Straight-Line Knowledge Extraction for Multi-Round Protocols

Lior Rotem* and Stefano Tessaro**

Abstract. The Fiat-Shamir (FS) transform is the standard approach to compiling interactive proofs into non-interactive ones. However, the fact that knowledge extraction typically requires rewinding limits its applicability without having to rely on further heuristic conjectures. A better alternative is a transform that guarantees straight-line knowledge extraction. Two such transforms were given by Pass (CRYPTO '03) and Fischlin (CRYPTO '05), respectively, with the latter giving the most practical parameters. Pass's approach, which is based on cut-and-choose, was also adapted by Unruh (EUROCRYPT '12, '14, '15) to the quantum setting, where rewinding poses a different set of challenges. All of these transforms are tailored at the case of three-round Sigma protocols, and do not apply to a number of popular paradigms for building succinct proofs (e.g., those based on folding or sumcheck) which rely on multi-round protocols.

This work initiates the study of transforms with straight-line knowledge extraction for *multi-round* protocols. We give two transforms, which can be thought of as multi-round analogues of those by Fischlin and Pass. Our first transform leads to more efficient proofs, but its usage applies to a smaller class of protocols than the latter one. Our second transform also admits a proof of security in the Quantum Random Oracle Model (QROM), making it the first transform for multi-round protocols which does not incur the super-polynomial security loss affecting the existing QROM analysis of the FS transform (Don et al., CRYPTO '20).

1 Introduction

The Fiat-Shamir (FS) transform [33] is the most popular approach to making zero-knowledge proofs *non-interactive*. The FS transform incurs no overhead, in that if the underlying protocol is a public-coin interactive proof, the resulting proof size equals the bit length of the communication from the prover to the verifier. While the transformation was initially cast in the context of three-round Sigma protocols, it easily generalizes to an arbitrary number of rounds.

The main technical issue with the FS transform is that it requires *rewinding* to achieve knowledge extraction (in the random oracle model (ROM) [8]), i.e., one needs to run a successful prover multiple times (at least twice) to extract a witness for the given instance–this argument typically relies on the well-known Forking Lemma [58, 7]. This is not only an issue with tightness and concrete parameters, but also comes with other undesirable features, in that rewinding is often at odds with proving security under composition, either in the concurrent execution of protocols (e.g., when proving their UC security [21]) or in recursive composition of zero-knowledge proofs [10]. Shoup and Gennaro [62] also were first to point out other issues unrelated to composition: They observed that the single-ciphertext variant of the Naor-Yung paradigm [56] due to Rackoff and Simon [59] cannot be proved secure when extraction for the underlying NIZK requires rewinding.

The situation is even *worse* with multi-round protocols. Only recently, a non-trivial analysis of the FS transform was given [6, 68] by using a very complex rewinding argument. The resulting soundness guarantees are still weaker than the tight ones obtained in ideal-model proofs [38].

* Stanford University. Email: `lrotem@cs.stanford.edu`.
** University of Washington. Email: `tessaro@cs.washington.edu`.

Straight-line extraction. A more flexible alternative is a transformation that offers *straight-line extractability.* In the context of the ROM, this means that the RO queries of a successful prover, which are observable by a knowledge extractor, already provide enough information to reconstruct the witness, *without* the need of rewinding. Extending the study of transforms with straight-line extractability is the main objective of this paper, but before stating our contributions, we briefly review prior work.

Pass [57] used cut-and-choose techniques to devise the first transform with straight-line extractability, which applies in particular to any Sigma protocol with *special soundness*, i.e., a Sigma protocol with the property that a witness can be extracted from any two accepting transcripts $(a, c, z), (a, c', z')$ such that $c \neq c'$. In Pass's transform, the prover initially commits to 2^ℓ successful transcripts (a, c_i, z_i) of the protocol, all sharing the first message a, but with different challenges c_i's. For succinctness, this can be be done with a Merkle Tree (although Pass's version did not), and the value of the root of the tree is hashed to obtain an index $i^* \in \{1, \ldots, 2^\ell\}$. The final proof then contains the root of the tree (i.e., the commitment to the 2^ℓ transcripts), along with (a, c_{i^*}, z_{i^*}), and the corresponding opening proof. Intuitively, *either* the prover commits to at least two accepting transcripts (with different challenges), which can be extracted for the prover's RO queries, *or* the prover can only convince of the proof validity with probability at most $2^{-\ell}$. One then needs to parallel repeat the proof λ/ℓ times to achieve soundness error $2^{-\lambda}$. If the original Sigma protocol generates B-bit transcripts, this leads in particular to proof size of $(\lambda\ell + B) \cdot \lambda/\ell = \lambda^2 + B\lambda/\ell$ bits. Unruh [63, 64, 65] gives a version of this transform with security against quantum attackers, in the quantum random oracle model (QROM) [12], which however does not allow for a succinct commitment. The commitment was then made succinct by Don et al. [31].

A more efficient transformation has been proposed by Fischlin [34]. To generate a valid proof, the prover is asked to solve a "proof of work," i.e., given the initial message a, the prover attempts to find a transcript (a, c, z) whose hash starts with ℓ 0's, and this transcript will end up being the final proof. A malicious prover that only queries a *single* transcript (a, c, z) will only succeed with probability $2^{-\ell}$, and therefore, any prover succeeding with higher probability needs to make queries corresponding to two valid transcripts associated with different challenges. Then, one repeats λ/ℓ times to achieve soundness error $2^{-\lambda}$, and the proof size is now $B \cdot \lambda/\ell$, i.e, compared to the cut-and-choose approach, we have now removed the λ^2 term, which is usually dominating.

This paper: Straight-line extraction for multi-round protocols. A key point is that all aforementioned existing transformations only apply to Sigma protocols with *three rounds.* However, several prominent proof systems are not in this class, and specifically require a larger number of rounds. A typical example consists of protocols based on *folding*, initially proposed by Bootle et al. [14], further optimized in [18], and then extended to a number of settings such as bilinear pairings [48, 20], lattices [17, 11], unknown order groups [19], and then generalized to general homomorphisms [3, 4, 5]. These protocols recursively reduce, at each step, proving a statement of size n into proving a smaller statement of size n/k, and the resulting round complexity is typically $O(\log(n)/\log(k))$. Another important class of multi-round proof systems stem from the Sumcheck [50] protocol, which underlies a number of succinct proof systems (see [67, 69, 60, 15, 23, 46] and the references therein). Ideas from both Sumcheck and folding protocols where then abstracted by a common abstraction of sumcheck arguments [16].

It is natural to want to use these protocols in applications where straight-line extractability is necessary. The lack of a suitable transform raises a natural question, namely:

Can we design a general transforms for multi-round protocols that achieves straight-line knowledge extraction?

This paper answers this question in the affirmative. We provide in fact two transforms. The first one is a generalization of Fischlin's transform to multi-round protocols which satisfy a generalized notion of special soundness. It yields the smallest proofs, but also only applies to protocols with slightly lower round complexity than those to which our second transform applies, which is a generalization of Pass's transform to multiple rounds. For this latter transformation, we also give an analysis in the Quantum Random Oracle model (QROM), and show that it applies to a broader range of parameters than our generalized Fischlin transform. In particular, this is the first transformation for multi-round protocols in the QROM without a super-polynomial security loss for protocols with super-constant number of rounds.

First transformation. Our first transformation targets $(2s+1)$-round protocols that achieve a generalized notion of special soundness, which can be cast as follows: Imagine we are given transcripts of form $(a_1, c_1, a_2, c_2, ..., c_s, a_{s+1})$. We can naturally arrange them in a so-called *tree of transcripts*, where nodes are labeled by prover messages and edges by challenges. Also, it helps here to think of responses as being unique, i.e., there is a unique prover response a_{i+1} that extends a partial transcript $(a_1, c_1, a_2, c_2, \ldots, a_i, c_i)$. Then, we say a protocol achieves $(n_1, \ldots, n_s)$-special soundness if a witness can be efficiently extracted from any transcript tree where all nodes at depth $i \in \{1, \ldots, s\}$ have degree n_i.

One could try to first naïvely extend Fischlin's idea, in that the prover generates a number of transcripts $(a_1, c_1, a_2, c_2, ..., c_s, a_{s+1})$, all starting with the same value a_1, and hashes them until one of them produces a hash starting with ℓ 0's. This transcript would then be the actual proof. This prover will be successful, in expectation, after 2^ℓ attempts. Unfortunately, extraction fails when adopting this strategy, and this is due to the much stronger soundness requirement. Indeed, a malicious prover can simply fix the first $2s-1$ rounds of a transcript to some sequence $(a_1, c_1, a_2, c_2, ..., c_{s-1}, a_s)$,, and then create about 2^ℓ completions $(a_1, c_1, a_2, c_2, ..., c_{s-1}, a_s, c_s^{(i)}, a_{s+1}^{(i)})$ for $i = 1, 2, \ldots$, until one of them hashes to 0^ℓ. Clearly, it is not possible to extract a tree of transcripts from the RO queries, which in turn would allow us to compute a witness. Somehow, we have to instead force the prover to follow a tree structure when querying the RO.

Our key insight here is to bound the range from which the challenges are drawn, in particular the i-th challenge should be drawn from a set of size $k_i > n_i$, where the k_i's are chosen to be suitably small, e.g., $n_i = ck_i$ for some constant c. Through a combinatorial lemma, we are going to show that surprisingly, carefully tuning these parameters leads to both an efficient prover as well a desirable level of knowledge soundness error, which we then amplify using parallel repetition. One caveat of this transformation is that, in order for the prover to remain efficient, the round complexity of the protocol can be at most $\Theta(\log \lambda / \log \log \lambda)$. Many of the folding schemes can be instantiated to achieve this round complexity (by choosing a suitable k, in the notation we used above), but not all. This fact motivates in part looking at our second transformation.

Second transformation. Our second transformation is in some sense the natural generalization of Pass's transformation [57] to multi-round protocols. Here, the idea is to commit to a tree T of transcripts with degrees $n_1, n_2, \ldots, n_s$ using a vector commitment. Here, crucially, we interpret the tree as a vector of prover messages, with the corresponding challenges being generated deterministically (either by following a pre-defined deterministic pattern, or from a random oracle to achieve zero-knowledge). Then, from the commitment, we deterministically derive the index of a path from the root to the leaves, which is then opened, and included in the proof, along with the valid proof of opening.

The advantage of this transformation is that the prover complexity only scales with the size of the tree, which can generally be polynomial for protocols with $O(\log \lambda)$ rounds. The disadvantage is that it gives a larger proof size due to the inclusion of the opening proofs. However, we also abstract the transformation as using a generic vector commitment, as opposed to using Merkle Trees, and one could reduce the proof size by using pairing-based vector commitments (e.g., based on KZG [43]), at the cost of introducing a non-transparent setup.

QROM security. We also show that our second transformation, when the vector commitment is specificially instantiated via a RO-based Merkle Tree, also admits a proof of security in the QROM, which we give using techniques by Chiesa et al. [26] and Zhandry [70]. This implies in particular the proof for the special case of the succinct Unruh transform for three-round Sigma protocols proved by Don et al. [31]. In contrast, our first transform is subject to the same limitations as Fischlin's transform in the QROM [2].

This is particularly relevant because we do not have any good multi-round transformation in the QROM to start with, regardless of the issue of straight-line extractability (although, in the quantum setting, we can benefit even more heavily from straight-line extractability due to the challenges in handling rewinding). Indeed, Unruh's transform [63, 64, 65], as originally stated, only applies to Sigma protocols, and was later only extended to 5-round protocols [24]. In contrast, while the FS transform has been analyzed in the QROM [30, 29], the concrete security loss is of the order q^{2s+1}, for a $(2s+1)$-round protocols and q is the number of RO queries, and this gives no security guarantees for super-constant number of rounds. In particular, no quantum analogue of the tighter analysis of Attema et al. [6] is known.

Other related work. There have been works aimed at improving the concrete complexity of the Fischlin transform for the special case of Sigma protocols. These works are orthogonal to ours, and aim at obtaining better practical parameters and/or achieving other goals thanks to straight-line extraction. Kondi and shelat [45] give an improvement where the inversion proof-of-work is replaced by a collision-finding one, thus achieving lower prover complexity. Chen and Lindell [25] also study the concrete parameters of the Fischlin transform, and adapt it to n-special soundness, i.e., the case where n transcripts are needed to extract a witness. Lysyanskaya and Rosenblum [51] used a variant of Fischlin's transform to formally implement a NIZK functionality in the UC framework [21]. Dagdelen and Venturi [28] revisited the security of digital signatures built from Fischlin's transform.

Straight-line extraction has been used in ideal models other than the ROM. Ghoshal and Tessaro [38] leverage straight-line extraction to give tight bounds for folding-type arguments using the Algebraic Group Model (AGM) [35]. Earlier on, Fuchsbauer, Plouviez, and Seurin [36] used a simpler version of this technique to prove tight bounds for Schnorr signatures in the AGM.

In the QROM, Chiesa, Manohar, and Spooner [26], building on techniques from Zhandry [70], proved that the the Micali [55] and BCS [9] non-interactive succinct arguments (build from PCPs and IOPs, respectively) are knowledge sound. They did not consider, however, the task of compiling general interactive protocols into non-interactive ones in the QROM.

2 Preliminaries

2.1 Special-Sound Multi-Round Proofs

We start by defining multi-round generalizations of special sound arguments. Our definitions follow those of Attema, Fehr, and Klooß [6]. We assume familiarity with the basic notions of interactive proofs and arguments, public coin protocols and knowledge soundness, and the reader is referred to Goldreich [41] and Boneh and Shoup [13] for the necessary background.

In this paper, we will consider $(2s+1)$-round[1] public coin protocols for $s \geq 1$, with the following canonical structure: In all odd rounds, the prover P sends a message to the verifer V, and in all even rounds, V sends a random challenge from a corresponding challenge space to P. Finally, the verifier applies a deterministic function onto the input statement and the transcript of the interaction to determine its final output. We will overload notation and use V to denote this function. That is, the output of the verifier on input x and transcript t will be denoted by $V(x,t)$. If $V(x,t) = 1$, we say that t is an accepting transcript for x.

To define special soundness for such protocols, we will first need to introduce the notion of a "transcript tree".

Definition 1. *Let $s \geq 1$ and let (P,V) be an $(2s+1)$-round public coin protocol with challenge spaces $\mathcal{C}_1, \ldots, \mathcal{C}_s$, and let $n_1, \ldots, n_s \in \mathbb{N}$. Let $\mathcal{T} = (t_{i_1,\ldots,i_s})_{\forall j \in [s],\ i_j \in [n_j]}$ be a vector of N transcripts for (P,V). We say that $\mathcal{T}$ is a $(n_1, \ldots, n_s)$-*transcript tree *if for any $(i_1, \ldots, i_s) \neq (i'_1, \ldots, i'_s)$ in $[n_1] \times \cdots \times [n_s]$, the following conditions hold:*

1. $(a_1, c_1, \ldots, c_j, a_{j+1}) = (a'_1, c'_1, \ldots, c'_j, a'_{j+1})$.
2. $c_{j+1} \neq c'_{j+1}$.

where $t_{i_1,\ldots,i_s} = (a_1, c_1, \ldots, a_s, c_s, a_{s+1})$ and $t_{i'_1,\ldots,i'_s} = (a'_1, c'_1, \ldots, a'_s, c'_s, a'_{s+1})$ and $j \in \{0, \ldots, s-1\}$ is the maximal index for which $(i_1, \ldots, i_j) = (i'_1, \ldots, i'_j)$.

