\mathbf{Q}^\mathsf{f}_\mathsf{P}$ be the result of removing redundant queries from $\mathbf{Q}_\mathsf{P}$ by retaining only the lowest-indexed occurrence of any queried value, i.e.

$$\mathbf{Q}^\mathsf{f}_\mathsf{P} = \{Q_i : Q_i \in \mathbf{Q}_\mathsf{P} \ \wedge \ \nexists Q_j \in \mathbf{Q}_\mathsf{P} \text{ s.t. } (Q_j = Q_i \wedge j < i)\}$$

[1]Since V is a public algorithm, it can always be run by P to ensure this, if it is deterministic.
[2]That is, indices are preserved by any filtering operations that occur.

Let $\mathbf{Q}_{\mathsf{T}}$ be the list of queries made by $\mathsf{Trim}_{\mathsf{P},n}(1^\lambda, x, w)$ to H, and let $\mathbf{Q}_{\mathsf{T}}^{\neg\delta}$ be the list of queries in $\mathbf{Q}_{\mathsf{T}}$ that remain after the ones that also appear in $\mathbf{Q}_{\mathsf{P}}^{\mathsf{f}}$ and have indices with residue $\delta \pmod n$ are removed, i.e.

$$\mathbf{Q}_{\mathsf{T}}^{\neg\delta} = \{Q_i : Q_i \in \mathbf{Q}_{\mathsf{T}} \ \wedge \ \nexists Q_j \in \mathbf{Q}_{\mathsf{P}}^{\mathsf{f}} \text{ s.t. } (Q_i = Q_j \wedge j \equiv \delta \pmod n)\} \tag{1}$$

Claim 3.5. *For any $(x, w) \in L$, given $(\rho, H^*) \leftarrow \mathsf{Trim}_{\mathsf{P},n}^{H}(1^\lambda, x, w)$ and $\mathbf{Q}_{\mathsf{T}}^{\neg\delta}$ as defined in equation 1, H and H^* agree on the responses to all queries in $\mathbf{Q}_{\mathsf{T}}^{\neg\delta}$.*

The above claim follows from the fact that Trim only programs H^* to disagree with H at queries Q_i where i indexes the first time such a query is made, and $i \equiv \delta \pmod n$. The set $\mathbf{Q}_{\mathsf{P}}^{\mathsf{f}}$ retains only queries that are indexed by their first occurrence in $\mathbf{Q}_{\mathsf{P}}$, and $\mathbf{Q}_{\mathsf{T}}^{\neg\delta}$ filters out those whose indices have residue δ. Therefore, $\mathbf{Q}_{\mathsf{T}}^{\neg\delta}$ only contains queries at which Trim programs H^* to be the same as H.

Claim 3.6. $\mathsf{V}^H(1^\lambda, x, \rho) = \mathsf{V}^{H^*}(1^\lambda, x, \rho)$ *when $\mathbf{Q}_{\mathsf{V}} \subseteq \mathbf{Q}_{\mathsf{T}}^{\neg\delta}$, where $\mathbf{Q}_{\mathsf{V}}$ is the list of queries made by V to either H or H^*.*

Because we have assumed V to be deterministic, V^H and V^{H^*} can only diverge when V queries a point at which H and H^* disagree, and it follows from claim 3.5 that they are always in agreement on queries in $\mathbf{Q}_{\mathsf{T}}^{\neg\delta}$.

Claim 3.7. $\Pr[\mathbf{Q}_{\mathsf{V}} \subseteq \mathbf{Q}_{\mathsf{T}}^{\neg\delta}] \geq 1/n$

Observe that $\mathbf{Q}_{\mathsf{T}}^{\neg\delta}$ has been formed by partitioning $\mathbf{Q}_{\mathsf{P}}$ into n sets of queries, and deleting one set at random. The partition to which a query $Q_i \in \mathbf{Q}_{\mathsf{P}}$ belongs is the one indexed by $j \bmod n$ such that $Q_j \in \mathbf{Q}_{\mathsf{P}}^{\mathsf{f}}$ and $Q_i = Q_j$. If we use $\mathbf{Q}_{\Delta}$ to denote the deleted partition, then $\mathbf{Q}_{\Delta} = \mathbf{Q}_{\mathsf{P}} \setminus \mathbf{Q}_{\mathsf{T}}^{\neg\delta}$. Applying lemma 3.1 we have

$$\Pr[\mathbf{Q}_{\Delta} \cap \mathbf{Q}_{\mathsf{V}} = \emptyset] \geq 1 - |\mathbf{Q}_{\mathsf{V}}|/n \geq 1 - (n-1)/n = 1/n$$

and since $\mathbf{Q}_{\Delta} \cap \mathbf{Q}_{\mathsf{V}} = \emptyset \implies \mathbf{Q}_{\mathsf{V}} \subseteq \mathbf{Q}_{\mathsf{T}}^{\neg\delta}$, the above claim follows.

Finally, to derive the statement of the lemma, we observe that if $\mathsf{V}^{H^*}(1^\lambda, x, \rho) = 1$ and $\mathbf{Q}_{\mathsf{V}} \subset \mathbf{Q}_{\mathsf{T}}^{\neg\delta}$, then via claim 3.6 we have $\mathsf{V}^H(1^\lambda, x, \rho) = 1$. To determine the probability that $\mathsf{V}^{H^*}(1^\lambda, x, \rho) = 1$ and $\mathbf{Q}_{\mathsf{V}} \subset \mathbf{Q}_{\mathsf{T}}^{\neg\delta}$ hold simultaneously, consider the complement of the case that *neither* event occurs. In particular,

$$\begin{aligned}
\Pr\left[\mathsf{V}^H(1^\lambda, x, \rho) = 1\right] &\geq \Pr\left[\mathsf{V}^{H^*}(1^\lambda, x, \rho) = 1 \ \wedge \ \mathbf{Q}_{\mathsf{V}} \subset \mathbf{Q}_{\mathsf{T}}^{\neg\delta}\right] \\
&= \Pr\left[\neg\left(\mathsf{V}^{H^*}(1^\lambda, x, \rho) \neq 1 \ \vee \ \mathbf{Q}_{\mathsf{V}} \not\subset \mathbf{Q}_{\mathsf{T}}^{\neg\delta}\right)\right] \\
&= 1 - \Pr\left[\mathsf{V}^{H^*}(1^\lambda, x, \rho) \neq 1 \ \vee \ \mathbf{Q}_{\mathsf{V}} \not\subset \mathbf{Q}_{\mathsf{T}}^{\neg\delta}\right] \\
&\geq 1 - \left(\Pr\left[\mathsf{V}^{H^*}(1^\lambda, x, \rho) \neq 1\right] + \Pr\left[\mathbf{Q}_{\mathsf{V}} \not\subset \mathbf{Q}_{\mathsf{T}}^{\neg\delta}\right]\right) \\
&\geq 1 - (\epsilon(\lambda) + (1 - 1/n)) \\
&\geq 1/n - \epsilon(\lambda)
\end{aligned}$$

Because V is polynomial-time, it must hold that $n \in \operatorname{poly}(\lambda)$, and $\epsilon(\lambda)$ is a negligible function per claim 3.4. We can conclude that

$$1/n - \epsilon(\lambda) \in \Omega(1/\operatorname{poly}(\lambda))$$

which proves the lemma. □

3.2 Distributed P in-the-Head

Suppose that there exists some straight-line extractable non-interactive proof scheme $(\mathsf{P}, \mathsf{V}, \mathsf{E})$, and there exists a protocol $\pi_{n\text{-}\mathsf{Dist}}$ that ORD-computes P among n parties. We define an alternative prover algorithm $\mathsf{P}_{n\text{-}\mathsf{Dist}}$ that runs the protocol $\pi_{n\text{-}\mathsf{Dist}}$ "in the head" (i.e. it emulates all parties in the protocol) while forwarding any random oracle queries that the parties make to H. We wish for queries to H by $\mathsf{P}_{n\text{-}\mathsf{Dist}}$ to adhere to the following format: the i^{th} query must come from party $\mathcal{P}_j$ such that $j \equiv i \pmod n$ in the emulated instance of $\pi_{n\text{-}\mathsf{Dist}}$ running inside of $\mathsf{P}_{n\text{-}\mathsf{Dist}}$. This is straightforward to enforce by inserting redundant queries as padding, and it allows us to use the Trim algorithm from the previous section to delete the queries from a single consistent party, without any out-of-band information. We also define a companion algorithm Shave to remove these redundant queries (if they have not otherwise been filtered). In figure 2 we illustrate the algorithm $\mathsf{P}_{n\text{-}\mathsf{Dist}}$, along with the sequences of oracle queries that it emits. In figure 3 we illustrate the effect that applying Trim and then Shave has upon the sequence of emitted queries.

Algorithm 3.8. $\mathsf{P}^{H}_{n\text{-}\mathsf{Dist}}(1^\lambda, x, w)$**: Distributed P in-the-Head**

1. Sample $\{w_i\}_{i\in[n]} \coloneqq \mathbf{w} \leftarrow \mathsf{Share}_n(w)$, and begin executing $\pi_{n\text{-}\mathsf{Dist}}(1^\lambda, x, \mathbf{w})$ among virtual parties $\{\mathcal{P}_i(1^\lambda, x, w_i)\}_{i\in[n]}$

2. Upon receiving a query Q from $\mathcal{P}_i$ on behalf of H, make a sequence of $2n$ queries to H, where the i^{th} queried value amongst them is 1, the $(n+i)^{\text{th}}$ queried value is Q, and the rest of the queried values are 0. The first n queries in this sequence are the *signaling* queries, and the last n queries are the *payload* queries.

