On the Security of Succinct Interactive Arguments from Vector Commitments [*]

Alessandro Chiesa
alessandro.chiesa@epfl.ch
EPFL

Marcel Dall'Agnol
dallagnol@princeton.edu
Princeton University

Ziyi Guan
ziyi.guan@epfl.ch
EPFL

Nicholas Spooner
nicholas.spooner@warwick.ac.uk
University of Warwick & NYU

September 14, 2024

Abstract

We study the security of a fundamental family of succinct interactive arguments in the standard model, stemming from the works of Kilian (1992) and Ben-Sasson, Chiesa, and Spooner ("BCS", 2016). These constructions achieve succinctness by combining probabilistic proofs and vector commitments.

Our first result concerns the succinct interactive argument of Kilian, realized with any probabilistically-checkable proof (PCP) and any vector commitment. We establish the tightest known bounds on the security of this protocol. Prior analyses incur large overheads, or assume restrictive properties of the underlying PCP.

Our second result concerns an interactive variant of the BCS succinct non-interactive argument, which here we call IBCS, realized with any public-coin interactive oracle proof (IOP) and any vector commitment. We establish the first security bounds for the IBCS protocol. Prior works rely upon this protocol without proving its security; our result closes this gap.

Finally, we study the capabilities and limitations of succinct arguments based on vector commitments. We show that a generalization of the IBCS protocol, which we call the *Finale protocol*, is secure when realized with any *public-query* IOP (a notion that we introduce) that satisfies a natural "random continuation sampling" (RCS) property. We also show a partial converse: if the Finale protocol satisfies the RCS property (which in particular implies its security), then so does the underlying public-query IOP.

Keywords: succinct interactive arguments; vector commitment schemes

[*]A subset of the material in this paper (that concering Kilian's protocol) has been included and further developed in `https://ia.cr/2024/1434`, subsuming some sections. The other sections in this paper (primarily concerning the IBCS protocol and the Finale protocol) remain relevant.

Contents

1 Introduction

A *succinct argument* for a relation R is a computationally-sound interactive proof where, for a given instance $\mathbb{x}$, an argument prover seeks to convince the argument verifier that there exists a witness $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in R$, while communicating much fewer than $|\mathbb{w}|$ bits. Succinct arguments are a central cryptographic object, with numerous theoretical and practical applications. In this paper, we study succinct *interactive* arguments for NP constructed in the standard model (i.e., without oracles) from falsifiable assumptions.[1]

In this work we investigate the power and limitations of the **VC-based approach**, a fundamental paradigm for constructing succinct arguments from two ingredients: a probabilistic proof and a vector commitment (VC). Specifically, we set out to understand *which* probabilistic proofs are amenable to the VC-based approach, and then to *quantitatively* relate the security of the succinct argument to the security of the underlying ingredients. In short, we ask:

What is the security of succinct interactive arguments obtained via the VC-based approach?

Kilian's protocol [Kil92] is the first known succinct argument, and the most notable example of a succinct interactive argument obtained via the VC-based approach. The argument prover commits to a probabilistically checkable proof (PCP) string via a vector commitment scheme (Kilian's presentation uses a Merkle tree, a VC built from collision-resistant hash functions), and sends the resulting commitment to the argument verifier; the argument verifier sends PCP verifier randomness to the argument prover; and finally the argument prover reveals the values of the queried locations of the PCP string and accompanies these values with opening information. The argument verifier accepts if the opening information is valid and the PCP verifier accepts.

Despite this simple template, the security of Kilian's protocol is not well understood. Kilian [Kil92] gives only an informal analysis. Barak and Goldreich [BG08] give a detailed analysis, but with limitations. First, their analysis applies only to PCPs that satisfy restrictive properties; second, their analysis incurs overheads; and third, they restrict their attention to Merkle trees rather than general VCs. This state of affairs motivates our first question: *What is the security of Kilian's protocol, for any given PCP and VC?*

While Kilian's protocol relies on only basic (and efficient) cryptography, its reliance on PCPs means that it is of limited practical value, as known PCP constructions have poor concrete efficiency. To address this, Ben-Sasson, Chiesa, and Spooner [BCS16] introduced a new paradigm for constructing succinct arguments that builds on Kilian's idea. To construct a succinct argument, it is not necessary to start with a *non-interactive* object such as a PCP; instead, they construct succinct arguments from an *interactive* generalization of a PCP called an *interactive oracle proof* (IOP) [BCS16; RRR16]. Since then, extensive study of IOPs has resulted in highly efficient IOP constructions, which have facilitated numerous applications of succinct arguments.

The aforementioned BCS protocol in fact compiles a (public-coin) IOP into a succinct *non-interactive* argument (SNARG) in the random oracle model. That protocol can be straightforwardly relaxed to an "interactive BCS" (IBCS) protocol in the standard model that, analogously to Kilian's protocol, uses a Merkle tree (more generally, a vector commitment scheme) to compile a public-coin IOP into a corresponding succinct interactive argument. The IBCS protocol is a key ingredient in a line of work on succinct arguments with highly efficient provers [BCG20; RR22; HR22] but its security has never been proved, leaving a gap in those results. This motivates our second question: *What is the security of IBCS, for any given public-coin IOP and VC?*

Finally, we observe that one can formulate a natural generalization of IBCS that, syntactically, can be realized from *any* IOP (i.e., including private-coin IOPs). This brings us to our final question: *for what class of IOPs is this VC-based approach secure?*

[1]Succinct *non-interactive* arguments for NP cannot be proved secure via black-box reductions to falsifiable assumptions [GW11].

1.1 Our results

Throughout this section, we fix a vector commitment scheme VC and denote by ϵ_{VC} its position binding error, an upper bound on the probability that an adversary outputs two openings for the same commitment that disagree in at least one position. In general, ϵ_{VC} is a function of the security parameter λ, length ℓ of the committed vector, number s of opened entries of the vector, and bound t_{VC} on the adversary size.

Succinct arguments from PCPs. We provide the tightest known security bounds for Kilian's protocol [Kil92], a seminal result that combines a PCP system and a vector commitment scheme to obtain a succinct interactive argument. Let PCP be a PCP system for a relation R with proof length l, query complexity q, and soundness error ϵ_{PCP} (resp. knowledge soundness error κ_{PCP}); these parameters can depend on the given instance $\mathbb{x}$.

Theorem 1 (informal)**.** *The soundness error ϵ_{ARG} of $\mathsf{Kilian}[\mathsf{PCP}, \mathsf{VC}]$ satisfies the following for every security parameter λ, instance $\mathbb{x} \notin L(R)$, adversary size bound t_{ARG}, and error tolerance $\epsilon > 0$:*

$$\epsilon_{\mathsf{ARG}}(\lambda, \mathbb{x}, t_{\mathsf{ARG}}) \leq \epsilon_{\mathsf{PCP}}(\mathbb{x}) + \epsilon_{\mathsf{VC}}\big(\lambda, \mathsf{l}(\mathbb{x}), \mathsf{q}(\mathbb{x}), t_{\mathsf{VC}}\big) + \epsilon, \text{ where } t_{\mathsf{VC}} = O\left(\frac{\mathsf{l}(\mathbb{x})}{\epsilon} \cdot t_{\mathsf{ARG}}\right) .$$

Similarly, the knowledge soundness error satisfies $\kappa_{\mathsf{ARG}}(\lambda, \mathbb{x}, t_{\mathsf{ARG}}) \leq \kappa_{\mathsf{PCP}}(\mathbb{x}) + \epsilon_{\mathsf{VC}}\big(\lambda, \mathsf{l}(\mathbb{x}), \mathsf{q}(\mathbb{x}), t_{\mathsf{VC}}\big) + \epsilon$.

In particular, if ϵ_{VC} is negligible when $t_{\mathsf{VC}} = \mathsf{poly}(\lambda)$, then $\epsilon_{\mathsf{ARG}}(\lambda, \mathbb{x}, \mathsf{poly}(\lambda)) \leq \epsilon_{\mathsf{PCP}}(\mathbb{x}) + \mathrm{negl}(\lambda)$ (and similarly for κ_{ARG}).

The above bound has an intuitive explanation. An adversary that commits to the PCP string $\widetilde{\Pi}$ with maximal acceptance probability (and opens accordingly) succeeds with probability at least ϵ_{PCP} in $\mathsf{Kilian}[\mathsf{PCP}, \mathsf{VC}]$. Moreover, an adversary that then tries to find a collision when $\widetilde{\Pi}$ is rejected achieves (under some mild conditions) a success probability of $\epsilon_{\mathsf{PCP}} + (1 - \epsilon_{\mathsf{PCP}}) \cdot \epsilon_{\mathsf{VC}}$. The $\frac{\mathsf{l}}{\epsilon}$ multiplicative loss in t_{VC} compared to t_{ARG} expresses the cost of rewinding: to reconstruct an almost full PCP string from small fragments revealed in each (accepting) execution, we must rewind the malicious argument prover sufficiently many times. We leave it as an open problem to establish whether the security bound in Theorem 1 is tight or can be improved.

Succinct arguments from public-coin IOPs. Interaction often leads to dramatic gains in efficiency for proof systems. PCPs are no exception: *interactive oracle proofs* (IOPs) enable such gains by communicating oracle proof strings across multiple rounds. (In particular, a PCP system is an IOP with a single round.)

Let IOP be an IOP system for a relation R with round complexity k and soundness error ϵ_{IOP} (resp. knowledge soundness error κ_{IOP}). Owing to its multiple proof strings, the proof length and query complexity in an IOP may vary round by round; we let $\mathsf{l}_{\mathsf{max}}$ and $\mathsf{q}_{\mathsf{max}}$ denote the maximum proof length and query complexity and let l and q denote the total proof length and query complexity.[2]

Our second main result provides the first security analysis for the *interactive BCS* protocol (inspired by [BCS16]), which extends Kilian's from the case of PCPs to that of *public-coin* IOPs. (An IOP is public-coin if verifier messages are independent uniformly random strings.) As discussed in Section 1.3, this closes a gap in prior works, which used the interactive BCS protocol as an ingredient (without proving its security).

Theorem 2 (informal)**.** *If IOP is public-coin, the soundness error ϵ_{ARG} of $\mathsf{IBCS}[\mathsf{IOP}, \mathsf{VC}]$ satisfies the following for every security parameter λ, instance $\mathbb{x} \notin L(R)$, adversary size bound t_{ARG}, and error tolerance $\epsilon > 0$:*

$$\epsilon_{\mathsf{ARG}}(\lambda, \mathbb{x}, t_{\mathsf{ARG}}) \leq \epsilon_{\mathsf{IOP}}(\mathbb{x}) + \epsilon_{\mathsf{VC}}\big(\lambda, \mathsf{l}_{\mathsf{max}}(\mathbb{x}), \mathsf{q}_{\mathsf{max}}(\mathbb{x}), t_{\mathsf{VC}}\big) + \epsilon, \text{ where } t_{\mathsf{VC}} = O\left(\frac{\mathsf{k}(\mathbb{x}) \cdot \mathsf{l}(\mathbb{x})}{\epsilon} \cdot t_{\mathsf{ARG}}\right) .$$

[2]If the IOP verifier $\mathbf{V}$ makes q_i queries to a proof string of length l_i in round i, then $\mathsf{l}_{\mathsf{max}} := \max_i \{\mathsf{l}_i\}$, $\mathsf{q}_{\mathsf{max}} := \max_i \{\mathsf{q}_i\}$, $\mathsf{l} := \sum_i \mathsf{l}_i$ and $\mathsf{q} := \sum_i \mathsf{q}_i$.

Similarly, the knowledge soundness error is $\kappa_{\mathsf{ARG}}(\lambda, \mathbb{x}, t_{\mathsf{ARG}}) \leq \kappa_{\mathsf{IOP}}(\mathbb{x}) + \epsilon_{\mathsf{VC}}\big(\lambda, \mathsf{l}_{\max}(\mathbb{x}), \mathsf{q}_{\max}(\mathbb{x}), t_{\mathsf{VC}}\big) + \epsilon$.

In particular, if ϵ_{VC} *is negligible when* $t_{\mathsf{VC}} = \mathsf{poly}(\lambda)$*, then* $\epsilon_{\mathsf{ARG}}(\lambda, \mathbb{x}, \mathsf{poly}(\lambda)) \leq \epsilon_{\mathsf{IOP}}(\mathbb{x}) + \mathrm{negl}(\lambda)$ *(and similarly for* κ_{ARG}*).*

The above parameters are analogous to those in Theorem 1 for PCPs, except for a dependence on the round complexity k of the IOP. Again, it is an open question if the loss in t_{VC} compared to t_{ARG} is tight.

How general is the VC-based approach? Kilian's protocol realizes query access to a PCP string via a vector commitment scheme: the values of positions queried by the PCP verifier are revealed alongside an opening proof attesting consistency with the commitment. In the IBCS protocol, this VC-based approach is applied round by round to a public-coin IOP. In general, however, an IOP may not be public-coin. We ask: *which IOPs can be compiled into a secure interactive argument via the VC-based approach?*

It is not hard to see that there exist private-coin IOPs for which the VC-based approach is insecure, i.e., for which the "natural" generalization of IBCS to private-coin IOPs yields insecure interactive arguments.[3] For example, consider a 2-round IOP where the verifier, after receiving the first proof string from the prover, queries the first proof string at a few locations in order to determine its first message to the prover but *must keep the queried locations secret* from the prover in order to ensure soundness.[4] In the VC-based approach, the argument verifier sends the desired queries to the argument prover in order to subsequently obtain corresponding answers and opening proofs. A malicious argument prover would then be able to use this information to break IOP soundness when choosing which second proof string to commit to.

The case of public-query IOPs. The above discussion implies that a *necessary* condition for an IOP to be compatible with the VC-based approach is being **public-query** (a notion that we introduce): soundness (or knowledge soundness) of the IOP must hold even if, whenever the verifier makes a query, the prover learns that this particular query has been made. (Of course, the IOP verifier's own randomness remains secret.)

A public-coin IOP is, in particular, public-query. One may conjecture that the VC-based approach works not only for every public-coin IOP, but also for every public-query IOP. Indeed, we describe a natural protocol that we call $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ that, intuitively, ought to be secure for every public-query IOP.

However, challenges arise when attempting to prove security. The "rewinding proof" of Theorem 2 crucially depends on the existence of an efficient algorithm for sampling IOP verifier randomness *conditioned on a partial transcript of interaction* (which includes prior verifier messages and queries). We call this algorithm a *random continuation sampler* (RCS). Public-coin IOPs have a (trivial) RCS, but there exist public-query IOPs that do not: for example, consider an IOP where, in the first round, the verifier cryptographically commits (via a statistically hiding commitment) to all of its subsequent messages.[5] Sampling random continuations for this proof system is computationally infeasible.

We show that if IOP has an efficient RCS then $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ is secure (for any VC). We proceed in two steps: first, we show that if IOP has an RCS then $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ has an RCS; then, we show that if $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ has an RCS then it is (knowledge) sound. We also show a partial converse: if $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ has an RCS, then so does IOP. This suggests that known proof approaches (all of which rely on random continuations) do not suffice to show the security of $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ for general public-query IOPs.

Theorem 3 (informal)**.** *Let* IOP *be a public-query IOP that admits a RCS with running time* $t_{\mathbf{S}}$*. The soundness error* ϵ_{ARG} *of* $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ *satisfies the following for every security parameter* λ*, instance* $\mathbb{x} \notin L(R)$*,*

[3]Of course, other approaches (e.g., using additional cryptography beyond VC schemes) could in principle work with every IOP.

[4]For example, one can modify an IOP so that the IOP verifier always accepts if the IOP prover guesses the query locations.

[5]The zero knowledge interactive proof of [GK96] has this form.

adversary size bound t_{ARG}, and error tolerance $\epsilon > 0$:

$$\epsilon_{\mathsf{ARG}}(\lambda, \mathbb{x}, t_{\mathsf{ARG}}) \leq \epsilon_{\mathsf{IOP}}(\mathbb{x}) + \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}_{\max}, \mathsf{q}_{\max}, t_{\mathsf{VC}}) + \epsilon, \text{ where } t_{\mathsf{VC}} = O\left(\frac{\mathsf{k}(\mathbb{x}) \cdot \mathsf{l}(\mathbb{x})}{\epsilon} \cdot (t_{\mathsf{ARG}} + t_{\mathbf{S}})\right) .$$

Similarly, the knowledge soundness error is $\kappa_{\mathsf{ARG}}(\lambda, \mathbb{x}, t_{\mathsf{ARG}}) \leq \kappa_{\mathsf{IOP}}(\mathbb{x}) + \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}_{\max}, \mathsf{q}_{\max}, t_{\mathsf{VC}}) + \epsilon$.

Moreover, Finale[IOP, VC] *admits an RCS if and only if* IOP *admits an RCS.*

We actually prove a stronger version of Theorem 3: our analysis works for every public-query IOP that admits a RCS with an arbitrary sampling error α, in which case the (knowledge) soundness error bound of Finale[IOP, VC] depends on both ϵ and α. (See Section 6 for the technical details of this case.)

Remark 1 (simulatable verifiers)**.** The notion of RCS in Theorem 3 is closely related to the notion of a "simulatable verifier" that appears in the literature on parallel repetition for interactive arguments [HPWP10; CL10]. Specifically, an IOP admits an RCS (with negligible sampling error α) if and only if the IOP verifier is (1-)simulatable (without verdict). An interesting research direction is to understand whether there is a deeper connection between standard-model succinct arguments and parallel repetition. For example, can counterexamples to parallel repetition [BIN97] lead us to public-query IOPs for which Finale is not sound?

Remark 2 (adaptive choice of $\mathbb{x}$)**.** The results of Theorems 1, 2 and 3 are stated, for simplicity, in the plain model (no trusted setups), where the argument verifier is responsible for sampling and sending public parameters pp for VC to the argument prover. However, we actually *prove* these results in the (adaptive) common reference string model, wherein public parameters pp for VC are sampled by a trusted party and *a malicious argument prover may adaptively choose the instance $\mathbb{x}$ after learning* pp. Since in these stronger theorems there is no pre-set instance $\mathbb{x}$, the analogous statements (for corresponding security properties) in the common reference model replace $\mathbb{x}$ with a size bound n (and hold for all instances such that $|\mathbb{x}| \leq n$). The plain model variants are straightforwardly implied (see Section 2.4 and Remark 3.6).

1.2 Discussion

On the price of rewinding. We compare the soundness of Kilian's protocol when analyzed via: (i) a rewinding extractor based on a collision resistant hash function; or (ii) a straightline extractor based on an ideal hash function (a random oracle). Our results enable, for the first time, an accounting of the "price of rewinding" for succinct arguments: the cost of a more expensive security reduction that works under weaker assumptions on the underlying cryptography.

(i) *Rewinding extractor.* Suppose that the vector commitment scheme VC is obtained from a collision-resistant hash function (via a Merkle tree) with security $\epsilon_{\mathsf{CRH}}(\lambda, t_{\mathsf{CRH}})$. By Remark 3 this means that

$$\epsilon_{\mathsf{VC}}(\lambda, \ell, s, t_{\mathsf{VC}}) \leq \epsilon_{\mathsf{CRH}}\Big(\lambda, t_{\mathsf{CRH}} = t_{\mathsf{VC}} + O(t_{h_\lambda} \cdot \mathsf{q} \cdot \log \mathsf{l})\Big) .$$

Suppose that $\epsilon_{\mathsf{CRH}}(\lambda, t_{\mathsf{CRH}}) \leq t_{\mathsf{CRH}}^2/2^\lambda$, which is what would be achieved by an ideal hash function. In this case, Theorem 1 gives the following upper bound on the soundness error for Kilian[PCP, VC]:

$$\epsilon_{\mathsf{ARG}}(\lambda, \mathbb{x}, t_{\mathsf{ARG}}) \leq \epsilon_{\mathsf{PCP}}(\mathbb{x}) + O\left(\frac{1}{2^\lambda} \cdot \left(\frac{\mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}} + t_{h_\lambda} \cdot \mathsf{q} \cdot \log \mathsf{l}\right)^2\right) + \epsilon .$$

Setting $\epsilon = \Theta((\mathsf{l} \cdot t_{\mathsf{ARG}})^{2/3} \cdot 2^{-\lambda/3})$ minimizes the right-hand side at $\epsilon_{\mathsf{PCP}}(\mathbb{x}) + \Theta(\mathsf{l}^{2/3} \cdot (t_{\mathsf{ARG}}^2 \cdot 2^{-\lambda})^{1/3})$.[6]

[6]Ignoring the lower-order term $t_{h_\lambda} \cdot \mathsf{q} \cdot \log \mathsf{l}$.

(ii) *Straightline extractor.* Suppose that we model the collision-resistant hash function as an ideal hash function, and analyze $\mathsf{Kilian}[\mathsf{PCP}, \mathsf{VC}]$ in the random oracle model. Prior work analyzing such protocols [BCS16] implies the following upper bound on soundness error:

$$\epsilon_{\mathsf{ARG}}(\lambda, \mathbb{x}, t_{\mathsf{ARG}}) \leq \epsilon_{\mathsf{PCP}}(\mathbb{x}) + O(t_{\mathsf{ARG}}^2 \cdot 2^{-\lambda}) \ .$$

This smaller upper bound is achieved thanks to a straightline (i.e., non-rewinding) extractor for the vector commitment scheme (which is a Merkle tree in the random oracle model).

Setting parameters. Observe that, neglecting factors in l, the rewinding analysis incurs a cube-root loss in the second soundness term. To understand better how this loss affects security, we work through an example of setting concrete parameters for Kilian's protocol according to each analysis.

Theorem 1 gives that, for every malicious argument prover $\widetilde{\mathcal{P}}$ of size t_{ARG} and instance $\mathbb{x} \notin L(R)$,

$$\Pr\left[\langle \widetilde{\mathcal{P}}, \mathcal{V}(1^\lambda, \mathbb{x}) \rangle = 1\right] \leq \epsilon_{\mathsf{PCP}}(\mathbb{x}) + \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}(\mathbb{x}), \mathsf{q}(\mathbb{x}), t_{\mathsf{VC}}) + \epsilon,$$

where $t_{\mathsf{VC}} \leq 4 \cdot \frac{\mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}$. (The constant 4 is obtained in our analysis in the technical sections.)

Say that we are targeting a soundness error of 2^{-40} against adversaries of size $t_{\mathsf{ARG}} = 2^{60}$. Suppose that the PCP underlying Kilian's protocol, for the chosen instance size, achieves soundness error $\epsilon_{\mathsf{PCP}} = 2^{-42}$ with a proof consisting of 2^{30} symbols. Further suppose that the vector commitment scheme has binding error $\epsilon_{\mathsf{VC}}(\lambda, \ell, s, t_{\mathsf{VC}}) \leq t_{\mathsf{VC}}^2/2^\lambda$ (the bound achieved by an ideal Merkle tree). Setting $\epsilon = 2^{-42}$ results in a bound of

$$t_{\mathsf{VC}} \leq 4 \cdot \frac{2^{30}}{2^{-42}} \cdot t_{\mathsf{ARG}} = 2^{74} \cdot t_{\mathsf{ARG}}$$

so

$$\epsilon_{\mathsf{VC}} \leq \frac{(2^{74} \cdot t_{\mathsf{ARG}})^2}{2^\lambda} = 2^{148-\lambda} \cdot t_{\mathsf{ARG}}^2 = 2^{268-\lambda} \ .$$

Then Theorem 1 indicates that we can set $\lambda = 309$ to achieve the desired security level (soundness error of 2^{-40} against adversaries of size 2^{60}).