We may sometimes also call a $(n_1, \ldots, n_s)$-transcript tree a $(n_1, \ldots, n_s)$-*special-soundness tree*.

Intuitively, it is helpful to think of a $(n_1, \ldots, n_s)$-transcript tree for a $(2s+1)$-round public-coin interactive protocol (P,V) as a set N transcripts arranged in the following tree structure. The nodes in this tree correspond to the prover's messages and the edges to the verifier's challenges. Every node at depth i has precisely n_i children corresponding to n_i pairwise distinct challenges. Every transcript corresponds to exactly one path from the root node to a leaf.

Definition 2. *Let $s \geq 1$ and let (P,V) be an $(2s+1)$-round public coin argument for a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$, and let $k_1, \ldots, k_s \in \mathbb{N}$. We say that (P,V) satisfies $(n_1, \ldots, n_s)$-special-soundness if there exists a deterministic polynomial-time algorithm E such that the following holds. On input $x \in \mathcal{X}$ and a tree of transcripts $\mathcal{T}$ that are all accepting for x, E outputs a witness w such that $(x,w) \in \mathcal{R}$.*

Finally, we need to define the notion of "quasi-unique responses" for multi-round protocols, extending a similar notion formalized by Fischlin [34] for Sigma protocols. Essentially, an interactive argument (P,V) has quasi-unique responses if it is hard to come up with an instance x and two transcripts t_1, t_2 that are accepting for x, such that the first point in which they diverge is a prover message (and not a verifier challenge).

[1] By "round" we mean a single message, either from the prover to the verifier or vice versa.

Definition 3. *Let (P, V) be a $(2s+1)$-round interactive argument. We say that (P, V) has* quasi-unique responses *if for every probabilistic-polynomial time algorithm A, there exists a negligible function $\nu(\cdot)$ such that*

$$\mathsf{Adv}^{\mathrm{qur}}_{A,(P,V)}(\lambda) := \Pr\left[\begin{array}{c} a_{i+1} \neq a'_{i+1} \\ \wedge V(x,\tau) = 1 \\ \wedge V(x,\tau') = 1 \end{array} \;:\; \begin{pmatrix} x, i, a_1, \ldots, a_i, \\ c_1, \ldots, c_i, \\ a_{i+1}, \ldots, a_{s+1}, \\ c_{i+1}, \ldots, c_s, \\ a'_{i+1}, \ldots, a'_{s+1}, \\ c'_{i+1}, \ldots, c'_s \end{pmatrix} \leftarrow A(1^\lambda)\right] \leq \nu(\lambda),$$

where $\tau = (x, a_1, c_1, \ldots, c_i, a_{i+1}, \ldots, c_s, a_{s+1})$ and $\tau' = (x, a_1, c_1, \ldots, c_i, a'_{i+1}, \ldots, c'_s, a'_{s+1})$.

2.2 Non-Interactive Random Oracle Arguments

In this section, we define non-interactive arguments in the random oracle model (ROM), following the definitions of Ben-Sasson, Chiesa, and Spooner [9]. A non-interactive arguments in the random oracle model is a natural generalization of non-interactive arguments, where both the prover and verifier are augmented with oracle access to a random oracle $\mathcal{O}$, chosen uniformly at random from the set of all functions mapping inputs from some domain $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ to some range $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$.

More precisely, let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ be a relation and let $\mathcal{L}_\mathcal{R} = \{x \in \mathcal{X} \;:\; \exists w \in \mathcal{W},\ (x, w) \in \mathcal{R}\}$ be the language induced by $\mathcal{R}$. A non-interactive random oracle argument for $\mathcal{R}$ is a pair (P, V), where P is a probabilistic oracle-aided algorithm, and V is a deterministic algorithm. In this paper, both algorithms will run in polynomial time in the length $|x|$ of the input and the security parameter $\lambda \in \mathbb{N}$. We ask that a non-interactive random oracle argument satisfy the standard notion of completeness.

Definition 4. *Let (P, V) be a non-interactive random oracle argument for a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$. We say that (P, V) is* complete *if there exists a negligible function $\nu(\cdot)$ such that*

$$\Pr\left[V^{\mathcal{O}}(1^\lambda, x, \pi) = 1 \;:\; P^{\mathcal{O}}(1^\lambda, x)\right] \geq 1 - \nu(\lambda)$$

where the probability is taken the randomness of P and the choice of $\mathcal{O}$.

As for soundness, we require that a non-interactive random oracle argument satisfy the standard notion of *knowledge soundness*. Intuitively, this means that a prover P^* that can output an accepting proof for $x \in \mathcal{X}$, must "know" a corresponding witness $w \in \mathcal{W}$ such that $(x, w) \in \mathcal{R}$. This is formalized by the existence of an extractor E, that given access to P^* can find a witness w for x. Looking ahead, for our purposes, the extractor will not need to decide on the responses of the random oracle to P^* (i.e., we will work in the "non-programable" random oracle model [42]), but it will be important the E *observes* the queries issued by P^* to the oracle.

Definition 5. *Let (P, V) be a non-interactive random oracle argument for a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$. We say that (P, V) is* knowledge sound *if there exists an algorithm E such that for every $x \in \mathcal{X}$ and every algorithm P^*, there exists a negligible function $\nu(\cdot)$ such that*

$$\Pr\left[(x, E^{P^*}(1^\lambda, x)) \in \mathcal{R}\right] \geq \epsilon_{P^*}(1^\lambda, x) - \nu(\lambda)$$

where $\epsilon_{P^*}(1^\lambda, x) = \Pr\left[V^{\mathcal{O}}(1^\lambda, x, \pi) = 1 \;:\; \pi \leftarrow\!\$\, (P^*)^{\mathcal{O}}(1^\lambda, x)\right]$*. Moreover,* E *runs in time which is polynomial in* λ*, in* $|x|$ *and in the number of random oracle queries issued by* P^* *on these inputs. We call* E *a* straight-line extractor *if it only invokes* P^* *once.*

In the above definition, the notation $E^{P^*}(1^\lambda, x))$ means that E invokes P as a black-box, and — since P^* is an oracle-aided algorithm — E simulates the oracle to P^*.

Zero Knowledge. For succinctness, we forgo formally defining zero knowledge here, and the reader is referred to [41, 13, 9, 26] and the references therein.

3 A Communication Efficient-Transformation

In this section, we present our communication-efficient transformation for multi-round special-sound protocols, that is inspired by the Fischilin transformation for Sigma protocols [34]. We begin by presenting the transformation and arguing its completeness, and then prove that it compiles special sound protocols (Definition 2) into non-interactive random oracles arguments.

3.1 Overview

Before presenting the transformation and its analysis in detail, we first highlight the main ideas that underlie it. The starting point for our transformation is Fischlin's transformation from Sigma protocols to non-interactive arguments [34] that is sketched in Section 1. Recall that in Fischlin's transformation, given their initial message a, the prover attempts to find a transcript (a, c, z) whose hash starts with ℓ 0's, and this transcript will end up being the final proof. This is repeated in parallel in order to amplify the knowledge soundness of the resulting protocol.

Now, as a simple example, suppose that instead of a three-round Sigma protocol, we want to compile a five-round protocol that satisfies $(2, 2)$-special soundness. Recall that this means that to extract a witness, we need four transcripts of the form

$$\begin{aligned}\tau_1 &= (a_1, c_1, a_2, c_2, a_3)\\ \tau_2 &= (a_1, c_1, a_2, c_2', a_3')\\ \tau_3 &= (a_1, c_1', a_2', c_2'', a_3'')\\ \tau_4 &= (a_1, c_1', a_2', c_2''', a_3'''),\end{aligned}$$

where the a_is and the c_is correspond to prover messages and verifier challenges, respectively.

A first attempt to transform such a protocol into a non-interactive one with a straight-line extractor, would be to apply Fischlin's compiler essentially as is: the prover must output a transcript $(a_1, c_1, a_2, c_2, a_3)$ whose hash starts with ℓ 0's, and this is repeated t times over in parallel. If ℓ and t are set properly and the hash function is modeled as a random oracle, it is indeed the case that to produce an accepting proof, a prover must query the random oracle $\mathcal{O}$ on at least four different accepting transcripts with the same first message a_1 (in at least one of the parallel repetitions). However, this in and of itself is insufficient for extraction. It is not enough that the prover queries $\mathcal{O}$ on four different transcripts. For extraction to go through, these four transcripts must also induce a specific tree topology; in our example, a complete binary tree with four leaves.

Indeed, the above idea by itself **does not** preserve the knowledge soundness of the interactive protocol. To see why, observe that there is nothing stopping the prover from just keeping (a_1, c_1, a_2)

fixed, and resampling (c_2, a_3) over and over until it finds a transcript whose hash begins with ℓ 0's. This will produce four transcripts of the form

$$\begin{aligned}\tau_1 &= (a_1, c_1, a_2, c_2, a_3)\\ \tau_2 &= (a_1, c_1, a_2, c_2', a_3')\\ \tau_3 &= (a_1, c_1, a_2, c_2'', a_3'')\\ \tau_4 &= (a_1, c_1, a_2, c_2''', a_3'''),\end{aligned}$$

which does not meet the $(2, 2)$-special soundness criterion.

A natural idea to counter this specific attack is to limit the number of "admissible" values that c_2 can take. If, for example, we only allow c_2 to take values in $\{1, 2, 3\}$, then the above attack goes away. Still, it is not immediately clear that limiting the number of permissible challenges at each level forces a successful prover to query the random oracle on a complete binary tree of transcripts. Surprisingly, we manage to show that if the parameters are set correctly, then in at least one of the parallel repetitions, a successful prover does indeed have to query the random oracle on four transcripts that form such a tree.

More generally, we show that the above idea generalizes to $(2s + 1)$-round $(n_1, \ldots, n_s)$-special sound interactive arguments, for a wide range of parameters. Our compiler requires that the prover finds a transcript $(a_1, c_1, \ldots, a_s, c_s, a_{s+1})$ whose hash start with ℓ zeroes, and in addition $c_i \leq k_i$ for some integers $k_i \geq n_i$ that parameterize the compiler.[2]

At the heart of analysis is a new combinatorial lemma, that bounds that number of leaves that a tree of bounded arity can have, without inducing an $(n_1, \ldots, n_s)$-special soundness tree. In particular, we show that a tree of height $s + 1$, whose nodes at depth i have at most k_{i+1} children either (1) induces an $(n_1, \ldots, n_s)$-special soundness tree; or (2) has at most ϕ leaves, where ϕ is some function that depends on $n_1, \ldots, n_s$ and $k_1, \ldots, k_s$. Intuitively, this means that if we cannot extract a witness from the random oracles queries of a prover P^*, then this prover has a bounded number of trials — namely ϕ trials — to find a transcript whose hash starts with ℓ 0's. Fortunately, we show that for carefully chosen parameters, the function ϕ takes small enough values so that P^* has only a negligible probability of producing an accepting proof, while an honet prover (from whom a witness can be extracted) finds an accepting proof with high probability.

3.2 The Transformation

Let (P, V) be a $(2s+1)$-round public coin interactive argument for a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$. Without loss of generality, suppose that the ith verifier challenge is drawn uniformly at random from the set $[B_i]$ for some $B_i \in \mathbb{N}$. This is not a necessary assumption, but it will help us simplify the presentation. Let $\mathcal{T}$ denote the space of transcripts of (P, V) and let $\mathcal{A}_1$ denote the space of first messages by P in the protocol. The compiler is parameterized by $2s + 1$ integers $k_1, \ldots, k_s \in \mathbb{N}$, and integeres $\ell \in \mathbb{N}$ and $t \in \mathbb{N}$. It also uses a hash function H, mapping inputs in $\mathcal{A}_1^t \times [t] \times \mathcal{X} \times \mathcal{T}$ to outputs in $\{0, 1\}^\ell$.

To present our compiler we introduce the following notation. For a vector $\boldsymbol{k} = (k_1, \ldots, k_s) \in \mathbb{N}^s$ of s integers, we define the operator $\mathsf{increment}_{\boldsymbol{k}}$ as follows. Informally, $\mathsf{increment}_{\boldsymbol{k}}$ takes as input a vector $\boldsymbol{c}$ and increments it to the next vector in lexicographical order, under the condition that the ith entry of the output must not exceed k_i for every $i \in [s]$. More formally, on input $\boldsymbol{c} = (c_1, \ldots, c_s) \in [k_1] \times [k_2] \times \cdots \times [k_s]$, the operator $\mathsf{increment}_{\boldsymbol{k}}$ does:

[2] Suppose for simplicity of presentation that challenges can take arbitrary integer values. Moreover, in this presentation, we ignore zero knowledge, which we discuss later in the section.

1. Find the last position $i \in [s]$ for which $c_i < k_i$. If no such position exists, return $\bot$.
2. Return $\boldsymbol{c'} = (c_1, \ldots, c_{i-1}, c_i + 1, 1, \ldots, 1)$.

Equipped with this notation, the the compiler for transforming (P, V) into a non-interactive argument is presented in Fig. 1. The presentation uses the following observation: Note that the transcript of an interaction between P and V is completely determined by P's inputs — the instance x and the witness w and randomness r — and V's challenges $c_1, \ldots, c_s$. Hence, when defining the compiled protocol, we say that we invoke P on (x, w), a vector $\boldsymbol{c}$ of s challenges, and randomness r to obtain a proof π. By that, we mean that π is the aforementioned induced transcript of the interaction between P and V. The presentation of the compiler is Fig. 1 also assumes that (P, V) is perfectly complete, and does not necessarily satisfy zero-knowledge (this depends on the exact honest-verifier zero knowledge guarantees of (P, V)). In Section 3.6 we discuss how to slightly modify the construction to lift both of these restrictions.

<u>The non-interactive prover P_{ni}</u>

1. Sample random coins $r_1, \ldots, r_t \leftarrow\$ \{0,1\}^{\rho}$ for t executions of P.
2. For $i = 1, \ldots, t$, invoke P on input (x, w) and randomness r_i to obtain a first messages $a_{1,i}$. Denote $\boldsymbol{a_1} = (a_{1,1}, \ldots, a_{1,t})$.
3. For $i = 1, \ldots, t$:
   (a) Initialize $\boldsymbol{c_i} \leftarrow (1, 1, \ldots, 1) \in \mathbb{N}^s$ and $\mathsf{flag}_i \leftarrow \mathsf{false}$.
   (b) While $\mathsf{flag}_i = \mathsf{false}$ do:
      i. Invoke P on (x, w), challenges $\boldsymbol{c_i}$, and randomness r_i to obtain a proof π_i.
      ii. If $H(\boldsymbol{a_1}, i, x, \pi_i) = 0^{\ell}$, then set $\mathsf{flag}_i \leftarrow \mathsf{true}$. Otherwise, increment $\boldsymbol{c_i}$ by setting $\boldsymbol{c_i} \leftarrow \mathsf{increment}_{\boldsymbol{k}}(\boldsymbol{c_i})$.
      iii. If $\boldsymbol{c_i} = \bot$, then output $\bot$ and abort.
4. Output $\pi \leftarrow (\pi_1, \ldots, \pi_t)$.