3. Output whatever $\pi_{n\text{-}\mathsf{Dist}}$ outputs.

Algorithm 3.9. $\mathsf{Shave}^{H}_{\mathsf{P},n}(1^\lambda, x, w)$**: Redundant Query Remover**

1. Initialize $i = 1$ and begin executing $\mathsf{P}(1^\lambda, x, w)$.

2. Upon receiving an oracle query from P,

   - Let Q_i be the value that was queried. If $\lceil i/n \rceil$ is *odd*, return 0 to P on behalf of the random oracle.
   - If $\lceil i/n \rceil$ is *even* and $Q_{i-n} = 0$, then return 0 to P on behalf of the random oracle.

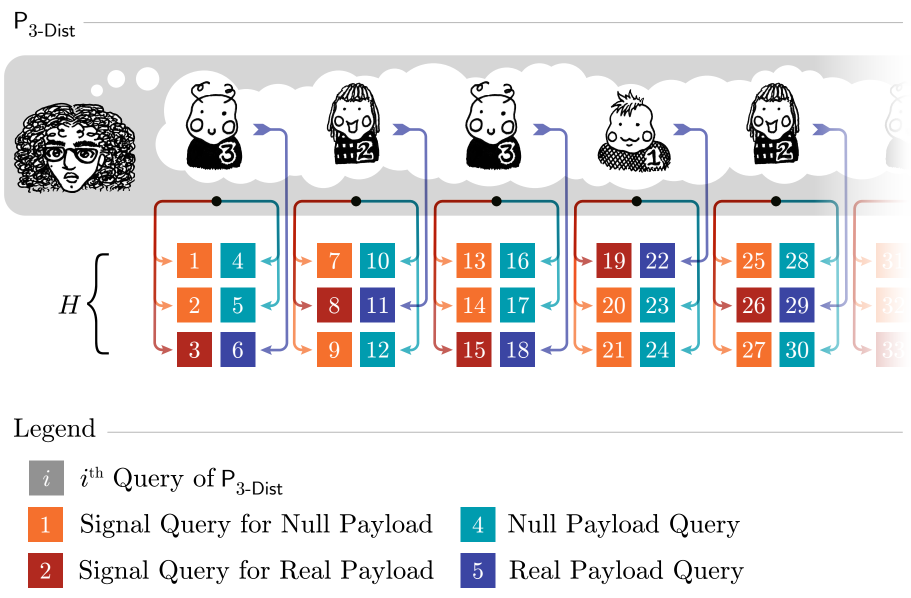


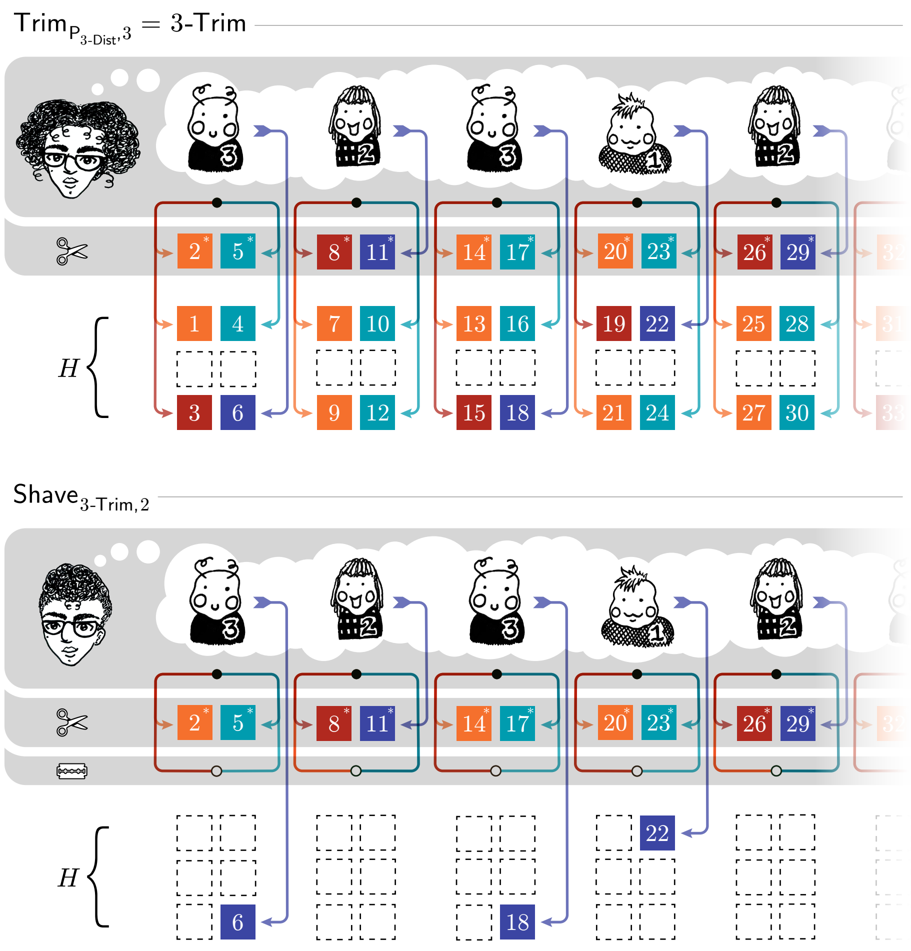
Figure 2: As in figure 1, we illustrate the three-party case. $P_{3\text{-Dist}}$ runs $\pi_{3\text{-Dist}}$ "in the head," and embeds the original sequence of queries into a larger, padded sequence. Each query made by a party in $\pi_{3\text{-Dist}}$ becomes a set of six queries emitted by $P_{3\text{-Dist}}$: three *signal* queries followed by three *payload* queries. If the original query is made by $\mathcal{P}_i$, then it appears as the i^{th} payload query, and the other two payload queries in the set are null. The signal queries serve to indicate which of the payloads are null, and which are real.

- If $\lceil i/n \rceil$ is *even* and $Q_{i-n} = 1$, then return $H(Q_i)$ to P on behalf of the random oracle. In other words, query H on behalf of P and forward its response.

After responding, increment i.

3. Output whatever P outputs.

We observe about $P_{n\text{-Dist}}$, Trim, and Shave that they pass the output of $\pi_{n\text{-Dist}}$ through unaltered by construction. We prove two lemmas about them: the first lower-bounds the problem that extraction is still successful if a straight-line extractable prover is distributed in-the-head, trimmed, and then the redundant queries are shaved. The second lemma relates the distribution of queries emitted by the trimmed distributed-in-the-head prover to the distribution of queries emitted by the underlying protocol.


$\mathsf{Trim}_{\mathsf{P}_{\text{3-Dist}},3} = \text{3-Trim}$
$\mathsf{Shave}_{\text{3-Trim},2}$
H
H

Legend

- $\boxed{i}$ i^{th} Query of $\mathsf{P}_{\text{3-Dist}}$
- $\boxed{i^*}$ Query Diverted to H^* by Trim
- 🟧 1 Signal Query for Null Payload
- 🟦 4 Null Payload Query
- 🟥 2 Signal Query for Real Payload
- 🟦 5 Real Payload Query

Figure 3: As in figures 1 and 2, we illustrate the three-party case. Trim wraps $\mathsf{P}_{\text{3-Dist}}$ and diverts query $3i+j$ to the internal random oracle H^*, for every $i \in \mathbb{N}$ and some random $j \in [3]$. Shave wraps both $\mathsf{P}_{\text{3-Dist}}$ and Trim and removes the padding queries that $\mathsf{P}_{\text{3-Dist}}$ added to the set of queries produced by the instance of $\pi_{\text{3-Dist}}$ in its head. The distribution of queries emitted by Shave is identical to the distribution observed by an adversary randomly corrupting two parties in $\pi_{\text{3-Dist}}$, as illustrated in figure 1. The output distributions are also identical.

Lemma 3.10. *If*

- $(\mathsf{P}^H, \mathsf{V}^H, \mathsf{E}^H)$ *is a straight-line extractable non-interactive proof scheme for a language L in the random oracle model, per definitions 2.1 and 2.2*
- P *makes at least n queries to H and V checks at most $n-1$*
- $\pi_{n\text{-}\mathsf{Dist}}$ *is a protocol that* ORD*-computes P among n parties per definition 2.4*
- $\mathsf{P}^H_{n\text{-}\mathsf{Trim}}$ *is a prover that runs $(\rho, H^*) \leftarrow \mathsf{Trim}^H_{\mathsf{P}_{n\text{-}\mathsf{Dist}},n}$ and then outputs ρ*

then for any $(x, w) \in L$,

$$\Pr\left[(x, w') \in L \quad : \quad \begin{array}{r} \rho \leftarrow \mathsf{Shave}^H_{\mathsf{P}_{n\text{-}\mathsf{Trim}},n-1}(1^\lambda, x, w), \\ w' \leftarrow \mathsf{E}^H(1^\lambda, x, \rho, \mathbf{Q}_\mathsf{S}) \end{array}\right] \in \Omega(1/\operatorname{poly}(\lambda))$$

where $\mathbf{Q}_\mathsf{S}$ denotes the ordered list of queries made by Shave *to H and the probability is taken over the coins of* Shave*,* Trim*,* $\mathsf{P}_{n\text{-}\mathsf{Dist}}$*,* E*, and H.*

Proof. As above, we structure our proof as a series of claims.