On the other hand, for the same overall security level, and with the same underlying PCP, the straightline analysis indicates that setting $\lambda \approx 160$ suffices. Hence, if we are willing to make a qualitatively stronger assumption about the cryptography, we can obtain succinct arguments which are roughly a factor 2 more efficient. Note that, for the setting $\lambda = 160$, our rewinding analysis cannot give *any* nontrivial bound against adversaries of size 2^{60}.

Of course, known PCPs are wildly inefficient; in practice, one would use an IOP instead. The explicit security bound in (the detailed analysis underlying) Theorem 2 for the IBCS protocol enables reasoning similarly to the above for the case of public-coin IOPs.

Remark 3 (security of underlying components)**.** We derive security bounds for argument systems *as a function of the security bounds of the underlying components.* In short, we take ϵ_{VC}, ϵ_{PCP}, κ_{PCP} (and ϵ_{IOP}, κ_{IOP}) as given. While statistical soundness bounds on PCPs and IOPs can be calculated (they are information-theoretic components), the position binding errors for VC must be derived from some (concrete) computational assumption. For example, if VC is a Merkle tree obtained from a collision-resistant hash function $h_\lambda \colon \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ computable in time t_{h_λ} whose collision probability against t_{CRH}-size adversaries is bounded by $\epsilon_{\mathsf{CRH}}(\lambda, t_{\mathsf{CRH}})$ then VC has binding error $\epsilon_{\mathsf{VC}}(\lambda, \ell, s, t_{\mathsf{VC}}) \leq \epsilon_{\mathsf{CRH}}(\lambda, t_{\mathsf{CRH}})$ where $t_{\mathsf{CRH}} = t_{\mathsf{VC}} + O(t_{h_\lambda} \cdot \mathsf{q} \cdot \log \mathsf{l})$ for a small hidden constant that can be derived from the security reduction. (The reduction transforms a t_{VC}-size adversary A_{VC} against the Merkle tree into a t_{CRH}-size adversary A_{CRH} against the collision-resistant hash function. Briefly, A_{CRH} runs A_{VC} and then looks for a collision among the authentication paths output by A_{VC}, resulting in the additive increase of $O(t_{h_\lambda} \cdot \mathsf{q} \cdot \log \mathsf{l})$ in size.)

Improved understanding of [BCS16]. Theorem 2 is a step towards a fuller understanding of the (non-interactive) BCS protocol. While [BCS16] gives a detailed security analysis of the BCS protocol as a whole, they crucially consider only the specific setting where the VC is implemented as a Merkle tree *and* the hash function used in the tree is modeled as a random oracle. It is believed that the BCS protocol is secure when realized with any vector commitment (in particular, with VCs in the standard model) but this has not been shown thus far. A natural way to prove this would be to combine (an appropriate strengthening of) Theorem 2 with an analysis of the Fiat–Shamir transformation applied to the resulting argument.[7] Note that, while such an analysis would apply to standard-model VCs, the overall scheme would be proven secure in the random oracle model (under computational assumptions); indeed, there are (contrived) IOPs for which the BCS transformation is provably insecure in the standard model [BBHMR19].

A reflection on succinct arguments. Succinct arguments are a rare example of an "advanced" cryptographic primitive that can be achieved from simple cryptography. Indeed, it is remarkable that, based solely on the existence of a collision resistant hash function (even given as a black box), one can achieve cryptographic proof systems with such exceptional efficiency. On the other hand, the security reduction of a succinct argument is tasked with a challenging goal: find a "long" witness when given a malicious argument prover that only outputs "short" messages in any given interaction. This naturally leads to *rewinding*, a fundamental method of analysis in cryptography. In light of this, Kilian's protocol occupies a central place in cryptography: the simplest succinct argument, and its security analysis is a prominent example of extracting a long witness from sufficiently many short messages obtained via rewinding. This work contributes a long overdue general security analysis of a central cryptographic paradigm.

The preprocessing case. We omit discussion of the *preprocessing setting* for succinct arguments, and believe that future work can build on our security analyses to cover this case. Informally, the preprocessing setting is an offline-online model that enables succinct verification even for "non-uniform" computations. The canonical way to construct succinct arguments in the preprocessing setting is to combine a *holographic* probabilistic proof and a vector commitment scheme [CHMMVW20; COS20] — this is a direct generalization of the VC-based approach that we study in this paper. Specifically, each protocol that we study in this paper (Kilian's protocol, the IBCS protocol, the Finale protocol) has a straightforward extension to the preprocessing setting, where the vector commitment scheme is also used in the offline computation phase to commit to the holographic part of the probabilistic proof. We leave to future work extending our analyses to cover this case.

1.3 Related work

The literature on succinct arguments presents a vast landscape of constructions exhibiting complex tradeoffs between efficiency, expressiveness and security. The goal of this work is to study the security of the key family of VC-based constructions. This family occupies a special place in the landscape: it alone demonstrates that succinct arguments are a standard-model "minicrypt" primitive.[8] Moreover, the VC-based approach is the the only known method for obtaining succinct arguments in the standard model with linear-time provers. Below, we summarize only the most relevant prior work: security analyses for VC-based succinct arguments.

Succinct arguments from collision-resistant functions. The first construction of a succinct argument is due to Kilian [Kil92], and follows the VC-based approach (the underlying vector commitment is a Merkle

[7]The appropriate strengthening refers to showing that IBCS is a succinct interactive argument that satisfies (a suitable computational notion of) state-restoration soundness (which makes the interactive argument compatible with the Fiat–Shamir transformation) if the underlying IOP also satisfies state-restoration soundness.

[8]This is known to be true only of succinct *interactive* arguments; indeed, succinct *non-interactive* arguments cannot be proven secure in the standard model via black-box reductions from *any* falsifiable assumption [GW11].

tree constructed from a collision-resistant hash function). The security reduction in [Kil92] is informal, and does not provide any asymptotic (nor explicit) security bounds.

Barak and Goldreich [BG08] provide a formal analysis of a variant of Kilian's construction, towards their goal of constructing zero-knowledge arguments with a non-black-box simulator. Due to their setting, they restrict their result to the case where the PCP is *non-adaptive* and *reverse-samplable*. While the former restriction is mild (many known PCP constructions are non-adaptive, with few exceptions such as [KPT97]), the latter restriction is a non-standard strong property of the query algorithm, which has not been shown to hold for a number of PCP constructions of interest (e.g., the short PCPs in [BS06; BKKMS13]). Under these conditions, they establish that Kilian's protocol achieves non-adaptive knowledge soundness, with a constant multiplicative factor loss in soundness versus the PCP soundness. In contrast, our analysis applies to *all* PCPs (including adaptive PCPs) and establishes the tightest known bound for adaptive knowledge soundness.[9]

Lai and Malavolta [LM19, Appendix C] prove that Kilian's protocol is secure when realized with any PCP and vector commitment. In fact, they prove a more general result: a variant of Kilian's protocol is secure when realized with any *linear PCP* and *linear map commitment*. This generality can lead to shorter proofs. However, their proof applies only to PCPs with negligible soundness error and does not quantify the security of the succinct argument in terms of the security of the underlying cryptography.

Chiesa, Ma, Spooner, and Zhandry [CMSZ21] prove *post-quantum* security of Kilian's protocol. As part of their analysis, they give a proof of security for Kilian that also applies to the classical setting. Their analysis differs significantly from ours due to challenges unique to the quantum setting, and incurs a multiplicative soundness loss. In this work we consider soundness against classical adversaries only.

Several works [BCG20; RR22; HR22] obtain succinct arguments with very strong efficiency properties by applying the IBCS protocol to their highly-efficient IOP constructions (along with a Merkle tree based on linear-time CRHFs). They do not show soundness of the IBCS protocol itself (their focus is on novel IOPs). This omission has remained a gap in that line of work, and our work closes this gap.

Succinct arguments from ideal hash functions. A line of work studies security reductions for succinct *non-interactive* arguments in the random oracle model (ROM) [Mic00; Val08; BCS16; CMS19; CY21a; CY21b; BGTZ23]. These works take advantage of the ROM in two key ways. First, they use the observability of oracle queries to construct a vector commitment with a *straightline* (i.e., non-rewinding) extractor (a Merkle tree in the ROM). As shown in Section 1.2, this leads to tighter security bounds. In fact, since these constructions are *unconditionally* secure in the ROM, it is often possible to compute their *exact* soundness. Second, these constructions use the Fiat–Shamir transformation to convert an underlying *interactive* argument into a *non-interactive* one; the general security of this transformation has been shown only in the ROM.

Special-sound protocols. Interactive protocols with *special soundness* are an important and well-studied family of public-coin protocols. In the three-message public-coin (Σ-protocol) setting, k-special soundness means that a witness can be efficiently extracted from any k accepting protocol transcripts with distinct verifier challenges. A line of works extends this notion to multiple rounds [AC20; ACK22; AF22]. The concrete security of general special sound protocols is relatively well-understood.

However, as observed in [CMSZ21], for reasonable choices of PCP, Kilian's protocol is *not* k-special sound for any polynomial k (for example, one can find a set of transcripts that includes only queries to a small fraction of the PCP).[10] We are therefore not able to apply results about special soundness directly.

[9]Formally, our result is incomparable with the one of Barak and Goldreich. In more detail, they use the reverse samplability property of the PCP to obtain a collision-finder whose running time does not depend on the PCP length. This is necessary in their setting, as there the size of an extracted PCP is not *a priori* bounded by any polynomial. It is open whether such a reduction is possible for (even polynomial-size) PCPs that are not reverse samplable.

[10]Towards a tighter security proof for Kilian in the post-quantum setting, Lombardi, Ma, and Spooner [LMS22] introduce the

In concurrent work, Attema, Fehr, and Resch [AFR23] relax the notion of special soundness further: for an access structure Γ, a protocol is Γ-special sound if there is an efficient procedure to compute a witness from any set of transcripts whose challenges form an authorized set in Γ. They use this framework to show that the IBCS protocol applied to the FRI protocol (an IOP of proximity for Reed–Solomon codes [BBHR18]) is a proof of knowledge of a committed codeword (with a quasipolynomial-time extractor). In contrast, rather than analyzing any specific instantiation, our goal is to give a *general* security guarantee for the Kilian (resp. IBCS) protocol, which applies to any PCP (resp. IOP). It seems that Γ-special soundness may not be the right approach for such generality, as Γ will depend on the underlying PCP/IOP. Our guarantee depends, of course, on the *soundness* of the PCP/IOP, which must be analyzed separately.

notion of *probabilistic special soundness* (PSS), a relaxation of special soundness, and show that Kilian's protocol is PSS. We do not follow this approach, as we do not expect it to yield tight security bounds in the classical setting.

2 Techniques

We overview the main ideas underlying our results. In Section 2.1 we review Kilian's protocol and sketch our security analysis for it. In Section 2.2 we review the IBCS protocol and discuss our security analysis for it, comparing it to that for Kilian's protocol. In Section 2.3 we overview the Finale protocol (an extension of the IBCS protocol that is formulated for *public-query* IOPs) and discuss capabilities and limitations of the VC-based approach for succinct interactive arguments. In Section 2.4 we discuss adaptive security.

Vector commitment schemes. We fix a vector commitment scheme VC throughout this technical overview, whose interface and properties are sketched below; see Section 3.2 for formal definitions. Here we omit the algorithm that samples public parameters (and suppress these parameters in the interfaces of VC).[11]

- $\mathsf{VC.Commit}$: On input a message m, $\mathsf{VC.Commit}$ outputs a commitment cm and auxiliary state aux.
- $\mathsf{VC.Open}$: On input the auxiliary state aux and a query set $\mathcal{Q}$, $\mathsf{VC.Open}$ outputs an opening proof pf.
- $\mathsf{VC.Check}$: On input a commitment cm, query set $\mathcal{Q}$, answers ans, and opening proof pf, $\mathsf{VC.Check}$ determines if pf is valid for ans being the restriction to $\mathcal{Q}$ of the message committed in cm.

The property of *perfect completeness* ensures that $\mathsf{VC.Check}$ always accepts if pf is output by $\mathsf{VC.Open}$ given the auxiliary information produced by $\mathsf{VC.Commit}$. The security property of VC is *position binding*: VC has *position binding error* $\epsilon_{\mathsf{VC}}(\lambda, \ell, s, t_{\mathsf{VC}})$ if, when VC is instantiated with security parameter λ for messages of length ℓ, every adversary of size t_{VC} that outputs $(\mathsf{cm}, \mathsf{ans}, \mathsf{ans}', \mathcal{Q}, \mathcal{Q}', \mathsf{pf}, \mathsf{pf}')$ with $|\mathcal{Q}| = |\mathcal{Q}'| = s$ satisfies the following predicate with probability at most $\epsilon_{\mathsf{VC}}(\lambda, \ell, s, t_{\mathsf{VC}})$ (over VC's public parameters):

$$\begin{array}{l} \exists\, i \in \mathcal{Q} \cap \mathcal{Q}' : \mathsf{ans}[i] \neq \mathsf{ans}'[i] \\ \wedge\ \mathsf{VC.Check}(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 \\ \wedge\ \mathsf{VC.Check}(\mathsf{cm}, \mathcal{Q}', \mathsf{ans}', \mathsf{pf}') = 1 \end{array} \quad .$$

In other words, position binding makes it hard to produce two incompatible openings to the same commitment.

Stateful algorithms. Throughout this section, the interactive algorithms that participate in protocols are stateful. When it is important to distinguish different computation phases of a stateful algorithm, we make explicit the state passed from one phase to the next.

2.1 Succinct arguments based on PCPs

We review Kilian's protocol and sketch the main ideas behind Theorem 1; see Section 4 for details.

2.1.1 Kilian's protocol

Kilian's protocol [Kil92] obtains a succinct interactive argument by combining two ingredients: a probabilistically checkable proof (PCP) and a vector commitment scheme VC (fixed above). Let $\mathsf{PCP} = (\mathbf{P}, \mathbf{V})$ be a PCP system for a relation R with alphabet Σ, proof length l, query complexity q, and verifier randomness complexity r. $\mathsf{Kilian}[\mathsf{PCP}, \mathsf{VC}]$ is an interactive argument $\mathsf{ARG} = (\mathcal{P}, \mathcal{V})$ in which the argument prover $\mathcal{P}$ receives an instance $\mathbb{x}$ and a witness $\mathbb{w}$, and the argument verifier $\mathcal{V}$ receives the instance $\mathbb{x}$. Then $\mathcal{P}$ and $\mathcal{V}$ interact, exchanging 3 messages, as follows.

1. $\mathcal{P}$ computes the PCP string $\Pi \leftarrow \mathbf{P}(\mathbb{x}, \mathbb{w})$, computes the commitment $(\mathsf{cm}, \mathsf{aux}) \leftarrow \mathsf{VC.Commit}(\Pi)$, and sends cm to $\mathcal{V}$.

[11]For example, if VC is based on a Merkle tree, the public parameters are the (randomly sampled) collision-resistant function to be used for hashing the given message down to the Merkle root.

2. $\mathcal{V}$ samples PCP verifier randomness $\rho \leftarrow \{0,1\}^{\mathsf{r}}$ and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ deduces the set $\mathcal{Q}$ of queries that $\mathbf{V}(\mathbb{x};\rho)$ makes to Π, sets the query answers $\mathsf{ans} := \Pi[\mathcal{Q}]$, generates an opening proof $\mathsf{pf} \leftarrow \mathsf{VC.Open}(\mathsf{aux}, \mathcal{Q})$, and sends the tuple $(\mathcal{Q}, \mathsf{ans}, \mathsf{pf})$ to $\mathcal{V}$.
4. $\mathcal{V}$ performs the following checks.
   (a) $\mathsf{VC.Check}(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1$ (i.e., ans are valid answers for positions $\mathcal{Q}$ relative to cm);
   (b) $\mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho) = 1$ (i.e., the PCP verifier $\mathbf{V}(\mathbb{x};\rho)$ accepts the answers ans on $\mathcal{Q}$).

Above, the notation $\mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho)$ refers to the decision bit of the PCP verifier $\mathbf{V}$, given instance $\mathbb{x}$ and PCP randomness ρ, when each query $j \in \mathcal{Q}$ is answered with $\mathsf{ans}[j] \in \Sigma$. (If $\mathbf{V}$ queries outside the set $\mathcal{Q}$ then $\mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho) = 0$.)

2.1.2 Security reduction

Intuitively, the soundness error of $\mathsf{Kilian}[\mathsf{PCP}, \mathsf{VC}]$ should be at most the (statistical) soundness error of PCP plus the position binding error of VC. The key lemma below formalizes this intuition.

Consider a malicious argument prover $\widetilde{\mathcal{P}}$ whose first message is the commitment cm. Intuitively, by the position binding property of VC, $\widetilde{\mathcal{P}}$ is "bound" to open locations of at most a single underlying PCP string $\widetilde{\Pi}$. By *rewinding* $\widetilde{\mathcal{P}}$ sufficiently many times to recover the underlying PCP string $\widetilde{\Pi}$, we can relate the probability of $\widetilde{\mathcal{P}}$ convincing the argument verifier $\mathcal{V}$ to the probability of $\widetilde{\Pi}$ convincing the PCP verifier $\mathbf{V}$.

Lemma 1 (informal)**.** *There exists a probabilistic algorithm $\mathcal{R}$ (the* ***reductor****) that, for every instance $\mathbb{x}$, error parameter $\epsilon > 0$, adversary size bound $t_{\mathsf{ARG}} \in \mathbb{N}$, and t_{ARG}-size adversary $\widetilde{\mathcal{P}}$, satisfies*

$$\Pr\left[\begin{array}{l} \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho) \neq 1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 \end{array} \;\middle|\; \begin{array}{l} \mathsf{cm} \leftarrow \widetilde{\mathcal{P}} \\ (\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}}(\mathsf{cm}, \epsilon) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\rho) \end{array}\right] \leq \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) + \epsilon \ ,$$

where $t_{\mathsf{VC}} = O\left(\frac{1}{\epsilon} \cdot t_{\mathsf{ARG}}\right)$.

The reductor $\mathcal{R}$ handles the aforementioned rewinding process. It constructs a proof string $\widetilde{\Pi} \in \Sigma^{\mathsf{l}}$ whose convincing probability is approximately the same as that of the argument prover (up to the position binding error of VC and an arbitrary error term ϵ). Note that $\mathcal{R}$ requires only black-box access to $\widetilde{\mathcal{P}}$.

In the lemma above, the PCP verifier and the argument verifier are "coupled" in that they receive the same randomness ρ. The lemma states that it is unlikely, for a randomly-chosen ρ, that the argument verifier $\mathcal{V}$ accepts the answers provided by $\widetilde{\mathcal{P}}$ but the PCP verifier $\mathbf{V}$ rejects $\widetilde{\Pi}$ under the same randomness. Intuitively, this will allow us to approximately equate the probability that $\widetilde{\mathcal{P}}$ convinces the argument verifier $\mathcal{V}$ to the probability that $\widetilde{\Pi}$ convinces the PCP verifier $\mathbf{V}$.

First we discuss how to use Lemma 1 to establish soundness and knowledge soundness of $\mathsf{Kilian}[\mathsf{PCP}, \mathsf{VC}]$ in Sections 2.1.3 and 2.1.4. Then in Section 2.1.5 we sketch the proof of Lemma 1.

2.1.3 Soundness analysis

We wish to upper bound the soundness error of $\mathsf{Kilian}[\mathsf{PCP}, \mathsf{VC}]$. As claimed in Theorem 1, we argue that for every instance $\mathbb{x} \notin L(R)$, size bound $t_{\mathsf{ARG}} \in \mathbb{N}$, and t_{ARG}-size adversary $\widetilde{\mathcal{P}}$,

$$\Pr\left[\langle \widetilde{\mathcal{P}}, \mathcal{V}(\mathbb{x}) \rangle = 1\right] \leq \epsilon_{\mathsf{PCP}}(\mathbb{x}) + \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) + \epsilon \ .$$

By construction of the argument verifier $\mathcal{V}$, the above probability is equivalent to the following:

$$\Pr\left[\begin{array}{l} \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho)=1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1 \end{array} \middle| \begin{array}{l} \mathsf{cm}\leftarrow\widetilde{\mathcal{P}} \\ \rho\leftarrow\{0,1\}^{\mathsf{r}} \\ (\mathcal{Q},\mathsf{ans},\mathsf{pf})\leftarrow\widetilde{\mathcal{P}}(\rho) \end{array}\right] .$$

Below we fix the following experiment (which is the experiment above augmented with an invocation of $\mathcal{R}$):

$$\left[\begin{array}{l} \mathsf{cm}\leftarrow\widetilde{\mathcal{P}} \\ (\widetilde{\mathcal{Q}},\widetilde{\Pi})\leftarrow\mathcal{R}^{\widetilde{\mathcal{P}}}(\mathsf{cm},\epsilon) \\ \rho\leftarrow\{0,1\}^{\mathsf{r}} \\ (\mathcal{Q},\mathsf{ans},\mathsf{pf})\leftarrow\widetilde{\mathcal{P}}(\rho) \end{array}\right]$$

Using the law of total probability,

$$\begin{aligned} &\Pr\left[\begin{array}{l} \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho)=1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1 \end{array}\right] \\ &= \Pr\left[\begin{array}{l} \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho)=1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho)=1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1 \end{array}\right] + \Pr\left[\begin{array}{l} \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho)\neq 1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho)=1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1 \end{array}\right] . \end{aligned}$$

The term on the right is bounded from above by $\epsilon_{\mathsf{VC}}(\lambda,\mathsf{l},\mathsf{q},t_{\mathsf{VC}})+\epsilon$, due to Lemma 1.

The term on the left is bounded by $\epsilon_{\mathsf{PCP}}(\mathbb{x})$ (the soundness error of PCP). Indeed, we can view the first message of $\widetilde{\mathcal{P}}$ (cm in the experiment above) and the reductor $\mathcal{R}$ as a malicious PCP prover $\widetilde{\mathbf{P}}$ that outputs a PCP string $\widetilde{\Pi}$:

$\widetilde{\mathbf{P}}$:
1. Run $\mathsf{cm}\leftarrow\widetilde{\mathcal{P}}$.
2. Run $(\widetilde{\mathcal{Q}},\widetilde{\Pi})\leftarrow\mathcal{R}^{\widetilde{\mathcal{P}}}(\mathsf{cm},\epsilon)$.
3. Output $\widetilde{\Pi}$.