<u>The non-interactive verifier V_{ni}</u>

1. Parse π as $(\pi_1, \ldots, \pi_t)$ and parse π_i as $(a_{1,i}, c_{1,i}, \ldots, a_{s,i}, c_{s,i}, a_{s+1,i})$. Let $\boldsymbol{a_1} \leftarrow (a_{1,1}, \ldots, a_{1,t})$.
2. For $i = 1, \ldots, t$:
   (a) If for some $j \in [s]$, $c_{j,i} > k_i$, then output 0 and terminate.
   (b) If $V(x, \pi_i) = 0$, then output 0 and terminate.
   (c) If $H(\boldsymbol{a_1}, i, x, \pi_i) \neq 0^{\ell}$, then output 0 and terminate.
3. Output 1.

Fig. 1. The non-interactive argument $(P_{\mathsf{ni}}, V_{\mathsf{ni}})$ that results from applying our transformation to a $(n_1, \ldots, n_s)$-special sound $(2s+1)$-round protocol (P, V). We use $\rho = \rho(|x|, \lambda)$ do denote the number of random coins used by P.

Completeness. The following lemma establishes the completeness of the transform. As mentioned, we first assume that the underlying protocol (P, V) is perfectly-complete, and discuss the imperfect completeness case later in this section.

Lemma 1. *Let (P, V) be a perfectly-complete $(2s+1)$-round public coin interactive argument for a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$. Then, the non-interactive protocol $(P_{\mathsf{ni}}, V_{\mathsf{ni}})$ is $(1 - t \cdot e^{-2^{-\ell} \cdot K})$-complete, where $K = \prod_{j \in [s]} k_j$ and H is modeled as a random oracle.*

Proof. Let $(x, w) \in \mathcal{R}$. Say that a vector $\boldsymbol{r} = (r_1, \ldots, r_t)$ is (x, w)-*good* if for every $i \in [t]$ there exists a vector $(c_{1,i}, \ldots, c_{s,i}) \in [k_1] \times \cdots \times [k_s]$ such that $H(\boldsymbol{a_1}, i, x, \pi_i) = 0^{\ell}$. Here, $\boldsymbol{a_1} = (a_{1,1}, \ldots, a_{1,t})$, where $a_{1,i}$ is the first message of P on input (x, w) and randomness r_i, and π_i is the transcript of (P, V) induced by x, w, the randomness r_i and the challenges $(c_{1,i}, \ldots, c_{s,i})$. Then, an honest prover P_{ni}, running on input $(x, w) \in \mathcal{R}$ produces an accepting proof as long as it samples randomness $(r_1, \ldots, r_t)$ that is (x, w)-good in Step 1. In what follows we bound the probability for this event.

Let $i \in [t]$. Since H is modeled as a random oracle, it is, in particular, independent of the randomness $\boldsymbol{r}$. Hence, we can first fix the randomness $\boldsymbol{r}$ and then sample the oracle H. For every possible vector $\boldsymbol{c_i} = (c_{1,i}, \ldots, c_{s,i})$ of challenges, the probability that $H(\boldsymbol{a_1}, i, x, \pi_i) = 0^\ell$ is $2^{-\ell}$. Moreover, these events are independent for all different choices of $\boldsymbol{c_i}$ (since $\boldsymbol{c_i}$ is included in the proof π_i, given as input to H). There are a total of $K := \prod_{j\in[s]} k_j$ possible vectors $\boldsymbol{c_i}$ and hence

$$\Pr[\boldsymbol{r} \text{ is not } (x, w)\text{-good}] \le t \cdot \left(1 - 2^{-\ell}\right)^K \tag{1}$$

$$\le t \cdot e^{-2^{-\ell}\cdot K}, \tag{2}$$

where Eq. (1) follows from a union bound over $i \in [t]$. □

3.3 Knowledge Soundness

The following theorem asserts that a non-interactive random oracle argument that is obtained by applying our transformation from Section 3.2 to a multi-round special-sound protocol, is indeed knowledge sound.

Theorem 1. *Let (P, V) be a $(2s + 1)$-round $(n_1, \ldots, n_s)$-special sound argument for a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$, and let $(P_{\mathsf{ni}}, V_{\mathsf{ni}})$ be the non-interactive random oracle argument obtained from it using the transformation from Section 3.2, when the hash function H is modeled as a random oracle. Then, there exist a straight-line extractor E such that for every $x \in \mathcal{X}$, and every algorithm P^* there is an algorithm A such that*

$$\Pr\left[(x, E^{P^*}(1^\lambda, x)) \in \mathcal{R}\right] \ge \epsilon_{P^*}(1^\lambda, x) - \mathsf{Adv}^{\mathrm{qur}}_{A,(P,V)}(\lambda) - Q_{P^*} \cdot (2^{-\ell} \cdot L)^t$$

where $\epsilon_{P^*}(1^\lambda, x) = \Pr\left[V^{\mathcal{O}}(1^\lambda, x, \pi) = 1 \ : \ \pi \leftarrow\$ (P^*)^{\mathcal{O}}(1^\lambda, x)\right]$, $Q_{P^*} = Q_{P^*}(\lambda, |x|)$ is a bound on the number of random oracle queries issued by P^*, and $L = \sum_{i=1}^{s}(n_i - 1) \cdot \prod_{j=i+1}^{s} k_j \cdot \prod_{h=1}^{i-1}(k_h - n_h + 1)$. Moreover, E and A run in time which is polynomial in λ, in $|x|$ and in the number of random oracle queries issued by P^* on these inputs.

Our knowledge soundness analysis relies on the following combinatorial lemma, Lemma 2 below. To state the lemma, we first need to define the notions of a *complete tree* and a *bounded tree.*

Definition 6. *Let $s \in \mathbb{N}$ and $n_1, \ldots, n_s \in \mathbb{N}$ be integers. We say that a tree T is $(n_1, \ldots, n_s)$-*complete*, if for each $i \in \{0, \ldots, s - 1\}$, each node in depth i has n_{i+1} children.*

Definition 7. *Let $s \in \mathbb{N}$ and $k_1, \ldots, k_s \in \mathbb{N}$ be integers. We say that a tree T is $(k_1, \ldots, k_s)$-*bounded*, if for each $i \in \{0, \ldots, s - 1\}$, each node in depth i has* at most *k_{i+1} children.*

With these two notions defined, the following lemma bounds the number of leaves that a $(k_1, \ldots, k_s)$-bounded tree can have, if it does not contain a $(n_1, \ldots, n_s)$-complete sub tree.

Lemma 2. *Let $s \in \mathbb{N}$, let $n_1, \dots, n_s \geq 2$, and let $k_1, \dots, k_s$ be integers such that $k_i \geq n_i$ for every $i \in [s]$. Let T be a $(k_1, \dots, k_s)$-bounded tree that does not contain a $(n_1, \dots, n_s)$-complete sub-tree. Then, T has at most $\phi(n_1, \dots, n_s, k_1, \dots, k_s)$ leaves, where*

$$\phi(n_1, \dots, n_s, k_1, \dots, k_s) = \sum_{i=1}^{s} (n_i - 1) \cdot \prod_{j=i+1}^{s} k_j \cdot \prod_{h=1}^{i-1} (k_h - n_h + 1).$$

Proof. The proof is by induction on the height s of the tree. For $s = 1$, the tree T consists of the root and its children. The root cannot have n_1, since otherwise, T contains a n_1-complete sub-tree (which is just a root with n_1 children). Indeed, it holds that $\phi(n_1, k_1) = n_1 - 1$.

For $s > 1$, observe that if n_1 of the root's children are themselves roots of $(n_2, \dots, n_s)$-complete trees, then T contains an $(n_1, \dots, n_s)$-complete sub-tree. Hence, at most $n_1 - 1$ of the root's children can be roots of $(n_2, \dots, n_s)$-complete trees. Call such children "heavy" and all other children "light". Since T is $(k_1, \dots, k_s)$-bounded, any heavy child can be the root of a sub-tree with at most $\prod_{j=2}^{s} k_j$ leaves. On the other hand, by the induction hypothesis, each light child can serve as the root of a sub-tree with at most $\phi(n_2, \dots, n_s, k_2, \dots, k_s)$, where

$$\phi(n_2, \dots, n_s, k_2, \dots, k_s) = \sum_{i=2}^{s} (n_i - 1) \cdot \prod_{j=i+1}^{s} k_j \cdot \prod_{h=2}^{i-1} (k_h - n_h + 1).$$

Putting everything together, we obtain that the number of leaves in T is bounded by

$$\begin{aligned}
(n_1 - 1) \cdot \prod_{j=2}^{s} k_j + (k_1 - n_1 + 1) \cdot \sum_{i=2}^{s} (n_i - 1) \cdot \prod_{j=i+1}^{s} k_j \cdot \prod_{h=2}^{i-1} (k_h - n_h + 1) \\
= (n_1 - 1) \cdot \prod_{j=2}^{s} k_j + \sum_{i=2}^{s} (n_i - 1) \cdot \prod_{j=i+1}^{s} k_j \cdot \prod_{h=1}^{i-1} (k_h - n_h + 1) \\
= \sum_{i=1}^{s} (n_i - 1) \cdot \prod_{j=i+1}^{s} k_j \cdot \prod_{h=1}^{i-1} (k_h - n_h + 1).
\end{aligned}$$

This completes the proof of the lemma. □

Equipped with Lemma 2, we are now ready to prove Theorem 1, establishing the knowledge soundness of $(P_{\mathsf{ni}}, V_{\mathsf{ni}})$.

Proof (of Theorem 1). Let $x \in \mathcal{X}$, $\lambda \in \mathbb{N}$, and a prover P^*, making at most $Q = Q(|x|, \lambda)$ queries to the random oracle. Assume without loss of generality, that before outputting a proof $\pi = (\pi_1, \dots, \pi_t)$ for an instance x, P^* queries the oracle on $\mathcal{O}(\boldsymbol{a_1}, i, x, \pi_i)$ for every $i \in [t]$, where $\boldsymbol{a_1} = (a_1, \dots, a_t)$ and a_j is the first prover message in π_j for every $j \in [t]$. Moreover, assume without loss of generality that P^* only queries the oracle on inputs $(\boldsymbol{a_1}, i, x, \pi_i)$ such that the transcript π_i is accepting for x.

Fix some vector $\boldsymbol{a_1} = (a_{1,1}, \dots, a_{1,t})$ of first prover messages. For every $i \in [t]$, we define a tree $T_{\boldsymbol{a_1},i}$ rooted at $a_{1,i}$, induced by the queries issued by $(P^*)^{\mathcal{O}}(1^\lambda, x)$ to $\mathcal{O}$. The nodes of $T_{\boldsymbol{a_1},i}$ are labeled by prover messages, and the edges are labeled by V's challenges. Then, a path from the root to another node defines a partial transcript for (P, V) in a natural manner, by letting the root's label specify the first prover message, the label of the edge going out of the root specify the first verifier's challenge, and so forth. Generally, the label of the ith node on the path specifies the ith prover message in the partial transcript, and the label on the jth edge on the path specifies the jth verifier's challenge.

We define $T_{\boldsymbol{a_1},i}$ inductively over the queries of P^*. Whenever P^* issues a new query of the form $(\boldsymbol{a_1}, i, x, \pi_i)$, we append π_i to $T_{\boldsymbol{a_1},i}$ as follows:

1. Parse π_i as $(a_{1,i}, c_{1,i}, \ldots, c_{s,i}, a_{s+1,i})$.
2. Find the longest prefix of π_i that corresponds to a path in T_i starting from the root, and ends with a prover message. Let $(a_{1,i}, c_{1,i}, \ldots, a_{j,i})$ be that prefix.
3. Extend $T_{\boldsymbol{a_1},i}$ by appending the path $(c_{j,i}, \ldots, c_{s,i}, a_{s+1,i})$ to the node labeled with $a_{j,i}$.

We are now ready to describe the straight-line extractor E. On input 1^λ and an instance $x \in \mathcal{X}$, and one-time oracle access to P^*, the extractor E does the following:

1. Run P^* on input $(1^\lambda, x)$, recording its random oracle queries.
2. Eventually P^* terminates and outputs a proof π for x. If $V_{\mathsf{ni}}(1^\lambda, x, \pi) = 0$, then output $\perp$ and abort.
3. For every vector $\boldsymbol{a_1}$ of first prover messages included as an argument in a query issued by P^*, construct the t trees $T_{\boldsymbol{a_1},1}, \ldots T_{\boldsymbol{a_1},t}$ from P^* oracle queries as described above.
4. If there exists a tree among the constructed trees that contains a $(n_1, \ldots, n_s)$-transcript tree (recall Definition 1) as a sub-tree, let T^* be that sub-tree. Invoke the special soundness extractor on (x, T^*) to obtain a witness w. Output w.
5. If there is no such sub-tree, output $\perp$ and abort.

We now turn to analyze the success probability of the extractor. Let acc denote the event in which P^* outputs an accepting proof for x and let tree denote the event in which its random oracle queries induce a $(n_1, \ldots, n_s)$-transcript tree (i.e., that E does not abort in Step 5). Clearly, by $(n_1, \ldots, n_s)$-special soundness, the probability that E outputs in accepting transcript is $\Pr[\mathsf{acc} \wedge \mathsf{tree}]$. First, observe that

$$\begin{aligned} \Pr[\mathsf{acc} \wedge \mathsf{tree}] &= \Pr[\mathsf{acc}] - \Pr[\mathsf{acc} \wedge \neg\mathsf{tree}] \\ &= \epsilon_{P^*}(1^\lambda, x) - \Pr[\mathsf{acc} \wedge \neg\mathsf{tree}], \end{aligned} \tag{3}$$

and we are left with bounding $\Pr[\mathsf{acc} \wedge \neg\mathsf{tree}]$. Note, that for the event $\neg\mathsf{tree}$ to occur at least one of two cases must occur:

1. No tree constructed by E contains a $(n_1, \ldots, n_s)$-complete sub-tree; or
2. There is a $(n_1, \ldots, n_s)$-complete sub-tree in in the trees constructed by E, but it is not a valid transcript tree per Definition 1.

Denote the first case by small and the second case by $\mathsf{collision}$. Then, it holds that

$$\begin{aligned} \Pr[\mathsf{acc} \wedge \neg\mathsf{tree}] &\leq \Pr[\mathsf{acc} \wedge \mathsf{small}] + \Pr[\mathsf{acc} \wedge \mathsf{collision}] \\ &\leq \Pr[\mathsf{acc} \wedge \mathsf{small}] + \Pr[\mathsf{collision}] \end{aligned}$$

We argue that there exists a probabilistic polynomial-time algorithm A for which

$$\Pr[\mathsf{collision}] \leq \mathsf{Adv}^{\mathrm{qur}}_{A,(P,V)}(\lambda). \tag{4}$$

The algorithm A gets x as non-uniform advice, and is defined as E for Steps 1–3. It then checks if there is a sub-tree in the trees constructed by E, that is $(n_1, \ldots, n_s)$-complete but it is not a valid transcript tree. If so, then it must be that this sub-tree contains two transcripts of the form $t_1 = (a_1, c_1, \ldots, c_i, a_{i+1}, \ldots, c_s, a_{s+1})$ and $t_2 = (a_1, c_1, \ldots, c_i, a'_{i+1}, \ldots, c'_s, a'_{s+1})$, where $a_{i+1} \neq a'_{i+1}$. Otherwise, this sub-tree would have been a valid transcript tree. Hence, A can output t_1 and t_2 and break the quasi-unique response property of (P, V).