Claim 3.11. $(\mathsf{P}^H_{n\text{-}\mathsf{Dist}}, \mathsf{V}^H)$ *is a non-interactive proof scheme for language L in the random oracle model per definition 2.1.*

Because V^H belongs to a non-interactive proof scheme as specified in definition 2.1 and because

$$\Pr\left[\mathsf{V}^H(1^\lambda, x, \rho) = 1 : \rho \leftarrow \pi^H_{n\text{-}\mathsf{Dist}}(1^\lambda, x, \mathsf{Share}_n(w))\right] \in \Omega(1 - \operatorname{negl}(\lambda))$$

for any $(x, w) \in L$ as specified by definition 2.4, and because $\mathsf{P}^H_{n\text{-}\mathsf{Dist}}$ simply runs $\pi^H_{n\text{-}\mathsf{Dist}}(1^\lambda, x, \mathsf{Share}_n(w))$ internally, we can conclude that $(\mathsf{P}^H_{n\text{-}\mathsf{Dist}}, \mathsf{V}^H)$ is a non-interactive proof scheme for language L per definition 2.1.

Claim 3.12. *For any $(x, w) \in L$,*

$$\Pr\left[\mathsf{V}^H(1^\lambda, x, \rho) = 1 : \rho \leftarrow \mathsf{Trim}^H_{\mathsf{P}_{n\text{-}\mathsf{Dist}},n}(1^\lambda, x, w)\right] \in \Omega(1/\operatorname{poly}(\lambda))$$

The above claim follows from the conjunction of claim 3.11, which establishes that $(\mathsf{P}^H_{n\text{-}\mathsf{Dist}}, \mathsf{V}^H)$ is a non-interactive proof scheme, and lemma 3.3, which establishes that when Trim is applied to a non-interactive proof scheme with at most $n-1$ verifier-checked queries, the resulting algorithm has a noticeable chance of producing proofs that are accepted by an unaltered verifier.

Claim 3.13. *Let $\mathsf{P}^H_{n\text{-}\mathsf{Trim}}$ be a prover that runs $(\rho, H^*) \leftarrow \mathsf{Trim}^H_{\mathsf{P}_{n\text{-}\mathsf{Dist}},n}$ and then outputs ρ. For any $(x, w) \in L$, the outputs of $\mathsf{Shave}^H_{\mathsf{P}_{n\text{-}\mathsf{Trim}},n-1}(1^\lambda, x, w)$ and $\mathsf{P}^H_{n\text{-}\mathsf{Trim}}(1^\lambda, x, w)$ are identically distributed.*

We make three observations about the structure of oracle queries through the various algorithms:

1. The queries emitted by $\mathsf{P}_{n\text{-}\mathsf{Dist}}$ can be partitioned into alternating *signaling* and *payload* subsets of length n, and the response to a query in a payload subset will only be forwarded to a virtual party in $\mathsf{P}_{n\text{-}\mathsf{Dist}}$'s emulated instance of $\pi_{n\text{-}\mathsf{Dist}}$ if the corresponding query value in the proceeding signaling subset is 1. This implies that the value of ρ produced by $\mathsf{P}_{n\text{-}\mathsf{Dist}}$ is *completely independent* of the responses to any signaling query or any payload query that has a corresponding signaling query with value 0.

2. Trim preserves the signal/payload relationship. Since $\mathsf{Trim}_{\mathsf{P}_{n\text{-}\mathsf{Dist}},n}$ omits every n^{th} query produced by $\mathsf{P}_{n\text{-}\mathsf{Dist}}$ (starting with some random index δ), and the signal query corresponding to payload query Q_i always has index $i-n$, Trim omits a payload query if and only if it also omits the corresponding signal query. The query set emitted by Trim can be partitioned into alternating signaling and payload subsets of length $n-1$, matching the parameter of Shave.

3. Shave forwards all payload queries with a signal value of 1 to H, and does not query H when it receives a signal query or a payload query that has an associated signal of 0. Since we have established that ρ produced by $\mathsf{P}_{n\text{-}\mathsf{Dist}}$ or $\mathsf{P}_{n\text{-}\mathsf{Trim}}$ depends *only* on payload queries that have an associated signal query of 1, this implies that the output of $\mathsf{P}^H_{n\text{-}\mathsf{Trim}}(1^\lambda, x, w)$ is distributed identically to the output of $\mathsf{Shave}^H_{\mathsf{P}_{n\text{-}\mathsf{Trim}},n-1}(1^\lambda, x, w)$.

   The conjunction of claims 3.12 and 3.13 yields

$$\Pr\left[\mathsf{V}^H(1^\lambda, x, \rho) = 1 : \rho \leftarrow \mathsf{Shave}^H_{\mathsf{P}_{n\text{-}\mathsf{Trim}},n-1}(1^\lambda, x, w)\right] \in \Omega(1/\operatorname{poly}(\lambda))$$

for any $(x, w) \in L$. Since V^H belongs to a straight-line extractable non-interactive proof scheme with an extractor E^H per definition 2.2 and noticeable probabilities are closed under multiplication, definition 2.2 implies that lemma 3.10 must hold. $\square$

Lemma 3.14. *Let*

- $(\mathsf{P}^H, \mathsf{V}^H)$ *be a non-interactive proof scheme for a language L in the random oracle model, per definition 2.1*
- $\pi_{n\text{-}\mathsf{Dist}}$ *be a protocol that* ORD*-computes* P *among n parties per definition 2.4.*
- $\mathsf{P}^H_{n\text{-}\mathsf{Trim}}$ *be a prover that runs $(\rho, H^*) \leftarrow \mathsf{Trim}^H_{\mathsf{P}_{n\text{-}\mathsf{Dist}},n}$ and then outputs ρ*

and for any $(x, w) \in L$, let

- $\mathbf{Q}_{\mathsf{S}}$ *be the list of queries made by $\mathsf{Shave}^H_{\mathsf{P}_{n\text{-}\mathsf{Trim}},n-1}(1^\lambda, x, w)$ to H*
- $\mathbf{Q}_\pi$ *be the list of queries made to H by all parties in an execution of $\pi^H_{n\text{-}\mathsf{Dist}}(1^\lambda, n, x, \mathsf{Share}_n(w))$,*
- $\mathbf{Q}_{\mathcal{P}_i}$ *be the list of queries that are made* exclusively *by party $\mathcal{P}_i$ in the aforementioned execution of $\pi^H_{n\text{-}\mathsf{Dist}}$*

and finally let $\delta \leftarrow [n]$ be uniformly sampled and let $\mathbf{Q}_{\neg\delta} = \mathbf{Q}_\pi \setminus \mathbf{Q}_{\mathcal{P}_\delta}$. It follows that $\mathbf{Q}_{\neg\delta}$ is distributed identically to $\mathbf{Q}_{\mathsf{S}}$.

Proof. We begin with an intermediate claim about the set of queries observed by the internally emulated random oracle of Trim.

Claim 3.15. *Let $\mathbf{Q}_\pi$ be the list of queries made to H by all parties in an execution of $\pi^H_{n\text{-}\mathsf{Dist}}(1^\lambda, x, \mathsf{Share}_n(w))$, and let $\mathbf{Q}^*_{\mathsf{P}}$ be the list of queries made by $\mathsf{P}^{H^*}_{n\text{-}\mathsf{Dist}}$ to the emulated oracle H^* in an execution of $\mathsf{Trim}_{\mathsf{P}_{n\text{-}\mathsf{Dist}},n}(1^\lambda, x, w)$. $\mathbf{Q}_\pi$ is distributed identically to $\mathbf{Q}^*_{\mathsf{P}}$, up to redundant padding queries.*

The above claim follows from the fact the oracle H^* emulated to $\mathsf{P}_{n\text{-}\mathsf{Dist}}$ by Trim has a distribution identical to H, which implies that the views of all parties in $\pi^H_{n\text{-}\mathsf{Dist}}$ are distributed identically to the views of the corresponding parties in the instance of $\pi^{H^*}_{n\text{-}\mathsf{Dist}}$ that is emulated internally by $\mathsf{P}_{n\text{-}\mathsf{Dist}}$.