Since $\mathbb{x}\notin L(R)$, by the definition of soundness error of PCP,

$$\Pr\left[\begin{array}{l} \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho)=1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho)=1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1 \end{array}\right] \leq \Pr\left[\mathbf{V}^{\widetilde{\Pi}}(\mathbb{x};\rho)=1 \middle| \begin{array}{l} \rho\leftarrow\{0,1\}^{\mathsf{r}} \\ \widetilde{\Pi}\leftarrow\widetilde{\mathbf{P}} \end{array}\right] \leq \epsilon_{\mathsf{PCP}}(\mathbb{x}) .$$

2.1.4 Knowledge soundness analysis

We wish to upper bound the knowledge soundness error of Kilian[PCP, VC]. As claimed in Theorem 1, we argue that there exists a polynomial-time probabilistic extractor $\mathcal{E}$ such that for every instance $\mathbb{x}$, size bound $t_{\mathsf{ARG}}\in\mathbb{N}$, and t_{ARG}-size adversary $\widetilde{\mathcal{P}}$,

$$\Pr\left[\begin{array}{l} b=1 \\ \wedge\,(\mathbb{x},\mathbb{w})\notin R \end{array} \middle| \begin{array}{l} b\leftarrow\langle\widetilde{\mathcal{P}},\mathcal{V}(\mathbb{x})\rangle \\ \mathbb{w}\leftarrow\mathcal{E}^{\widetilde{\mathcal{P}}}(\mathbb{x}) \end{array}\right] \leq \kappa_{\mathsf{PCP}}(\mathbb{x})+\epsilon_{\mathsf{VC}}(\lambda,\mathsf{l},\mathsf{q},t_{\mathsf{VC}})+\epsilon .$$

By construction of the argument verifier $\mathcal{V}$, the above probability is equivalent to the following:

$$\Pr\left[\begin{array}{l} \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho)=1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1 \\ \wedge\, (\mathbb{x},\mathbb{w})\notin R \end{array}\;\middle|\; \begin{array}{l} \mathsf{cm}\leftarrow\widetilde{\mathcal{P}} \\ \rho\leftarrow\{0,1\}^{\mathsf{r}} \\ (\mathcal{Q},\mathsf{ans},\mathsf{pf})\leftarrow\widetilde{\mathcal{P}}(\rho) \\ \mathbb{w}\leftarrow\mathcal{E}^{\widetilde{\mathcal{P}}}(\mathbb{x}) \end{array}\right] .$$

We construct $\mathcal{E}$ using the PCP prover $\widetilde{\mathbf{P}}$ described in Section 2.1.3 and the PCP extractor $\mathbf{E}$ (which is given by the underlying PCP system):

$\mathcal{E}^{\widetilde{\mathcal{P}}}(\mathbb{x})$:
1. Run $\widetilde{\Pi}\leftarrow\widetilde{\mathbf{P}}$.
2. Output $\mathbb{w}\leftarrow\mathbf{E}(\mathbb{x},\widetilde{\Pi})$.

Using the law of total probability,

$$\Pr\left[\begin{array}{l} \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho)=1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1 \\ \wedge\, (\mathbb{x},\mathbb{w})\notin R \end{array}\;\middle|\; \begin{array}{l} \mathsf{cm}\leftarrow\widetilde{\mathcal{P}} \\ \rho\leftarrow\{0,1\}^{\mathsf{r}} \\ (\mathcal{Q},\mathsf{ans},\mathsf{pf})\leftarrow\widetilde{\mathcal{P}}(\rho) \\ \mathbb{w}\leftarrow\mathcal{E}^{\widetilde{\mathcal{P}}}(\mathbb{x}) \end{array}\right]$$
$$=\Pr\left[\begin{array}{l} \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho)=1 \\ \wedge\,\mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho)=1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1 \\ \wedge\, (\mathbb{x},\mathbb{w})\notin R \end{array}\right]+\Pr\left[\begin{array}{l} \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho)\neq 1 \\ \wedge\,\mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho)=1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1 \\ \wedge\, (\mathbb{x},\mathbb{w})\notin R \end{array}\right] ,$$

where the last two probabilities are with respect to

$$\left[\begin{array}{l} \mathsf{cm}\leftarrow\widetilde{\mathcal{P}} \\ \rho\leftarrow\{0,1\}^{\mathsf{r}} \\ (\mathcal{Q},\mathsf{ans},\mathsf{pf})\leftarrow\widetilde{\mathcal{P}}(\rho) \\ \widetilde{\Pi}\leftarrow\widetilde{\mathbf{P}} \\ \mathbb{w}\leftarrow\mathbf{E}(\mathbb{x},\widetilde{\Pi}) \end{array}\right] .$$

The term on the right is bounded by $\epsilon_{\mathsf{VC}}(\lambda,\mathsf{l},\mathsf{q},t_{\mathsf{VC}})+\epsilon$ due to Lemma 1.

The term on the left is bounded by $\kappa_{\mathsf{PCP}}(\mathbb{x})$ (the knowledge soundness error of PCP) as shown below:

$$\Pr\left[\begin{array}{l} \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho)=1 \\ \wedge\,\mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho)=1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1 \\ \wedge\, (\mathbb{x},\mathbb{w})\notin R \end{array}\right]\leq\Pr\left[\begin{array}{l} \mathbf{V}^{\widetilde{\Pi}}(\mathbb{x};\rho)=1 \\ \wedge\,(\mathbb{x},\mathbb{w})\notin R \end{array}\;\middle|\; \begin{array}{l} \rho\leftarrow\{0,1\}^{\mathsf{r}} \\ \widetilde{\Pi}\leftarrow\widetilde{\mathbf{P}} \\ \mathbb{w}\leftarrow\mathbf{E}(\mathbb{x},\widetilde{\Pi}) \end{array}\right]\leq\kappa_{\mathsf{PCP}}(\mathbb{x}) .$$

2.1.5 Proof sketch of Lemma 1

We are left to sketch the proof of Lemma 1. To do so, we present a reductor algorithm $\mathcal{R}$.

The goal of $\mathcal{R}$ is to piece together a PCP string $\widetilde{\Pi}$ obtained from the argument prover $\widetilde{\mathcal{P}}$. Intuitively, $\widetilde{\Pi}$ is "fixed" after $\widetilde{\mathcal{P}}$ outputs a commitment cm, and $\mathcal{R}$ attempts to obtain information about $\widetilde{\Pi}$ by *rewinding* the second phase of $\widetilde{\mathcal{P}}$, when given freshly sampled choices of PCP randomness ρ. Each such execution (if it outputs a valid opening) reveals a fragment of $\widetilde{\Pi}$. By repeating this process sufficiently many times, $\mathcal{R}$ obtains enough locations of the string $\widetilde{\Pi}$. Below we denote by $\mathsf{N}=\mathsf{N}(\epsilon)$ the number of samples (set later).

```
$\mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux},\cdot)}(\mathsf{cm},\epsilon)$:
1. Initialize a proof string: $\widetilde{\Pi} := (\sigma)^{\mathsf{l}}$, where $\sigma$ is an arbitrary element in $\Sigma$.
2. Initialize an empty set $\widetilde{\mathcal{Q}}$ to track which locations of $\widetilde{\Pi}$ are filled in.
3. Repeat the following $\mathsf{N}$ times:
   (a) Sample PCP verifier randomness $\rho \leftarrow \{0,1\}^{\mathsf{r}}$.
   (b) Ask $\widetilde{\mathcal{P}}$ for answers to this randomness: $(\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho)$.
   (c) If $\mathsf{VC.Check}(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1$, set $\widetilde{\Pi}[\mathcal{Q}] := \mathsf{ans}$ and update $\widetilde{\mathcal{Q}} := \widetilde{\mathcal{Q}} \cup \mathcal{Q}$.
4. Output $(\widetilde{\mathcal{Q}}, \widetilde{\Pi})$.
```

We make explicit the two computation phases of the (stateful) malicious argument prover $\widetilde{\mathcal{P}}$:

$$(\mathsf{cm}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}} \quad \text{and} \quad (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho) \ ,$$

where aux is the auxiliary state passed across the two computation phases of $\widetilde{\mathcal{P}}$. The reductor $\mathcal{R}$ needs to rerun only the second phase of $\widetilde{\mathcal{P}}$, so the oracle for $\mathcal{R}$ is $\widetilde{\mathcal{P}}(\mathsf{aux}, \cdot)$.

As stated in Lemma 1, with the above notation we wish to bound the following probability:

$$\Pr\left[\begin{array}{l|l} \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho) \neq 1 & (\mathsf{cm},\mathsf{aux}) \leftarrow \widetilde{\mathcal{P}} \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho) = 1 & (\widetilde{\mathcal{Q}},\widetilde{\Pi}) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux},\cdot)}(\mathsf{cm},\epsilon) \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf}) = 1 & \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ & (\mathcal{Q},\mathsf{ans},\mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux},\rho) \end{array}\right] .$$

Observe that, if $\mathsf{VC.Check}(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1$ then $\mathbf{V}^{\widetilde{\Pi}}(\mathbb{x};\rho) \neq 1 \wedge \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho) = 1$ implies either: (i) $\widetilde{\Pi}$ and ans disagree at a position $q \in \mathcal{Q} \cap \widetilde{\mathcal{Q}}$; or (ii) there is query q in $\mathcal{Q}$ but not in $\widetilde{\mathcal{Q}}$. We analyze the two events separately, which bounds the probability above by a union bound.

(i) Valid openings with disagreeing answers. We informally argue that

$$\Pr\left[\begin{array}{l|l} & (\mathsf{cm},\mathsf{aux}) \leftarrow \widetilde{\mathcal{P}} \\ \exists\, q \in \mathcal{Q} \cap \widetilde{\mathcal{Q}} : \mathsf{ans}[q] \neq \widetilde{\Pi}[q] & (\widetilde{\mathcal{Q}},\widetilde{\Pi}) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux},\cdot)}(\mathsf{cm},\epsilon) \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf}) = 1 & \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ & (\mathcal{Q},\mathsf{ans},\mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux},\rho) \end{array}\right] \leq \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) \ .$$

The reductor $\mathcal{R}$ checks the validity of the opening for each position it fills into $\widetilde{\Pi}$. Therefore, the event above implies that there are valid openings to two different values at the same query position; equivalently, the following adversary A_{VC}, which has size $t_{\mathsf{VC}} = O(\mathsf{N} \cdot t_{\mathsf{ARG}})$, breaks VC's position binding:

```
$A_{\mathsf{VC}}$:
1. Run $(\mathsf{cm}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}}$.
2. Run the reductor $\mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux},\cdot)}(\mathsf{cm},\epsilon)$ and collect the valid query-answer-opening tuples $(\mathcal{Q}, \mathsf{ans}, \mathsf{pf})$ it sampled in a set $\mathcal{K}$.
3. Sample fresh randomness $\rho \leftarrow \{0,1\}^{\mathsf{r}}$ and obtain another tuple $(\mathcal{Q}', \mathsf{ans}', \mathsf{pf}') \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho)$.
4. For every tuple $(\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \in \mathcal{K}$, if there exists a query $q \in \mathcal{Q} \cap \mathcal{Q}'$ such that $\mathsf{ans}[q] \neq \mathsf{ans}'[q]$, then output $(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}, \mathcal{Q}', \mathsf{ans}', \mathsf{pf}')$.[12]
```

[12]To conform with the algorithm's syntax, A_{VC} outputs $(\mathsf{cm}, \mathcal{Q}', \mathsf{ans}', \mathsf{pf}', \mathcal{Q}', \mathsf{ans}', \mathsf{pf}')$ if no disagreeing openings are found.

Note that A_{VC} only checks inconsistencies between $(\mathcal{Q}', \mathsf{ans}', \mathsf{pf}')$ and tuples in $\mathcal{K}$. In other words, A_{VC} does not attempt to detect inconsistencies among tuples in $\mathcal{K}$. Even so, A_{VC} captures the bound we wish to prove:

$$\Pr\left[\begin{array}{l|l} \exists\, q \in \mathcal{Q} \cap \widetilde{\mathcal{Q}} : \mathsf{ans}[q] \neq \widetilde{\Pi}[q] & (\mathsf{cm}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}} \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 & (\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux},\cdot)}(\mathsf{cm}, \epsilon) \\ & \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ & (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho) \end{array}\right]$$
$$\leq \Pr\left[\begin{array}{l|l} \exists\, q \in \mathcal{Q} \cap \mathcal{Q}' : \mathsf{ans}[q] \neq \mathsf{ans}'[q] & \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 & (\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}, \mathcal{Q}', \mathsf{ans}', \mathsf{pf}') \leftarrow A_{\mathsf{VC}} \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm}, \mathcal{Q}', \mathsf{ans}', \mathsf{pf}') = 1 & \end{array}\right]$$
$$\leq \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) \ .$$

(ii) Missing position in $\widetilde{\Pi}$. We show that

$$\Pr\left[\begin{array}{l|l} \mathcal{Q} \setminus \widetilde{\mathcal{Q}} \neq \emptyset & (\mathsf{cm}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}} \\ \wedge\, \mathsf{VC.Check}(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 & (\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux},\cdot)}(\mathsf{cm}, \epsilon) \\ & \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ & (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho) \end{array}\right] \leq \frac{\mathsf{l}}{\mathsf{N}} \ .$$

To upper bound the probability of a query $q \in \mathcal{Q}$ not having been filled in by $\mathcal{R}$, we use the probability that a given position $q \in [\mathsf{l}]$ is queried. The *weight* $\delta(q)$ of a query $q \in [\mathsf{l}]$ is the probability that it is queried by the argument verifier with uniformly sampled randomness. We can write:

$$\begin{aligned} &\Pr\left[\mathcal{Q} \setminus \widetilde{\mathcal{Q}} \neq \emptyset\right] \\ &= \Pr\left[\exists\, q \in [\mathsf{l}] : q \in \mathcal{Q} \wedge q \notin \widetilde{\mathcal{Q}}\right] \\ &\leq \sum_{q \in [\mathsf{l}]} \delta(q) \cdot (1 - \delta(q))^{\mathsf{N}} \ , \end{aligned}$$

where the inequality follows from the fact that $\mathcal{Q}$ and all query sets used to generate $\widetilde{\mathcal{Q}}$ correspond to independently sampled verifier randomness. Note that, for every $\delta \in [0,1]$, $\delta \cdot (1-\delta)^{\mathsf{N}} \leq 1/\mathsf{N}$.[13] Hence, the target probability is upper bounded by $\frac{\mathsf{l}}{\mathsf{N}}$.

In fact, the proof for this case is more delicate than sketched above. If a position $q \in [\mathsf{l}]$ has weight $\delta(q)$, we cannot conclude that $q \notin \widetilde{\mathcal{Q}}$ with probability at most $(1-\delta(q))^{\mathsf{N}}$, because $\widetilde{\mathcal{P}}$ may often output invalid openings for q while $\mathcal{R}$ only includes valid openings. To fix this issue, we use a refined notion: $\delta(q)$ is the probability that during the execution of the interactive argument, the verifier $\mathcal{V}$ samples randomness that corresponds to a query set containing *q and the prover $\mathcal{P}$ outputs a valid VC opening for the query set.*

Setting parameters. By an union bound, the desired probability can be upper bounded by

$$\epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) + \frac{\mathsf{l}}{\mathsf{N}} \ .$$

Setting $\mathsf{N} := \frac{\mathsf{l}}{\epsilon}$, we get $t_{\mathsf{VC}} = O(\mathsf{N} \cdot t_{\mathsf{ARG}}) = O\left(\frac{\mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}\right)$ and $\frac{\mathsf{l}}{\mathsf{N}} = \epsilon$, yielding the bound stated in Lemma 1.

[13] A simple derivation of the inequality is the following: with $f(x) = x \cdot (1-x)^{\mathsf{N}}$, we have $\frac{d}{dx} f(\delta) = 0 \iff \delta = \frac{1}{\mathsf{N}+1}$. As $f(0) = f(1) = 0$ and δ is the only critical point in $[0,1]$, it achieves the maximum: $\max_{x \in [0,1]} \{f(x)\} = f(\delta) \leq 1/\mathsf{N}$.

Remark 2.1. Superficially one might hope for an improved analysis showing that one only needs $\frac{\mathsf{l}}{\mathsf{q}\cdot\epsilon}$ rewindings rather than $\frac{\mathsf{l}}{\epsilon}$. Indeed, each rewinding that leads to an accepting transcript yields a freshly sampled fragment of the PCP containing q locations. However such a bound is unrealistic because, in general, a PCP may have many dummy queries. For example, consider a PCP where only $O(1)$ of the q queries are "real", while all others are dummy queries to fixed locations of the PCP string. That said, there may be other metrics through which the factor $\frac{\mathsf{l}}{\epsilon}$ can be slightly improved. We leave this intriguing question to future work.

2.2 Succinct arguments based on public-coin IOPs

We review the IBCS protocol and sketch the main ideas behind Theorem 2; see Section 5 for details.

2.2.1 IBCS protocol

The IBCS protocol (the interactive variant of [BCS16]) obtains a succinct interactive argument by combining two ingredients: a public-coin interactive oracle proof (IOP) and a vector commitment scheme VC (fixed above). Let $\mathsf{IOP} = (\mathbf{P}, \mathbf{V})$ be a public-coin IOP for a relation R with alphabet Σ, round complexity k, proof lengths $(\mathsf{l}_i)_{i\in[\mathsf{k}]}$ and query complexities $(\mathsf{q}_i)_{i\in[\mathsf{k}]}$ (with maximal values $\mathsf{l}_{\mathsf{max}}, \mathsf{q}_{\mathsf{max}}$ and total values l, q) and verifier randomness complexity r (r_i per round). $\mathsf{IBCS}[\mathsf{IOP}, \mathsf{VC}]$ is an interactive argument $\mathsf{ARG} = (\mathcal{P}, \mathcal{V})$ in which the argument prover $\mathcal{P}$ receives an instance $\mathbb{x}$ and a witness $\mathbb{w}$, and the argument verifier $\mathcal{V}$ receives the instance $\mathbb{x}$. Then $\mathcal{P}$ and $\mathcal{V}$ interact, across $\mathsf{k}+1$ rounds (exchanging $2\mathsf{k}+1$ messages), as follows.

1. For every round $i \in [\mathsf{k}]$ of the IOP system:
   (a) $\mathcal{P}$ computes the IOP string $\Pi_i \leftarrow \mathbf{P}(\mathbb{x}, \mathbb{w}, \rho_{i-1})$, computes the commitment $(\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \mathsf{VC.Commit}(\Pi_i)$, and sends cm_i to $\mathcal{V}$.
   (b) $\mathcal{V}$ samples the i-th IOP verifier randomness $\rho_i \leftarrow \{0,1\}^{\mathsf{r}_i}$ and sends it to $\mathcal{P}$.
2. $\mathcal{P}$ runs the IOP verifier $\mathbf{V}^{(\Pi_i)_{i\in[\mathsf{k}]}}\big(\mathbb{x}, (\rho_i)_{i\in[\mathsf{k}]}\big)$ to deduce the query sets $(\mathcal{Q}_i)_{i\in[\mathsf{k}]} \subseteq [\mathsf{l}_1] \times \cdots \times [\mathsf{l}_\mathsf{k}]$ of $\mathbf{V}$ (where $\mathcal{Q}_i$ is the set of queries to the i-th IOP string), computes opening proofs $\mathsf{pf}_i \leftarrow \mathsf{VC.Open}(\mathsf{aux}_i, \mathcal{Q}_i)$, and sets $\mathsf{ans}_i := \Pi_i[\mathcal{Q}_i]$ for each $i \in [\mathsf{k}]$, then sends $((\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i))_{i\in[\mathsf{k}]}$ to $\mathcal{V}$.
3. $\mathcal{V}$ performs the following checks.
   (a) $\mathsf{VC.Check}(\mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i) = 1$ for every $i \in [\mathsf{k}]$ (i.e., all answers have valid openings);
   (b) $\mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big) = 1$ (i.e., the IOP verifier $\mathbf{V}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big)$ accepts the answers).

Similarly to Section 2.1.1, $\mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big)$ refers to the decision bit of the IOP verifier $\mathbf{V}$, given instance $\mathbb{x}$ and IOP randomness $(\rho_i)_{i\in[\mathsf{k}]}$, when each query $q \in \mathcal{Q}_i$ to the i-th oracle is answered with $\mathsf{ans}_i[q]$. (If $\mathbf{V}$ queries an oracle outside the corresponding query set then the decision bit is 0.)

2.2.2 Security reduction

The analysis in Section 2.1 extends to the case of public-coin IOPs with a more complicated reductor $\mathcal{R}$, which is responsible for producing, for each round, an appropriate IOP string for that round, given a partial transcript of interaction so far. Analogously to Lemma 1, the key step of the security reduction is this lemma.

Lemma 2 (informal)**.** *There exists a probabilistic algorithm $\mathcal{R}$ (the **reductor**) that, for every instance $\mathbb{x}$,* *error parameter $\epsilon > 0$, adversary size bound t_{ARG}, and t_{ARG}-size adversary $\widetilde{\mathcal{P}}$, satisfies*

$$\Pr\left[\begin{array}{l|l} \mathbf{V}^{([\widetilde{\mathcal{Q}}_i,\widetilde{\Pi}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x};(\rho_i)_{i\in[\mathsf{k}]}\big) \neq 1 & \text{For } i \in [\mathsf{k}]: \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i,\mathsf{ans}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x};(\rho_i)_{i\in[\mathsf{k}]}\big) = 1 & \quad (\mathsf{cm}_i,\mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1},\rho_{i-1}) \\ \wedge\, \Big(\bigwedge_{i\in[\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{cm}_i,\mathcal{Q}_i,\mathsf{ans}_i,\mathsf{pf}_i\big)\Big) = 1 & \quad (\widetilde{\mathcal{Q}}_i,\widetilde{\Pi}_i) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)}\big((\mathsf{cm}_j)_{j\le i},(\rho_j)_{j<i},\epsilon\big) \\ & \quad \rho_i \leftarrow \{0,1\}^{\mathsf{r}_i} \\ & \big((\mathcal{Q}_i,\mathsf{ans}_i,\mathsf{pf}_i)\big)_{i\in[\mathsf{k}]} \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{\mathsf{k}},\rho_{\mathsf{k}}) \end{array}\right]$$

$$\le \epsilon_{\mathsf{VC}}(\lambda,\mathsf{l}_{\mathsf{max}},\mathsf{q}_{\mathsf{max}},t_{\mathsf{VC}}) + \epsilon \ ,$$

where $t_{\mathsf{VC}} = O\left(\frac{\mathsf{k}\cdot\mathsf{l}}{\epsilon}\cdot t_{\mathsf{ARG}}\right)$.

Observe that Lemma 2 generalizes Lemma 1, which corresponds to the case $\mathsf{k} = 1$.