We are left with bounding $\Pr[\mathsf{acc} \wedge \mathsf{small}]$. For this event to occur, it must be that P^* has produced t accepting transcripts of the form $t_i = (a_{1,i}, c_{1,i}, \ldots, c_{s,i}, a_{s+1,i})$ such that $H(\boldsymbol{a_1}, i, x, t_i) = 0^\ell$ for every $i \in [t]$. Moreover, it is the case that for every $i \in [t]$, the tree $T_{\boldsymbol{a_1},i}$ does not contain a $(n_1, \ldots, n_s)$-complete sub-tree. Fix some vector $\boldsymbol{a_1}$ that was included as the first argument in a query made by P^* to $\mathcal{O}$. By Lemma 2, each $T_{\boldsymbol{a_1},i}$ has at most $L := \phi(n_1, \ldots, n_s, k_1, \ldots, k_s)$ leaves, where ϕ is as in Lemma 2. This implies that for every $i \in [t]$, P^* was able to query $\mathcal{O}$ on at most L queries of the form $(\boldsymbol{a_1}, i, x, t)$ for some transcript t, and one have these queries resulted in a reply which is 0^ℓ. By a union bound, for every $i \in [t]$, this can occur with probability at most $2^{-\ell} \cdot L$. As this occurs independently for each $i \in [t]$, it follows that the probability that P^* finds an accepting proof for $\boldsymbol{a_1}$ without its queries inducing a transcript tree for x is at most $(2^{-\ell} \cdot L)^t$. Since P^* queries $\mathcal{O}$ with at most Q different vectors $\boldsymbol{a_1}$, it follows that

$$\Pr[\mathsf{acc} \wedge \mathsf{small}] \leq Q \cdot (2^{-\ell} \cdot L)^t. \tag{5}$$

Putting Equations (3), (4), and (5), we obtain that the extractor E succeeds with probability at elast

$$\Pr[\mathsf{acc} \wedge \mathsf{tree}] \geq \epsilon_{P^*}(1^\lambda, x) - \mathsf{Adv}^{\mathrm{qur}}_{A,(P,V)}(\lambda) - Q \cdot (2^{-\ell} \cdot L)^t.$$

This concludes the proof of the theorem. □

3.4 Parameter Selection

The completeness and knowledge soundness bounds proved above provide a large design space in which one can select specific parameters to instantiate the compiler. To make them more concrete, we now provide an example to one manner in which the parameters can be set, but many more options are possible.

Let (P, V) be a $(2s+1)$-round $(n_1, \ldots, n_s)$-special sound interactive argument for some relation $\mathcal{R}$. To simplify the discussion, suppose that (P, V) has perfectly-unique responses; that is, that for any adversary A it holds that $\mathsf{Adv}^{\mathrm{qur}}_{A,(P,V)}(\lambda) = 0$. The parameters need setting are: the integers $k_1, \ldots, k_s$ bounding the number of possible challenges at every round; the integer $\ell \in \mathbb{N}$, which determines the probability that a transcript hashes to 0^ℓ and hence can be included in a proof; and the integer t which determines that number of parallel repetitions of (P, V).

As a special case of interest, we consider the case where $k_i = cn_i$ for every $i \in [s]$, for some parameter c to be chosen later in the discussion. In this case, the combinatorial bound from Lemma 2 can be bounded as follows:

$$\begin{aligned}\sum_{i=1}^{s}(n_i-1)\cdot\prod_{j=i+1}^{s}k_j\cdot\prod_{h=1}^{i-1}(k_h-n_h+1) &= \sum_{i=1}^{s}(n_i-1)\cdot\prod_{j=i+1}^{s}cn_j\cdot\prod_{h=1}^{i-1}(cn_h-n_h+1)\\ &\le \sum_{i=1}^{s}n_i\cdot\prod_{j\in[s]\setminus\{i\}}cn_j\\ &= c^{s-1}\cdot s\cdot\prod_{i=1}^{n}n_i\\ &= \frac{K\cdot s}{c},\end{aligned}$$

where $K=\prod_{i=1}^{s}k_i$.

Plugging this in to Theorem 1, we obtain a knowledge soundness error of

$$\left(\frac{K\cdot s}{c}\cdot 2^{-\ell}\right)^t.$$

Now suppose that we set the parameters such that

$$c \ge 2\cdot s\cdot K\cdot 2^{-\ell}. \tag{6}$$

In this case, the knowledge soundness error is at most 2^{-t}.

On the other hand, recall that the guarantee of Lemma 1 gave a completeness error of $t\cdot e^{-2^{-\ell}\cdot K}$. Hence, setting ℓ and c such that

$$2^{-\ell}\cdot K \ge \ln(2t) \tag{7}$$

yields a completeness error of $1/2$. Completeness can be amplified be repeatedly invoking P_{ni} until an accepting proof is found. In expectation, P_{ni} will have to invoked twice. If we wish to have a strict upper bound on the running time of the prover, we can invoke P_{ni} for α times with a completeness error of $2^{-\alpha}$.

If we plug the requirement that $2^{-\ell}\cdot K\ge\ln(2t)$ into Eq. (6), we obtain that $c\ge 2\cdot s\cdot\ln(2t)$.

Example parameters. With the above constraints, we can set the following parameters. First, we can set t to be equal to the security parameter λ, which will result in a knowledge soundness error of $2^{-\lambda}$. Then, setting $c\approx s\log t = s\log\lambda$ and ℓ such that $2^{-\ell}\approx\frac{\log\lambda}{N\cdot\log^s\lambda}$ satisfies the constraints from Eq. (6) and (7) above, where $N=\prod_{i=1}^{s}n_i$.

To further understand the implications of this parameter choice, consider two examples:

1. If (P,V) is a constant-round protocol, we obtain $t=\lambda$, $c\approx\log\lambda$, and $\ell\approx\log\left(\frac{\log\lambda}{N\cdot\log^s\lambda}\right)$.
2. The above choice of parameters can also handle the case where (P,V) is has a super-constant number of rounds. Now, however, we must be careful not to end up with a prover P_{ni} that is inefficient. Note that at worst, P_{ni} has to construct a $(k_1,\dots,k_s)$-complete tree of transcripts. This requires invoking the underlying prover P for $\prod_{i=1}^{s}k_i=c^s\cdot N$ times over. For this to be polynomially-bounded, we need c^s to be polynomially-bounded.[3] Since we set $c\approx s\log\lambda$, then as long as $s=O(\frac{\log\lambda}{\log\log\lambda})$, the prover P_{ni} is indeed polynomial-time.

[3] We also need N to be polynomially-bounded, but generalized special soundness is typically only interesting when this is the case.

3.5 Strong Special Soundness

In the context of Sigma protocols and Fischlin's original transform, Kondi and shelat [45] observed that the quasi-unique response property can be lifted, if the original Sigma protocol satisfies a property which they call "strong special soundness". A Sigma protocol is strongly special sound, if extraction is possible from any two distinct transcripts (a, c, z) and (a, c', z'), even if $c = c'$ (but $z \neq z'$). We observe that a similar claim holds in our case. We first generalize the notion of strong special soundness to the multi-round setting.

Definition 8. *Let $s \geq 1$ and let (P, V) be an $(2s+1)$-round public coin protocol. Let $n_1, \ldots, n_s \in \mathbb{N}$, and let $N = \prod_{i=1}^{s} n_i$. Let $\mathcal{T} = (t_{i_1,\ldots,i_s})_{\forall j \in [s],\ i_j \in [n_j]}$ be a vector of N transcripts for (P, V). We say that $\mathcal{T}$ is a* weak $(n_1, \ldots, n_s)$-transcript tree *if it satisfies Definition 1, where Condition 2 is replaced with*

$$a_{j+2} \neq a'_{j+2}.$$

We say that (P, V) satisfies $(n_1, \ldots, n_s)$-strong-special-soundness if there exists a deterministic polynomial-time algorithm E such that the following holds. On input $x \in \mathcal{X}$ and a weak tree of transcripts $\mathcal{T}$ that are all accepting for x, E outputs a witness w such that $(x, w) \in \mathcal{R}$.

The proof of Theorem 1 immediately implies that our compiler results in a knowledge sound non-interactive protocol, whenever it is applied to a strongly-special-sound interactive protocol, even if the latter does not satisfy quasi-unique responses. The reason is that if the event $\mathsf{collision}$ from the proof occurs, than we can directly apply the strong-special-soundness extractor.

A prime example for protocols that do not satisfy quasi-unique response, but do satisfy strong special soundness, is the OR composition of special sound protocols (see [27, 1, 37, 39, 40] and the references therein).

3.6 Zero-Knowledge and Imperfect Completeness

Similarly to Fischlin's transform [34], our compiler results in a zero-knowledge non-interactive argument, if the underlying protocol (P, V) is simulatable in the following sense: Given an instance x in the language, and challenges $(c_1, \ldots, c_s)$, there is an efficient simulator that samples $(a_1, \ldots, a_{s+1})$ such that $(x, a_1, c_1, \ldots, a_s, c_s, a_{s+1})$ are distributed as in an honest execution of the protocol, conditioned on $(c_1, \ldots, c_s)$ (and hence, in particular, $V(x, a_1, \ldots, a_{s+1}) = 1$ if the protocol is perfectly complete). Protocols satisfying quasi-unique response typically satisfy this requirement.

However, our transform also applies to other protocols as well, as discussed above. Therefore, we would like for it to provide zero-knowledge, even if the underlying protocol (P, V) is just honest-verifier zero knowledge, and not necessarily simulatable as above. In the context of Sigma protocol, Kondi and shelat [45] observed that in Fischlin's protocol, if the prover chooses the challenges uniformly at random from the set of all possible challenges (instead of iterating over them in a deterministic manner) than zero knowledge is guaranteed. This is also true in our case, but in our transformation, a problem arises: we need to bound the number of admissible challenges at every round, and so we cannot allow the prover to sample arbitrary challenges. To fix this issue, we can expropriate the choice of challenges from the prover, and use a hash function to sample them instead.

Concretely, let H_{ChalSet} be a hash function from $\mathcal{X} \times [t] \times [s] \times \mathbb{N}$ to the challenge space (suppose for simplicity that the challenge space is the same for all rounds). Then, the set of challenges that is admissible it the jth round of the ith parallel repetition is

$$\{H_{\mathsf{ChalSet}}(x, i, j, m) \ : \ m = 1, \ldots, k_j\} .$$

If H_{ChalSet} is modeled as a random oracle, then the compiled protocol is zero knowledge, similarly to the case in [45].

Observe that an added benefit of this modification is that now the compiled protocol $(P_{\mathsf{ni}}, V_{\mathsf{ni}})$ is complete, even if the underlying protocol (P, V) suffers from some negligible completeness error. This is because now transcripts sample by P_{ni} are distributed as in a random execution of (P, V).

4 A Transformation Based on Cut-and-Choose

In this Section we present a different transformation from multi-round special-sound protocols to non-interactive protocols in the random oracle model. The transformation generalizes Pass's cut-and-choose-based transformation for Sigma protocols [57] to the multi-round setting. It also further abstract a basic building block — extractable vector commitments — in a way that allows for more efficient instantiations.

As promised, the key building block on which the transformation relies is a vector commitment scheme [49, 22]. Here, we recall the standard syntax of vector commitments, and we postpone the discussion of security properties to a later stage.

Definition 9. *A vector commitment scheme over a domain* $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ *is a tuple* $\mathcal{VC} = (\mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ *of algorithms defined as follows:*

- *The algorithm* $\mathsf{VC.Commit}$ *is a probabilistic algorithm that receives as input the security parameter* $\lambda \in \mathbb{N}$ *and a vector* $(x_1, \ldots, x_q) \in (\mathcal{X}_\lambda)^q$*, and outputs a commitment* vcom *and a state* state*.*
- *The algorithm* $\mathsf{VC.Open}$ *is a probabilistic algorithm that receives as input the security parameter* $\lambda \in \mathbb{N}$*, a commitment* vcom*, a state* state *and an index* $i \in [q]$*, and outputs a proof* π*.*
- *The algorithm* $\mathsf{VC.Verify}$ *is a deterministic algorithm that receives as input the security parameter* $\lambda \in \mathbb{N}$*, a commitment* vcom*, an index* $i \in [q]$*, an element* $x \in \mathcal{X}_\lambda$ *and a proof* π*, and outputs a bit* $b \in \{0, 1\}$*.*

Correctness. A vector commitment scheme $\mathcal{VC} = (\mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ is correct if for any $\lambda \in \mathbb{N}$, for any polynomial $q = q(\lambda)$, for any vector $(x_1, \ldots, x_q) \in (\mathcal{X}_\lambda)^q$, and for any index $i \in [q]$, it holds that

$$\Pr\left[\mathsf{VC.Verify}\left(1^\lambda, \mathsf{vcom}, i, x_i, \pi\right) = 1\right] = 1,$$

where $(\mathsf{vcom}, \mathsf{state}) \leftarrow \mathsf{VC.Commit}(1^\lambda, (x_1, \ldots, x_q))$ and $\pi \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{vcom}, \mathsf{state}, i)$; and the probability is taken over the randomness of all algorithms.

Looking ahead, in terms of security, the knowledge soundness of the compiled argument will require that the underlying vector commitment schemes satisfies a strong extractability notion, as well as the standard notion of position binding. We will formalize this notions in Section 4.2 below, when discussing the knowledge soundness of non-interactive random oracle argument that results from the transformation. Zero-knowledge further requires the standard notion of hiding, as discussed later in Section 4.2.

4.1 The Transformation

We now present our cut-and-choose-based transformation. As in Section 3, the compiled protocol is defined with respect to an interactive argument (P, V), which is a $(2s+1)$-round public coin argument for a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$. We will also continue to assume for simplicity of presentation that the ith verifier challenge is drawn uniformly at random from the set $[B_i]$ for some $B_i \in \mathbb{N}$. Since we are interested in protocols that are $(n_1, \ldots, n_s)$-special sound per Definition 2, we assume that $B_i \geq n_i$ for ever i. Similarly to the previous seciton, we first consider the case in which (P, V) is perfectly complete, and ignore zero knowledge, and later discuss how to get zero knowledge and handle imperfectly-complete protocols.

The presentation of the compiler will rely on the following notation. For randomness r for P, we will refer to *the* $(n_1, \ldots, n_s)$*-tree of transcripts for* (P, V) *with randomness* r. By that, we will mean a $(n_1, \ldots, n_s)$-complete tree (recall Definition 6), whose nodes correspond to prover messages in the following way. We associate each edge at depth i with a verifier challenge from the set $[n_i]$: the edges going out of roots are assigned the integers $1, \ldots, n_1$ in a pairwise-distinct manner; for each child of the root, we assign to its outgoing edges the integers $1, \ldots, n_2$ in a pairwise-distinct manner; and so forth. Then, for $i \in \{0, \ldots, s-1\}$ a node v at depth i is labeled with the prover message at the $i+1$ round of (P, V) that is induced by the prover randomness r and the verifier challenges that are the labels of the edges along the path from the root to v. Since the verifier challenges are the same for all trees, the tree can be represented by the concatenation of all $1 + n_1 + n_1 n_2 + \ldots + n_1 \cdots n_s$ prover messages, ordered in some canonical way (for example, pre-order traversal).