Now we derive the statement of the lemma. Recall that the j^{th} random oracle query made by $\mathsf{P}^{H^*}_{n\text{-}\mathsf{Dist}}$ corresponds to either a query by party $\mathcal{P}_i$ such that $i \equiv j \pmod{n}$ in the emulated instance of $\pi^{H^*}_{n\text{-}\mathsf{Dist}}$ internal to $\mathsf{P}^{H^*}_{n\text{-}\mathsf{Dist}}$, or to a redundant padding query. The j^{th} query Q_j made by $\mathsf{P}^{H^*}_{n\text{-}\mathsf{Dist}}$ in $\mathsf{Trim}^H_{\mathsf{P}_{n\text{-}\mathsf{Dist}},n}(1^\lambda, x, w)$ is forwarded to H if and only if $j \not\equiv \delta \pmod{n}$, which implies that the list of queries made by Trim to H is exactly the set of queries made by all emulated parties except party $\mathcal{P}_\delta$ for some randomly sampled $\delta \leftarrow [n]$, interleaved with redundant padding queries. As we previously argued in the context of claim 3.13, Shave perfectly omits redundant padding queries, even when intermediated by Trim, which implies that only queries originating from emulated parties other than $\mathcal{P}_\delta$ are emitted by $\mathsf{Shave}^H_{\mathsf{P}_{n\text{-}\mathsf{Trim}},n-1}(1^\lambda, x, w)$. Claim 3.15 establishes that the distribution of queries produced by the complete set of parties that is emulated by Trim is equivalent to the distribution of queries in a standalone instance of the protocol, which yields lemma 3.14 when combined with the foregoing facts. $\square$

3.3 Distributed NIZKs are Sometimes Witness-Revealing

Finally, we apply the above lemmas to well known straight-line extractable NIZK schemes. Notice that in lemma 3.10, ρ is effectively produced with one oracle H^* and extracted with another oracle H. The oracles H and H^* diverge at the queries

$$\mathbf{Q}_\Delta = \{Q : Q \in \mathbf{Q}_\pi \wedge \mathcal{P}_\delta \text{ is the first party to query } Q \text{ in } \pi^{H^*}_{n\text{-}\mathsf{Dist}}\}$$

since the responses to these queries are programmed in H^* by Trim, independently of H. The intersection $\mathbf{Q}_{\Delta\cap\neg\delta} = \mathbf{Q}_\Delta \cap \mathbf{Q}_{\neg\delta}$ is non-empty if some query Q initially made by $\mathcal{P}_\delta$ is later repeated by another party. We wish to show that there exists an extractor that is successful given the queryset $\mathbf{Q}_{\neg\delta}$ and the proof produced by $\pi^{H^*}_{n\text{-}\mathsf{Dist}}$ and access to H^* (whereas we are currently only guaranteed an extractor that is successful with access to H). We make two different restrictions upon the original straight-line-extractable proof that resolve this discrepancy, yielding two theorems.

Theorem 3.16. *If* $(\mathsf{P}^H, \mathsf{V}^H, \mathsf{E})$ *is a non-interactive proof scheme for a language* L *with response-independent straight-line extractability in the random oracle model, per definitions 2.1 and 2.3, such that* P *makes at least* n *queries to* H *and* V *checks at most* $n-1$*, then any protocol* $\pi^H_{n\text{-}\mathsf{Dist}}(1^\lambda, x, \mathbf{w})$ *that* ORD*-computes* P *among* n *parties per definition 2.4 is witness-revealing per definition 2.6 in the presence of an adversary that corrupts* $n-1$ *parties.*

Proof. Because lemma 3.10 preserves the original extractor, applying it to $(\mathsf{P}^H, \mathsf{V}^H, \mathsf{E})$ yields

$$\Pr\left[(x, w') \in L \quad : \quad \begin{matrix} \rho \leftarrow \mathsf{Shave}^H_{\mathsf{P}_{n\text{-}\mathsf{Trim}}, n-1}(1^\lambda, x, w), \\ w' \leftarrow \mathsf{E}(1^\lambda, x, \rho, \mathbf{Q}_\mathsf{S}) \end{matrix}\right] \in \Omega(1/\operatorname{poly}(\lambda)) \tag{2}$$

for any $(x, w) \in L$, where $\mathbf{Q}_\mathsf{S}$ denotes the ordered list of queries made by Shave to H. Notice that in equation 2 the extractor does not have access to H, which implies that it cannot observe the divergence between H and oracle H^* emulated by Trim toward $\mathsf{P}_{n\text{-}\mathsf{Dist}}$.

We can construct an adversary for $\pi^H_{n\text{-}\mathsf{Dist}}$ as follows: the adversary corrupts $n-1$ parties at random and instructs them to behave as though they were honest, but records the queries that they make to H as $\mathbf{Q}_{\neg\delta}$. When the protocol is complete with the output ρ, the adversary runs $w' \leftarrow \mathsf{E}(1^\lambda, x, \rho, \mathbf{Q}_{\neg\delta})$ and outputs w'. By lemma 3.14, $\mathbf{Q}_{\neg\delta}$ and $\mathbf{Q}_\mathsf{S}$ are identically distributed, and by construction we know that $\mathsf{Shave}^H_{\mathsf{P}_{n\text{-}\mathsf{Trim}}, n-1}(1^\lambda, x, w)$ and $\pi^H_{n\text{-}\mathsf{Dist}}(1^\lambda, x, \mathsf{Share}_n(w))$ have identical output distributions. Thus equation 2 implies that the extractor produces (and the adversary outputs) a valid witness with probability $\Omega(1/\operatorname{poly}(\lambda))$; this satisfies definition 2.6. □

Corollary 3.17. *When the Pass [Pas03] transform is applied to any sigma protocol, and the ideal commitments are realized via the folkloric random-oracle based commitment protocol wherein a commitment is simply the oracle's output on the committed value concatenated with a salt,*[3] *then any protocol that* ORD*-computes the resulting non-interactive proof scheme among* n *parties is witness revealing in the presence of an adversary that corrupts* $n-1$ *parties, where* $n-1$ *is the number of random oracle queries made by the verifier.*

Proof. The explicit extractor given by Pass in the extended version of his work [Pas04] is specified to receive a list of the responses to the oracle queries made by the prover, in addition to the values queried, but we observe that when his scheme is constructed using the random-oracle-based straight-line commitment scheme that he suggests, a response-independent straight-line extractor is also implicitly possible, with some computational overhead. In Pass's scheme, the inputs to the oracle are the transcripts of a sigma protocol of the form (a, b, c), which has 2-special soundness. If the verifier accepts the proof, then with overwhelming probability the prover has made made queries on two transcripts (a, b, c) and (a, b', c') with the same statement x and first message, but

[3] This commitment scheme is specified by Pass himself.

different challenges and responses. Thus our response-independent extractor $\mathsf{E}_{\mathsf{Pass}}$ for the non-interactive proof can simply feed all pairs of query values into the sigma protocol's extractor, and each time check whether the output of the sigma protocol's extractor is a valid witness for the statement. When a valid witness is found (which it must be, with overwhelming probability), $\mathsf{E}_{\mathsf{Pass}}$ outputs it and halts. The total running time of $\mathsf{E}_{\mathsf{Pass}}$ is quadratic in the number of random oracle queries made by the prover, and thus it is PPT if the prover is. In order for theorem 3.16 to apply, it is sufficient for any response-independent extraction algorithm to exist; thus corollary 3.17 holds. □

Corollary 3.18. *When the Fischlin [Fis05], or Kondi-shelat [Ks22] transform is applied to any sigma protocol, then any protocol that* ORD*-computes the resulting non-interactive proof scheme among n parties is witness revealing in the presence of an adversary that corrupts $n-1$ parties, where $n-1$ is the number of random oracle queries made by the verifier.*

Proof. Fischlin's transform is formalized with response-independent straight-line extractability; thus, this corollary is a direct consequence of theorem 3.16. Likewise, Kondi and shelat specify an explicit extractor for their transform which uses only the queried values, and not the oracle's responses. □

Corollary 3.19. *When the Unruh [Unr15] transform is applied to any sigma protocol, then any protocol that* ORD*-computes the resulting non-interactive proof scheme among n parties is witness revealing in the presence of a* classical *adversary that corrupts $n-1$ parties, where $n-1$ is the number of random oracle queries made by the verifier.*

Proof. Unruh's transform is effectively the same as Pass's transform, except that it uses a different extraction technique and a different analysis in the presence of a malicious prover with quantum access to the random oracle. Against a classical adversarial prover, it is possible to use the same extractor as we described in our proof of corollary 3.17, and so this corollary follows from that one. □

Theorem 3.20. *If $(\mathsf{P}^H, \mathsf{V}^H, \mathsf{E}^H)$ is a straight-line extractable non-interactive proof scheme for a language L in the random oracle model, per definitions 2.1 and 2.2 such that P makes at least n queries to H and V checks at most $n-1$, then any protocol $\pi^H_{n\text{-}\mathsf{Dist}}(1^\lambda, x, \mathbf{w})$ that* ORD*-computes P among n parties with traceable query allocation under the mapping function M per definition 2.5 is witness-revealing per definition 2.6 in the presence of an adversary that corrupts $n-1$ parties.*

Proof. We can construct an adversary for $\pi^H_{n\text{-}\mathsf{Dist}}$ as follows: the adversary corrupts $n-1$ parties at random and instructs them to behave as though they were honest, but records the queries that they make to H as $\mathbf{Q}_{\neg\delta}$. When the protocol is complete with the output ρ, the adversary runs $w' \leftarrow \mathsf{E}^{H^\dagger}(1^\lambda, x, \rho, \mathbf{Q}_{\neg\delta})$ and emulates for it an instance of the oracle $H^\dagger$. On receiving a query Q on behalf of $H^\dagger$,