Similar to the reductor in the PCP case, $\mathcal{R}$ rewinds the IOP adversary to extract an IOP string for each round. In particular, for each round i, the reductor $\mathcal{R}$ is given oracle access to $\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)$ (where aux_i is the auxiliary state output along with the i-th commitment cm_i) and has the following syntax. For round i, $\mathcal{R}$ receives as input commitments $(\mathsf{cm}_j)_{j\le i}$, (partial) verifier randomness $(\rho_j)_{j<i} \in \{0,1\}^{\mathsf{r}_1}\times\cdots\times\{0,1\}^{\mathsf{r}_{i-1}}$, and error tolerance $\epsilon > 0$; then $\mathcal{R}$ outputs an IOP string $\widetilde{\Pi}_i$ whose entries in $\widetilde{\mathcal{Q}}_i$ are valid openings.

With Lemma 2, we can obtain the soundness and knowledge soundness error bounds of Theorem 2 via an analysis analogous to Sections 2.1.3 and 2.1.4. In this overview we omit this straightforward generalization, and only sketch the proof of Lemma 2.

2.2.3 Proof sketch of Lemma 2

The reductor $\mathcal{R}$ is faced with two challenges. On the one hand, for each round i, the IOP string $\widetilde{\Pi}_i$ to be extracted depends on prior randomness $(\rho_j)_{j<i}$ (since the corresponding commitment cm_i depends on these). On the other hand, for each round i, queries to $\widetilde{\Pi}_i$ may depend on prior randomness $(\rho_j)_{j<i}$ and subsequent randomness $(\rho_j)_{i\le j\le\mathsf{k}}$. (Indeed, all queries by the IOP verifier may occur after the interaction because we consider public-coin IOPs.) Therefore, $\mathcal{R}$ must sample $(\rho_j)_{i\le j\le\mathsf{k}}$ (i.e., IOP verifier randomness for all rounds $j \ge i$) to see a fragment of $\widetilde{\Pi}_i$. We use $\mathsf{N} = \mathsf{N}(\epsilon)$ to denote the number of samples that we set later.

For every $i \in [\mathsf{k}]$, the reductor algorithm $\mathcal{R}$ works as follows.

```
$\mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)}\big((\mathsf{cm}_j)_{j\le i},(\rho_j)_{j<i},\epsilon\big)$:
1. Initialize an IOP string with an arbitrary symbol $\sigma \in \Sigma$: set $\widetilde{\Pi}_i := (\sigma)^{\mathsf{l}_i}$.
2. Initialize an empty set $\widetilde{\mathcal{Q}}_i$ to store filled locations of $\widetilde{\Pi}_i$.
3. Repeat the following $\mathsf{N}$ times:
   (a) Sample IOP verifier randomness from round $i$ to round $\mathsf{k}$: $(\rho_i,\ldots,\rho_{\mathsf{k}}) \leftarrow \{0,1\}^{\mathsf{r}_i+\cdots+\mathsf{r}_{\mathsf{k}}}$.
   (b) Run the prover $\widetilde{\mathcal{P}}$ using the sampled randomness $(\rho_i,\ldots,\rho_{\mathsf{k}})$ until the end of interaction, and
       obtain the query-answer-opening tuple $(\mathcal{Q}_i,\mathsf{ans}_i,\mathsf{pf}_i)$ for the $i$-th IOP string.
   (c) If $\mathsf{VC.Check}(\mathsf{cm}_i,\mathcal{Q}_i,\mathsf{ans}_i,\mathsf{pf}_i) = 1$, set $\widetilde{\Pi}_i[\mathcal{Q}_i] := \mathsf{ans}_i$ and update $\widetilde{\mathcal{Q}}_i := \widetilde{\mathcal{Q}}_i \cup \mathcal{Q}_i$.
4. Output $(\widetilde{\mathcal{Q}}_i,\widetilde{\Pi}_i)$.
```

With the above construction, we can prove Lemma 2 with a similar case analysis as Section 2.1.5: if the IOP verifier $\mathbf{V}$, upon querying the proof strings output by $\mathcal{R}$, disagrees with the argument verifier $\mathcal{V}$, then either:

(i) for some $i \in [\mathsf{k}]$, $\widetilde{\Pi}_i$ and ans_i disagree at a position $q \in \mathcal{Q}_i \cap \widetilde{\mathcal{Q}}_i$; or
(ii) for some $i \in [\mathsf{k}]$, there is query q in $\mathcal{Q}_i$ but not in $\widetilde{\mathcal{Q}}_i$.

We bound the probability of the two events separately. We fix this experiment for the rest of the proof:

$$\left[\begin{array}{l} \text{For } i \in [\mathsf{k}] : \\ \quad (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \rho_{i-1}) \\ \quad (\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i, \cdot)}\big((\mathsf{cm}_j)_{j \le i}, (\rho_j)_{j<i}, \epsilon\big) \\ \quad \rho_i \leftarrow \{0,1\}^{\mathsf{r}_i} \\ \big((\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i)\big)_{i \in [\mathsf{k}]} \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_\mathsf{k}, \rho_\mathsf{k}) \end{array}\right] .$$

(i) Valid openings with disagreeing answers. We informally argue that

$$\Pr\left[\begin{array}{l} \exists\, i \in [\mathsf{k}], q \in \mathcal{Q}_i \cap \widetilde{\mathcal{Q}}_i : \mathsf{ans}_i[q] \neq \widetilde{\Pi}_i[q] \\ \wedge \left(\bigwedge_{i \in [\mathsf{k}]} \mathsf{VC.Check}(\mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i)\right) = 1 \end{array}\right] \le \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}_{\mathsf{max}}, \mathsf{q}_{\mathsf{max}}, t_{\mathsf{VC}}) .$$

We construct an adversary A_{VC} for VC with $\mathcal{R}$ and $\widetilde{\mathcal{P}}$ as in Section 2.1.5.

```
A_VC:
1. Sample $(\rho_i)_{i \in [\mathsf{k}]} \leftarrow \{0,1\}^{\mathsf{r}_1 + \cdots + \mathsf{r}_\mathsf{k}}$.
2. For every $i \le \mathsf{k}$:
   (a) Run $(\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \rho_{i-1})$.
   (b) Run $\mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i, \cdot)}((\mathsf{cm}_j)_{j \le i}, (\rho_j)_{j<i}, \epsilon)$ and collect the valid query-answer-opening tuples in a set $\mathcal{K}_i$.
3. Run $((\mathcal{Q}'_i, \mathsf{ans}'_i, \mathsf{pf}'_i))_{i \in [\mathsf{k}]} \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_\mathsf{k}, \rho_\mathsf{k})$.
4. If there exists $i \in [\mathsf{k}]$ and $(\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i) \in \mathcal{K}_i$ such that $q \in \mathcal{Q}_i \cap \mathcal{Q}'_i$ and $\mathsf{ans}_i[q] \neq \mathsf{ans}'_i[q]$, output
   $(\mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i, \mathcal{Q}'_i, \mathsf{ans}'_i, \mathsf{pf}'_i)$. (Otherwise, simply output $(\mathsf{cm}_1, \mathcal{Q}'_1, \mathsf{ans}'_1, \mathsf{pf}'_1, \mathcal{Q}'_1, \mathsf{ans}'_1, \mathsf{pf}'_1)$.)
```

Once again, the VC adversary A_{VC} only needs to check for inconsistent answers between $((\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i))_{i \in [\mathsf{k}]}$ and $(\mathcal{K}_i)_{i \in [\mathsf{k}]}$, rather than also checking inconsistencies within each $\mathcal{K}_i$.

The VC adversary A_{VC} simulates round i of $\widetilde{\mathcal{P}}$ for $i \cdot (\mathsf{N} + 1)$ times (see construction of $\mathcal{R}$), so A_{VC} runs at most kN full executions of $\widetilde{\mathcal{P}}$. Since one full execution of $\widetilde{\mathcal{P}}$ has size t_{ARG}, the size of A_{VC} is at most $O(\mathsf{kN} \cdot t_{\mathsf{ARG}})$. Hence setting $t_{\mathsf{VC}} = O(\mathsf{kN} \cdot t_{\mathsf{ARG}})$ bounds the probability above by $\epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}_{\mathsf{max}}, \mathsf{q}_{\mathsf{max}}, t_{\mathsf{VC}})$.

(ii) Missing query in $\widetilde{\mathcal{Q}}_i$. By extending the approach in Section 2.1.5, we show that, for every $\delta \in [0,1]$,

$$\Pr\left[\begin{array}{l} \left(\exists\, i \in [\mathsf{k}] : \mathcal{Q}_i \setminus \widetilde{\mathcal{Q}}_i \neq \emptyset\right) \\ \wedge \left(\bigwedge_{i \in [\mathsf{k}]} \mathsf{VC.Check}(\mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i)\right) = 1 \end{array}\right] \le \frac{\mathsf{l}}{\mathsf{N}} .$$

For each $i \in [\mathsf{k}]$ and a given position $q \in [\mathsf{l}_i]$, q has weight $\delta(q)$ with respect to a randomness prefix $\rho_1, \ldots, \rho_{i-1}$ if it is queried by the argument verifier $\mathcal{V}$ with probability $\delta(q)$ (over $\rho_{i+1}, \ldots, \rho_\mathsf{k}$).[14] Here we consider the *residual* probability after fixing a randomness prefix; this is necessary as this is the distribution that $\mathcal{R}$ samples from.

Therefore:

$$\begin{aligned} &\Pr\left[\exists\, i \in [\mathsf{k}] : \mathcal{Q}_i \setminus \widetilde{\mathcal{Q}}_i \neq \emptyset\right] \\ &= \Pr\left[\exists\, i \in [\mathsf{k}], q \in [\mathsf{l}_i] : q \in \mathcal{Q}_i \wedge q \notin \widetilde{\mathcal{Q}}_i\right] \end{aligned}$$

[14]Similarly to Section 2.1.5, here the definition of weight is simplified. In the full proof, we need to consider the fact that $\widetilde{\mathcal{P}}$ might output invalid openings.

$$
\begin{aligned}
&\leq \sum_{i=1}^{\mathsf{k}} \Pr\left[q \in [\mathsf{l}_i] : q \in \mathcal{Q}_i \wedge q \notin \widetilde{\mathcal{Q}}_i\right] \\
&\leq \sum_{i=1}^{\mathsf{k}} \sum_{q \in [\mathsf{l}_i]} \delta(q) \cdot (1 - \delta(q))^{\mathsf{N}} \\
&\leq \frac{\mathsf{l}}{\mathsf{N}} \ .
\end{aligned}
$$

Setting parameters. Setting $\mathsf{N} := \frac{\mathsf{l}}{\epsilon}$, we have that $t_{\mathsf{VC}} = O(\mathsf{kN} \cdot t_{\mathsf{ARG}}) = O\left(\frac{\mathsf{k} \cdot \mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}\right)$ and $\frac{\mathsf{l}}{\mathsf{N}} = \epsilon$ as desired.

2.3 Why doesn't the same analysis work for public-query IOPs?

We have discussed how the VC-based approach works for PCPs (in Section 2.1) and for public-coin IOPs (in Section 2.2). Here we discuss the capabilities and limitations of the this approach towards succinct interactive arguments. We begin by giving an informal definition of a general (private-coin) IOP.[15]

Definition 1 (informal)**.** *An* $\mathsf{IOP} = (\mathbf{P}, \mathbf{V})$ *with round complexity* k *and randomness complexity* r *works as follows.*

1. $\mathbf{V}$ *samples its private random string* $\rho \leftarrow \{0,1\}^{\mathsf{r}}$.
2. *For every round* $i \in [\mathsf{k}]$*:*
   (a) $\mathbf{P}$ *computes the* i*-th IOP string* $\Pi_i \leftarrow \mathbf{P}(\mathbb{x}, \mathbb{w}, m_{i-1})$ *and sends it to* $\mathbf{V}$.
   (b) $\mathbf{V}$ *computes the* i*-th query set* $\mathbf{Q}_i = (\mathcal{Q}_{i,j})_{j \leq i} \leftarrow \mathbf{V}_{\mathsf{q}}\big(\mathbb{x}, \rho, (\mathbf{ans}_j)_{j<i}\big)$*, where* $\mathcal{Q}_{i,j}$ *is the set of queries to the* j*-th IOP string in round* i*.*
   (c) $\mathbf{V}$ *obtains the answers* $\mathbf{ans}_i$ *to* $\mathbf{Q}_i$ *by querying* $(\Pi_j)_{j \leq i}$*.*
   (d) $\mathbf{V}$ *computes the* i*-th message* $m_i \leftarrow \mathbf{V}_{\mathsf{m}}\big(\mathbb{x}, \rho, (\mathbf{ans}_j)_{j \leq i}\big)$ *and sends it to* $\mathbf{P}$.
3. $\mathbf{V}$ *checks that* $\mathbf{V}_{\mathsf{d}}\big(\mathbb{x}, \rho, (\mathbf{ans}_i)_{i \in [\mathsf{k}]}\big) = 1$.

We use $\mathbf{V}_{\mathsf{q}}, \mathbf{V}_{\mathsf{m}}, \mathbf{V}_{\mathsf{d}}$ *to denote the query, message, and decision functions of the IOP verifier* $\mathbf{V}$*, respectively.*

We cannot hope for the VC-based approach to work for *every* IOP. The argument verifier needs the argument prover's help to obtain the answers to its queries; hence, the argument prover learns each round's queries as they are made. Unless the underlying IOP remains secure even when queries are revealed to the IOP prover, the VC-based approach is insecure.

Therefore, at best we can hope for the approach to work for *public-query IOPs*. These are IOPs in which soundness (and knowledge soundness) holds even if the IOP prover knows the queries made by the IOP verifier, or, equivalently, where the latter sends its i-th round query set $\mathbf{Q}_i$ along with the message m_i. This leads to the following (narrowed) question: *Does the VC-based approach work for every public-query IOP?*

In Section 2.3.1, we describe a succinct interactive argument that appears secure provided that the underlying IOP is public-query. In Section 2.3.2, we explain the challenges towards a proof of security for this protocol. In Section 2.3.3, we identify a property of public-query IOPs (which is trivially satisfied by public-coin IOPs) that suffices for security. In Section 2.3.4, we partially characterize the class of public-query IOPs that admit this property. Technical details are in Section 6.

[15]For notational simplicity, we assume that, within each round of interaction, the IOP verifier queries non-adaptively. All discussions in this paper directly extend to adaptive queries within each round.

2.3.1 The Finale protocol

We describe the *Finale* protocol, a succinct interactive argument obtained from any public-query IOP. It is a generalization of the IBCS protocol in Section 2.2.1 from public-coin IOPs to public-query IOPs. The main difference is that, while in the IBCS protocol the argument prover answers queries by the IOP verifier after the IOP interaction, the argument prover in the Finale protocol answers queries within each round.

Let $\mathsf{IOP} = (\mathbf{P}, \mathbf{V})$ be a public-query IOP for a relation R with alphabet Σ, round complexity k, maximal proof length $\mathsf{l}_{\mathsf{max}}$, maximal query complexity $\mathsf{q}_{\mathsf{max}}$, and verifier randomness complexity r. $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ is an interactive argument $\mathsf{ARG} = (\mathcal{P}, \mathcal{V})$ in which the argument prover $\mathcal{P}$ receives an instance $\mathbb{x}$ and a witness $\mathbb{w}$, and the argument verifier $\mathcal{V}$ receives the instance $\mathbb{x}$. Then $\mathcal{P}$ and $\mathcal{V}$ interact, across $2\mathsf{k}$ rounds (exchanging $4\mathsf{k}$ messages), as follows.

1. $\mathcal{V}$ samples the verifier randomness $\rho \leftarrow \{0,1\}^{\mathsf{r}}$ for $\mathbf{V}$.
2. For every round $i \in [\mathsf{k}]$ of the IOP system:
   (a) $\mathcal{P}$ computes the i-th IOP string $\Pi_i \leftarrow \mathbf{P}(\mathbb{x}, \mathbb{w}, m_{i-1})$, computes the corresponding commitment $(\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \mathsf{VC.Commit}(\Pi_i)$, and sends cm_i to $\mathcal{V}$.
   (b) $\mathcal{V}$ sends the i-th query set $\mathbf{Q}_i = (\mathcal{Q}_{i,j})_{j \le i} \leftarrow \mathbf{V}_{\mathsf{q}}\big(\mathbb{x}, \rho, (\mathbf{ans}_j)_{j<i}\big)$ to $\mathcal{P}$, where $\mathcal{Q}_{i,j}$ is the set of queries to Π_j in round i.
   (c) For each $j \le i$, $\mathcal{P}$ computes an opening proof $\mathsf{pf}_{i,j} \leftarrow \mathsf{VC.Open}(\mathsf{aux}_j, \mathcal{Q}_{i,j})$.
   (d) $\mathcal{P}$ sends $\big(\mathbf{ans}_i := (\Pi_i[\mathcal{Q}_{i,j}])_{j \le i}, \mathbf{pf}_i := (\mathsf{pf}_{i,j})_{j \le i}\big)$ to $\mathcal{V}$.
   (e) Then $\mathcal{V}$ sends the i-th message $m_i \leftarrow \mathbf{V}_{\mathsf{m}}\big(\mathbb{x}, \rho, (\mathbf{ans}_j)_{j \le i}\big)$ to $\mathcal{P}$.
3. $\mathcal{V}$ checks if the following conditions hold:
   (a) $\mathbf{V}_{\mathsf{d}}\big(\mathbb{x}, \rho, (\mathbf{ans}_i)_{i \in [\mathsf{k}]}\big) = 1$.
   (b) For every $i \in [\mathsf{k}]$ and $j \le i$, $\mathsf{VC.Check}(\mathsf{cm}_j, \mathcal{Q}_{i,j}, \mathsf{ans}_{i,j}, \mathsf{pf}_{i,j}) = 1$.

2.3.2 Is there a security reduction that works for every public-query IOP?

It would be natural to conjecture that the security analysis of the IBCS protocol (for public-coin IOPs) directly extends to a security analysis for the Finale protocol (for public-query IOPs). However, that is not the case.

The key object in the security analysis is the reductor: $\mathcal{R}$ rewinds the malicious argument prover $\widetilde{\mathcal{P}}$ multiple times, each time simulating a fresh partial interaction between $\widetilde{\mathcal{P}}$ and the (honest) argument verifier $\mathcal{V}$. Let us attempt to construct such a reductor $\mathcal{R}$ for the Finale protocol.

Consider the goal of the reductor $\mathcal{R}$ for the first round. The reductor $\mathcal{R}$ receives oracle access to $\widetilde{\mathcal{P}}$ and receives as input the first commitment cm_1 (and error parameter ϵ); the (first) goal of $\mathcal{R}$ is to output a (partially filled) IOP string $\widetilde{\Pi}_1$ consistent with cm_1. As before, $\mathcal{R}$ can rewind $\widetilde{\mathcal{P}}$ for $\mathsf{N}(\epsilon)$ times, each time sampling fresh private randomness for $\mathcal{V}$ and simulating a full interaction between $\widetilde{\mathcal{P}}$ and $\mathcal{V}$.

The problem arises in the second round, where $\mathcal{R}$ must simulate partial interactions of the interactive argument. Now $\mathcal{R}$ receives as input the first two commitments $(\mathsf{cm}_1, \mathsf{cm}_2)$, the first round's verifier queries $\mathbf{Q}_1$, the corresponding query answers $\mathbf{ans}_1$ and opening proofs $\mathbf{pf}_1$, and the first round's verifier message m_1 (as well as the error parameter ϵ). The new goal of $\mathcal{R}$ is to output a (partially filled) IOP string $\widetilde{\Pi}_2$ consistent with cm_2. To do so, $\mathcal{R}$ must simulate multiple partial interactions between $\widetilde{\mathcal{P}}$ and $\mathcal{V}$ *starting from the second round*. This means that $\mathcal{R}$ needs to sample "consistent" private randomness ρ for the argument verifier $\mathcal{V}$: ρ such that $\mathcal{V}$'s first query set is $\mathbf{Q}_1$ and, given answers $\mathbf{ans}_1$ to these queries, $\mathcal{V}$'s first message to the argument prover is m_1. More generally, $\mathcal{R}$ must have the capability of sampling consistent private randomness starting from any round (defined by a partial transcript of the interactive argument).

How may we sample from this conditional distribution? For the IBCS protocol, sampling a random continuation for a partial transcript is trivial because it is public coin: given first round randomness ρ_1, sample randomness $\rho_2, \ldots, \rho_{\mathsf{k}}$ for the remaining rounds; and similarly if starting from a later round. In contrast, in the Finale protocol, the argument verifier is private-coin: it samples all of its private randomness ρ at the beginning of the interaction and then uses ρ to deterministically compute queries and messages.

An attempt to sample from the conditional distribution would be for $\mathcal{R}$ to repeatedly sample choices of ρ until one is consistent with $((\mathsf{cm}_1, \mathsf{cm}_2), (\mathbf{Q}_1, \mathbf{ans}_1, m_1))$. However, such a sampling strategy would be inefficient, and would imply (in the resulting reduction) an adversary against VC whose similarly large runtime makes ϵ_{VC} trivial (recall that in Section 2.1.5 the VC adversary A_{VC} uses $\mathcal{R}$ as a subroutine).

In sum, the bottleneck of the security reduction approach in Section 2.2.2 is the ability to *efficiently and consistently sample argument verifier randomness consistent with a partial interaction transcript*, which in turn reduces to the ability to sample consistent IOP verifier randomness for the underlying public-query IOP (as we discuss soon in Section 2.3.3). We are not aware of alternative approaches, and the problem of proving the Finale protocol secure for every public-query IOP (or proving it insecure for some public-query IOP) **remains open**. In the meantime, to achieve a security reduction, we restrict our attention to public-query IOPs that possess efficient *random continuation samplers*, which we discuss next.

2.3.3 IOP random continuation sampler leads to security reduction

We sketch how, as claimed in Theorem 3, if the underlying public-query IOP has an efficient random continuation sampler then the Finale protocol is secure.

Transcripts for general IOPs. A complete interaction transcript for an IOP has the following form:

$$\mathbf{tr} := \big((\mathbf{Q}_i, \mathbf{ans}_i, m_i)\big)_{i \in [\mathsf{k}]} \ .$$

For every $i \in [\mathsf{k}]$, a partial interaction transcript $\mathbf{tr}_i$ for an IOP can have the following forms:

- $\mathbf{tr}_i = \big(\big((\mathbf{Q}_j, \mathbf{ans}_j, m_j)\big)_{j<i}, (\mathbf{Q}_i, \mathbf{ans}_i)\big)$, if the IOP verifier is about to send its i-th message m_i;
- $\mathbf{tr}_i = \big(\big((\mathbf{Q}_j, \mathbf{ans}_j, m_j)\big)_{j \le i}\big)$, if the IOP verifier is about to output its $(i+1)$-th query set $\mathbf{Q}_{i+1}$ ($i < \mathsf{k}$).

An IOP interaction transcript includes only the values of queried locations, and not entire IOP strings.