The compiler is presented in Figure 2. It is parameterized by an integer $t \in \mathbb{N}$ and it makes use of the following two ingredients:

- A vector commitment scheme $\mathcal{VC} = (\mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ with security properties to be presented later. Denote the space of commitments of $\mathcal{VC}$ by $\mathcal{C}$.
- A hash function H mapping inputs in $\mathcal{X} \times \mathcal{C}$ to outputs in $[N]^t$, where $N = \prod_{i=1}^{s} n_i$. This function will be modeled as a random oracle in the security proof.

4.2 Knowledge Soundness in the ROM

We now prove that when our cut-and-choose transformation is applied to a special sound protocol and H is modeled as a random oracle, then the resulting non-interactive random-oracle argument is straight-line knowledge sound. To this end, we first define the security properties that we need the underlying vector commitment to satisfy.

First, we need $\mathcal{VC}$ to satisfy the standard notion of position binding. Informally, position binding means that an efficient adversary cannot open a position in a vector commitment to two different values.

Definition 10. *A vector commitment scheme* $\mathcal{VC} = (\mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ *over a domain* $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ *is* position binding *if for any polynomial* $q = q(\lambda)$ *and for any probabilistic polynomial-time algorithm* A *there exists a negligible function* $\nu(\cdot)$ *such that*

$$\mathsf{Adv}^{\mathsf{posbind}}_{\mathcal{VC},q,A}(\lambda) := \Pr\left[\begin{array}{c} x_i \neq x'_i \\ \wedge\ \mathsf{VC.Verify}\left(1^\lambda, \mathsf{vcom}, i, x_i, \pi\right) = 1 \\ \wedge\ \mathsf{VC.Verify}\left(1^\lambda, \mathsf{vcom}, i, x'_i, \pi'\right) = 1 \end{array} : \ (\mathsf{vcom}, i, x_i, x'_i, \pi, \pi') \leftarrow A(1^\lambda, q)\right] \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$.

The non-interactive prover P_{cnc}

1. Sample random coins $r_1, \ldots, r_t \leftarrow\$ \{0,1\}^\rho$ for t executions of P.
2. For $i = 1, \ldots, t$, generate the $(n_1, \ldots, n_s)$-tree T_i of transcripts for (P, V) with randomness r_i.
3. Commit to $(T_1, \ldots, T_t)$ by $(\mathsf{vcom}, \mathsf{state}) \leftarrow \mathsf{VC.Commit}(1^\lambda, (T_1, \ldots, T_t))$, where each T_i is interpreted as $1 + n_1 + n_2 n_1 + \ldots + n_1 \cdots n_s$ prover messages.
   // overall, vcom is a commitment to a vector of size $t \cdot (1 + n_1 + n_1 n_2 + \ldots + n_1 \cdots n_s)$
4. Compute $(j_1, \ldots, j_t) \leftarrow H(x, \mathsf{vcom})$.
5. For $i = 1, \ldots, t$, open the prover messages that correspond to the j_ith path in T_i:
   Let $a_{i,1}, \ldots, a_{i,s+1}$ be these prover messages and let $f_{i,1}, \ldots, f_{i,t}$ be their positions in the vector $(T_1, \ldots, T_t)$.
   Then, compute $\pi_{i,1} \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{vcom}, \mathsf{state}, f_{i,1}), \ldots, \pi_{i,s+1} \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{vcom}, \mathsf{state}, f_{i,t})$.
6. Output $\pi = (\mathsf{vcom}, (a_{i,g}, \pi_{i,g})_{i \in [t], g \in [s+1]})$.

The non-interactive verifier V_{cnc}

1. Parse π as $(\mathsf{vcom}, (a_{i,g}, \pi_{i,g})_{i \in [t], g \in [s+1]})$.
2. Compute $(j_1, \ldots, j_t) \leftarrow H(x, \mathsf{vcom})$, and for each $i \in [t]$, interpret j_i as a vector $(c_{i,1}, \ldots, c_{i,s}) \in [k_1] \times \cdots \times [k_s]$.
3. For each $(i, g) \in [t] \times [s+1]$, verify $\pi_{i,g}$ by running $\mathsf{VC.Verify}\left(1^\lambda, \mathsf{vcom}, f_{i,g}, a_{i,g}, \pi_{i,g}\right)$, where $f_{i,g}$ is the location of $a_{i,g}$ in the committed vector $(T_1, \ldots, T_t)$. If any of the verifications fail, output 0 and terminate.
   // note that $f_{i,g}$ can be computed from j_i and $(n_1, \ldots, n_s)$
4. For each $i \in [t]$, verify that the transcript $\tau_i = (a_{i,1}, c_{i,1}, \ldots, a_{i,s}, c_{i,s}, a_{i,s+1})$ is accepting for x by invoking $V(x, \tau_i)$. If any of the transcripts is not accepting, output 0 and terminate.
5. If reached, output 1.

Fig. 2. The non-interactive argument $(P_{\mathsf{cnc}}, V_{\mathsf{cnc}})$ that results from applying our cut-and-choose-based transformation to a $(n_1, \ldots, n_s)$-special sound $(2s + 1)$-round protocol (P, V).

In addition to position binding, we will require that vcom is *straight-line extractable*. Intuitively, this means that there is an extractor E such that whenever an algorithm A outputs a commitment vcom, E (potentially given some special access to A) can output a vector $\boldsymbol{x} = (x_1, \ldots, x_q)$ and corresponding proofs $\pi_1, \ldots, \pi_q$ that are consistent with vcom. Importantly, E is straight-line, and does not rewind the algorithm A. This notion is formalized by definition 11 below. The definition defines extractability with respect to an oracle $\mathcal{F}$. Cases of interest are those in which $\mathcal{F}$ is a random oracle, a generic-group oracle [61, 52], or an empty oracle (in which case, the vector commitment is extractable in the standard model).

Definition 11. *A vector commitment scheme* $\mathcal{VC} = (\mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ *is* straight-line extractable *with respect to an oracle* $\mathcal{F}$ *if there exists a polynomial-time algorithm* $\mathsf{VC.Extract}$ *such that for any polynomial* $q = q(\lambda)$ *and for every probabilistic polynomial-time algorithm* A*, there exists a negligible function* $\nu(\cdot)$ *such that*

$$\mathsf{Adv}^{\mathsf{ext}}_{\mathcal{VC},q,A}(\lambda) := \Pr\left[\begin{array}{c} \mathsf{VC.Verify}^{\mathcal{F}}\left(1^\lambda, \mathsf{vcom}, i, x_i, \pi_i\right) = 1 \\ \wedge\ \mathsf{VC.Verify}^{\mathcal{F}}\left(1^\lambda, \mathsf{vcom}, i, x_i', \pi_i'\right) = 0 \end{array} : \begin{array}{c} (i, \mathsf{vcom}, x_i, \pi_i) \leftarrow\$\ A^{\mathcal{F}}(1^\lambda, q) \\ (x_i', \pi_i') \leftarrow \mathsf{VC.Extract}(1^\lambda, \mathsf{vcom}, i, \mathsf{lst}) \end{array}\right] \le \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$*, where and* lst *is the list of all oracle queries issued by* A*.*

To simplify notation, we will write $\mathsf{VC.Extract}(1^\lambda, \mathsf{vcom}, \mathsf{lst})$ as a shorthand for invoking $\mathsf{VC.Extract}$ on vcom and lst on all indices. That is $\mathsf{VC.Extract}(1^\lambda, \mathsf{vcom}, \mathsf{lst})$ is defined to be:

$$(\mathsf{VC.Extract}(1^\lambda, \mathsf{vcom}, 1, \mathsf{lst}), \ldots, \mathsf{VC.Extract}(1^\lambda, \mathsf{vcom}, q, \mathsf{lst}))$$

Instantiations. We discuss two possible instantiations of straight-line extractable vector commitments. The first instantiation is hash-based Merkle-trees [54]. It was previously observed in the context of arguments of knowledge that Merkle trees are extractable in the random oracle model, where the hash function used to construct the tree is modeled as an observable random oracle [44, 55, 66, 9]. Another instantiation, relying on heavier cryptographic machinery, is vector commitments in idealized group models. As a concrete example, consider KZG commitments [43], that are extractable in the generic group model [61, 53] and in the algebraic group model [35].[4] One the one hand, relying on KZG commitments reduces the proof size: it is possible to provide an constant-size opening proof for t positions out of a length q committed vector, in contrast to an opening proof of size $t \cdot \log q$ in the case of Merkle trees. On the other hand, KZG commitments require pairing groups, and are hence more computation intensive compared to the hash-based Merkle trees. They also require a trusted setup, and are not post-quantum secure.

Theorem 2. *Let* (P, V) *be a* $(2s+1)$*-round* $(n_1, \ldots, n_s)$*-special sound argument for a relation* $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$*, and let* $(P_{\mathsf{cnc}}, V_{\mathsf{cnc}})$ *be the non-interactive random oracle argument obtained from it using the transformation from Section 4.1, when the hash function* H *is modeled as a random oracle. Then, there exist a straight-line extractor* E *such that for every* $x \in \mathcal{X}$*, and every algorithm* P^**, there are algorithms* A *and* B *such that*

$$\Pr\left[(x, E^{P^*}(1^\lambda, x)) \in \mathcal{R}\right] \ge \epsilon_{P^*}(1^\lambda, x) - \mathsf{Adv}^{\mathsf{posbind}}_{\mathcal{VC},q,A}(\lambda) - \mathsf{Adv}^{\mathsf{ext}}_{\mathcal{VC},q,B}(\lambda) - Q_{P^*} \cdot \left(\frac{N-1}{N}\right)^t$$

[4] Tehcnically, the algebraic group model (AGM) was not defined as an oracle model, but it can be rendered as such by requiring that whenever an algorithm outputs a group element, it queries an oracle with a representation of it in terms of the previously-seen elements. In this formalization of the AGM, KZG commitments are indeed extractable per our definition.

where $\epsilon_{P^*}(1^\lambda, x) = \Pr\left[V^{\mathcal{O}}(1^\lambda, x, \pi) = 1 \ : \ \pi \leftarrow\$ (P^*)^{\mathcal{O}}(1^\lambda, x)\right]$*,* $Q_{P^*} = Q_{P^*}(\lambda, |x|)$ *is a bound on the number of random oracle queries issued by* P^**, and* $N = \prod_{i=1}^{s} n_i$ *and* $q = t \cdot (1 + n_1 + n_1 n_2 + \ldots + n_1 \cdots n_s)$*. Moreover,* E*,* A*, and* B *run in time which is polynomial in* λ*, in* $|x|$ *and in the number of random oracle queries issued by* P^* *on these inputs.*

Proof. Let P^* be a prover as in the theorem statement and let $x \in \mathcal{X}$. Assume without loss of generality that before outputting a proof of the form $(\mathsf{vcom}, (a_{i,g}, \pi_{i,g})_{i \in [t], g \in [s]})$ for x, P^* queries $\mathcal{O}$ on (x, vcom).

Consider the following extractor E, running on input $1^\lambda, x$ and observing the oracle queries made by P^* to the random oracle $\mathcal{O}$:

1. Invoke $\pi \leftarrow\$ P^*(1^\lambda, x)$. Let $\mathsf{lst}_{\mathcal{O}}$ denote the set of random oracle queries issued by P^* and let $\mathsf{lst}_{\mathcal{F}}$ denote the set of $\mathcal{F}$-queries issued by P^*.
2. For every vcom_i such that $(x, \mathsf{vcom}_i) \in \mathsf{lst}_{\mathcal{O}}$:
   (a) Run $((T_{i,1}, \ldots, T_{i,t}), (\pi_{i,1}, \ldots, \pi_{i,M})) \leftarrow \mathsf{VC.Extract}(\mathsf{vcom}_i, \mathsf{lst}_{\mathcal{F}})$, where $M = t \cdot (1 + n_1 + n_1 n_2 + \ldots + n_1 \cdots n_s)$.
   (b) If for some $j \in [t]$, $T_{i,j}$ is a $(n_1, \ldots, n_s)$-tree of all accepting transcripts, then invoke the special soundness extractor (Definition 2) E' on x and $T_{i,j}$, and output the witness w outputted by E'.
3. If reached, output $\perp$ and terminate.

We now turn to analyze the success probability of the extractor E. Denote by $\mathsf{goodtree}$ the event in which there is some $\mathcal{O}$-query (x, vcom_i) made by P^*, such that one of its extracted trees $T_{i,j}$ is a $(n_1, \ldots, n_s)$-tree of all accepting transcripts. Denote by acc the event in which $V^{\mathcal{O}}(1^\lambda, x, \pi) = 1$ defined over $\pi \leftarrow\$ (P^*)^{\mathcal{O}}(1^\lambda, x)$. Clearly,

$$\begin{aligned}
\Pr\left[(x, E^{P^*}(1^\lambda, x)) \in \mathcal{R}\right] &= \Pr[\mathsf{goodtree}] \\
&\geq \Pr[\mathsf{acc}] - \Pr[\mathsf{acc} \ \wedge \ \neg\mathsf{goodtree}] \\
&= \epsilon_{P^*}(1^\lambda, x) - \Pr[\mathsf{acc} \ \wedge \ \neg\mathsf{goodtree}] \,. \qquad (8)
\end{aligned}$$

In the remainder of the proof, we will bound $\Pr[\mathsf{acc} \mid \neg\mathsf{goodtree}]$. Let $I(T)$ be a mapping that takes in a $(n_1, \ldots, n_s)$-tree of transcripts, and maps it to the first index $p \in [N]$ such that the transcript induced by the path from root to the pth leaf in $T_{i,j}$ is not accepting. If all transcripts are accepting $I(T) = 0$. Suppose the event $\neg\mathsf{goodtree}$ occurs. This means that for every vcom_i included in a query (x, vcom_i) to $\mathcal{O}$, and every tree $T_{i,j}$ extracted by E for that vcom_i, $I(T_{i,j}) \neq 0$.

Let avoid_i denote the event in which the response $(h_1, \ldots, h_t) \leftarrow \mathcal{O}(x, \mathsf{vcom}_i)$ to the ith query (x, vcom_i) made by P^* to $\mathcal{O}$ satisfies $h_j \neq I(T_{i,j})$ for every $j \in [t]$. Let $\mathsf{avoid} = \cup_{i \in [Q_{p^*}]} \mathsf{avoid}_i$. Then, by a union bound, it holds that

$$\begin{aligned}
\Pr[\mathsf{avoid} \ \wedge \ \neg\mathsf{goodtree}] &\leq \Pr[\mathsf{avoid} \mid \neg\mathsf{goodtree}] \\
&\leq Q_{p^*} \cdot \Pr[\mathsf{avoid}_i \mid \neg\mathsf{goodtree}] \\
&\leq Q_{p^*} \cdot \left(\frac{N-1}{N}\right)^t .
\end{aligned}$$

Now, by total probability, we have that

$$
\begin{aligned}
&\Pr[\mathsf{acc} \wedge \neg\mathsf{goodtree}] \\
&\qquad = \Pr[\mathsf{acc} \wedge \neg\mathsf{goodtree} \wedge \mathsf{avoid}] + \Pr[\mathsf{acc} \wedge \neg\mathsf{goodtree} \wedge \neg\mathsf{avoid}] \\
&\qquad \leq Q_{p^*} \cdot \left(\frac{N-1}{N}\right)^t + \Pr[\mathsf{acc} \wedge \neg\mathsf{avoid}] \qquad (9)
\end{aligned}
$$

Consider the event $\mathsf{acc} \wedge \neg\mathsf{avoid}$. In this event, there exist $i \in [Q_{p^*}]$ and $j \in [t]$, such that the following holds. Let $(h_1, \ldots, h_t) = \mathcal{O}(x, \mathsf{vcom}_i)$. Then, two conditions are satisfied:

- The h_jth path in $T_{i,j}$, as extracted by E, is not accepting.
- The proof outputted by P^* containes vcom_i, and t vectors of prover messages $(\boldsymbol{a_1}, \ldots, \boldsymbol{a_t})$ together with corresponding opening proofs, such that: (1) $\boldsymbol{a_j}$, together with the challenges that correspond to the h_j path, constitutes is an accepting (P, V)-proof for x; and (2) the corresponding opening proofs are valid.