- If $Q \notin \mathbf{Q}_{\neg\delta}$ and Q is fresh, the adversary programs $H^\dagger(Q)$ to be a uniformly-sampled value
- If $Q \in \mathbf{Q}_{\neg\delta}$ and $\mathsf{M}(n, x, Q) = \delta$ and Q is fresh, the adversary programs $H^\dagger(Q)$ to be a uniformly-sampled value
- If $Q \in \mathbf{Q}_{\neg\delta}$ and $\mathsf{M}(n, x, Q) \neq \delta$ and Q is fresh, the adversary forwards the query to H and programs $H^\dagger(Q) \mapsto H(Q)$

and when $\mathsf{E}^{H^\dagger}$ halts, the adversary outputs w'. By lemma 3.14, $\mathbf{Q}_{\neg\delta}$ and $\mathbf{Q}_{\mathsf{S}}$ are identically distributed, and by construction we know that $\mathsf{Shave}^{H}_{\mathsf{P}_{n\text{-Trim}}, n-1}(1^\lambda, x, w)$ and $\pi^{H}_{n\text{-Dist}}(1^\lambda, x, \mathsf{Share}_n(w))$ have identical output distributions. In lemma 3.10, ρ is generated under the oracle H^* and extraction occurs under H, whereas when our adversary interacts with $\pi^{H}_{n\text{-Dist}}$, ρ is generated under H and extraction occurs under $H^\dagger$; thus we must show that the distributions of (H^*, H) and $(H, H^\dagger)$ are identical. This is easy to see: by construction, both pairs of oracles diverge on the queries in

$$\mathbf{Q}_\Delta = \{Q : Q \in \mathbf{Q}_\pi \wedge \mathcal{P}_\delta \text{ is the first party to query } Q \text{ in } \pi^{H}_{n\text{-Dist}} \text{ or } \pi^{H^*}_{n\text{-Dist}}\}$$

and in both cases, the two oracles return independently sampled uniform values. On all other queries, both pairs of oracles agree. Thus lemma 3.10 implies that the extractor produces (and the adversary outputs) a valid witness with probability $\Omega(1/\operatorname{poly}(\lambda))$; this satisfies definition 2.6. $\square$

4 A Constant-Fraction Construction

In section 3, we considered the case that an adversary is able to corrupt all but one of the parties in protocol that ORD-computes some prover. Given the inherent barriers that we demonstrated, it is natural to ask whether a more restricted adversary retains the same power, and in what regime it is possible to construct multi-prover non-interactive straight-line zero-knowledge schemes that are not witness revealing. In this section, we consider the next-most-powerful natural corruption threshold, which is corruption of a constant fraction of parties. Specifically, we consider the case that a semi-honest adversary corrupts $c \cdot n$ of the provers, for some positive constant $c < 1$. To our knowledge, no prior work has ever given an oracle-respecting distributed zero-knowledge prover in this setting such that the verifier must perform fewer than $n - 1$ oracle queries and a meaningful privacy notion is achieved. Working by example, we construct precisely such a scheme.

We focus our attention on a specific language—discrete logarithm—and for this language we construct a non-interactive zero-knowledge proof of knowledge that requires a number of verification queries dependent only upon the security parameter. We then give a protocol that ORD-computes our prover among any number of parties, and prove that so long as a constant fraction of those parties are honest, the honest parties' messages can be simulated *without knowledge of their shares of the witness* (see definition 4.3), which contradicts witness-revealingness (definition 2.6) if the discrete logarithm assumption holds.

4.1 Additional Definitions

The discrete logarithm relation $\mathcal{R}_{\mathsf{DL}}$ is defined over a group $\mathbb{G}$ of order q generated by some element G. We write group operations additively and by convention we use capitalized variables for group elements. Thus $\mathcal{R}_{\mathsf{DL}}(X, w) \mapsto 1$ if $w \cdot G = X$ or 0 otherwise.

Since we will be constructing a protocol in this section, we must give a richer definition of secret sharing than we have heretofore, which is satisfied by simple additive secret-sharing.

Definition 4.1. All-but-One Secret-Sharing

A pair of algorithms $(\mathsf{Share}_n, \mathsf{Recon}_n)$ is a secret sharing scheme for n parties with security against $n-1$ corruptions if they satisfy three properties:

1. Efficiency: Share_n is a probabilistic polynomial-time algorithm and Recon_n is a deterministic polynomial time algorithm.

2. Correctness and Completeness: For any x in the domain of Share_n

$$\Pr\left[\mathsf{Recon}_n(\mathsf{Share}_n(x)) = x\right] = 1$$

   where the probability is over the coins of Share_n.

3. Perfect Secrecy: If the range of Share_n is $\mathbb{S}^n$, then for any x in the domain of Share_n and any $\mathbf{C} \subset [n]$ such that $|\mathbf{C}| = n-1$,

$$\{\{s_i\}_{i\in\mathbf{C}} : \mathsf{Share}_n(x)\} = \mathcal{U}_{\mathbb{S}^{n-1}}$$

   where $\mathcal{U}_{\mathbb{S}^{n-1}}$ is the uniform distribution over $\mathbb{S}^{n-1}$.

In order to show that definition 2.6 is contradicted, we must give a formal notion of security for our protocol. We begin with a standard definition of zero-knowledge for single provers.

Definition 4.2. Zero-Knowledge [GMR85, GMW91]

Let L be a language in NP where membership of an instance $x \in \mathbb{X}$ can be verified by a witness $w \in \mathbb{W}$, and let $(\mathsf{P}^H, \mathsf{V}^H)$ be an efficient proof scheme where both P and V have access to the random oracle H. We say that $(\mathsf{P}^H, \mathsf{V}^H)$ is *zero-knowledge in the random oracle model* if for every PPT adversary $\mathcal{A}^H$ with oracle access to H there exists a PPT simulator algorithm $\mathsf{S}^{\mathcal{A}}$ that may reactively *program* H such that for any $(x, w) \in L$ and any non-uniform advice $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$,

$$\mathsf{View}_{\mathcal{A}}^{H,\mathsf{P}^H(1^\lambda,x,w)}(1^\lambda, x, z) \approx_{\mathrm{c}} \mathsf{S}^{\mathcal{A}}(1^\lambda, x, z)$$

over the random coins of $\mathcal{A}$, P, S, and H, where $\mathsf{View}_{\mathcal{A}}^{H,\mathsf{P}^H(1^\lambda,x,w)}(1^\lambda, x, z)$ represents the view of $\mathcal{A}^H(1^\lambda, x, z)$ during its interaction with $\mathsf{P}^H(1^\lambda, x, w)$.

If the two distributions are $\approx_{\mathrm{s}}$ rather than $\approx_{\mathrm{c}}$, then the protocol is said to be *statistically* zero-knowledge. If the two distributions are identical, then it is *perfectly* zero-knowledge.

We note that the above definition does *not* insist that the proof scheme be non-interactive, and that it permits the simulator to program the oracle. Pass [Pas03] proved that such programming is necessary. We extend the above definition in a minimal way to accommodate protocols with multipler provers, of which some threshold may be corrupt.

Definition 4.3. Multi-Prover ZK against t Corruptions

Let L be a language in NP where membership of an instance $x \in \mathbb{X}$ can be verified by a witness $w \in \mathbb{W}$, and let $(\mathsf{P}^H, \mathsf{V}^H)$ be an efficient proof scheme where both P and V have access to the random oracle H, and let $\pi^H_{n\text{-}\mathsf{Dist}}$ be a protocol among n parties that ORD-computes P^H per definition 2.4. We say that $\pi^H_{n\text{-}\mathsf{Dist}}$ has *multi-prover zero-knowledge in the random oracle model against t corruptions* if for every $\mathbf{C} \subset [n]$ such that $|\mathbf{C}| = t$ and every PPT adversary $\mathcal{A}^H_{\mathbf{C}}$ with oracle access to H that corrupts the protocol parties indexed by $\mathbf{C}$ there exists a PPT simulator algorithm $\mathsf{S}^{\mathcal{A}_{\mathbf{C}}}$ that may reactively *program* H such that for any $(x, w) \in L$ and $\{w_i\}_{i\in[n]} \leftarrow \mathsf{Share}_n(w)$, and any non-uniform advice $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$,

$$\mathsf{View}^{H,\left\{\mathcal{P}^H_j(1^\lambda,x,w_j)\right\}_{j\in[n]\setminus\mathbf{C}}}_{\mathcal{A}_{\mathbf{C}}}\left(1^\lambda, x, \{w_i\}_{i\in\mathbf{C}}, z\right) \approx_{\mathrm{c}} \mathsf{S}^{\mathcal{A}_{\mathbf{C}}}\left(1^\lambda, x, \{w_i\}_{i\in\mathbf{C}}, z\right)$$

over the random coins of $\mathcal{A}$, S, H, and $\mathcal{P}_j$ for $j \in [n] \setminus \mathbf{C}$, where

$$\mathsf{View}^{H,\left\{\mathcal{P}^H_j(1^\lambda,x,w_j)\right\}_{j\in[n]\setminus\mathbf{C}}}_{\mathcal{A}_{\mathbf{C}}}\left(1^\lambda, x, \{w_i\}_{i\in\mathbf{C}}, z\right)$$

represents the view of $\mathcal{A}^H_{\mathbf{C}}\left(1^\lambda, x, \{w_i\}_{i\in\mathbf{C}}, z\right)$ during its interaction with the honest parties $\mathcal{P}^H_j(1^\lambda, x, w_j)$ for $j \in [n] \setminus \mathbf{C}$ who participate in $\pi^H_{n\text{-}\mathsf{Dist}}$. If the two distributions are $\approx_{\mathrm{s}}$ rather than $\approx_{\mathrm{c}}$, then the protocol is said to be *statistically* zero-knowledge. If the two distributions are identical, then it is *perfectly* zero-knowledge.