What is an IOP random continuation sampler? An IOP random continuation sampler (IOP-RCS) receives as input a partial interaction transcript $\mathbf{tr}_i$ for an IOP and samples the next message for $\mathbf{tr}_i$ at random among all next messages consistent with $\mathbf{tr}_i$. (In the technical sections, we also allow for a sampling error α.) Formally, an IOP-RCS $\mathbf{S}$ works as follows:

- If $\mathbf{tr}_i = \big(\big((\mathbf{Q}_j, \mathbf{ans}_j, m_j)\big)_{j<i}, (\mathbf{Q}_i, \mathbf{ans}_i)\big)$, then $\mathbf{S}(\mathbf{tr}_i)$ samples the next message m_i;
- If $\mathbf{tr}_i = \big(\big((\mathbf{Q}_j, \mathbf{ans}_j, m_j)\big)_{j \le i}\big)$ and $i < \mathsf{k}$, then $\mathbf{S}(\mathbf{tr}_i)$ samples the next query set $\mathbf{Q}_{i+1}$.

How does IOP-RCS help with proving security? As discussed in Section 2.3.2, the key step in the security reduction is efficiently sampling "consistent random continuations" of a given partial interaction transcript of the argument system. In other words, we need an *ARG random continuation sampler* (ARG-RCS) for the Finale protocol. The notion of ARG-RCS was implicit in prior settings: the construction of $\mathcal{R}$ for Kilian's protocol (in Section 2.1.5) and for the IBCS protocol (in Section 2.2.3) implicitly relies on an ARG-RCS, which is trivial for arguments built from PCPs or public-coin IOPs (it suffices to sample uniform random strings for future rounds). In the Finale protocol, based on public-query IOPs, an efficient ARG-RCS is not trivial. Fortunately, we can show that if IOP admits an IOP-RCS then $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ admits an ARG-RCS.

Step 1: IOP-RCS implies ARG-RCS. An ARG-RCS can be defined analogously to an IOP-RCS. A complete interaction transcript for $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ can be written as:

$$\mathsf{tr} := \big((\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i, m_i)\big)_{i\in[\mathsf{k}]} \ .$$

Such an interaction transcript is syntactically identical to an IOP interaction transcript, but for the additional inclusion of commitments cm_i and openings $\mathbf{pf}_i$. Moreover, the argument verifier's behavior in the interaction is independent of the commitments and openings (only after the interaction the argument verifier uses them to check the validity of the answers to the query sets). Hence the transformation from an IOP-RCS $\mathbf{S}$ to an ARG-RCS $\mathcal{S}$ is trivial: given a partial argument transcript tr_i, discard the commitments and openings and pass the resulting partial IOP transcript to $\mathbf{S}$. More specifically, $\mathcal{S}$ is obtained from $\mathbf{S}$ as follows:

```
$\mathcal{S}(\mathsf{tr}_i)$:
1. If $\mathsf{tr}_i = \big(\big((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j<i}, (\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i)\big)$, output $\mathbf{S}\big(\big(\big((\mathbf{Q}_j, \mathbf{ans}_j, m_j)\big)_{j<i}, (\mathbf{Q}_i, \mathbf{ans}_i)\big)\big)$;
2. If $\mathsf{tr}_i = \big(\big((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j\le i}\big)$ and $i < \mathsf{k}$, output $\mathbf{S}\big(\big(\big((\mathbf{Q}_j, \mathbf{ans}_j, m_j)\big)_{j\le i}\big)\big)$.
```

It is clear that $\mathcal{S}$ samples valid randomness if $\mathbf{S}$ does, and the running time of $\mathcal{S}$ is $O(t_{\mathbf{S}})$.

Step 2: ARG-RCS enables a security reduction. We can use an ARG-RCS $\mathcal{S}$ for the Finale protocol to construct an efficient reductor $\mathcal{R}$ for the Finale protocol. Indeed, we can use $\mathcal{S}$ to randomly sample the next query set and verifier message in the transcript.

```
$\mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)}((\mathsf{cm}_j)_{j\le i}, ((\mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j))_{j<i}, \epsilon)$:
1. Initialize an IOP string with an arbitrary symbol $\sigma \in \Sigma$: set $\widetilde{\Pi}_i := (\sigma)^{\mathsf{l}_i}$.
2. Initialize an empty set $\widetilde{\mathcal{Q}}_i$ to store filled locations of $\widetilde{\Pi}_i$.
3. Compute $\mathsf{N} := \mathsf{N}(\epsilon)$.
4. Repeat the following $\mathsf{N}$ times: For $i \le j \le \mathsf{k}$:
    (a) If $j \ne i$, compute the $j$-th commitment: $(\mathsf{cm}_j, \mathsf{aux}_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{j-1}, \mathbf{Q}_{j-1}, m_{j-1})$.
    (b) Sample the $j$-th IOP query set: $\mathbf{Q}_j \leftarrow \mathcal{S}\big(\big(\big((\mathsf{cm}_\ell, \mathbf{Q}_\ell, \mathbf{ans}_\ell, \mathbf{pf}_\ell, m_\ell)\big)_{\ell<j}\big)\big)$.
    (c) Get the $j$-th answer and VC openings from $\widetilde{\mathcal{P}}$: $(\mathbf{ans}_j, \mathbf{pf}_j, \mathsf{aux}_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_j, \mathbf{Q}_j)$.
    (d) Sample the $j$-th IOP verifier message: $m_j \leftarrow \mathcal{S}\big(\big(\big((\mathsf{cm}_\ell, \mathbf{Q}_\ell, \mathbf{ans}_\ell, \mathbf{pf}_\ell, m_\ell)\big)_{\ell<j}, (\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j)\big)\big)$.
    (e) If $\mathsf{VC.Check}(\mathsf{cm}_i, \mathcal{Q}_{j,i}, \mathsf{ans}_{j,i}, \mathsf{pf}_{j,i}) = 1$, set $\widetilde{\Pi}_i[\mathcal{Q}_{j,i}] := \mathsf{ans}_{j,i}$ and update $\widetilde{\mathcal{Q}}_i := \widetilde{\mathcal{Q}} \cup \mathcal{Q}_{j,i}$.
5. Output $(\widetilde{\mathcal{Q}}, \widetilde{\Pi}_i)$.
```

If the ARG-RCS $\mathcal{S}$ is efficient then the reductor $\mathcal{R}$ is efficient. We can then prove security by adapting Lemma 2 to this setting; we sketch how the proof changes to account for public queries.

As before (Sections 2.1.5 and 2.2.3), the security reduction claim can be split into two parts.

- *Valid openings with disagreeing answers.* We construct an adversary A_{VC} for VC with $\mathcal{R}$ and $\widetilde{\mathcal{P}}$ as in Section 2.1.5. The construction is similar, except that the verifier's messages are now denoted by m rather than ρ and the running time of A_{VC} needs to take into account the running time of the ARG-RCS $\mathcal{S}$ (in $\mathcal{R}$).

- *Missing query in $\widetilde{\Pi}_i$.* For every i-round partial transcript tr_i, we can define the weight of a position $q \in [\mathsf{l}_i]$ with respect to tr_i analogously as in Section 2.2.3, and the same analysis follows:

$$\Pr\left[\exists\, q \in [\mathsf{l}_i] : (\exists\, j \in [i, \mathsf{k}] : q \in \mathcal{Q}_{j,i}) \wedge q \notin \widetilde{\mathcal{Q}}_i\right] \le \frac{\mathsf{l}_i}{\mathsf{N}} \ .$$

Using a union bound over the rounds, we upper bound the target probability of this case as before by

$$\sum_{i=1}^{\mathsf{k}} \frac{\mathsf{l}_i}{\mathsf{N}} = \frac{\mathsf{l}}{\mathsf{N}} \ .$$

We conclude that when the Finale protocol (based on a public-query IOP) has an efficient random continuation sampler, we can show (knowledge) soundness via a similar strategy to Sections 2.1.3 and 2.1.4.

2.3.4 ARG-RCS implies IOP-RCS

We show that IOP has an efficient IOP-RCS if $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ admits an efficient ARG-RCS. Combining this with the discussion from Section 2.3.3, we conclude that the existence of an efficient ARG-RCS for the Finale protocol is equivalent to the existence of an efficient IOP-RCS for the underlying public-query IOP.

Let $\mathcal{S}$ be an ARG-RCS for $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ with running time $t_{\mathcal{S}}$. We wish to construct an IOP-RCS $\mathbf{S}$ for IOP. Let $\mathbf{tr}_i$ be a partial interaction transcript for the IOP. To invoke $\mathcal{S}$, the IOP sampler $\mathbf{S}$ needs to augment the IOP transcript to an argument transcript that includes commitments to each IOP string and openings to the commitments. For example, in order for $\mathcal{S}$ to output the i-th query set, the argument transcript given as input should include cm_j for every $j \leq i$. However, $\mathbf{S}$ has no information about the i-th IOP string.

We circumvent this issue by observing that the (honest) verifier in the Finale protocol ignores the VC commitments and openings until the interaction ends. Therefore, the IOP-RCS $\mathbf{S}$ can construct from $\mathbf{tr}_i$ an argument transcript to pass to $\mathcal{S}$ by providing dummy values for the commitments and openings, as follows:

$\mathbf{S}(\mathbf{tr}_i)$:
1. If $\mathbf{tr}_i = \big(\big((\mathbf{Q}_j, \mathbf{ans}_j, m_j)\big)_{j<i}, (\mathbf{Q}_i, \mathbf{ans}_i)\big)$, output $\mathcal{S}\big(\big(\big((\perp, \mathbf{Q}_j, \mathbf{ans}_j, \perp, m_j)\big)_{j<i}, (\perp, \mathbf{Q}_i, \mathbf{ans}_i, \perp)\big)\big)$.
2. If $\mathbf{tr}_i = \big(\big((\mathbf{Q}_j, \mathbf{ans}_j, m_j)\big)_{j\leq i}\big)$ and $i < \mathsf{k}$, output $\mathcal{S}\big(\big(\big((\perp, \mathbf{Q}_j, \mathbf{ans}_j, \perp, m_j)\big)_{j\leq i}\big)\big)$.

The correctness of $\mathbf{S}$ follows straightforwardly from the correctness of $\mathcal{S}$, and the running time of $\mathbf{S}$ is $O(t_{\mathcal{S}})$.

2.4 Succinct interactive arguments with adaptive security

For simplicity, all results and discussions in Sections 1 and 2 are in the *plain model*, where there are no public parameters available to all parties (so the argument verifier is responsible to sample and send VC's public parameters to the argument prover). However, in the technical sections (Sections 4 to 6) we show stronger versions of Theorems 1 to 3 that hold with adaptive security in the common reference string (CRS) model.

An interactive argument in the CRS model includes an additional algorithm: a trusted *generator algorithm* that samples public parameters pp for the argument prover and argument verifier (which can be used any number of times across different interactions). After that, based on pp, a malicious argument prover can choose the instance on which to interact with the argument verifier. This setting necessitates appropriate definitions of *adaptive* soundness and knowledge soundness (see Section 3.1), which require error bounds to hold for any instance $\mathbb{x}$ chosen by the malicious argument prover up to an instance size bound n.[16] In particular, the (soundness and knowledge soundness) error bounds depend on n rather than $\mathbb{x}$.

Our security analyses to achieve adaptive security in the CRS model follow the same structure as the discussions in the sections above, with only syntactic modifications due to the different target definitions.

[16]We also rely on formulations of soundness and knowledge soundness for PCPs and IOPs in which the malicious prover chooses the instance (see Sections 3.3 and 3.4). This is only for convenience, because, due to the information-theoretic setting, these definitions are straightforwardly implied by (standard) definitions for fixed instances.

(E.g., modifying the experiments to replace a fixed instance $\mathbb{x}$ with an instance size bound n, and letting the malicious argument prover choose the instance.)

Overall, the (formal) statements provided in the technical sections (Sections 4 to 6) are stronger than the (informal) statements in Theorems 1 to 3 because we achieve adaptive security in the CRS model.[17]

[17]Adaptive security in the CRS model directly implies security in the plain model. Since no CRS is allowed, the argument verifier can begin the interaction by running itself the generator algorithm and sending the public parameters for the argument system to the argument prover. See Remark 3.6.

3 Preliminaries

Definition 3.1. *A* **relation** R *is a set of pairs* $(\mathbb{x}, \mathbb{w})$ *where* $\mathbb{x}$ *is an instance and* $\mathbb{w}$ *a witness. The corresponding* **language** $L(R)$ *is the set of instances* $\mathbb{x}$ *for which there exists a witness* $\mathbb{w}$ *such that* $(\mathbb{x}, \mathbb{w}) \in R$.

3.1 Interactive arguments

An *interactive argument* (in the common reference string model) for a relation R is a tuple of polynomial-time algorithms $\mathsf{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ that satisfies the following properties.

Definition 3.2 (Perfect completeness)**.** $\mathsf{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *for a relation* R *has* **perfect completeness** *if for every security parameter* $\lambda \in \mathbb{N}$, *instance size bound* $n \in \mathbb{N}$, *public parameter* $\mathsf{pp} \in \mathcal{G}(1^\lambda, n)$*, and instance-witness pair* $(\mathbb{x}, \mathbb{w}) \in R$ *with* $|\mathbb{x}| \leq n$*,*

$$\Pr\left[\left\langle \mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \right\rangle = 1\right] = 1 \ .$$

Definition 3.3 (Adaptive soundness)**.** $\mathsf{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *for a relation* R *has* **(adaptive) soundness error** ϵ_{ARG} *if for every security parameter* $\lambda \in \mathbb{N}$*, instance size bound* $n \in \mathbb{N}$*, auxiliary input distribution* $\mathcal{D}$*, circuit size bound* $t_{\mathsf{ARG}} \in \mathbb{N}$*, and* t_{ARG}*-size circuit* $\widetilde{\mathcal{P}}$*,*

$$\Pr\left[\begin{array}{l|l} |\mathbb{x}| \leq n & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \wedge\, \mathbb{x} \notin L(R) & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, b = 1 & (\mathbb{x}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ & b \leftarrow \left\langle \widetilde{\mathcal{P}}(\mathsf{aux}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \right\rangle \end{array}\right] \leq \epsilon_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) \ .$$

Definition 3.4 (Adaptive knowledge soundness)**.** $\mathsf{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *for a relation* R *has* **(adaptive) knowledge soundness error** κ_{ARG} **with extraction time** $t_{\mathcal{E}}$ *if there exists a probabilistic algorithm* $\mathcal{E}$ *such that for every security parameter* $\lambda \in \mathbb{N}$*, instance size bound* $n \in \mathbb{N}$*, auxiliary input distribution* $\mathcal{D}$*, circuit size bound* $t_{\mathsf{ARG}} \in \mathbb{N}$*, and* t_{ARG}*-size circuit* $\widetilde{\mathcal{P}}$*,*

$$\Pr\left[\begin{array}{l|l} & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ |\mathbb{x}| \leq n & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R & (\mathbb{x}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \wedge\, b = 1 & b \xleftarrow{\mathsf{tr}} \left\langle \widetilde{\mathcal{P}}(\mathsf{aux}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \right\rangle \\ & \mathbb{w} \leftarrow \mathcal{E}^{\widetilde{\mathcal{P}}(\mathsf{aux})}(\mathsf{pp}, \mathbb{x}, \mathsf{tr}) \end{array}\right] \leq \kappa_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) \ ;$$

moreover, $\mathcal{E}$ *runs in time* $t_{\mathcal{E}}(\lambda, n, t_{\mathsf{ARG}})$.

Above, $b \xleftarrow{\mathsf{tr}} \langle \widetilde{\mathcal{P}}(\mathsf{aux}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \rangle$ denotes the fact that tr is the transcript of the interaction (i.e., public parameters and messages exchanged between $\widetilde{\mathcal{P}}$ and $\mathcal{V}$). Moreover, $\mathcal{E}^{\widetilde{\mathcal{P}}}$ means that $\mathcal{E}$ has black-box access to (each next-message function of) $\widetilde{\mathcal{P}}$; in particular $\mathcal{E}$ can send verifier messages to $\widetilde{\mathcal{P}}$ in order to obtain the next message of $\widetilde{\mathcal{P}}$ (for a partial interaction where $\mathcal{V}$ sent those messages).

Moreover, we can assume, without loss of generality, that $\widetilde{\mathcal{P}}$ is deterministic relative to auxiliary input ai (as the internal coin flips of a probabilistic $\widetilde{\mathcal{P}}$ can be incorporated into the auxiliary input distribution $\mathcal{D}$).

Remark 3.5. The argument generator $\mathcal{G}$ receives two inputs: the security parameter λ and an instance size bound n. This means that the public parameter sampled by $\mathcal{G}$ may work only for instances of size at most n. However, one could consider the stronger notion where the sampled public parameter works for all instance sizes; in this case $\mathcal{G}$ receives only λ as input. Our analysis works for both cases; see Remark 4.4.

Remark 3.6 (plain model variant)**.** The above definitions consider interactive arguments in the *common reference string model*, where a generator samples a public parameter used by the argument prover and the argument verifier. One could also consider interactive arguments in the *plain model*, where there is no generator. This latter notion is implied, at the cost of an additional verifier message, as we now explain.

Suppose that $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is an interactive argument in the common reference string model. We describe an interactive argument $(\mathcal{P}', \mathcal{V}')$ in the plain model with an additional verifier message. The argument prover $\mathcal{P}'$ receives as input an instance $\mathbb{x}$ and witness $\mathbb{w}$, and the argument verifier $\mathcal{V}'$ receives as input the instance $\mathbb{x}$; both also receive as input the security parameter λ (in unary). They interact as follows:

- $\mathcal{V}'$ samples a public parameter $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, |\mathbb{x}|)$ and sends pp to $\mathcal{P}'$;
- $\mathcal{P}'$ and $\mathcal{V}'$ simulate an interaction of $\mathcal{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ and $\mathcal{V}(\mathsf{pp}, \mathbb{x})$.

It is straightforward to see that $(\mathcal{P}', \mathcal{V}')$ satisfies the standard definitions of completeness, soundness, and knowledge soundness for interactive arguments in the plain model.[18] In fact, it would suffice for $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ to satisfy the non-adaptive relaxations of soundness and knowledge soundness.

3.2 Vector commitments

A (static) *vector commitment scheme* [CF13] over alphabet Σ is a tuple of algorithms

$$\mathsf{VC} = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Check})$$

with the following syntax.

- $\mathsf{VC.Gen}(1^\lambda, \ell) \to \mathsf{pp}$: On input a security parameter $\lambda \in \mathbb{N}$ and message size bound $\ell \in \mathbb{N}$, $\mathsf{VC.Gen}$ samples public parameter pp.
- $\mathsf{VC.Commit}(\mathsf{pp}, m) \to (\mathsf{cm}, \mathsf{aux})$: On input a public parameter pp and a message $m \in \Sigma^\ell$, $\mathsf{VC.Commit}$ produces a commitment cm and the corresponding auxiliary state aux.
- $\mathsf{VC.Open}(\mathsf{pp}, \mathsf{aux}, \mathcal{Q}) \to \mathsf{pf}$: On input a public parameter pp, an auxiliary state aux, and a query set $\mathcal{Q} \subseteq [\ell]$, $\mathsf{VC.Open}$ outputs an opening proof string pf attesting that $m[\mathcal{Q}]$ is a restriction of m to $\mathcal{Q}$.
- $\mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \to \{0,1\}$: On input a public parameter pp, a commitment cm, a query set $\mathcal{Q} \subseteq [\ell]$, an answer string $\mathsf{ans} \in \Sigma^{\mathcal{Q}}$, and an opening proof string pf, $\mathsf{VC.Check}$ determines if pf is a valid proof for $\mathsf{ans} \in \Sigma^{\mathcal{Q}}$ being a restriction of the message committed in cm to $\mathcal{Q}$.

The vector commitment scheme VC must satisfy perfect completeness and position binding.

Definition 3.7 (Completeness)**.** $\mathsf{VC} = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Check})$ *has* **perfect completeness** *if for every security parameter $\lambda \in \mathbb{N}$, message length $\ell \in \mathbb{N}$, message $m \in \Sigma^\ell$, and query set $\mathcal{Q} \subseteq [\ell]$,*

$$\Pr\left[\mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, m[\mathcal{Q}], \mathsf{pf}) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{VC.Gen}(1^\lambda, \ell) \\ (\mathsf{cm}, \mathsf{aux}) \leftarrow \mathsf{VC.Commit}(\mathsf{pp}, m) \\ \mathsf{pf} \leftarrow \mathsf{VC.Open}(\mathsf{pp}, \mathsf{aux}, \mathcal{Q}) \end{array}\right] = 1 \enspace .$$

Definition 3.8 (Position binding)**.** $\mathsf{VC} = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Check})$ *has* **position binding error** ϵ_{VC} *if for every security parameter $\lambda \in \mathbb{N}$, message length $\ell \in \mathbb{N}$, query set size $s \in \mathbb{N}$ with $s \leq \ell$, auxiliary input distribution $\mathcal{D}$, adversary size bound $t_{\mathsf{VC}} \in \mathbb{N}$, and t_{VC}-size circuit A_{VC},*

$$\Pr\left[\begin{array}{l} |\mathcal{Q}| = |\mathcal{Q}'| = s \\ \wedge\ \exists\, i \in \mathcal{Q} \cap \mathcal{Q}' : \mathsf{ans}[i] \neq \mathsf{ans}'[i] \\ \wedge\ \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 \\ \wedge\ \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}', \mathsf{ans}', \mathsf{pf}') = 1 \end{array} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{VC.Gen}(1^\lambda, \ell) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ \begin{pmatrix} \mathsf{cm}, \mathsf{ans}, \mathsf{ans}', \\ \mathcal{Q}, \mathcal{Q}', \mathsf{pf}, \mathsf{pf}' \end{pmatrix} \leftarrow A_{\mathsf{VC}}(\mathsf{pp}, \mathsf{ai}) \end{array}\right] \leq \epsilon_{\mathsf{VC}}(\lambda, \ell, s, t_{\mathsf{VC}}) \enspace .$$

[18]These standard definitions can be derived from Definitions 3.2 to 3.4 by setting pp to be empty.

Remark 3.9 (Monotonicity of ϵ_{VC})**.** We assume hereafter that the position binding error ϵ_{VC} is monotone in each coordinate in the natural direction:

- $\epsilon_{\mathsf{VC}}(\cdot, \ell, s, t_{\mathsf{VC}})$ is non-increasing (larger security parameters decrease an adversary's success);
- $\epsilon_{\mathsf{VC}}(\lambda, \cdot, s, t_{\mathsf{VC}})$ is non-decreasing (opening some set in a string is easier than opening in a substring);
- $\epsilon_{\mathsf{VC}}(\lambda, \ell, \cdot, t_{\mathsf{VC}})$ is non-decreasing (finding a collision in a set is easier than finding one in a subset); and
- $\epsilon_{\mathsf{VC}}(\lambda, \ell, s, \cdot)$ is non-decreasing (the success of an adversary increases with its computational power).