There are two options. Either the opening proofs provided by E for the h_jth path in $T_{i,j}$ are all valid or not. Denote by valid the event in which they are. Then,

$$\Pr[\mathsf{acc} \wedge \neg\mathsf{avoid}] = \Pr[\mathsf{acc} \wedge \neg\mathsf{avoid} \wedge \mathsf{valid}] + \Pr[\mathsf{acc} \wedge \neg\mathsf{avoid} \wedge \neg\mathsf{valid}] \qquad (10)$$

Observe that since P^* provides valid openings for the same positions of the vector committed to by vcom_i, the event $\mathsf{acc} \wedge \neg\mathsf{avoid} \wedge \neg\mathsf{valid}$ corresponds to breaking extractability. Hence, this immediately gives an adversary B for which

$$\mathsf{Adv}^{\mathsf{ext}}_{\mathcal{VC},q,B}(\lambda) = \Pr[\mathsf{acc} \wedge \neg\mathsf{avoid} \wedge \neg\mathsf{valid}].$$

Moreover, since the prover messages extracted by E do not map to an accepting transcript, they are in particular different than the P messages to which P^* opens the same positions of the vector committed to by vcom_i. Hence, the event $\Pr[\mathsf{acc} \wedge \neg\mathsf{avoid} \wedge \mathsf{valid}]$ corresponds to breaking position binding. Hence, this immediately gives an adversary A for which

$$\mathsf{Adv}^{\mathsf{posbind}}_{\mathcal{VC},q,A}(\lambda) = \Pr[\mathsf{acc} \wedge \neg\mathsf{avoid} \wedge \mathsf{valid}].$$

Together with Eq. (8), (9), and (10) yields the theorem.

Zero-Knowledge and Imperfect Completeness. Achieving zero knowledge and handling interactive protocols with negligible completness error can be done analogously to as discussed in Section 3.6. One additional consideration here, is that since we are committing to entire trees of transcripts via a vecotr commitment scheme, we need this scheme to be hiding in the standard sense: Seeing a vector commitment vcom and and openings to a set $\mathcal{L} \subseteq [q]$ of positions in the vector, reveals no information about the committed values in positions $[q] \setminus \mathcal{L}$. See, for example, Catalano and Fiore [22] for a formal definition.

Improved efficiency via subvector commitments. In our compiler, the prover is required to open $(2s+1) \cdot t$ positions at the same time. Hence, we can reduce the proof size by relying on extractable "subvector commitments" [47]; this are vector commitments that allow for more efficient batch openings of many positions. In particular, KZG commitments admit such batch openings, where an opening proof for multiple vector entries has size that is independent of the number of opened positions, as it consists of just one group element (see, for example, [32]).

5 Security in the QROM

In this section, we prove that the a specific instantiation of the cut-and-choose transformation from Section 4 is secure in the quantum random oracle model (the QROM). In the QROM (see [12] for a definition and discussion), a quantum adversary can query the oracle in superposition, which greatly complicates the extraction argument. The fact that the aforementioned transformations admit straight-line extractors is a step in the right direction, since they get rid of elaborate rewinding arguments (recall the discussion in the introduction) that are hard to carry out in the quantum setting without overly disturbing the state of the prover from which we wish to extract the witness. However, note that although these (classical) extractors are straight-line does not mean that they carry over to the quantum setting. The issue is that these extractors heavily rely on the *observability* of oracle queries issued by the prover. In the quantum setting, where queries can be quantum states, observing a query means measuring it, which could disturb the prover. To circumvent this issue, we rely on a toolkit developed in the works of Zhandry [70] and Chiesa, Manohar, and Spooner [26], and prove that a concrete instantiation of the transformation from Section 4 is indeed knowledge sound in the QROM.

5.1 The Transformation

We first begin by explicitly defining the transformation that is secure in the quantum random oracle model. The transformation is a concrete realization of the cut-and-choose based transformation from Section 4, where the vector commitment scheme $\mathcal{VC}$ is instantiated using a Merkle tree [54]. In the proof of knowledge soundness, the hash function underlying the Merkle tree will be treated a (quantum) random oracle.

Merkle trees. We briefly recall the instantiation of a vector commitment scheme using Merkle trees. Such commitments are defined with respect to hash function $H_{\mathsf{com}} : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$, where $\mathcal{X}$ is the domain of vector entries.[5] Suppose that we wish to commit to a vector $\boldsymbol{x} = (x_1, \ldots, x_q)$ of length $q = 2^d$ for some $d \in \mathbb{N}$; if q is not a power of 2, the vector is padded with dummy elements. Then, the $\mathsf{VC.Commit}$ algorithm computes a list $(v_{k,i})_{k\in[d],i\in[2k]}$ of elements defined as recursively as follows:

$$\forall i \in [q],\ v_{d,i} := x_i$$

$$\forall h \in \{d-1, \ldots, 0\},\ \forall i \in [2^h],\ v_{h,i} := H_{\mathsf{com}}(v_{h+1,2i-1}, v_{h+1,2i})$$

The commitment is then $\mathsf{rt} := v_{0,1}$. Visually, we think of this list of values as arranged in the form of a complete binary tree: $v_{0,1}$ is the root of the tree, and each $v_{h,i}$ for $h \in [d]$ and i$\in [2^h]$ has $v_{h+1,2i-1}$ as its left child and $v_{h+1,2i}$ as its right.

If hiding is required, then $\mathsf{VC.Commit}$ first pads $\boldsymbol{x}$ with dummy elements between each two consequtive elements of $\boldsymbol{x}$. For example, if $\mathcal{X} = \{0,1\}^\lambda$, then $\mathsf{VC.Commit}$ first constructs $\boldsymbol{x'} = (x_1, 0^\lambda, x_2, 0^\lambda, \ldots, x_q, 0^\lambda)$, and then commits to $\boldsymbol{x'}$ as before.

An opening proof that x_i is the ith entry in the vector committed to by rt consists of the values along the co-path from $v_{d,i}$ to the root rt; that is, the list of siblings of all nodes on the path from $v_{d,i}$ to rt. To verify this proof, $\mathsf{VC.Verify}$ computes the values along the path from $v_{d,i}$ to the root, using x_i, the co-path values included the proof and the hash function H_{com}. It accepts if and only if the computed root value $v_{0,1}$ is equal to the commitment value rt.

[5] We implicitly assume that $\mathcal{X}$ is sufficiently large so that collision are hard to find.

When the cut-and-choose transformation with Merkle trees is applied to a $(2s+1)$-round interactive argument, we denote the resulting non-interactive argument by $(P_{\mathsf{mcnc}}, V_{\mathsf{mcnc}})$.

5.2 Detour: Games and Their Instability

Before proving that $(P_{\mathsf{mcnc}}, V_{\mathsf{mcnc}})$ is knowledge sound in the QROM, we first need to introduce technical tools from Chiesa et al. [26] and Zhandry [70] on which we rely. We focus here only on the preliminaries necessary for our result, and present them in a way that is specifically tailored to our needs. For a more comprehensive and detail account of the framework of Chiesa et al. and Zhandry, the reader is referred to their works [70, 26].

First, we briefly recall the notions of *oracle games* and *database games*.

Oracle games. Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}, \mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be sets (where for each $\lambda \in \mathbb{N}$, the projections $\mathcal{X}_\lambda, \mathcal{Y}_\lambda$, and $\mathcal{C}_\lambda$ are finite) and let $q = q(\lambda) \in \mathbb{N}$. A *game* is a subset of the product $\mathcal{X}^q \times \mathcal{Y}^q \times \mathcal{C}$. We say that a quantum adversary A wins the game G if it outputs $(\boldsymbol{x}, \boldsymbol{y}, c) \in G$.

An *oracle game* G, defined with respect to a random oracle $\mathcal{O} : \mathcal{X} \to \mathcal{Y}$, additionally requires that $\boldsymbol{x}, \boldsymbol{y}$ outputted by the adversary correspond to input-output pairs under $\mathcal{O}$. That is, a quantum adversary A wins the game G if it outputs $(\boldsymbol{x}, \boldsymbol{y}, c) \in G$, and in addition $\mathcal{O}(x_i) = y_i$ for all $i \in [q]$. We define the advantage of an adversary A in an oracle game G as

$$\mathsf{Adv}_A^{\mathrm{G,o}}(\lambda) := \Pr\left[\begin{matrix}(\boldsymbol{x}, \boldsymbol{y}, c) \in G \\ \forall i \in [q],\ \mathcal{O}(x_i) = y_i\end{matrix} \;:\; (\boldsymbol{x}, \boldsymbol{y}, c) \leftarrow\!\$\, A^{\mathcal{O}}(1^\lambda)\right]$$

where the probability is also over the choice of $\mathcal{O}$ uniformly at random from the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$.

Database games. We first introduce the notion of databases and some related notation. A database D in an associative mapping from values from some domain $\mathcal{X}$ to some range $\mathcal{Y}$. The support of D, $\mathrm{Supp}(D)$ is the set of all $\mathcal{X}$ elements who appear in D, and the image of D, $Im(D)$, is the set of all $y \in \mathcal{Y}$ such that there is an $x \in \mathrm{Supp}(D)$ for which $D(x) = y$. The notation $D + [x \to y]$ is defined as: if $x \in \mathrm{Supp}(D)$, the the value of $D(x)$ is changed to y. Otherwise, the mapping $D(x) = y$ is added to D.

We now define database games. A database game is defined similarly to an oracle game, but with two modifications: first, the random oracle is simulated to A by a simulator; and second, the output $(\boldsymbol{x}, \boldsymbol{y})$ is not the output of A, but rather a record of the simulated query-response pattern.

More precisely, a *database game* is additionally defined by a simulator Sim. For a quantum adversary A, the process $\mathsf{Sim}(A, 1^\lambda)$ is a quantum algorithm that simulates the random oracle to $A(1^\lambda)$ using the compressed phase oracle of Zhandry [70]. Details omitted, the simulator maintains a "database register" D that, intuitively speaking, records the state of the simulated phase oracle throughout the simulation.

When A halts, the simulation $\mathsf{Sim}(A)$ then outputs the result of measuring the database registers in the computational basis. We will overload notation and use D to denote the result of this measurement. We say that A wins the database game G, if there are vectors $\boldsymbol{x'}$ and $\boldsymbol{y'}$ that are consistent with the measured database D, and an element c', such that $(\boldsymbol{x'}, \boldsymbol{y'}, c) \in G$. Formally, we define the advantage of A to be:

$$\mathsf{Adv}_A^{\mathrm{G,db}}(\lambda) := \Pr\left[\begin{matrix}\exists(\boldsymbol{x'}, \boldsymbol{y'}, c') \in G \\ \text{s.t. } \forall i \in [q],\ D(x_i) = y_i\end{matrix} \;:\; D \leftarrow\!\$\, \mathsf{Sim}(A, 1^\lambda)\right]$$

The following lemma was proven by Zhandry [70]. Roughly, it says that an adversary that wins in an oracle game for a base game G also wins in the related database game.

Lemma 3. *Let $\mathcal{X}, \mathcal{Y}, \mathcal{C}$ be sets, let $q = q(\lambda) \in \mathbb{N}$, and let $G \subseteq \mathcal{X}^q \times \mathcal{Y}^q \times \mathcal{C}$ be a base game. For any quantum adversary A and every $\lambda \in \mathbb{N}$, it holds that*

$$\sqrt{\mathsf{Adv}_A^{\mathrm{G,o}}(\lambda)} \leq \sqrt{\mathsf{Adv}_A^{\mathrm{G,db}}(\lambda)} + \sqrt{q/|\mathcal{Y}|}.$$

The instability of database games. Lemma 3 above reduces the task of bounding the advantage of an adversary A in winning an oracle game to the task of bounding its advantage in winning the corresponding database game. In order to bound the latter, Zhandry [70] and Chiesa et al. [26] have come up with a lifting theorems that help tie the maximum adversarial advantage in a database game to a combinatorial property of the *classical variant* of the same database game.

Following Chiesa et al., we first define the notion of a *database property*. Our definition uses the following notation: For sets $\mathcal{X}$ and $\mathcal{Y}$, we denote by $\mathcal{D}(\mathcal{X}, \mathcal{Y})$ the set of all databases mapping inputs from $\mathcal{X}$ to outputs in $\mathcal{Y}$.

Definition 12. *A* database property *$\mathcal{P}$ for databases mapping inputs in $\mathcal{X}$ to outputs in $\mathcal{Y}$ is a subset of $\mathcal{D}(\mathcal{X}, \mathcal{Y})$. The negation of $\mathcal{P}$ is denoted by $\neg\mathcal{P}$ and is defined by $\neg\mathcal{P} := \mathcal{D}(\mathcal{X}, \mathcal{Y}) \setminus \mathcal{P}$.*

Equipped with this definition, we can define the "flip" probability of a property. Essentially, the flip probability of a property with respect to an integer q, is the probability that one additional query can take a database with q queries from $\mathcal{P}$ to $\neg\mathcal{P}$ or vice versa. The definition uses the notation $\mathcal{D}_q(\mathcal{X}, \mathcal{Y})$ to denote the set of all databases of size at most q from $\mathcal{X}$ to $\mathcal{Y}$.

Definition 13. *The* flip probability *from database property $\mathcal{P}$ to property $\mathcal{Q}$, with resecpt to $q \in \mathbb{N}$, denoted $\mathsf{flip}(\mathcal{P} \to \mathcal{Q}, q)$, is defined by*

$$\mathsf{flip}(\mathcal{P}, q) := \max_{D \in \mathcal{D}_q(\mathcal{X},\mathcal{Y}) \cap \mathcal{P}} \left\{ \max_{x \in \mathcal{X} \setminus \mathrm{Supp}(D)} \left\{ \Pr_{y \leftarrow\$ \mathcal{Y}} [(D + [x \to y]) \in \mathcal{Q}] \right\} \right\}$$

The instability of $\mathcal{P}$ with respect to q queries is then defined as follows.