Informally, if a protocol is both multi-prover zero-knowledge against t corruptions and witness-revealing in the presence of t corruptions, and Share_n information-theoretically hides w when only t shares are known (as definition 4.1 insists that it must), then an extractor E can be used to recover a witness w from the statement x, which may imply breaking a hardness assumption. In the case of our example in the next section, it would imply breaking the discrete logarithm assumption.

4.2 Proof by Committee

We begin by assuming the existence of a non-interactive response-independent straight-line extractable zero-knowledge proof of knowledge for the discrete logarithm relation. That is, let $(\mathsf{P}^H_{\mathsf{DL}}, \mathsf{V}^H_{\mathsf{DL}}, \mathsf{E}_{\mathsf{DL}}, \mathsf{S}_{\mathsf{DL}})$ satisfy definitions 2.1, 2.3, and 4.2 for the language L defined by $\mathcal{R}_{\mathsf{DL}}$. Such a scheme is straightforward to instantiate with Schnorr's protocol compiled with any straight-line extractable NIZK transform [Pas03, Fis05, Ks22]. We also assume the existence of an *additive* secret sharing scheme $(\mathsf{Share}, \mathsf{Recon})$ that satisfies definition 4.1, and we assume it can be applied over $\mathbb{Z}_q$ and $\mathbb{G}$ in a homomorphic fashion, always producing shares in the same domain as the input.[4] From any such basis we construct the following willfully-inefficient NIZK, which creates λ secret shares of the original witness and statement and proves knowledge of them individually:

Algorithm 4.4. Secret-Shared NIZKPoK of Discrete Logarithm

These algorithms are implicitly parameterized by the NIZK $(\mathsf{P}^H_{\mathsf{DL}}, \mathsf{V}^H_{\mathsf{DL}}, \mathsf{E}_{\mathsf{DL}}, \mathsf{S}_{\mathsf{DL}})$ and the additive secret-sharing scheme $(\mathsf{Share}_\lambda, \mathsf{Recon}_\lambda)$, and they have access to the random oracle H.

$\mathsf{P}^H_{\mathsf{DL}\lambda}(1^\lambda, X, w)$:

1. Sample $\{w_i\}_{i\in[\lambda]} \leftarrow \mathsf{Share}_\lambda(w)$ and compute $X_i := w_i \cdot G$ for every $i \in [\lambda]$.
2. For every $i \in [\lambda]$, compute $\rho_i \leftarrow \mathsf{P}^{H_i}_{\mathsf{DL}}(1^\lambda, X_i, w_i)$, where H_i is simply H with i prefixed to all queries.
3. Output $\{X_i, \rho_i\}_{i\in[\lambda]}$.

$\mathsf{V}^H_{\mathsf{DL}\lambda}(1^\lambda, X, \{X_i, \rho_i\}_{i\in[\lambda]})$:

1. Output 1 if $\sum_{i\in[\lambda]} X_i = X$ and $\mathsf{V}^{H_i}_{\mathsf{DL}}(1^\lambda, X_i, \rho_i) = 1$ for every $i \in [\lambda]$. Otherwise, output 0.

$\mathsf{E}_{\mathsf{DL}\lambda}(1^\lambda, \{X_i, \rho_i\}_{i\in[\lambda]}, \mathbf{Q})$:

1. For every $i \in [\lambda]$, let $\mathbf{Q}_i$ be the subset of queries in $\mathbf{Q}$ prefixed by i.
2. For every $i \in [\lambda]$, compute $w'_i \leftarrow \mathsf{E}_{\mathsf{DL}}(1^\lambda, X_i, \rho_i, \mathbf{Q}_i)$.
3. Output $w' := \mathsf{Recon}_\lambda\left(\{w'_i\}_{i\in[\lambda]}\right)$.

$\mathsf{S}_{\mathsf{DL}\lambda}(1^\lambda, X)$:

1. Initialize H as a uniformly random oracle.

[4] i.e. for any $w \in \mathbb{Z}_q$ and any $n \in \mathbb{N}^+$,

$$w \cdot G = \mathsf{Recon}_n\left(\{w_i \cdot G\}_{i\in[n]}\right) : \{w_i\}_{i\in[n]} \leftarrow \mathsf{Share}_n(w)$$

2. Compute $\{X_i\}_{i\in[\lambda]} \leftarrow \mathsf{Share}_\lambda(X)$.

3. For every $i \in [\lambda]$, compute $\rho_i \leftarrow \mathsf{S}_{\mathsf{DL}}(1^\lambda, X_i)$, programming H_i as required.

4. Output $\{X_i, \rho_i\}_{i\in[\lambda]}$.

Lemma 4.5. *If* $(\mathsf{P}^H_{\mathsf{DL}}, \mathsf{V}^H_{\mathsf{DL}}, \mathsf{E}_{\mathsf{DL}}, \mathsf{S}_{\mathsf{DL}})$ *is a response-independent straight-line extractable NIZK in the random oracle model for the discrete logarithm relation in* $\mathbb{G}$*, per definitions 2.1, 2.3, and 4.2, and* $(\mathsf{Share}_\lambda, \mathsf{Recon}_\lambda)$ *is an additive all-but-one secret sharing scheme per definition 4.1, then* $(\mathsf{P}^H_{\mathsf{DL}\lambda}, \mathsf{V}^H_{\mathsf{DL}\lambda}, \mathsf{E}_{\mathsf{DL}\lambda}, \mathsf{S}_{\mathsf{DL}\lambda})$ *is also a response-independent straight-line extractable NIZK in the random oracle model for the discrete logarithm relation in* $\mathbb{G}$*.*

Proof Sketch. Completeness follows directly from the completeness of $(\mathsf{P}^H_{\mathsf{DL}}, \mathsf{V}^H_{\mathsf{DL}})$ and $(\mathsf{Share}_\lambda, \mathsf{Recon}_\lambda)$. Soundness follows directly from the correctness of $(\mathsf{Share}_\lambda, \mathsf{Recon}_\lambda)$ over both $\mathbb{Z}_q$ and $\mathbb{G}$ and the soundness of $(\mathsf{P}^H_{\mathsf{DL}}, \mathsf{V}^H_{\mathsf{DL}})$.

Response-independent straight-line extraction follows from a straightforward reduction to the matching property of $(\mathsf{P}^H_{\mathsf{DL}}, \mathsf{V}^H_{\mathsf{DL}}, \mathsf{E}_{\mathsf{DL}})$. An adversary for $(\mathsf{P}^H_{\mathsf{DL}\lambda}, \mathsf{V}^H_{\mathsf{DL}\lambda}, \mathsf{E}_{\mathsf{DL}\lambda})$ who produces a proof with at least one trio $(X_i, \rho_i, \mathbf{Q}_i)$ such that $\mathsf{E}_{\mathsf{DL}}(1^\lambda, X_i, \rho_i, \mathbf{Q}_i)$ fails to produce a valid witness for X_i can trivially be transformed into an adversary for $(\mathsf{P}^H_{\mathsf{DL}}, \mathsf{V}^H_{\mathsf{DL}}, \mathsf{E}_{\mathsf{DL}})$.

Zero-knowedge follows from the fact that the outputs of $\mathsf{S}_{\mathsf{DL}\lambda}$ and $\mathsf{P}^H_{\mathsf{DL}\lambda}$ differ only in that the former computes ρ_i for every $i \in [\lambda]$ using S_{DL}, whereas the latter uses $\mathsf{P}^H_{\mathsf{DL}}$. The zero-knowledge property of $(\mathsf{P}^H_{\mathsf{DL}}, \mathsf{V}^H_{\mathsf{DL}}, \mathsf{S}_{\mathsf{DL}})$ implies that these distributions are indistinguishable. □

Next, we introduce an ideal functionality $\mathcal{F}^{\mathsf{Share}_\lambda,\mathsf{Recon}_n}_{\mathsf{RandReshare}}$ that will be used as a building block for the protocol that ORD-computes $\mathsf{P}^H_{\mathsf{DL}}$. The functionality takes n secret shares of a value from n parties, reconstructs the shared value, and then distributes new shares of the value to a randomly chosen size-λ subset of the parties.

Functionality 4.6. $\mathcal{F}^{\mathsf{Share}_\lambda,\mathsf{Recon}_n}_{\mathsf{RandReshare}}(n, \lambda, q)$

This functionality interacts with $n > \lambda$ parties, and is also parameterized by a prime q. It has oracle access to the reconstruction algorithm Recon_n and the sharing algorithm Share_λ, both over $\mathbb{Z}_q$.

Upon receiving $(\texttt{reshare}, \mathsf{sid}, w_i)$ from party $\mathcal{P}_i$ for all $i \in [n]$:

1. Compute $w := \mathsf{Recon}(\{w_i\}_{i\in[n]})$.

2. Sample $\{w'_j\}_{j\in[\lambda]} \leftarrow \mathsf{Share}_\lambda(w)$.

3. Sample $\{\mathcal{C}_j\}_{j\in[\lambda]} \subset \{\mathcal{P}_i\}_{i\in[n]}$ uniformly.

4. For each $j \in [\lambda]$ send $(\texttt{share}, \mathsf{sid}, w'_j)$ to party $\mathcal{C}_j$ as adversarially-delayed private output.