The last condition is trivially satisfied, while the first should also hold in any reasonable commitment scheme. The remaining two are natural (and satisfied in the case of Merkle trees); in any case, otherwise one may replace, in our computations, expressions of the type $\epsilon_{\mathsf{VC}}(\lambda, \ell_{\mathsf{max}}, s_{\mathsf{max}}, t_{\mathsf{VC}})$, when $\ell_{\mathsf{max}} = \max_i \{\ell_i\}$ and $s_{\mathsf{max}} = \max_j \{s_j\}$, with

$$\max_{i,j} \{\epsilon_{\mathsf{VC}}(\lambda, \ell_i, s_j, t_{\mathsf{VC}})\} \ .$$

3.3 Probabilistically checkable proofs

A *probabilistically checkable proof* (PCP) is an information-theoretic proof system where a probabilistic verifier has oracle access to a proof string.

Definition 3.10 (Completeness)**.** $\mathsf{PCP} = (\mathbf{P}, \mathbf{V})$ *for a relation R has* **perfect completeness** *if, for every instance-witness pair $(\mathbb{x}, \mathbb{w}) \in R$,*

$$\Pr\left[\mathbf{V}^{\Pi}(\mathbb{x}) = 1 \,\middle|\, \Pi \leftarrow \mathbf{P}(\mathbb{x}, \mathbb{w})\right] = 1 \ .$$

Definition 3.11 (Soundness)**.** $\mathsf{PCP} = (\mathbf{P}, \mathbf{V})$ *for a relation R has* **soundness error** ϵ_{PCP} *if, for every (unbounded) circuit $\widetilde{\mathbf{P}}$ and auxiliary input distribution $\mathbf{D}$,*

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathbf{V}^{\widetilde{\Pi}}(\mathbb{x}) = 1 \end{array} \,\middle|\, \begin{array}{l} \mathbf{ai} \leftarrow \mathbf{D} \\ (\mathbb{x}, \widetilde{\Pi}) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \end{array}\right] \le \epsilon_{\mathsf{PCP}}(n) \ .$$

Definition 3.12 (Knowledge soundness)**.** $\mathsf{PCP} = (\mathbf{P}, \mathbf{V})$ *for a relation R has* **knowledge soundness error** κ_{PCP} **with extraction time** $t_{\mathbf{E}}$ *if there exists a probabilistic algorithm $\mathbf{E}$ such that, for every circuit $\widetilde{\mathbf{P}}$ and auxiliary input distribution $\mathbf{D}$,*

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{\widetilde{\Pi}}(\mathbb{x}) = 1 \end{array} \,\middle|\, \begin{array}{l} \mathbf{ai} \leftarrow \mathbf{D} \\ (\mathbb{x}, \widetilde{\Pi}) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \\ \mathbb{w} \leftarrow \mathbf{E}(\mathbb{x}, \widetilde{\Pi}) \end{array}\right] \le \kappa_{\mathsf{PCP}}(n) \ ;$$

moreover, $\mathbf{E}$ runs in time $t_{\mathbf{E}}(n)$.

We consider several efficiency measures for a PCP:

- the *proof alphabet* Σ is the alphabet over which a PCP string is written;
- the *proof length* l is the number of alphabet symbols in the PCP string;
- the *query complexity* $\mathsf{q} \in [\mathsf{l}]$ is the number of queries that the PCP verifier makes to the PCP string (each query is an index in $[\mathsf{l}]$ and is answered by the corresponding symbol in Σ in the PCP string);
- the *randomness complexity* r is the number of random bits used by the PCP verifier.

An efficiency measure may be a function of the instance $\mathbb{x}$ (e.g., of its size $|\mathbb{x}|$).

3.4 Interactive oracle proofs

An *interactive oracle proof* (IOP) system [BCS16; RRR16] is a generalization of PCPs to multiple rounds. An IOP system for a relation R is formally defined as follows.

Definition 3.13 (Completeness). $\mathsf{IOP} = (\mathbf{P}, \mathbf{V})$ *for a relation R has* **perfect completeness** *if for every instance-witness pair $(\mathbb{x}, \mathbb{w}) \in R$,*

$$\Pr[\langle \mathbf{P}(\mathbb{x}, \mathbb{w}), \mathbf{V}(\mathbb{x})\rangle = 1] = 1 \ .$$

Definition 3.14 (Soundness). $\mathsf{IOP} = (\mathbf{P}, \mathbf{V})$ *for a relation R has* **soundness error** ϵ_{IOP} *if for every circuit $\widetilde{\mathbf{P}}$,*

$$\Pr\left[\begin{array}{l|l} |\mathbb{x}| \leq n & \mathbf{ai} \leftarrow \mathbf{D} \\ \wedge\, \mathbb{x} \notin L(R) & (\mathbb{x}, \mathbf{aux}) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \\ \wedge\, b = 1 & b \leftarrow \big\langle \widetilde{\mathbf{P}}(\mathbf{aux}), \mathbf{V}(\mathbb{x}) \big\rangle \end{array}\right] \leq \epsilon_{\mathsf{IOP}}(n) \ .$$

Definition 3.15 (Knowledge soundness). $\mathsf{IOP} = (\mathbf{P}, \mathbf{V})$ *for a relation R has* **knowledge soundness error** κ_{IOP} **with extraction time** $t_{\mathbf{E}}$ *if there exists a probabilistic algorithm $\mathbf{E}$ such that, for every circuit $\widetilde{\mathbf{P}}$,*

$$\Pr\left[\begin{array}{l|l} & \mathbf{ai} \leftarrow \mathbf{D} \\ |\mathbb{x}| \leq n & (\mathbb{x}, \mathbf{aux}) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R & b \xleftarrow{\mathbf{tr}} \big\langle \widetilde{\mathbf{P}}(\mathbf{aux}), \mathbf{V}(\mathbb{x}) \big\rangle \\ \wedge\, b = 1 & \mathbb{w} \leftarrow \mathbf{E}^{\widetilde{\mathbf{P}}(\mathbf{aux})}(\mathbb{x}, \mathbf{tr}) \end{array}\right] \leq \kappa_{\mathsf{IOP}}(n) \ ;$$

moreover, $\mathbf{E}$ runs in time $t_{\mathbf{E}}(n)$.

We consider several efficiency measures for an IOP:

- the *proof alphabet* Σ is the alphabet over which each IOP string is written;
- the *proof length* l is the total number of alphabet symbols across all IOP strings sent by the IOP prover; moreover, l_i is the length of the proof sent by $\mathbf{P}$ in round i and $\mathsf{l}_{\mathsf{max}} := \max_i \{\mathsf{l}_i\}$;
- the *query complexity* $\mathsf{q} \in [\mathsf{l}]$ is the total number of queries that the IOP verifier makes to any IOP string (each query specifies a round a location in the IOP string of that round and is answered by the corresponding symbol in the IOP string);
- the *randomness complexity* r is the number of random bits used by the IOP verifier;
- the *round complexity* k is the number of interaction rounds (back-and-forth interactions) between the IOP prover and IOP verifier.

Any efficiency measure may be a function of the instance $\mathbb{x}$ (e.g., of the instance size $|\mathbb{x}|$).

Public-coin IOPs. We focus on IOPs that are *public-coin*, which informally means that the IOP verifier is a public-coin interactive algorithm.

In other words, in every round, the verifier sends a uniformly random message, independent of other messages they send, to the prover. We provide the formal definition below.

Definition 3.16. $\mathsf{IOP} = (\mathbf{P}, \mathbf{V})$ *for a relation R is* **public-coin** *if, for every $i \in [\mathsf{k}]$, the i-th message of the IOP verifier $\mathbf{V}$ is a freshly-sampled uniform random string ρ_i of a prescribed length r_i (which may depend on the instance). In particular, during an interaction, the IOP prover knows all the randomness sampled by the IOP verifier so far (as the messages received by the IOP prover are all that randomness so far).*

Queries in a public-coin IOP can be postponed until after the interaction. (This is without loss of generality as, during the interaction, the IOP verifier simply sends a fresh random message in each round.) Hence the IOP can be viewed in two parts: an interaction phase (the prover sends oracles and the verifier sends random messages) and then a query phase (the verifier queries the oracles and outputs a decision). We denote by $b = \mathbf{V}^{(\Pi_i)_{i\in[\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]})$ *the IOP verifier's decision in the query phase, where* k *is the IOP's round complexity; then the two-step experiment* $(\mathbb{x}, \mathbf{aux}) \leftarrow \widetilde{\mathbf{P}}$ *followed by* $b \leftarrow \langle \widetilde{\mathbf{P}}(\mathbf{aux}), \mathbf{V}(\mathbb{x}) \rangle$ *can be written as*

$$\left[\begin{array}{l} \mathbf{ai} \leftarrow \mathbf{D} \\ (\mathbb{x}, \mathbf{aux}_0) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \\ (\widetilde{\Pi}_1, \mathbf{aux}_1) \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux}_0) \\ \rho_1 \leftarrow \{0,1\}^{\mathsf{r}_1} \\ \text{For } i \in [\mathsf{k}-1] \setminus \{1\} : \\ \quad (\widetilde{\Pi}_i, \mathbf{aux}_i) \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux}_{i-1}, \rho_{i-1}) \\ \quad \rho_i \leftarrow \{0,1\}^{\mathsf{r}_i} \\ \widetilde{\Pi}_{\mathsf{k}} \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux}_{\mathsf{k}-1}, \rho_{\mathsf{k}-1}) \\ b := \mathbf{V}^{(\Pi_i)_{i\in[\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}) \end{array}\right] = \left[\begin{array}{l} \mathbf{ai} \leftarrow \mathbf{D} \\ (\mathbb{x}, \mathbf{aux}_0) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \\ \rho_0 := \bot \\ \text{For } i \in [\mathsf{k}] : \\ \quad (\widetilde{\Pi}_i, \mathbf{aux}_i) \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux}_{i-1}, \rho_{i-1}) \\ \quad \rho_i \leftarrow \{0,1\}^{\mathsf{r}_i} \\ b := \mathbf{V}^{(\Pi_i)_{i\in[\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}) \end{array}\right] . \tag{1}$$

Remark 3.17. We often make use of simplifications as in Experiment 1, where the right-hand side sets ρ_0 to the empty string $\bot$. (And does so, implicitly, for $\mathbf{aux}_{\mathsf{k}}$ as well.)

We shall additionally make use of notation for *partial* (or full) executions of an interactive algorithm: for example, an execution of $\widetilde{\mathbf{P}}$ until its second message is denoted $(\mathbb{x}, \widetilde{\Pi}_1, (\widetilde{\Pi}_2, \mathbf{aux}_2)) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}, \rho_1)$ (we omit auxiliary outputs that are not used in the remainder of the experiment). Therefore, Experiment 1 is also equivalent to

$$\left[\begin{array}{l} \mathbf{ai} \leftarrow \mathbf{D} \\ (\rho_i)_{i\in[\mathsf{k}]} \leftarrow \{0,1\}^{\mathsf{r}_1+\cdots+\mathsf{r}_{\mathsf{k}}} \\ (\mathbb{x}, \widetilde{\Pi}_1, \ldots, \widetilde{\Pi}_{\mathsf{k}}) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}, \rho_1, \ldots, \rho_{\mathsf{k}-1}) \\ b := \mathbf{V}^{(\Pi_i)_{i\in[\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}) \end{array}\right] .$$

Public-query IOPs. In Section 6 we discuss IOPs that are *public-query*, which informally means that security (soundness or knowledge soundness) holds even if the IOP prover can "see" the queries that the IOP verifier makes. In contrast, the general definition of an IOP implicitly assumes that an (honest or malicious) IOP prover has no information about the queries that the IOP verifier makes during the interaction.

Definition 3.18. $\mathsf{IOP} = (\mathbf{P}, \mathbf{V})$ *for a relation* R *is* **public-query** *if the soundness (and knowledge soundness) condition holds even if the malicious IOP prover* $\widetilde{\mathbf{P}}$ *receives, during the interaction, the index of each location queried by the IOP verifier* $\mathbf{V}$ *(the moment it happens). In particular, for each* $i \in [\mathsf{k}]$*, the IOP string* $\widetilde{\Pi}_i$ *sent by* $\widetilde{\mathbf{P}}$ *in the* i*-th round may depend on every query made by the IOP verifier in prior rounds (to prior IOP strings), in addition to depending on messages sent by the IOP verifier so far.*

Note that a public-coin IOP is a public-query IOP, but the converse need not hold.

4 Interactive arguments based on PCPs

Theorem 4.1. *Consider these two ingredients:*

- $\mathsf{PCP} = (\mathbf{P}, \mathbf{V})$*, a PCP system for a relation* R *with alphabet* Σ*, proof length* l*, and query complexity* q*; and*
- $\mathsf{VC} = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Check})$*, a vector commitment scheme over alphabet* Σ*.*

Then $\mathsf{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V}) := \mathsf{Kilian}[\mathsf{PCP}, \mathsf{VC}]$ *(Construction 4.3) is a three-message public-coin interactive argument system for* R*, whose soundness error* ϵ_{ARG} *and knowledge soundness error* κ_{ARG} *satisfy the following for every* $\epsilon > 0$ *and* $t_{\mathsf{ARG}} \geq t_{\mathbf{V}} + t_{\mathsf{VC.Check}} + \log|\Sigma| + \log \mathsf{l}$*:*

$$\begin{aligned}\epsilon_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) &\leq \epsilon_{\mathsf{PCP}}(n) + \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) + \epsilon \quad \textit{and} \\ \kappa_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) &\leq \kappa_{\mathsf{PCP}}(n) + \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) + \epsilon \ \ .\end{aligned}$$

Above, ϵ_{PCP} *and* κ_{PCP} *are the soundness and knowledge soundness errors of* PCP*, and* $t_{\mathsf{VC}} = O\left(\frac{\mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}\right)$*. Moreover, the knowledge extractor runs in time* $t_{\mathcal{E}} = O(t_{\mathbf{E}} + t_{\mathsf{VC}})$*.*

Corollary 4.2. *Let* ARG *be as in Theorem 4.1. Assume that for any* $n \in \mathbb{N}$*,* $\epsilon_{\mathsf{VC}}(\cdot, \cdot, \cdot, t_{\mathsf{VC}}) = \mathrm{negl}(n)$ *if* $t_{\mathsf{VC}} = \mathsf{poly}(n)$*. Then, given that* $t_{\mathsf{ARG}} = \mathsf{poly}(n)$*, we have*

$$\begin{aligned}\epsilon_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) &\leq \epsilon_{\mathsf{PCP}}(n) + \mathrm{negl}(n) \quad \textit{and} \\ \kappa_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) &\leq \kappa_{\mathsf{PCP}}(n) + \mathrm{negl}(n) \ \ .\end{aligned}$$

Proof. Let $p(n)$ be an arbitrary polynomial. We set ϵ to be $\frac{1}{2p(n)} > 0$. Hence, $t_{\mathsf{VC}} = O\left(\frac{\mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}\right) = \mathsf{poly}(n)$, which implies that $\epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) = \mathrm{negl}(n)$. Therefore,

$$\epsilon_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) \leq \epsilon_{\mathsf{PCP}}(n) + \mathrm{negl}(n) + \frac{1}{2p(n)} < \epsilon_{\mathsf{PCP}}(n) + \mathrm{negl}(n) + \frac{1}{p(n)} \ \ .$$

Since p is an arbitrary polynomial, we conclude that

$$\epsilon_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) \leq \epsilon_{\mathsf{PCP}}(n) + \mathrm{negl}(n) \ \ .$$

An analogous argument holds for κ_{ARG}. $\square$

4.1 Construction

The construction of $(\mathcal{G}, \mathcal{P}, \mathcal{V}) := \mathsf{Kilian}[\mathsf{PCP}, \mathsf{VC}]$ is specified below.

Construction 4.3. The argument generator $\mathcal{G}$ receives as input a security parameter $\lambda \in \mathbb{N}$ and an instance size bound $n \in \mathbb{N}$, and works as follows.

$\mathcal{G}(\lambda, n)$:
1. Sample public parameter for the VC scheme: $\mathsf{pp}_{\mathsf{VC}} \leftarrow \mathsf{VC.Gen}(1^{\lambda}, \mathsf{l}(n))$.
2. Set public parameter for the interactive argument: $\mathsf{pp} := \mathsf{pp}_{\mathsf{VC}}$.
3. Output pp.

The argument prover $\mathcal{P}$ receives as input the public parameter pp, an instance $\mathbb{x}$ and a witness $\mathbb{w}$, and the argument verifier $\mathcal{V}$ receives as input the public parameter pp and the instance $\mathbb{x}$. Then $\mathcal{P}$ and $\mathcal{V}$ interact as follows.

1. $\mathcal{P}$'s commitment.
    (a) Compute a PCP string: $\Pi \leftarrow \mathbf{P}(\mathbb{x}, \mathbb{w})$.
    (b) Compute a vector commitment to the PCP string: $(\mathsf{cm}, \mathsf{aux}) \leftarrow \mathsf{VC.Commit}(\mathsf{pp}, \Pi)$.
    (c) Send cm to $\mathcal{V}$.
2. $\mathcal{V}$'s challenge.
    (a) Sample PCP verifier randomness: $\rho \leftarrow \{0,1\}^{\mathsf{r}}$.
    (b) Send ρ to $\mathcal{P}$.
3. $\mathcal{P}$'s response.
    (a) Run the PCP verifier $\mathbf{V}^{\Pi}(\mathbb{x}; \rho)$ to deduce its query set $\mathcal{Q} \subseteq [\mathsf{l}]$.
    (b) Compute a VC opening proof: $\mathsf{pf} \leftarrow \mathsf{VC.Open}(\mathsf{pp}, \mathsf{aux}, \mathcal{Q})$.
    (c) Set $\mathsf{ans} := \Pi[\mathcal{Q}]$.
    (d) Send $(\mathcal{Q}, \mathsf{ans}, \mathsf{pf})$ to $\mathcal{V}$.
4. $\mathcal{V}$'s decision: check that $\mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x}; \rho) = 1$ and $\mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1$.

The interactive argument consists of three messages: a prover message; a verifier message; and a prover message. The interactive argument is public-coin since the verifier's (only) message is a uniform random string. The efficiency measures of interactive arguments are as follows:

- the generator outputs public parameter of size $|\mathsf{pp}_{\mathsf{VC}}|$ bits;
- the prover-to-verifier communication consists of $|\mathsf{cm}| + \mathsf{q} \cdot (\log \mathsf{l} + \log|\Sigma|) + |\mathsf{pf}|$ bits;
- the verifier-to-prover communication consists of r bits;
- the time complexity of the argument generator is $t_{\mathsf{VC.Gen}}$.
- the time complexity of the argument prover is $t_{\mathbf{P}} + t_{\mathsf{VC.Commit}} + t_{\mathbf{V}} + t_{\mathsf{VC.Open}}$;
- the time complexity of the argument verifier is $t_{\mathbf{V}} + t_{\mathsf{VC.Check}}$.

Remark 4.4. There are vector commitments for which $\mathsf{VC.Gen}$ needs only the security parameter λ as input (i.e., $\mathsf{VC.Gen}$ works for every message size); for example, vector commitments based on Merkle trees have this property, because the public parameter consist of (the description of) a hash function, which suffices for every message size. In this case, the argument generator $\mathcal{G}$ in Construction 4.3 also requires only λ as input and works for every instance size. This leads the notion of an interactive argument discussed in Remark 3.5.

Remark 4.5. In the plain-model variant of Construction 4.3 (see Remark 3.6), the public parameters $\mathsf{pp} := \mathsf{pp}_{\mathsf{VC}}$ are sampled and sent by the argument verifier (resulting in a four-message protocol). Hence the plain-model variant is public-coin if (and only if) $\mathsf{VC.Gen}$ is a public-coin algorithm (its output includes all of its randomness).

4.2 Security reduction

To analyze the soundness and knowledge soundness for the argument system of Construction 4.3, it is important to understand how the argument system is related to the PCP system. The core of the security analysis is the construction of a PCP prover $\widetilde{\mathbf{P}}$ from an argument prover $\widetilde{\mathcal{P}}$ (which may or may not be malicious). More precisely, given a convincing argument prover $\widetilde{\mathcal{P}}$, we want to obtain a convincing PCP prover $\widetilde{\mathbf{P}}$, which we achieve via the *reductor* algorithm $\mathcal{R}$ in Construction 4.8.

Recall that if $\mathcal{V}$ accepts if and only if both $\mathbf{V}$ and $\mathsf{VC.Check}$ accept. Hence, Lemma 4.6 shows that PCP strings generated by the reductor $\mathcal{R}$ are, up to small errors, as convincing to the PCP verifier $\mathbf{V}$ as the argument prover $\mathcal{P}$ is to the argument verifier $\mathcal{V}$; in other words, $\mathcal{R}$ transforms an argument prover $\widetilde{\mathcal{P}}$ into a

PCP prover $\widetilde{\mathbf{P}}$.[19] We later use this lemma to prove (adaptive) soundness and knowledge soundness of ARG in Sections 4.3 and 4.4, respectively.

Lemma 4.6. *There exists a probabilistic algorithm $\mathcal{R}$ which, for every $\epsilon > 0$, auxiliary input distribution $\mathcal{D}$, size bound $t_{\mathsf{ARG}} \geq t_{\mathbf{V}} + t_{\mathsf{VC.Check}} + 2\mathsf{q} \cdot (\log|\Sigma| + \log \mathsf{l})$, and t_{ARG}-size circuit $\widetilde{\mathcal{P}}$, satisfies*

$$\Pr\left[\begin{array}{l|l} & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x}; \rho) \neq 1 & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x}; \rho) = 1 & \big(\mathbb{x}, (\mathsf{cm}, \mathsf{aux})\big) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 & (\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux},\cdot)}(\mathsf{pp}, \mathsf{cm}, \epsilon) \\ & \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ & (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho) \end{array}\right]$$
$$\leq \epsilon_{\mathsf{VC}}\left(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}\right) + \epsilon \enspace ,$$

where $t_{\mathsf{VC}} = O\left(\frac{\mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}\right)$. Moreover, $\mathcal{R}$ makes l/ϵ queries to $\widetilde{\mathcal{P}}$ and runs in $O(t_{\mathsf{VC}})$ time.

We stress that in the experiment above the reductor $\mathcal{R}$ is *independent* of ρ, since it does not receive the verifier randomness as input. (Otherwise, the lemma would be satisfied trivially with $\widetilde{\mathcal{Q}} := \mathcal{Q}$ and $\widetilde{\Pi}[\widetilde{\mathcal{Q}}] := \mathsf{ans}$.)

We now construct $\mathcal{R}$, which will be convenient to separate into two parts: a sampling subroutine $\mathcal{S}$ followed by a post-processing layer $\mathcal{R}_{\text{post}}$ that deterministically pieces together a PCP string $\widetilde{\Pi}$ from the samples obtained by $\mathcal{S}$ (and outputs the set $\widetilde{\mathcal{Q}}$ of "filled-in" coordinates along with $\widetilde{\Pi}$).