Definition 14. *The instability $\mathcal{I}(\mathcal{P})$ of a database property $\mathcal{P}$ with respect to an integer $q \in \mathbb{N}$ is defined by*

$$\mathcal{I}(\mathcal{P}, q) := \max \{\mathsf{flip}(\mathcal{P} \to \neg\mathcal{P}, q), \mathsf{flip}(\neg\mathcal{P} \to \mathcal{P}, q)\}$$

We now get to the main technical tool that we will use in the proof of knowledge soundness for our compiler. The following lemma is implicit in Zhandry [70] and extended and made explicit by Chiesa et al. [26]. It relates the probability that the output D of a database game with an adversary making q oracle queries belongs to a database property $\mathcal{P}$ to the instability $\mathcal{I}(\mathcal{P}, q)$ of this property.

Lemma 4. *Let $q = q(\lambda)$, and let A be a quantum algorithm making at most q queries to the random oracle. Then, for any database property $\mathcal{P}$, it holds that*

$$\Pr\left[D \in \mathcal{P} \;:\; D \leftarrow\$ \mathsf{Sim}(A, 1^\lambda)\right] \leq q^2 \cdot 6\mathcal{I}(\mathcal{P}, q).$$

Conditional instability. Chiesa et al. have generalized the notion of instability as follows.

Definition 15. *Let $\mathcal{P}$ and $\mathcal{Q}$ be two database properties and let $q \in \mathbb{N}$. The* conditional flip probability *for a database property $\mathcal{P}$, conditioned on $\mathcal{Q}$, with resecpt to $q \in \mathbb{N}$, denoted* $\mathsf{flip}(\mathcal{P}|\mathcal{Q}, q)$*, is defined by*

$$\mathsf{flip}(\mathcal{P}|\mathcal{Q}, q) := \mathsf{flip}(\neg\mathcal{P} \cap \mathcal{Q} \to \mathcal{P} \cap \mathcal{Q}, q).$$

The conditional instability $\mathcal{I}(\mathcal{P}|\mathcal{Q}, q)$ *is defined as*

$$\mathcal{I}(\mathcal{P}|\mathcal{Q}, q) := \max \{\mathsf{flip}(\mathcal{P}|\mathcal{Q}, q), \mathsf{flip}(\neg\mathcal{P}|\mathcal{Q}, q)\}.$$

We will need the following proposition from Chiesa et al. [26].

Proposition 1. *Let $\mathcal{P}$ and $\mathcal{Q}$ be two database properties and let $q \in \mathbb{N}$. Then,*

$$\mathcal{I}(\mathcal{P} \cup \mathcal{Q}, q) \leq \mathcal{I}(\mathcal{P}|\neg\mathcal{Q}, q) + \mathcal{I}(\mathcal{Q}, q).$$

5.3 Knowledge Soundness

We now turn to prove the knowledge soundness of the Merkle-tree based transformation when it comes to *quantum* provers in the quantum random oracle model (QROM). The definition of knowledge soundness in the QROM is the same as in the classical ROM (Definition 5), allowing both the prover P^* and the extractor E to be quantum algorithms. Jumping ahead, the simulator of the random oracle by E to P^* will now be done using Zhandry's compressed phase oracle technique [70].

Let s and $n_1, \ldots, n_s$ be integers, let (P, V) be a $(2s+1)$-round $(n_1, \ldots, n_s)$-special sound interactive argument for a relation $\mathcal{R} \subset \mathcal{X} \times \mathcal{Y}$, and let $(P_{\mathsf{mcnc}}, V_{\mathsf{mcnc}})$ be the non-interactive argument obtained by applying the Merkle-tree-based cut-and-choose transformation described in Section 5.1 to (P, V). With the above technical tools, we are ready to argue that the knowledge soundness of $(P_{\mathsf{mcnc}}, V_{\mathsf{mcnc}})$ in the QROM. As a first step, we define an oracle game that corresponds to a prover P^* convincing the verifier in $(P_{\mathsf{mcnc}}, V_{\mathsf{mcnc}})$ to accept.

In $(P_{\mathsf{mcnc}}, V_{\mathsf{mcnc}})$, the prover outputs a proof π that consists of:

1. t vectors $(\boldsymbol{a_1}, \ldots, \boldsymbol{a_t})$ of P-messages.
2. The root rt of the Merkle tree commitment to these $(s+1) \cdot t$ messages.
3. Corresponding opening proofs for each of the $(s+1) \cdot t$ prover messages $\{a_{i,g}\}_{i \in [t], g \in [s+1]}$. Each of these opening proofs is the co-path from $a_{i,g}$ to rt.

A prover P^* convinces the verifier V_{mcnc} to accept if:

1. All opening proofs check out. For each opening proof, this means that the chain of s hash evaluations, induced by $a_{i,g}$ and the elements in the opening proof, results in rt.
2. For each $i \in [t]$, V accepts $(x, a_{i,1}, c_{i,1}, \ldots, a_{i,s}, c_{i,s}, a_{i,s+1})$. Here, $(j_1, \ldots, j_s) := H(x, \mathsf{rt})$, and $(c_{i,1}, \ldots, c_{i,s})$ is j_i interpreted as a vector in $[n_1] \times \cdots \times [n_s]$.

The oracle game G_{mcnc} is defined very similarly, but we additionally impose the restriction, that the adversary participating in the game does not only output a proof π, but also explicitly outputs all of the query-answer pairs occurring in the verification of the proof by V_{mcnc}. It is easy to see that a prover P^* that makes q' random oracle queries and convinces the verifier V_{mcnc} to accept with probability ϵ immediately gives rise to an algorithm A wins G_{mcnc} with probability ϵ

and makes $q = q' + O(t \cdot \log(t \cdot s))$ queries. The adversary A just invokes P^* on the instance x to obtain a proof π, and then outputs the set of query-answer pairs occurring in the verification of π.

Hence, in the proof of knowledge soundness, we will be interested in extracting a witness w for x from an adversary A making q queries to the oracle and wins in G_{mcnc}. The existence of such an extractor is established in the following theorem.

Theorem 3. *Let $(P_{\mathsf{mcnc}}, V_{\mathsf{mcnc}})$ be a non-interactive argument as described above. Then, $(P_{\mathsf{mcnc}}, V_{\mathsf{mcnc}})$ is knowledge sound in the quantum random oracle model.*

Before proving the theorem, let us briefly remark that although the protocol is defined with respect to two hash functions (H_{com} implementing the random oracle and H implementing the challenge generation from Section 4), both of which will be modeled as random oracles in the security proof, the framework described in Section 5.2 still immediately applies. This is beacuse we can think of both random oracles as being implemented via a single random oracle, with a selection bit as input, specifying which of the two hash functions is invoked and separating their domains. For concreteness, the bound that we will prove will also assume that H_{com} has outputs of length at least λ bits, and that prover messages in (P, V) are at least λ-bits long (if the latter is not satisfied, we can pad them when committing to them using a Merkle tree).

We now prove Theorem 3, adapting the techniques developed in Chiesa et al. [26] to our setting.

Proof. We prove that there exists a polynomial-time quantum extractor E, such that for every $x \in \mathcal{X}$ and every polynomial-time quantum algorithm A participating in G_{mcnc}, it holds that

$$\Pr\left[(x, E^A(1^\lambda, x)) \in \mathcal{R}\right] \geq \mathsf{Adv}_A^{\mathrm{G_{MCNC}},\mathrm{o}}(\lambda) - 2\sqrt{q/2^\lambda} - q^2 \cdot 6\left(2q/2^\lambda + ((N-1)/N)^t\right)$$

To define the extractor E, we first need to define an helper (classical) algorithm, which we will call E_{tree}. This algorithm takes in a database $D : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$ of oracle query-answer pairs, and an element $\mathsf{rt} \in \mathcal{X}$, and a depth parameter d, and outputs the binary tree of height at most d of elements that hash to rt, if such a tree uniquely exists.

Specifically, $E_{\mathsf{tree}}(D, \mathsf{rt}, d)$ does:

1. If rt is not in the image of D, output $\perp$ and terminate.
2. Set $h \leftarrow 1$, $\mathcal{S} \leftarrow \{(\mathsf{rt}, \varepsilon)\}$, and $\mathcal{V} \leftarrow (\mathsf{rt}, \varepsilon)$.
3. While $\mathcal{S}$ contains an element (v, h) with $h \in \{0,1\}^{<d}$:
   (a) Remove an element (v, h) with $h \in \{0,1\}^{<d}$ from $\mathcal{S}$.
   (b) If v is the result of collision, output $\perp$ and terminate.
   (c) Otherwise, if $v \in Im(D)$, then let (u_0, u_1) be the unique pre-image of v in D.
   (d) Update $\mathcal{S} \leftarrow \mathcal{S} \cup \{(u_0, h\|0), (u_1, h\|1)\}$ and $\mathcal{V} \leftarrow \mathcal{V} \cup \{(u_0, h\|0), (u_1, h\|1)\}$.
4. Output the set $\mathcal{V}$.

Observe that the set $\mathcal{V}$ encodes the entire tree: for each element (v, h), v encodes the label of the hash value corresponding to that node, and h encodes its position in the tree.

We can now define the extractor E. On input $(1^\lambda, x)$ and access to a quantum adversary A, E does:

1. Compute the quantum state $|\mathsf{Sim}(A)\rangle$ by simulating the A and its oracle access.
2. Measure the database register of $|\mathsf{Sim}(A)\rangle$ to obtain classical database D.
3. For every rt in the image of the database:

(a) Extract the tree $T \leftarrow E_{\mathsf{tree}}(D, \mathsf{rt}, \log(M \cdot t))$.

(b) Interpret the leaves at depth $\log(M \cdot t)$ of T as a collection of t trees $T_1, \ldots, T_t$ (some of them may be empty, if there are no corresponding leaves at this depth). That is, leaves $jM, \ldots, jM + M - 1$ at depth d, for $j = 1, \ldots, t$, are interpreted as the nodes of the tree T_j, in the order in which they are committed to according to the compiler (e.g., pre-order traversal).

(c) If any T_i corresponds to a complete $(n_1, \ldots, n_s)$-tree of accepting transcript, then use the special soundness extractor to extract a witness w. Output w and terminate.

We now turn to analyze the success probability of the extractor E.

We define the following database properties:

- Note that E finds a witness whenever the database D measured at the end of the simulation of A induces a $(n_1, \ldots, n_s)$-transcript tree. Denote the set of all such databases, where the transcript tree can be obtained by E by querying $E_{\mathsf{tree}}(D, \mathsf{rt}, \log(M \cdot t))$, by $\mathcal{P}_{E,\mathsf{rt}}$.
- Denote the set of all databases that win the database game of G_{mcnc} by $\mathcal{P}_{\mathsf{G}}$. That is, $\mathcal{P}_{\mathsf{G}}$ is the property that the database contains query-answer pairs that result in V_{mcnc} accepting.
- Denote by $\mathcal{P}_{\mathsf{v,rt}}$ the set of all databases in which the verifier accepts a proof that contains rt as the Merkle tree commitment. Moreover, this proof is contained in the output of the tree extractor $E_{\mathsf{tree}}(D, \mathsf{rt}, \log(M \cdot t))$.
- Finally, let $\mathcal{P}_{\mathsf{col}}$ denote the set of all databases containing a collision.

Note that

$$\mathcal{P}_{\mathsf{G}} \subseteq \mathcal{P}_{\mathsf{col}} \cup \left(\bigcup_{\mathsf{rt}} \mathcal{P}_{\mathsf{v,rt}} \right). \tag{11}$$

This is true, since if $D \in \mathcal{P}_{\mathsf{G}}$, then this means that it contains a root rt and the opening proofs corresponding to $H(x, \mathsf{rt})$. If additionally, $D \in \neg\mathcal{P}_{\mathsf{col}}$, then these opening proofs will be found and outputted by $E_{\mathsf{tree}}(D, \mathsf{rt}, \log(M \cdot t))$.

Hence, we can deduce that

$$\begin{aligned}
&\Pr\left[(x, E^A(1^\lambda, x)) \in \mathcal{R}\right] \\
&\quad = \Pr\left[D \in \bigcup_{\mathsf{rt}} \mathcal{P}_{E,\mathsf{rt}}\right] \\
&\quad \geq \Pr\left[D \in \mathcal{P}_{\mathsf{G}} \cap \left(\bigcup_{\mathsf{rt}} \mathcal{P}_{E,\mathsf{rt}}\right)\right] \\
&\quad = \Pr\left[D \in \mathcal{P}_{\mathsf{G}}\right] - \Pr\left[D \in \mathcal{P}_{\mathsf{G}} \cap \left(\bigcap_{\mathsf{rt}} \neg\mathcal{P}_{E,\mathsf{rt}}\right)\right] \\
&\quad \geq \Pr\left[D \in \mathcal{P}_{\mathsf{G}}\right] - \Pr\left[D \in \left(\mathcal{P}_{\mathsf{col}} \cup \left(\bigcup_{\mathsf{rt}} \mathcal{P}_{\mathsf{v},\mathsf{rt}}\right)\right) \cap \left(\bigcap_{\mathsf{rt}} \neg\mathcal{P}_{E,\mathsf{rt}}\right)\right] && (12) \\
&\quad \geq \Pr\left[D \in \mathcal{P}_{\mathsf{G}}\right] - \Pr\left[D \in \mathcal{P}_{\mathsf{col}} \cup \left(\bigcup_{\mathsf{rt}} \neg\mathcal{P}_{E,\mathsf{rt}} \cap \mathcal{P}_{\mathsf{v},\mathsf{rt}}\right)\right] && (13) \\
&\quad = \mathsf{Adv}_A^{\mathrm{G}_{\mathsf{MCNC}},\mathrm{db}}(\lambda) - \Pr\left[D \in \mathcal{P}_{\mathsf{col}} \cup \left(\bigcup_{\mathsf{rt}} \neg\mathcal{P}_{E,\mathsf{rt}} \cap \mathcal{P}_{\mathsf{v},\mathsf{rt}}\right)\right] \\
&\quad \geq \mathsf{Adv}_A^{\mathrm{G}_{\mathsf{MCNC}},\mathrm{db}}(\lambda) - q^2 6\mathcal{I}\left(\mathcal{P}_{\mathsf{col}} \cup \left(\bigcup_{\mathsf{rt}} \neg\mathcal{P}_{E,\mathsf{rt}} \cap \mathcal{P}_{\mathsf{v},\mathsf{rt}}\right), q\right), && (14)
\end{aligned}$$

where Eq. (12) is by total probability, Eq. (13) follwos from Eq. (11), and Eq. (14) follows from Lemma 4.