5. Send $(\texttt{reshared}, \mathsf{sid})$ to all parties as adversarially-delayed output, and ignore any future messages with the same sid.

There may be efficient custom protocols to realize $\mathcal{F}^{\mathsf{Share}_\lambda,\mathsf{Recon}_n}_{\mathsf{RandReshare}}$, but for our purposes it suffices that it can be realized generically with resilience to $n-1$ corruptions [GMW87]. We are now ready to construct a protocol that ORD-computes $\mathsf{P}^H_{\mathsf{DL}\lambda}$ with multi-prover zero-knowledge. To simplify our protocol description and analysis, we make the assumption that $n > \lambda/(1-c)$. This assumption is easy to remove: in case $n \leq \lambda/(1-c)$ parties wish to run the protocol, they can instead run a version of the protocol where each real party controls m virtual parties (so that $m \cdot n > \lambda/(1-c)$)—such a virtualization preserves the same fraction of corruptions, and therefore retains the security guarantees of the original protocol.

Protocol 4.7. $\pi^H_{\mathsf{DL}\lambda}(1^\lambda, X, \mathbf{w}, n, \mathsf{sid})$**: ORD-$\mathsf{P}^H_{\mathsf{DL}\lambda}$**

This protocol distributes the computation of $\mathsf{P}^H_{\mathsf{DL}\lambda}$ (while making oracle use of H) amongst n parties such that $n > \lambda/(1-c)$. Each party $\mathcal{P}_i$ begins with the common input $X \in \mathbb{G}$ and private input $w_i \in \mathbb{Z}_q$ such that $\mathbf{w} = \{w_i\}_{i\in[n]}$ is in the image of $\mathsf{Share}(w)$.

1. Every party $\mathcal{P}_i$ for $i \in [n]$ sends $(\texttt{reshare}, \mathsf{sid}, w_i)$ to $\mathcal{F}^{\mathsf{Share}_\lambda,\mathsf{Recon}_n}_{\mathsf{RandReshare}}(n, \lambda, q)$, and waits for a response.

2. If $\mathcal{P}_i$ receives $(\texttt{share}, \mathsf{sid}, w'_j)$ from $\mathcal{F}^{\mathsf{Share}_\lambda,\mathsf{Recon}_n}_{\mathsf{RandReshare}}(n, \lambda, q)$, it computes $X_j := w'_j \cdot G$ and $\rho_j \leftarrow \mathsf{P}^H_{\mathsf{DL}}(1^\lambda, X_j, w_j)$ and then broadcasts $(\texttt{proof-shard}, \mathsf{sid}, X_j, \rho_j)$ to the other parties.

3. The parties collect and output $\{X_j, \rho_j\}_{j\in[\lambda]}$

Lemma 4.8. *For any positive rational $c < 1$ and any n such that $n > \lambda/(1-c)$, protocol $\pi^H_{\mathsf{DL}\lambda}$ ORD-computes $\mathsf{P}^H_{\mathsf{DL}\lambda}$ per definition 2.4 and achieves multi-prover zero-knowledge in the presence of $c \cdot n$ corrupt parties per definition 4.3.*

Proof. It is easy to verify by inspection that $\pi^H_{\mathsf{DL}\lambda}$ ORD-computes $\mathsf{P}^H_{\mathsf{DL}\lambda}$: the algorithms are exactly the same, except that in $\pi^H_{\mathsf{DL}\lambda}$ the various sub-proofs are computed by separate parties instead of by a single, unified prover.

To satisfy definition 4.3, we define a simulator $\mathcal{S}^{\mathcal{A}_\mathbf{C}}_{\mathsf{DL}\lambda}(1^\lambda, X, \{w_i\}_{i\in\mathbf{C}}, n, \mathsf{sid})$, which works in the following way:

1. Initialize H as a uniformly random oracle.

2. On receiving $(\texttt{reshare}, \mathsf{sid}, w_i)$ from all corrupt parties on behalf of $\mathcal{F}^{\mathsf{Share}_\lambda,\mathsf{Recon}_n}_{\mathsf{RandReshare}}$, sample $\{\mathcal{C}_j\}_{j\in[\lambda]} \subset \{\mathcal{P}_i\}_{i\in[n]}$ uniformly subject to $\mathcal{C}_h$ being an honest party, for at least one $h \in [\lambda]$.

3. For every $i \in [\lambda]$ such that $\mathcal{C}_i$ is corrupt, sample $w'_i \leftarrow \mathbb{Z}_q$ uniformly, compute $X_i := w'_i \cdot G$, and send $(\texttt{share}, \mathsf{sid}, w'_i)$ to $\mathcal{C}_i$ on behalf of $\mathcal{F}^{\mathsf{Share}_\lambda,\mathsf{Recon}_n}_{\mathsf{RandReshare}}$.

4. For every $j \in [\lambda]$ such that $\mathcal{C}_j$ is honest, sample $X_j \leftarrow \mathbb{G}$ uniformly subject to $\mathsf{Recon}_\lambda(\{X_k\}_{k\in[\lambda]}) = X$ and compute $\rho_j \leftarrow \mathsf{S}_{\mathsf{DL}}(1^\lambda, X_j)$, programming H_j as required.
5. For every $j \in [\lambda]$ such that $\mathcal{C}_j$ is honest, broadcast $(\texttt{proof-shard}, \mathsf{sid}, X_j, \rho_j)$ to the corrupt parties.

As in our proof of lemma 4.5, the zero-knowledge property of $(\mathsf{P}^H_{\mathsf{DL}}, \mathsf{V}^H_{\mathsf{DL}}, \mathsf{S}_{\mathsf{DL}})$ implies that these distributions of ρ_j for $j \in [\lambda]$ such that $\mathcal{C}_j$ is honest are indistinguishable in the adversary's view of the protocol (where they are generated by $\mathsf{P}^H_{\mathsf{DL}}$) and the adversary's view of the simulation (where they are generated by S_{DL}). The only remaining case in which the adversary can distinguish $\mathcal{S}_{\mathsf{DL}\lambda}$ from $\pi^H_{\mathsf{DL}\lambda}$ is the event in which $\mathcal{F}^{\mathsf{Share}_\lambda,\mathsf{Recon}_n}_{\mathsf{RandReshare}}$ chooses an entirely corrupt committee. By construction, this never happens when the adversary is interacting with $\mathcal{S}_{\mathsf{DL}\lambda}$, and it also cannot happen when the adversary interacts with $\pi^H_{\mathsf{DL}\lambda}$ and $c \cdot n < \lambda$. When the adversary interacts with $\pi^H_{\mathsf{DL}\lambda}$ and $c \cdot n \geq \lambda$, the probability that a fully-corrupt committee will be sampled is upper-bounded by

$$\frac{\binom{c\cdot n}{\lambda}}{\binom{n}{\lambda}} \leq \left(\frac{c\cdot n}{n-\lambda}\right)^\lambda = \left(\frac{c\cdot d\cdot\lambda}{d\cdot\lambda-\lambda}\right)^\lambda = \left(\frac{c}{1-1/d}\right)^\lambda \in \frac{1}{\mathsf{exp}(\lambda)}$$

where $d\cdot\lambda = n$ and we know by construction that $d > 1/(1-c)$. Since this event fully characterizes the separation between the real protocol from the simulation, and it occurs with negligible probability, the theorem is proven. □

Theorem 4.9. *If the discrete logarithm assumption holds in* $\mathbb{G}$, *then for any positive rational* $c < 1$ *and any* n *such that* $n > \lambda/(1-c)$, $\pi^H_{\mathsf{DL}\lambda}$ *is* not *witness revealing per definition 2.6 in the presence of* $c \cdot n$ *corrupt parties.*

Proof. Our proof is by contradiction. Suppose that $\pi^H_{\mathsf{DL}\lambda}$ were witness-revealing in the presence of $c \cdot n$ corrupt parties; this implies that with noticeable probability a valid witness could be extracted from the corrupt parties' views in any execution that yields an accepting proof. Since $\pi^H_{\mathsf{DL}\lambda}$ ORD-computes $\mathsf{P}^H_{\mathsf{DL}\lambda}$ with respect to the verifier $\mathsf{V}^H_{\mathsf{DL}\lambda}$, and it has multi-prover zero-knowledge in the presence of $c \cdot n$ corrupt parties as shown in lemma 4.8, there exists a simulator ($\mathcal{S}_{\mathsf{DL}\lambda}$) that can generate accepting proofs on any statement $X \in \mathbb{G}$ along with the views of $c \cdot n$ corrupt parties *without* access to a witness for X. Since the proofs and views generated by the simulator are indistinguishable from the ones produced by a protocol, applying the witness-revealing extractor must still yield a valid witness w such that $w \cdot G = X$ with noticeable probability. This contradicts the discrete logarithm assumption in $\mathbb{G}$; thus our theorem holds. □

Since we know that $\pi^H_{\mathsf{DL}\lambda}$ ORD-computes $\mathsf{P}^H_{\mathsf{DL}\lambda}$ with respect to the verifier $\mathsf{V}^H_{\mathsf{DL}\lambda}$, and $\mathsf{V}^H_{\mathsf{DL}\lambda}$ makes a number of oracle queries that is independent of n, we can conclude that no theorem equivalent to theorems 3.16 or 3.20 can hold *in general* for adversaries that only corrupt a constant fraction of provers.