Construction 4.7. We construct the *sampler* $\mathcal{S}$ as follows.

```
$\mathcal{S}^{\widetilde{\mathcal{P}}(\mathsf{aux},\cdot)}(\mathsf{pp}, \mathsf{cm}, \mathsf{N})$:
1. Initialize $\mathcal{K} := \emptyset$.
2. Repeat the following $\mathsf{N}$ times:
   (a) Sample PCP verifier randomness: $\rho' \leftarrow \{0,1\}^{\mathsf{r}}$.
   (b) Obtain $(\mathcal{Q}', \mathsf{ans}', \mathsf{pf}') \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho')$.
   (c) If $\mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}', \mathsf{ans}', \mathsf{pf}') = 1$, add $(\mathcal{Q}', \mathsf{ans}', \mathsf{pf}')$ to $\mathcal{K}$.
3. Output $\mathcal{K}$.
```

The algorithm $\mathcal{S}$ makes N queries to $\widetilde{\mathcal{P}}$, and runs in time[20]

$$t_{\mathcal{S}} \leq \mathsf{N} \cdot (t_{\mathsf{ARG}} + t_{\mathbf{V}} + t_{\mathsf{VC.Check}}) \leq 3\mathsf{N} \cdot t_{\mathsf{ARG}} \enspace .$$

Construction 4.8. The *reductor* $\mathcal{R}$ is defined as follows. (Below, σ is an arbitrary symbol in the alphabet Σ.)

```
$\mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux},\cdot)}(\mathsf{pp}, \mathsf{cm}, \epsilon)$:[21]
1. Set $\mathsf{N} := \frac{\mathsf{l}}{\epsilon}$ and run $\mathcal{K} \leftarrow \mathcal{S}^{\widetilde{\mathcal{P}}(\mathsf{aux},\cdot)}(\mathsf{pp}, \mathsf{cm}, \mathsf{N})$.
2. Run $(\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}_{\text{post}}(\mathcal{K}, \mathsf{l})$.
```

[19]Moreover, $\mathcal{R}$ preserves uniformity: if $\widetilde{\mathcal{P}}$ is a uniform algorithm, then so is $\widetilde{\mathbf{P}}$.

[20]Note that the $2\mathsf{q}(\log|\Sigma| + \log \mathsf{l})$ overhead incurred by copying $(\mathcal{Q}', \mathsf{ans}', \mathsf{pf}')$ into $\mathcal{K}$ is accounted for in the difference between t_{ARG} and the other terms.

[21]We also denote by $\mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux},\cdot)}\big(\mathsf{pp}, \mathsf{cm}, \epsilon; \boldsymbol{\rho}\big)$, where $\boldsymbol{\rho} = (\rho^{(\ell)})_{\ell \in [\mathsf{N}]}$ (and similarly for $\mathcal{S}$) the deterministic algorithm that uses $\rho^{(\ell)}$ as the randomness for $\mathcal{S}$'s ℓ-th sample. This allows the PCP and IOP provers of Constructions 4.10 and 5.8 to be deterministic.

```
3. Output $(\widetilde{\mathcal{Q}}, \widetilde{\Pi})$.
```

The reductor's second step is an execution of the following (deterministic) *post-processing* algorithm.

```
$\mathcal{R}_{\mathrm{post}}(\mathcal{K}, \mathsf{l})$:
1. Initialize $\widetilde{\Pi} := \sigma^{\mathsf{l}}$ and $\widetilde{\mathcal{Q}} := \emptyset$.
2. For every $(\mathcal{Q}', \mathsf{ans}', \mathsf{pf}') \in \mathcal{K}$:
   (a) Set $\widetilde{\mathcal{Q}} := \widetilde{\mathcal{Q}} \cup \mathcal{Q}'$.
   (b) For every $q \in \mathcal{Q}'$, set $\widetilde{\Pi}[q] := \mathsf{ans}'[q]$.
3. Output $(\widetilde{\mathcal{Q}}, \widetilde{\Pi})$.
```

Note that $\mathcal{R}$ makes $\mathsf{N} = \mathsf{l}/\epsilon$ queries to $\widetilde{\mathcal{P}}$ by construction, whose total $\mathsf{N} \cdot t_{\mathsf{ARG}}$ time dominates that of $\mathcal{R}$.

Proof. Throughout this proof, probabilistic expressions are with respect to the following experiment unless explicitly denoted otherwise:

$$\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ (\mathsf{cm}, \mathsf{aux}_1) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_0) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_1, \rho) \end{array}\right] = \left[\begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ \big(\mathbb{x}, (\mathsf{cm}, \mathsf{aux})\big) \leftarrow \widetilde{\mathcal{P}}\big(\mathsf{pp}, \mathsf{ai}\big) \\ (\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}, \cdot)}(\mathsf{pp}, \mathsf{cm}, \epsilon) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}\big(\mathsf{aux}, \rho\big) \end{array}\right] . \tag{2}$$

Our goal is to upper bound the probability of the following expression:

$$\left[\begin{array}{l} \mathbf{V}^{[\widetilde{\mathcal{Q}}, \widetilde{\Pi}]}(\mathbb{x}; \rho) \neq 1 \\ \wedge \mathbf{V}^{[\mathcal{Q}, \mathsf{ans}]}(\mathbb{x}; \rho) = 1 \\ \wedge \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 \end{array}\right] . \tag{3}$$

Observe that Eq. 3 implies that either (i) $\widetilde{\Pi}$ and ans disagree at a position $q \in \mathcal{Q} \cap \widetilde{\mathcal{Q}}$; or (ii) there is a query q in $\mathcal{Q} \setminus \widetilde{\mathcal{Q}}$. We analyze the two cases separately.

Valid openings with disagreeing answers. Our goal is to prove the following bound:

$$\Pr\left[\begin{array}{l} \exists\, q \in \mathcal{Q} \cap \widetilde{\mathcal{Q}} : \mathsf{ans}[q] \neq \widetilde{\Pi}[q] \\ \wedge \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 \end{array}\right] \leq \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) ,$$

where $t_{\mathsf{VC}} \leq 4\mathsf{N} \cdot t_{\mathsf{ARG}}$.

Consider the following adversary A_{VC} against the vector commitment scheme, which follows Experiment 2 (without executing $\mathcal{R}_{\mathrm{post}}$) and attempts to find a collision using the output $\mathcal{K}$ of the sampler $\mathcal{S}$.

```
$A_{\mathsf{VC}}(\mathsf{pp}, \mathsf{ai})$:
1. Run $\big(\mathbb{x}, (\mathsf{cm}, \mathsf{aux})\big) \leftarrow \widetilde{\mathcal{P}}\big(\mathsf{pp}, \mathsf{ai}\big)$.
2. Sample $\rho \leftarrow \{0,1\}^{\mathsf{r}}$.
3. Run $(\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho)$.
4. Run $\mathcal{K} \leftarrow \mathcal{S}^{\widetilde{\mathcal{P}}(\mathsf{aux}, \cdot)}(\mathsf{pp}, \mathsf{cm}, \mathsf{N})$, with $\mathsf{N} = \frac{\mathsf{l}}{\epsilon}$ (as in Construction 4.8).
5. If there are $(\mathcal{Q}', \mathsf{ans}', \mathsf{pf}') \in \mathcal{K}$ and $q \in \mathcal{Q}' \cap \mathcal{Q}$ with $\mathsf{ans}'[q] \neq \mathsf{ans}[q]$, output $(\mathsf{cm}, \mathsf{ans}, \mathsf{ans}', \mathcal{Q}, \mathcal{Q}', \mathsf{pf}, \mathsf{pf}')$.
6. Otherwise, output (the "dummy" tuple) $(\mathsf{cm}, \mathsf{ans}, \mathsf{ans}, \mathcal{Q}, \mathcal{Q}, \mathsf{pf}, \mathsf{pf})$.
```

The time complexity of the sampler $\mathcal{S}$ is at most $3\mathsf{N} \cdot t_{\mathsf{ARG}}$ and the collision-finding check (Step 5) runs in time $\mathsf{N} \cdot 2\mathsf{q}(\log|\Sigma| + \log \mathsf{l}) \leq \mathsf{N} \cdot t_{\mathsf{ARG}}$,[22] so the time complexity of A_{VC} is $t_{\mathsf{VC}} \leq 4\mathsf{N} \cdot t_{\mathsf{ARG}}$.

Therefore, according to Definition 3.8,

$$\begin{aligned}
&\Pr\left[\begin{array}{l} \exists\, q \in \mathcal{Q} \cap \widetilde{\mathcal{Q}} : \mathsf{ans}[q] \neq \widetilde{\Pi}[q] \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 \end{array}\right] \\
&= \Pr\left[\begin{array}{l|l} |\mathcal{Q}| = |\mathcal{Q}'| = \mathsf{q} & \mathsf{pp} \leftarrow \mathsf{VC.Gen}(1^\lambda, \mathsf{l}) \\ \wedge\, \exists\, q \in \mathcal{Q} \cap \mathcal{Q}' : \mathsf{ans}[q] \neq \mathsf{ans}'[q] & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 & \begin{pmatrix} \mathsf{cm}, \mathsf{ans}, \mathsf{ans}', \\ \mathcal{Q}, \mathcal{Q}', \mathsf{pf}, \mathsf{pf}' \end{pmatrix} \leftarrow A_{\mathsf{VC}}(\mathsf{pp}, \mathsf{ai}) \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}', \mathsf{ans}', \mathsf{pf}') = 1 & \end{array}\right] \\
&\leq \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) \ .
\end{aligned}$$

Missing positions in $\widetilde{\Pi}$. We now upper bound the probability that there is a missing position in $\widetilde{\Pi}$; we claim that

$$\Pr\left[\begin{array}{l} \mathcal{Q} \setminus \widetilde{\mathcal{Q}} \neq \emptyset \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 \end{array}\right] \leq \epsilon \ .$$

Fix a public parameter-auxiliary input pair $(\mathsf{pp}, \mathsf{ai})$ (recall that these are obtained in the first steps of Experiment 2), and define the *weight* of a coordinate $q \in [\mathsf{l}]$ with respect to $(\mathsf{pp}, \mathsf{ai})$ as

$$\delta_{\mathsf{pp},\mathsf{ai}}(q) := \Pr\left[\begin{array}{l|l} q \in \mathcal{Q} & (\mathbb{x}, \mathsf{cm}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 & \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ & (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho) \end{array}\right] .$$

Then, by a union bound over $q \in [\mathsf{l}]$,

$$\begin{aligned}
&\Pr\left[\begin{array}{l} \mathcal{Q} \setminus \widetilde{\mathcal{Q}} \neq \emptyset \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 \end{array}\right] \\
&= \Pr\left[\begin{array}{l} \exists\, q \in [\mathsf{l}] : q \in \mathcal{Q} \,\wedge\, q \notin \widetilde{\mathcal{Q}} \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 \end{array}\right] \\
&\leq \max_{(\mathsf{pp},\mathsf{ai})} \left\{ \Pr\left[\begin{array}{l|l} \exists\, q \in [\mathsf{l}] : q \in \mathcal{Q} \,\wedge\, q \notin \widetilde{\mathcal{Q}} & (\mathbb{x}, \mathsf{cm}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 & \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ & (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho) \end{array}\right] \right\} \\
&= \max_{(\mathsf{pp},\mathsf{ai})} \left\{ \sum_{q \in [\mathsf{l}]} \delta_{\mathsf{pp},\mathsf{ai}}(q) \cdot \left(1 - \delta_{\mathsf{pp},\mathsf{ai}}(q)\right)^{\mathsf{N}} \right\} \\
&\leq \frac{\mathsf{l}}{\mathsf{N}} \ ,
\end{aligned}$$

where the last inequality follows from $\delta \cdot (1-\delta)^{\mathsf{N}} \leq 1/\mathsf{N}$ for any $\delta \in [0,1]$.[23] Finally, plugging in $\mathsf{N} = \frac{\mathsf{l}}{\epsilon}$ bounds Eq. 3 as desired, concluding the proof:

$$\Pr\left[\begin{array}{l} \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho) \neq 1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 \end{array}\right]$$

[22] Searching for an intersection between $\mathcal{Q}'$ and $\mathcal{Q}$ takes $2\mathsf{q}\log\mathsf{l}$ time, while each check for a symbol mismatch takes $2\log|\Sigma|$. We omit the time required to produce the output (which can be accounted for in the runtime of $\mathcal{S}$).

[23] A simple derivation of the inequality is the following: with $f(x) = x \cdot (1-x)^{\mathsf{N}}$, we have $\frac{d}{dx} f(\delta) = 0 \iff \delta = \frac{1}{\mathsf{N}+1}$. As $f(0) = f(1) = 0$ and δ is the only critical point in $[0,1]$, it achieves the maximum $f(\delta) \leq 1/\mathsf{N}$.

$$\leq \Pr\left[\begin{array}{l}\exists\, q \in \mathcal{Q} \cap \widetilde{\mathcal{Q}} : \mathsf{ans}[q] \neq \widetilde{\Pi}[q] \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1\end{array}\right] + \Pr\left[\begin{array}{l}\mathcal{Q} \setminus \widetilde{\mathcal{Q}} \neq \emptyset \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1\end{array}\right]$$

$$\leq \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) + \epsilon \ . \qquad \square$$

4.3 Adaptive soundness

Lemma 4.9. *For every $\epsilon > 0$, security parameter $\lambda \in \mathbb{N}$, instance size bound $n \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}$, circuit size bound $t_{\mathsf{ARG}} \geq t_{\mathbf{V}} + t_{\mathsf{VC.Check}} + 2\mathsf{q} \cdot (\log|\Sigma| + \log \mathsf{l})$ and t_{ARG}-size circuit $\widetilde{\mathcal{P}}$, the soundness error of the argument system in Construction 4.3 satisfies*

$$\epsilon_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) \leq \epsilon_{\mathsf{PCP}}(n) + \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) + \epsilon \ ,$$

where $t_{\mathsf{VC}} = O\left(\frac{1}{\epsilon} \cdot t_{\mathsf{ARG}}\right)$.

Proof. Recall, from Definition 3.3 and Construction 4.3, that our goal is to upper bound

$$\Pr\left[\begin{array}{l|l}|\mathbb{x}| \leq n & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \wedge\, \mathbb{x} \notin L(R) & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, b = 1 & (\mathbb{x}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ & b \leftarrow \langle \widetilde{\mathcal{P}}(\mathsf{aux}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \rangle\end{array}\right]$$

$$= \Pr\left[\begin{array}{l|l}|\mathbb{x}| \leq n & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \wedge\, \mathbb{x} \notin L(R) & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, \mathbf{V}^{[\mathcal{Q}, \mathsf{ans}]}(\mathbb{x}; \rho) = 1 & \big(\mathbb{x}, (\mathsf{cm}, \mathsf{aux})\big) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 & \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ & (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho)\end{array}\right] .$$

(Since $\widetilde{\mathcal{P}}$ sends cm as the first message in Construction 4.3, the former experiment is equivalent to the latter, where we omit the auxiliary state of the choice of instance and aux denotes that of the first message.)

As in Lemma 4.6, we consider the following experiment (a restatement of Experiment 2), which augments the above by executing $\mathcal{R}$, and thus leaves the probability unchanged.

$$\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ \big(\mathbb{x}, (\mathsf{cm}, \mathsf{aux})\big) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ (\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}, \cdot)}(\mathsf{pp}, \mathsf{cm}, \epsilon) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho)\end{array}\right] .$$

By total probability,

$$\Pr\left[\begin{array}{l}|\mathbb{x}| \leq n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathbf{V}^{[\mathcal{Q}, \mathsf{ans}]}(\mathbb{x}; \rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1\end{array}\right]$$

$$= \Pr\left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho) = 1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp},\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf}) = 1 \end{array}\right] + \Pr\left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho) \neq 1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp},\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf}) = 1 \end{array}\right] .$$

We first bound the probability of the leftmost term by the PCP system's soundness error (i.e., Definition 3.11 with respect to PCP).

Construction 4.10. We define the auxiliary input distribution $\mathbf{D}$ of the PCP prover $\widetilde{\mathbf{P}}$ as follows:

$\mathbf{D}$:

1. Sample $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n)$ followed by $\mathsf{ai} \leftarrow \mathcal{D}$ and $\boldsymbol{\rho} := (\rho^{(\ell)})_{\ell\in[\mathsf{N}]} \leftarrow (\{0,1\}^{\mathsf{r}})^{\mathsf{N}}$.
2. Output $\mathbf{ai} := (\mathsf{pp}, \mathsf{ai}, \boldsymbol{\rho})$.

The PCP prover is then given by the following next message functions.

$\widetilde{\mathbf{P}}(\mathbf{ai})$:

1. Parse $\mathbf{ai}$ as $(\mathsf{pp}, \mathsf{ai}, \boldsymbol{\rho})$.
2. Run $(\mathbb{x}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai})$.
3. Set $\mathbf{aux} := (\mathsf{pp}, \mathsf{aux}_0, \boldsymbol{\rho})$.
4. Output $(\mathbb{x}, \mathbf{aux})$.

$\widetilde{\mathbf{P}}(\mathbf{aux})$:

1. Parse $\mathbf{aux}$ as $(\mathsf{pp}, \mathsf{aux}_0, \boldsymbol{\rho})$.
2. Run $(\mathsf{cm}, \mathsf{aux}_1) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_0)$.
3. Run $(\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_1,\cdot)}(\mathsf{pp}, \mathsf{cm}, \epsilon; \boldsymbol{\rho})$.
4. Output $\widetilde{\Pi}$.

Using Definition 3.11,[24]

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho) = 1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp},\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf}) = 1 \end{array}\right] \leq \Pr\left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho) = 1 \end{array}\right]$$

$$\leq \Pr\left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathbf{V}^{\widetilde{\Pi}}(\mathbb{x}) = 1 \end{array}\middle|\begin{array}{l} \mathbf{ai} \leftarrow \mathbf{D} \\ (\mathbb{x}, \mathbf{aux}) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \\ \widetilde{\Pi} \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux}) \end{array}\right]$$

$$\leq \epsilon_{\mathsf{PCP}}(n) \ .$$

Lastly, an application of Lemma 4.6 yields

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho) \neq 1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp},\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf}) = 1 \end{array}\right] \leq \Pr\left[\begin{array}{l} \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x};\rho) \neq 1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp},\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf}) = 1 \end{array}\right]$$

[24]Note that the prover in Definition 3.11 corresponds to the sequential execution of both steps in Construction 4.10.

$$\leq \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) + \epsilon \ ,$$

where $t_{\mathsf{VC}} \leq \frac{4\mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}$, which concludes the proof. □

4.4 Adaptive knowledge soundness

Lemma 4.11. *For every $\epsilon > 0$, security parameter $\lambda \in \mathbb{N}$, instance size bound $n \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}$, circuit size bound $t_{\mathsf{ARG}} \geq t_{\mathbf{V}} + t_{\mathsf{VC.Check}} + 2\mathsf{q} \cdot (\log|\Sigma| + \log \mathsf{l})$ and t_{ARG}-size circuit $\widetilde{\mathcal{P}}$, the knowledge soundness error of the argument system obtained by Construction 4.3 satisfies*

$$\kappa_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) \leq \kappa_{\mathsf{PCP}}(n) + \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) + \epsilon \ ,$$

where $t_{\mathsf{VC}} = O\left(\frac{\mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}\right)$. If the PCP extractor's runtime is $t_{\mathbf{E}}$, the argument system's extractor is $t_{\mathcal{E}} = t_{\mathbf{E}} + O(t_{\mathsf{VC}})$.

Construction 4.12. Let $\mathbf{E}$ be the extractor for PCP. We use $\widetilde{\mathbf{P}}$ (Construction 4.10) and $\mathbf{E}$ to construct the knowledge extractor $\mathcal{E}$ for ARG as follows.

```
$\mathcal{E}^{\widetilde{\mathcal{P}}(\mathsf{aux})}(\mathsf{pp}, \mathbb{x}, \mathsf{tr})$:
1. Sample $\boldsymbol{\rho} \leftarrow (\{0,1\}^{\mathsf{r}})^{\mathsf{N}}$.
2. Set $\mathbf{aux} := (\mathsf{pp}, \mathsf{aux}, \boldsymbol{\rho})$ and run $\widetilde{\Pi} \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux})$.[25]
3. Run $\mathbb{w} \leftarrow \mathbf{E}(\mathbb{x}, \widetilde{\Pi})$.
4. Output $\mathbb{w}$.
```

Note that $\mathcal{E}$ executes $\mathbf{E}$ once and $\widetilde{\mathcal{P}}$ for l/ϵ times (as $\widetilde{\mathbf{P}}$ runs the reductor $\mathcal{R}$, which in turn sets $\mathsf{N} = \mathsf{l}/\epsilon$ and repeats N executions of $\widetilde{\mathcal{P}}$). Therefore, $t_{\mathcal{E}} = t_{\mathbf{E}} + O(t_{\mathsf{VC}})$.

Proof. From Definition 3.4 and Construction 4.3, our goal is to upper bound

$$\Pr\left[\begin{array}{l|l} |\mathbb{x}| \leq n & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, b = 1 & (\mathbb{x}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ & b \xleftarrow{\mathsf{tr}} \langle \widetilde{\mathcal{P}}(\mathsf{aux}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \rangle \\ & \mathbb{w} \leftarrow \mathcal{E}^{\widetilde{\mathcal{P}}(\mathsf{aux})}(\mathsf{pp}, \mathbb{x}, \mathsf{tr}) \end{array}\right]$$

$$= \Pr\left[\begin{array}{l|l} & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ & \mathsf{ai} \leftarrow \mathcal{D} \\ |\mathbb{x}| \leq n & (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R & (\mathsf{cm}, \mathsf{aux}_1) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_0) \\ \wedge\, \mathbf{V}^{[\mathcal{Q}, \mathsf{ans}]}(\mathbb{x}; \rho) = 1 & \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 & (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_1, \rho) \\ & \mathsf{tr} := (\mathsf{cm}, \rho, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \\ & \mathbb{w} \leftarrow \mathcal{E}^{\widetilde{\mathcal{P}}(\mathsf{aux}_0)}(\mathsf{pp}, \mathbb{x}, \mathsf{tr}) \end{array}\right] .$$

Now, note that by Constructions 4.10 and 4.12, the experiment above is equivalent to the following.