By Lemma 3, we have that

$$\begin{aligned}
\mathsf{Adv}_A^{\mathrm{G}_{\mathsf{MCNC}},\mathrm{db}}(\lambda) &\geq \mathsf{Adv}_A^{\mathrm{G}_{\mathsf{MCNC}},\mathrm{o}}(\lambda) - 2\sqrt{\mathsf{Adv}_A^{\mathrm{G}_{\mathsf{MCNC}},\mathrm{o}}(\lambda) \cdot q/|\mathcal{X}|} \\
&\geq \mathsf{Adv}_A^{\mathrm{G}_{\mathsf{MCNC}},\mathrm{o}}(\lambda) - 2\sqrt{q/2^\lambda}
\end{aligned}$$

We are left with bounding $\mathcal{I}\left(\mathcal{P}_{\mathsf{col}} \cup \left(\bigcup_{\mathsf{rt}} \neg\mathcal{P}_{E,\mathsf{rt}} \cap \mathcal{P}_{\mathsf{v},\mathsf{rt}}\right), q\right)$. By Proposition 1, it holds that

$$\mathcal{I}\left(\mathcal{P}_{\mathsf{col}} \cup \left(\bigcup_{\mathsf{rt}} \neg\mathcal{P}_{E,\mathsf{rt}} \cap \mathcal{P}_{\mathsf{v},\mathsf{rt}}\right), q\right) \leq \mathcal{I}\left(\mathcal{P}_{\mathsf{col}}, q\right) + \mathcal{I}\left(\bigcup_{\mathsf{rt}} \neg\mathcal{P}_{E,\mathsf{rt}} \cap \mathcal{P}_{\mathsf{v},\mathsf{rt}} \middle| \neg\mathcal{P}_{\mathsf{col}}, q\right)$$

First, note that

$$\mathcal{I}\left(\mathcal{P}_{\mathsf{col}}, q\right) \leq q/2^\lambda.$$

This is because if $D \in \mathcal{P}_{\mathsf{col}}$ then it will forever remain in $\mathcal{P}_{\mathsf{col}}$. On the other hand, if $D \in \neg\mathcal{P}_{\mathsf{col}}$, then tue qth query brings it to $\mathcal{P}_{\mathsf{col}}$ if and only if it collides with one of the previous queries, which happens with probability less than $q/2^\lambda$.

We now conclude the proof by bounding $\mathcal{I}\left(\bigcup_{\mathsf{rt}} \neg\mathcal{P}_{E,\mathsf{rt}} \cap \mathcal{P}_{\mathsf{v},\mathsf{rt}} \middle| \neg\mathcal{P}_{\mathsf{col}}, q\right)$. Let D be a database with less than q queries, and denote by D' the data base $D' := D + [x \to y]$ after the qth query. Consider two cases:

– $D \in (\bigcup_{\mathsf{rt}} \neg\mathcal{P}_{E,\mathsf{rt}} \cap \mathcal{P}_{\mathsf{v},\mathsf{rt}}) \cap \neg\mathcal{P}_{\mathsf{coll}}$. In particular, this means that there exists a root $\hat{\mathsf{rt}}$ such that $D \in \neg\mathcal{P}_{E,\hat{\mathsf{rt}}} \cap \mathcal{P}_{\mathsf{v},\hat{\mathsf{rt}}}$. Since we are conditioning on the database remaining in $\neg\mathcal{P}_{\mathsf{coll}}$, it must be that after the qth query, it is still the case that D' is still in $\mathcal{P}_{\mathsf{v},\hat{\mathsf{rt}}}$. Hence, in order for D' to be in $\neg(\bigcup_{\mathsf{rt}} \neg\mathcal{P}_{E,\mathsf{rt}} \cap \mathcal{P}_{\mathsf{v},\mathsf{rt}}) \cap \neg\mathcal{P}_{\mathsf{coll}} = (\bigcap_{\mathsf{rt}} \mathcal{P}_{E,\mathsf{rt}} \cup \neg\mathcal{P}_{\mathsf{v},\mathsf{rt}}) \cap \neg\mathcal{P}_{\mathsf{coll}}$, it must be that $D' \in \mathcal{P}_{E,\hat{\mathsf{rt}}}$. In other words, the result of the qth query completes a transcript tree rooted at $\hat{\mathsf{rt}}$. Since $D \in \mathcal{P}_{\mathsf{v},\hat{\mathsf{rt}}}$, the hash tree rooted at $\hat{\mathsf{rt}}$ is defined before the qth query, and includes less than q nodes. For that tree to be complete after the qth query, the result of this query has to be one of these q nodes, which happens with probability at most $q/2^\lambda$.
– $D \in \neg(\bigcup_{\mathsf{rt}} \neg\mathcal{P}_{E,\mathsf{rt}} \cap \mathcal{P}_{\mathsf{v},\mathsf{rt}}) \cap \neg\mathcal{P}_{\mathsf{coll}} = (\bigcap_{\mathsf{rt}} \mathcal{P}_{E,\mathsf{rt}} \cup \neg\mathcal{P}_{\mathsf{v},\mathsf{rt}}) \cap \neg\mathcal{P}_{\mathsf{coll}}$. Then, in order for the qth query to put D' in $(\bigcup_{\mathsf{rt}} \neg\mathcal{P}_{E,\mathsf{rt}} \cap \mathcal{P}_{\mathsf{v},\mathsf{rt}}) \cap \neg\mathcal{P}_{\mathsf{coll}}$, there must be some $\hat{\mathsf{rt}}$ such that $D \in \left(\mathcal{P}_{E,\hat{\mathsf{rt}}} \cup \neg\mathcal{P}_{\mathsf{v},\hat{\mathsf{rt}}}\right) \cap \neg\mathcal{P}_{\mathsf{coll}}$ but $D' \in \left(\neg\mathcal{P}_{E,\hat{\mathsf{rt}}} \cap \mathcal{P}_{\mathsf{v},\hat{\mathsf{rt}}}\right) \cap \neg\mathcal{P}_{\mathsf{coll}}$. It cannot be that $D \in \mathcal{P}_{E,\hat{\mathsf{rt}}}$, since otherwise, $D' \in \mathcal{P}_{E,\hat{\mathsf{rt}}}$, since we are conditioning on D' having no collisions. Hence, it must be that $D \in \neg\mathcal{P}_{\mathsf{v},\hat{\mathsf{rt}}}$ but $D' \in \neg\mathcal{P}_{E,\hat{\mathsf{rt}}} \cap \mathcal{P}_{\mathsf{v},\hat{\mathsf{rt}}}$. In other words, the q query produced an accepting proof without there being a complete $(n_1, \dots, n_s)$-transcript tree in D'. If the qth query is to H, then the same argument from the proof of Theorem 2 shows that $D' \in D \in \mathcal{P}_{\mathsf{v},\hat{\mathsf{rt}}}$ with probability at most $((N-1)/N)^t$. If the qth query is to H_{com}, then it puts D' in $\neg\mathcal{P}_{E,\hat{\mathsf{rt}}} \cap \mathcal{P}_{\mathsf{v},\hat{\mathsf{rt}}}$ only if it hits an element already in D. This, as above, occurs with probability at most $q/2^\lambda$.

Taking everything together, we obtain that

$$\Pr\left[(x, E^A(1^\lambda, x)) \in \mathcal{R}\right] \geq \mathsf{Adv}_A^{\mathrm{G}^{\mathsf{MCNC},o}}(\lambda) - 2\sqrt{q/2^\lambda} - q^2 \cdot 6\left(2q/2^\lambda + ((N-1)/N)^t\right),$$

completing the proof of the theorem.

Acknowledgments

Lior Rotem is supported by the Simons Foundation and a research grant from Protocol Labs. Stefano Tessaro was supported in part by NSF grants CNS-2026774, CNS-2154174, a JP Morgan Faculty Award, a CISCO Faculty Award, and a gift from Microsoft.

References

1. M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In Y. Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 415–432. Springer, Berlin, Heidelberg, Dec. 2002.
2. A. Ambainis, A. Rosmanis, and D. Unruh. Quantum attacks on classical proof systems: The hardness of quantum rewinding. In *55th FOCS*, pages 474–483. IEEE Computer Society Press, Oct. 2014.
3. T. Attema and R. Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 513–543. Springer, Cham, Aug. 2020.
4. T. Attema, R. Cramer, and S. Fehr. Compressing proofs of k-out-of-n partial knowledge. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 65–91, Virtual Event, Aug. 2021. Springer, Cham.
5. T. Attema, R. Cramer, and M. Rambaud. Compressed Σ-protocols for bilinear group arithmetic circuits and application to logarithmic transparent threshold signatures. In M. Tibouchi and H. Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 526–556. Springer, Cham, Dec. 2021.
6. T. Attema, S. Fehr, and M. Klooß. Fiat-shamir transformation of multi-round interactive proofs. In E. Kiltz and V. Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 113–142. Springer, Cham, Nov. 2022.

7. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, Oct. / Nov. 2006.
8. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.
9. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Berlin, Heidelberg, Oct. / Nov. 2016.
10. N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th ACM STOC*, pages 111–120. ACM Press, June 2013.
11. D. Boneh and B. Chen. LatticeFold: A lattice-based folding scheme and its applications to succinct proof systems. Cryptology ePrint Archive, Report 2024/257, 2024.
12. D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. Random oracles in a quantum world. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Berlin, Heidelberg, Dec. 2011.
13. D. Boneh and V. Shoup. *A graduate course in applied cryptography (version 0.6)*. 2023. cryptobook.us.
14. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Berlin, Heidelberg, May 2016.
15. J. Bootle, A. Chiesa, Y. Hu, and M. Orrù. Gemini: Elastic SNARKs for diverse environments. In O. Dunkelman and S. Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 427–457. Springer, Cham, May / June 2022.
16. J. Bootle, A. Chiesa, and K. Sotiraki. Sumcheck arguments and their applications. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 742–773, Virtual Event, Aug. 2021. Springer, Cham.
17. J. Bootle, V. Lyubashevsky, N. K. Nguyen, and G. Seiler. A non-PCP approach to succinct quantum-safe zero-knowledge. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 441–469. Springer, Cham, Aug. 2020.
18. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.
19. B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Cham, May 2020.
20. B. Bünz, M. Maller, P. Mishra, N. Tyagi, and P. Vesely. Proofs for inner pairing products and applications. In M. Tibouchi and H. Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 65–97. Springer, Cham, Dec. 2021.
21. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, Oct. 2001.
22. D. Catalano and D. Fiore. Vector commitments and their applications. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, Berlin, Heidelberg, Feb. / Mar. 2013.
23. B. Chen, B. Bünz, D. Boneh, and Z. Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In C. Hazay and M. Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 499–530. Springer, Cham, Apr. 2023.
24. M.-S. Chen, A. Hülsing, J. Rijneveld, S. Samardjiska, and P. Schwabe. SOFIA: $\mathcal{MQ}$-based signatures in the QROM. In M. Abdalla and R. Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 3–33. Springer, Cham, Mar. 2018.
25. Y. Chen and Y. Lindell. Optimizing and implementing fischlin's transform for uc-secure zero knowledge. *IACR Commun. Cryptol.*, 1(2):11, 2024.
26. A. Chiesa, P. Manohar, and N. Spooner. Succinct arguments in the quantum random oracle model. In D. Hofheinz and A. Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 1–29. Springer, Cham, Dec. 2019.
27. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Berlin, Heidelberg, Aug. 1994.
28. Ö. Dagdelen and D. Venturi. A second look at Fischlin's transformation. In D. Pointcheval and D. Vergnaud, editors, *AFRICACRYPT 14*, volume 8469 of *LNCS*, pages 356–376. Springer, Cham, May 2014.

29. J. Don, S. Fehr, and C. Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 602–631. Springer, Cham, Aug. 2020.
30. J. Don, S. Fehr, C. Majenz, and C. Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 356–383. Springer, Cham, Aug. 2019.
31. J. Don, S. Fehr, C. Majenz, and C. Schaffner. Efficient NIZKs and signatures from commit-and-open protocols in the QROM. In Y. Dodis and T. Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 729–757. Springer, Cham, Aug. 2022.
32. D. Feist and D. Khovratovich. Fast amortized KZG proofs. Cryptology ePrint Archive, Paper 2023/033, 2023.
33. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, Aug. 1987.
34. M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Berlin, Heidelberg, Aug. 2005.
35. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, Aug. 2018.
36. G. Fuchsbauer, A. Plouviez, and Y. Seurin. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Cham, May 2020.
37. J. A. Garay, P. D. MacKenzie, and K. Yang. Strengthening zero-knowledge protocols using signatures. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, May 2003.
38. A. Ghoshal and S. Tessaro. Tight state-restoration soundness in the algebraic group model. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 64–93, Virtual Event, Aug. 2021. Springer, Cham.
39. A. Goel, M. Green, M. Hall-Andersen, and G. Kaptchuk. Stacking sigmas: A framework to compose Σ-protocols for disjunctions. In O. Dunkelman and S. Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 458–487. Springer, Cham, May / June 2022.
40. A. Goel, M. Hall-Andersen, G. Kaptchuk, and N. Spooner. Speed-stacking: Fast sublinear zero-knowledge proofs for disjunctions. In C. Hazay and M. Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 347–378. Springer, Cham, Apr. 2023.
41. O. Goldreich. *The Foundations of Cryptography, Volume 1: Basic Tools*. Cambridge University Press, Cambridge, UK, 2001.
42. D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 21–38. Springer, Berlin, Heidelberg, Aug. 2008.
43. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, Dec. 2010.
44. J. Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.
45. Y. Kondi and a. shelat. Improved straight-line extraction in the random oracle model with applications to signature aggregation. In S. Agrawal and D. Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 279–309. Springer, Cham, Dec. 2022.
46. A. Kothapalli and S. T. V. Setty. HyperNova: Recursive arguments for customizable constraint systems. In L. Reyzin and D. Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 345–379. Springer, Cham, Aug. 2024.
47. R. W. F. Lai and G. Malavolta. Subvector commitments with application to succinct arguments. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 530–560. Springer, Cham, Aug. 2019.
48. R. W. F. Lai, G. Malavolta, and V. Ronge. Succinct arguments for bilinear group arithmetic: Practical structure-preserving cryptography. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 2057–2074. ACM Press, Nov. 2019.
49. B. Libert and M. Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In D. Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 499–517. Springer, Berlin, Heidelberg, Feb. 2010.

50. C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *31st FOCS*, pages 2–10. IEEE Computer Society Press, Oct. 1990.
51. A. Lysyanskaya and L. N. Rosenbloom. Universally composable *Σ*-protocols in the global random-oracle model. In E. Kiltz and V. Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 203–233. Springer, Cham, Nov. 2022.
52. U. Maurer. Abstract models of computation in cryptography. In N. P. Smart, editor, *Cryptography and Coding*, pages 1–12, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
53. U. M. Maurer. Abstract models of computation in cryptography (invited paper). In N. P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Berlin, Heidelberg, Dec. 2005.
54. R. C. Merkle. A digital signature based on a conventional encryption function. In C. Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 369–378. Springer, Berlin, Heidelberg, Aug. 1988.
55. S. Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, Nov. 1994.
56. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.
57. R. Pass. On deniability in the common reference string and random oracle model. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Berlin, Heidelberg, Aug. 2003.
58. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.
59. C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, Berlin, Heidelberg, Aug. 1992.
60. S. Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737. Springer, Cham, Aug. 2020.
61. V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997.
62. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15(2):75–96, Mar. 2002.
63. D. Unruh. Quantum proofs of knowledge. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 135–152. Springer, Berlin, Heidelberg, Apr. 2012.
64. D. Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 755–784. Springer, Berlin, Heidelberg, Apr. 2015.
65. D. Unruh. Computationally binding quantum commitments. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 497–527. Springer, Berlin, Heidelberg, May 2016.
66. P. Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In R. Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 1–18. Springer, Berlin, Heidelberg, Mar. 2008.
67. R. S. Wahby, I. Tzialla, a. shelat, J. Thaler, and M. Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, May 2018.
68. D. Wikström. Special soundness in the random oracle model. Cryptology ePrint Archive, Report 2021/1265, 2021.
69. T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 733–764. Springer, Cham, Aug. 2019.
70. M. Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 239–268. Springer, Cham, Aug. 2019.