5 Related Work

In this section, we discuss work related to our bounds. Our overview begins with results that are loosely related to our problem setting, but that address different adversarial models and security goals. We then review the few constructions in the literature that match our setting, and show that our bounds on proof size and query complexity are respected. We conclude with a brief discussion of recent related work on straight-line extraction, emphasizing the impact of our bounds on various settings, results, and directions for future work.

Several recent works [CB17, BBC+19, AGJ+21] have constructed efficient non-interactive zero-knowledge proofs in settings where the *statement* is distributed among many verifiers, and not known in its entirety to any of them, but few works have addressed the setting where the *witness* is distributed among mutually-distrustful provers, as ours does. All existing works in the multi-prover setting fall into two categories: either they make additional trusted setup assumptions, such as the existance of a CRS for which the extractor knows a trapdoor, or they permit the size of their proofs (and thus their verification time) to grow linearly with the number of provers n, as our bound says they must.

Ozdemir and Boneh [OB22] adapt various zk-SNARK constructions based on polynomial interactive oracle proofs [BFS20, CHM+20] to the multi-prover setting. Of the four schemes they consider, only “Fractal” [COS20] avoids the use of a CRS, using hash-based vector commitments instead. Ozdemir and Boneh explicitly note that if the protocol is constructed in an oracle-respecting way, then its proof size and verifier computation cost grow linearly with n, and suggest that if it is not constructed in an oracle-respecting way, then the provers' computation costs are unacceptable. Cui et a. [CZC+21] also present an oracle-respecting multi-prover scheme; their proof size grows with n, as our bound predicts.

Wu et al. [WZC+18] describe a protocol that distributes Groth's CRS-based zk-SNARK [Gro16] and observe that the cost of the trusted setup procedure required for such schemes is substantial and grows with the circuit complexity of the statement to be proven, making their scheme and others with similar CRSes prohibitively expensive in many practical settings. Several works have introduced protocols for securely sampling such CRSes in a distributed fashion [BGM17, KMSV21, CDKs22], but the practicality of such techniques remains to be established when the number of participants or the size of the circuit to be proven is large.

Dziembowski, Faust, and Lizurej [DFL23] introduce the notion of *individual cryptography*, or proofs that cannot be efficiently distributed. We stress that their goal is much different than ours: we wish to show that any protocol that distributes a certain kind of proof must leak the witness in certain parameter regimes, and our notions of efficiency are purely asymptotic. They construct a specific hashing-based proof which they claim cannot be securely distributed among multiple honest (but mutually untrusting) parties at all, within some concrete time-bound.

Finally, several recent works have highlighted the importance of straight-line extraction, for instance in achieving universally composable NIZKs secure against static [LR22b] or adaptive [CSW22, LR22a] corruptions and constructing zk-SNARKs [GKO$^+$22] in the global random-oracle model, as well as NIZKs from lattices in the quantum random-oracle model [Kat21]. Straight-line extraction is also relevant in the context of non-linear proof systems, such as recursive arguments [KST21], where the rewind-and-fork proof technique of Pointcheval and Stern [PS96] causes a prohibitive blow-up in the complexity of the security reduction.

Acknowledgements

The authors of this work were supported by the ERC under projects NTSC (742754), SPEC (803096), and HSS (852952), by ISF grant 2774/2, by AFOSR award FA9550-21-1-0046, and by the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM). They wish to thank Eysa Lee for her participation in the early stages of this project.

References

[AGJ$^+$21] Surya Addanki, Kevin Garbe, Eli Jaffe, Rafail Ostrovsky, and Antigoni Polychroniadou. Prio+: Privacy preserving aggregate statistics via boolean shares. Cryptology ePrint Archive, Report 2021/576, 2021. https://eprint.iacr.org/2021/576.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017.

[BBC$^+$19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 67–97. Springer, Heidelberg, August 2019.

[BCR$^+$19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016.

[BFH+20] Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. Ligero++: A new optimized sublinear IOP. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 2025–2038. ACM Press, November 2020.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020.

[BG90] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 194–211. Springer, Heidelberg, August 1990.

[BGM17] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017. https://eprint.iacr.org/2017/1050.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In Aditya Akella and Jon Howell, editors, *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, pages 259–282. USENIX Association, 2017.

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1825–1842. ACM Press, October / November 2017.

[CDG+18] Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. The wonderful world of global random oracles. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 280–312. Springer, Heidelberg, April / May 2018.

[CDKs22] Ran Cohen, Jack Doerner, Yashvanth Kondi, and abhi shelat. Guaranteed output in $O(\sqrt{n})$ rounds for round-robin sampling protocols. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 241–271. Springer, Heidelberg, May / June 2022.

[CGG+20] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1769–1787. ACM Press, November 2020.

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

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 597–608. ACM Press, November 2014.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 769–793. Springer, Heidelberg, May 2020.

[CSW22] Ran Canetti, Pratik Sarkar, and Xiao Wang. Triply adaptive UC NIZK. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 466–495. Springer, Heidelberg, December 2022.

[CZC+21] Hongrui Cui, Kaiyi Zhang, Yu Chen, Zhen Liu, and Yu Yu. MPC-in-multi-heads: A multi-prover zero-knowledge proof system - (or: How to jointly prove any NP statements in ZK). In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *ESORICS 2021, Part II*, volume 12973 of *LNCS*, pages 332–351. Springer, Heidelberg, October 2021.

[DFL23] Stefan Dziembowski, Sebastian Faust, and Tomasz Lizurej. Individual cryptography. Cryptology ePrint Archive, Report 2023/088, 2023. https://eprint.iacr.org/2023/088.

[DKLs19] Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *2019 IEEE Symposium on Security and Privacy*, pages 1051–1066. IEEE Computer Society Press, May 2019.

[DOK$^+$20] Anders P. K. Dalskov, Claudio Orlandi, Marcel Keller, Kris Shrishak, and Haya Shulman. Securing DNSSEC keys via threshold ECDSA from generic MPC. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 654–673. Springer, Heidelberg, September 2020.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, August 2005.

[GKO$^+$22] Chaya Ganesh, Yashvanth Kondi, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Witness-succinct universally-composable SNARKs. Cryptology ePrint Archive, Report 2022/1618, 2022. https://eprint.iacr.org/2022/1618.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for Boolean circuits. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 1069–1083. USENIX Association, August 2016.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

[HMPs14] Susan Hohenberger, Steven Myers, Rafael Pass, and abhi shelat. ANONIZE: A large-scale anonymous survey system. In *2014 IEEE Symposium on Security and Privacy*, pages 375–389. IEEE Computer Society Press, May 2014.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

[Kat21] Shuichi Katsumata. A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure NIZKs. Cryptology ePrint Archive, Report 2021/927, 2021. https://eprint.iacr.org/2021/927.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018.

[KMSV21] Markulf Kohlweiss, Mary Maller, Janno Siim, and Mikhail Volkhov. Snarky ceremonies. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 98–127. Springer, Heidelberg, December 2021.

[Ks22] Yashvanth Kondi and abhi shelat. Improved straight-line extraction in the random oracle model with applications to signature aggregation. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 279–309. Springer, Heidelberg, December 2022.

[KST21] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. Cryptology ePrint Archive, Report 2021/370, 2021. https://eprint.iacr.org/2021/370.

[Lin22] Yehuda Lindell. Simple three-round multiparty schnorr signing with full simulatability. Cryptology ePrint Archive, Report 2022/374, 2022. https://eprint.iacr.org/2022/374.

[LN18] Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1837–1854. ACM Press, October 2018.

[LR22a] Anna Lysyanskaya and Leah Namisa Rosenbloom. Efficient and universally composable non-interactive zero-knowledge proofs of knowledge with security against adaptive corruptions. Cryptology ePrint Archive, Report 2022/1484, 2022. https://eprint.iacr.org/2022/1484.

[LR22b] Anna Lysyanskaya and Leah Namisa Rosenbloom. Universally composable Σ-protocols in the global random-oracle model. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 203–233. Springer, Heidelberg, November 2022.

[MPs19] Antonio Marcedone, Rafael Pass, and abhi shelat. Minimizing trust in hardware wallets with two factor signatures. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 407–425. Springer, Heidelberg, February 2019.

[OB22] Alex Ozdemir and Dan Boneh. Experimenting with collaborative zk-SNARKs: Zero-knowledge proofs for distributed secrets. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022*, pages 4291–4308. USENIX Association, August 2022.

[Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Heidelberg, August 2003.

[Pas04] Rafael Pass. Alternative variants of zero-knowledge proofs. `https://www.cs.cornell.edu/~rafael/papers/raf-lic.pdf`, 2004.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Heidelberg, May 1996.

[Unr15] Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 755–784. Springer, Heidelberg, April 2015.

[WZC$^+$18] Howard Wu, Wenting Zheng, Alessandro Chiesa, Raluca Ada Popa, and Ion Stoica. DIZK: A distributed zero knowledge proof system. In William Enck and Adrienne Porter Felt, editors, *USENIX Security 2018*, pages 675–692. USENIX Association, August 2018.