[25]Note that, by Construction 4.10, $\widetilde{\mathbf{P}}$ only executes $\widetilde{\mathcal{P}}(\mathsf{aux})$ (calls to $\widetilde{\mathcal{P}}(\mathsf{aux}', \rho)$ are continuations of executions of $\widetilde{\mathcal{P}}(\mathsf{aux})$). Therefore, an equivalent construction gives $\widetilde{\mathbf{P}}$ oracle access to $\widetilde{\mathcal{P}}(\mathsf{aux})$ (rather than to $\widetilde{\mathcal{P}}$), excluding aux from $\mathbf{aux}$.

$$\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ (\mathsf{cm}, \mathsf{aux}_1) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_0) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_1, \rho) \\ \boldsymbol{\rho} \leftarrow (\{0,1\}^{\mathsf{r}})^{\mathsf{N}} \\ \mathbf{aux} := \big(\mathsf{pp}, \mathsf{aux}_0, \boldsymbol{\rho}\big) \\ \widetilde{\Pi} \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux}) \\ \mathbb{w} \leftarrow \mathbf{E}(\mathbb{x}, \widetilde{\Pi})\end{array}\right] = \left[\begin{array}{l}\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ \big(\mathbb{x}, (\mathsf{cm}, \mathsf{aux})\big) \leftarrow \widetilde{\mathcal{P}}\big(\mathsf{pp}, \mathsf{ai}\big) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho) \\ (\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux},\cdot)}(\mathsf{pp}, \mathsf{cm}, \epsilon) \\ \mathbb{w} \leftarrow \mathbf{E}(\mathbb{x}, \widetilde{\Pi})\end{array}\right] .$$

We thus consider the above experiment, which augments that of Lemma 4.6 by appending an execution of $\mathbf{E}$, for the rest of the proof. Note that, as in Lemma 4.9, the experiment consisting of $(\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai})$ followed by $(\mathsf{cm}, \mathsf{aux}_1) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_0)$ can be replaced by $\big(\mathbb{x}, (\mathsf{cm}, \mathsf{aux})\big) \leftarrow \widetilde{\mathcal{P}}\big(\mathsf{pp}, \mathsf{ai}\big)$: since $\mathcal{R}$ only uses the second auxiliary state (which is obtained deterministically from the first), the former can be omitted. Note, moreover, that the explicit randomness for $\widetilde{\mathbf{P}}$ on the left-hand side is replaced with sampling by $\mathcal{R}$ on the right-hand side.

By total probability,

$$\Pr\left[\begin{array}{l}|\mathbb{x}| \le n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x}; \rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1\end{array}\right]$$
$$= \Pr\left[\begin{array}{l}|\mathbb{x}| \le n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x}; \rho) = 1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x}; \rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1\end{array}\right] + \Pr\left[\begin{array}{l}|\mathbb{x}| \le n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x}; \rho) \neq 1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x}; \rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1\end{array}\right] .$$

With the PCP prover $\widetilde{\mathbf{P}}$ in Construction 4.10 and using Definition 3.12, we have

$$\Pr\left[\begin{array}{l}|\mathbb{x}| \le n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x}; \rho) = 1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x}; \rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1\end{array}\right] \le \Pr\left[\begin{array}{l|l}|\mathbb{x}| \le n & \mathbf{ai} \leftarrow \mathbf{D} \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R & (\mathbb{x}, \widetilde{\Pi}) \leftarrow \widetilde{\mathbf{P}} \\ \wedge\, \mathbf{V}^{\widetilde{\Pi}}(\mathbb{x}) = 1 & \mathbb{w} \leftarrow \mathbf{E}(\mathbb{x}, \widetilde{\Pi})\end{array}\right] \le \kappa_{\mathsf{PCP}}(n) .$$

Finally, Lemma 4.6 implies

$$\Pr\left[\begin{array}{l}|\mathbb{x}| \le n \\ \wedge\, \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x}; \rho) \neq 1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x}; \rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1\end{array}\right] \le \Pr\left[\begin{array}{l}\mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbb{x}; \rho) \neq 1 \\ \wedge\, \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x}; \rho) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1\end{array}\right]$$
$$\le \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}, \mathsf{q}, t_{\mathsf{VC}}) + \epsilon ,$$

where $t_{\mathsf{VC}} \le \frac{4\mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}$, which concludes the proof. □

5 Interactive arguments based on public-coin IOPs

Theorem 5.1. *Consider these two ingredients:*

- $\mathsf{IOP} = (\mathbf{P}, \mathbf{V})$*, a public-coin IOP system for a relation* R *with round complexity* k*, alphabet* Σ*, proof length* l*, and query complexity* q*; and*
- $\mathsf{VC} = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Check})$*, a vector commitment scheme over alphabet* Σ*.*

Then $\mathsf{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V}) := \mathsf{IBCS}[\mathsf{IOP}, \mathsf{VC}]$ *(Construction 5.3) is a* $(2\mathsf{k}+1)$*-message public-coin interactive argument system for* R *whose soundness error* ϵ_{ARG} *and knowledge soundness error* κ_{ARG} *satisfy the following for every* $\epsilon > 0$ *and* $t_{\mathsf{ARG}} \geq t_{\mathbf{V}} + t_{\mathsf{VC.Check}} + \log|\Sigma| + \log \mathsf{l}_{\mathsf{max}}$*:*

$$\epsilon_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) \leq \epsilon_{\mathsf{IOP}}(n) + \epsilon_{\mathsf{VC}}\left(\lambda, \mathsf{l}_{\mathsf{max}}, \mathsf{q}_{\mathsf{max}}, t_{\mathsf{VC}}\right) + \epsilon \quad \textit{and}$$
$$\kappa_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) \leq \kappa_{\mathsf{IOP}}(n) + \epsilon_{\mathsf{VC}}\left(\lambda, \mathsf{l}_{\mathsf{max}}, \mathsf{q}_{\mathsf{max}}, t_{\mathsf{VC}}\right) + \epsilon \ ,$$

where $t_{\mathsf{VC}} = O\left(\frac{\mathsf{k} \cdot \mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}\right)$*. Moreover, the knowledge extractor runs in time* $t_{\mathcal{E}} = t_{\mathbf{E}} + O(t_{\mathsf{VC}})$*.*

Corollary 5.2. *Let* ARG *be as in Theorem 5.1. Assume that for any* $n \in \mathbb{N}$*,* $\epsilon_{\mathsf{VC}}(\cdot, \cdot, \cdot, t_{\mathsf{VC}}) = \mathrm{negl}(n)$ *if* $t_{\mathsf{VC}} = \mathsf{poly}(n)$*. Then, given that* $t_{\mathsf{ARG}} = \mathsf{poly}(n)$*, we have*

$$\epsilon_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) \leq \epsilon_{\mathsf{IOP}}(n) + \mathrm{negl}(n) \quad \textit{and}$$
$$\kappa_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) \leq \kappa_{\mathsf{IOP}}(n) + \mathrm{negl}(n) \ .$$

5.1 Construction

We describe below the construction of the interactive argument, which we denote $(\mathcal{P}, \mathcal{V}) := \mathsf{IBCS}[\mathsf{IOP}, \mathsf{VC}]$.

Construction 5.3. The argument generator $\mathcal{G}$ receives as input a security parameter $\lambda \in \mathbb{N}$ and an instance size bound $n \in \mathbb{N}$, and works as follows.

$\mathcal{G}(\lambda, n)$:
1. Sample public parameters for the VC scheme: $\mathsf{pp}_{\mathsf{VC}} \leftarrow \mathsf{VC.Gen}\big(1^{\lambda}, \mathsf{l}_{\mathsf{max}}(n)\big)$.[26]
2. Set public parameters for the interactive argument: $\mathsf{pp} := \mathsf{pp}_{\mathsf{VC}}$.
3. Output pp.

The argument prover $\mathcal{P}$ receives as input the public parameter pp, an instance $\mathbb{x}$ and a witness $\mathbb{w}$, and the argument verifier $\mathcal{V}$ receives as input the public parameter pp and the instance $\mathbb{x}$. Then $\mathcal{P}$ and $\mathcal{V}$ interact as follows.

1. $\mathcal{P}$'s commitments.
   For $i \in [\mathsf{k}]$:
   (a) $\mathcal{P}$'s i-th commitment.
      i. Compute the i-th IOP string $\Pi_i \in \Sigma^{\mathsf{l}_i}$ and auxiliary state:[27]

$$(\Pi_i, \mathbf{aux}_i) \leftarrow \begin{cases} \mathbf{P}(\mathbb{x}, \mathbb{w}) & \text{if } i = 1 \\ \mathbf{P}(\mathbf{aux}_{i-1}, \rho_{i-1}) & \text{if } i > 1 \end{cases} .$$

[26]Alternatively, $\mathsf{VC.Gen}$ could sample one set of public parameters $\mathsf{pp}_{\mathsf{VC},i}$ per proof length l_i and set $\mathsf{pp} := (\mathsf{pp}_{\mathsf{VC},i})_{i \in [\mathsf{k}]}$. For simplicity, we consider a single one and assume $\widetilde{\mathcal{P}}$ pads proofs where $\mathsf{l}_i < \mathsf{l}_{\mathsf{max}}$ with a fixed symbol $\sigma \in \Sigma$ where appropriate.

[27]Note the implicit setting of $\mathbf{aux}_{\mathsf{k}} := \perp$, since $\mathbf{P}$ only outputs the last proof string Π_{k} at the end of the interaction.

        ii. Compute a VC commitment to the IOP string: $(\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \mathsf{VC.Commit}(\mathsf{pp}, \Pi_i)$.
        iii. Send cm_i to $\mathcal{V}$.
    (b) $\mathcal{V}$'s i-th challenge.
        i. Sample the i-th IOP verifier randomness $\rho_i \leftarrow \{0,1\}^{\mathsf{r}_i}$.
        ii. Send ρ_i to $\mathcal{P}$.
2. $\mathcal{P}$'s response.
    (a) Run the IOP verifier $\mathbf{V}^{\Pi_1,\dots,\Pi_\mathsf{k}}(\mathbb{x}; \rho_1, \dots, \rho_\mathsf{k})$ to deduce $\mathcal{Q}_1, \dots, \mathcal{Q}_\mathsf{k}$, where $\mathcal{Q}_i \subseteq [\mathsf{l}_i]$ is the query set of $\mathbf{V}$ to Π_i.
    (b) For every $i \in [\mathsf{k}]$, compute an opening proof $\mathsf{pf}_i \leftarrow \mathsf{VC.Open}(\mathsf{pp}, \mathsf{aux}_i, \mathcal{Q}_i)$ and set $\mathsf{ans}_i := \Pi_i[\mathcal{Q}_i]$.
    (c) Send $\big((\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i)\big)_{i\in[\mathsf{k}]}$ to $\mathcal{V}$.
3. $\mathcal{V}$'s decision.
    Check that $\mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i\in[\mathsf{k}]}}(\mathbb{x}; \rho_1, \dots, \rho_\mathsf{k}) = 1$ and $\mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i) = 1$ for all $i \in [\mathsf{k}]$.[28]

The protocol has $\mathsf{k}+1$ rounds: the first k simulate the IOP, and in the last $\widetilde{\mathcal{P}}$ sends the query set assignments along with their opening proofs. Moreover, the protocol is public coin because the verifier's messages consist of random strings. We comment on the protocol's efficiency measures:

- the generator communication to prover and verifier consists of $|\mathsf{pp}_{\mathsf{VC}}|$ bits;
- the prover-to-verifier communication consists of $\sum_{i\in[\mathsf{k}]}(|\mathsf{cm}_i| + \mathsf{q} \cdot (\log \mathsf{l}_i + \log|\Sigma|) + |\mathsf{pf}_i|)$ bits;
- the verifier-to-prover communication consists of $\mathsf{r} = \sum_{i\in[\mathsf{k}]} \mathsf{r}_i$ bits;
- the time complexity of the argument generator is $t_{\mathsf{VC.Gen}}$.
- the time complexity of the argument prover is $t_{\mathbf{P}} + \mathsf{k} \cdot (t_{\mathsf{VC.Commit}} + t_{\mathsf{VC.Open}}) + t_{\mathbf{V}}$;
- the time complexity of the argument verifier is $t_{\mathbf{V}} + \mathsf{k} \cdot t_{\mathsf{VC.Check}}$.

5.2 Security reduction

As in Section 4.2, our analysis relies on the following security reduction lemma that relates the acceptance probability of $\mathsf{ARG} := \mathsf{IBCS}[\mathsf{IOP}, \mathsf{VC}]$ with that of IOP.

Lemma 5.4. *There exists a probabilistic algorithm $\mathcal{R}$ which, for every size bound $t_{\mathsf{ARG}} \geq t_{\mathbf{V}} + t_{\mathsf{VC.Check}} + \log|\Sigma| + \log \mathsf{l}_{\max}$, circuit $\widetilde{\mathcal{P}}$ of size t_{ARG} and $\epsilon > 0$, satisfies*

$$\Pr\left[\begin{array}{l|l} \mathbf{V}^{([\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i])_{i\in[\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}) \neq 1 & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big) = 1 & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, \Big(\bigwedge_{i\in[\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\Big) = 1 & (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ & \rho_0 := \bot \\ & \text{For } i \in [\mathsf{k}]: \\ & \quad (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \rho_{i-1}) \\ & \quad (\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i, \cdot)}\big(\mathsf{pp}, (\mathsf{cm}_j)_{j\in[i]}, (\rho_j)_{j<i}, \epsilon\big) \\ & \quad \rho_i \leftarrow \{0,1\}^{\mathsf{r}_i} \\ & \big((\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i)\big)_{i\in[\mathsf{k}]} \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_\mathsf{k}, \rho_\mathsf{k}) \end{array}\right]$$
$$\leq \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}_{\max}, \mathsf{q}_{\max}, t_{\mathsf{VC}}) + \epsilon \ ,$$

where t_{VC} and the total runtime of all executions of $\mathcal{R}$ are $O\left(\frac{\mathsf{k}\cdot\mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}\right)$.

[28] $\mathcal{V}$ also implicitly checks for appropriate padding, i.e., that $\mathsf{ans}_i[q] = \sigma$ for every $q \in \mathcal{Q}_i \setminus [\mathsf{l}_i]$.

We first construct the reductor $\mathcal{R}$, which, similarly to Construction 4.8, will include a sampling subroutine $\mathcal{S}$ and a post-processing subroutine $\mathcal{R}_{\mathrm{post}}$.

Construction 5.5. Given a number of iterations $\mathsf{N} \in \mathbb{N}$, we construct the sampler $\mathcal{S}$ as follows.

```
$\mathcal{S}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)}\big(\mathsf{pp}, (\mathsf{cm}_j)_{j\in[i]}, (\rho_j)_{j<i}, \mathsf{N}\big)$:
1. Initialize $\mathcal{K} := \emptyset$.
2. Repeat the following $\mathsf{N}$ times:
    (a) Sample IOP verifier randomness: $(\rho'_i, \dots, \rho'_{\mathsf{k}}) \leftarrow \{0,1\}^{\mathsf{r}_i} \times \cdots \times \{0,1\}^{\mathsf{r}_{\mathsf{k}}}$.
    (b) Obtain $\big((\mathcal{Q}'_i, \mathsf{ans}'_i, \mathsf{pf}'_i), \big((\mathsf{cm}'_j, \mathcal{Q}'_j, \mathsf{ans}'_j, \mathsf{pf}'_j)\big)_{i<j\leq \mathsf{k}}\big) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_i, \rho_1, \dots, \rho_{i-1}, \rho'_i, \dots, \rho'_{\mathsf{k}})$.
    (c) If $\mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}'_i, \mathsf{ans}'_i, \mathsf{pf}'_i\big) = 1$, add $(\mathcal{Q}'_i, \mathsf{ans}'_i, \mathsf{pf}'_i)$ to $\mathcal{K}$.
3. Output $\mathcal{K}$.
```

Construction 5.6. The reductor $\mathcal{R}$ is defined below.

```
$\mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)}\big(\mathsf{pp}, (\mathsf{cm}_j)_{j\in[i]}, (\rho_j)_{j<i}, \epsilon\big)$:
1. Set $\mathsf{N} := \mathsf{l}/\epsilon$.
2. Run $\mathcal{K} \leftarrow \mathcal{S}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)}\big(\mathsf{pp}, (\mathsf{cm}_j)_{j\in[i]}, (\rho_j)_{j<i}, \mathsf{N}\big)$.
3. Run $(\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}_{\mathrm{post}}(\mathcal{K}, \mathsf{l}_i)$.[29]
4. Output $(\widetilde{\mathcal{Q}}, \widetilde{\Pi})$.
```

Note that in each of its N iterations, $\mathcal{S}$ runs $\mathsf{VC.Check}$ (once) and makes at most k queries to $\widetilde{\mathcal{P}}$. Since $t_{\mathsf{ARG}} \geq t_{\mathsf{VC.Check}}$ (and the executions of $\widetilde{\mathcal{P}}$ dominate the remaining steps of $\mathcal{R}$), the total runtime of $\mathcal{R}$ across all k rounds is $O\left(\frac{\mathsf{k}\cdot\mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}\right)$.

Proof. Throughout this proof, probabilistic expressions are with respect to the following experiment:

$$\left[\begin{array}{l}
\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\
\mathsf{ai} \leftarrow \mathcal{D} \\
(\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\
\rho_0 := \bot \\
\text{For } i \in [\mathsf{k}]: \\
\quad (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \rho_{i-1}) \\
\quad (\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)}\big(\mathsf{pp}, (\mathsf{cm}_j)_{j\in[i]}, (\rho_j)_{j<i}, \epsilon\big) \\
\quad \rho_i \leftarrow \{0,1\}^{\mathsf{r}_i} \\
\big((\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i)\big)_{i\in[\mathsf{k}]} \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{\mathsf{k}}, \rho_{\mathsf{k}})
\end{array}\right],$$

[29] Recall that $\mathcal{R}_{\mathrm{post}}$ pieces together a proof string $\widetilde{\Pi}$ from a set of samples $\mathcal{K}$ (see Construction 4.8).

which by Construction 5.6 is equivalent to

$$\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \rho_0 \coloneqq \bot \\ \text{For } i \in [\mathsf{k}] : \\ \quad (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \rho_{i-1}) \\ \quad \mathcal{K}_i \leftarrow \mathcal{S}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i, \cdot)}\big(\mathsf{pp}, (\mathsf{cm}_j)_{j \in [i]}, (\rho_j)_{j<i}, \mathsf{N}\big) \\ \quad (\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i) \leftarrow \mathcal{R}_{\text{post}}(\mathcal{K}_i, \mathsf{l}_i) \\ \quad \rho_i \leftarrow \{0,1\}^{\mathsf{r}_i} \\ \big((\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i)\big)_{i \in [\mathsf{k}]} \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_\mathsf{k}, \rho_\mathsf{k}) \end{array}\right] . \tag{4}$$

The argument is analogous to Lemma 4.6. We first note that

$$\left[\begin{array}{l} \mathbf{V}^{([\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i])_{i \in [\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i \in [\mathsf{k}]}\big) \neq 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i \in [\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i \in [\mathsf{k}]}\big) = 1 \\ \wedge\, \Big(\bigwedge_{i \in [\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\Big) = 1 \end{array}\right]$$

implies, likewise, the existence of valid openings with disagreeing answers or a missing query in one of the IOP strings. We analyze the two cases separately.

Valid openings with disagreeing answers. We aim to show that

$$\Pr\left[\begin{array}{l} \exists\, i \in [\mathsf{k}], q \in \mathcal{Q}_i \cap \widetilde{\mathcal{Q}}_i : \mathsf{ans}_i[q] \neq \widetilde{\Pi}_i[q] \\ \wedge\, \Big(\bigwedge_{i \in [\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\Big) = 1 \end{array}\right] \leq \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}_{\mathsf{max}}, \mathsf{q}_{\mathsf{max}}, t_{\mathsf{VC}})$$

where $t_{\mathsf{VC}} = O\left(\frac{\mathsf{k} \cdot \mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}\right)$.

We consider the following adversary A_{VC} against the vector commitment scheme, which (essentially) follows Experiment 4:

```
$A_{\mathsf{VC}}(\mathsf{pp}, \mathsf{ai})$:
1. Run $(\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai})$.
2. Set $\mathsf{N} \coloneqq \frac{\mathsf{l}}{\epsilon}$, $\rho_0 \coloneqq \bot$ and sample $(\rho_i)_{i \in [\mathsf{k}]} \leftarrow \{0,1\}^{\mathsf{r}_1 + \cdots + \mathsf{r}_\mathsf{k}}$.
3. For $i \in [\mathsf{k}]$:
   (a) Run $(\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \rho_{i-1})$.
   (b) Run $\mathcal{K}_i \leftarrow \mathcal{S}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i, \cdot)}\big(\mathsf{pp}, (\mathsf{cm}_j)_{j \in [i]}, (\rho_j)_{j<i}, \mathsf{N}\big)$.
4. Run $\big((\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i)\big)_{i \in [\mathsf{k}]} \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_\mathsf{k}, \rho_\mathsf{k})$.
5. If there exist $i \in [\mathsf{k}]$ and $(\mathcal{Q}'_i, \mathsf{ans}'_i, \mathsf{pf}'_i) \in \mathcal{K}_i$ with $q \in \mathcal{Q}_i \cap \mathcal{Q}'_i$ and $\mathsf{ans}_i[q] \neq \mathsf{ans}'_i[q]$, output
   $(\mathsf{cm}_i, \mathsf{ans}_i, \mathsf{ans}'_i, \mathcal{Q}_i, \mathcal{Q}'_i, \mathsf{pf}_i, \mathsf{pf}'_i)$.
6. Otherwise, output (the "dummy" tuple) $(\mathsf{cm}_1, \mathsf{ans}_1, \mathsf{ans}_1, \mathcal{Q}_1, \mathcal{Q}_1, \mathsf{pf}_1, \mathsf{pf}_1)$.
```

The time complexity of k executions of the sampler $\mathcal{S}$ is $O(\mathsf{kN} \cdot t_{\mathsf{ARG}})$ and the collision-finding check (Step 5) runs in time $O\big(\mathsf{kN} \cdot (\log|\Sigma| + \log \mathsf{l}_i)\big) = O(\mathsf{kN} \cdot t_{\mathsf{ARG}})$, so the total time complexity of A_{VC} is $t_{\mathsf{VC}} = O(\mathsf{kN} \cdot t_{\mathsf{ARG}}) = O\left(\frac{\mathsf{k} \cdot \mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}\right)$.

Therefore, according to Definition 3.8,[30]

$$
\begin{aligned}
&\Pr\left[\begin{array}{l} \exists\, i \in [\mathsf{k}], q \in \mathcal{Q}_i \cap \widetilde{\mathcal{Q}}_i : \mathsf{ans}_i[q] \neq \widetilde{\Pi}_i[q] \\ \wedge \left(\bigwedge_{i \in [\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\right) = 1 \end{array}\right] \\
&\leq \Pr\left[\exists\, i \in [\mathsf{k}] : \left(\begin{array}{l} \exists\, q \in \mathcal{Q}_i \cap \widetilde{\mathcal{Q}}_i : \mathsf{ans}_i[q] \neq \widetilde{\Pi}_i[q] \\ \wedge\, \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big) = 1 \end{array}\right)\right] \\
&\leq \Pr\left[\begin{array}{l} |\mathcal{Q}| = |\mathcal{Q}'| \leq \mathsf{q}_{\mathsf{max}} \\ \wedge\, \exists\, q \in \mathcal{Q} \cap \mathcal{Q}' : \mathsf{ans}[q] \neq \mathsf{ans}'[q] \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}', \mathsf{ans}', \mathsf{pf}') = 1 \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{VC.Gen}(1^\lambda, \mathsf{l}_{\mathsf{max}}) \\ \begin{pmatrix} \mathsf{cm}, \mathsf{ans}, \mathsf{ans}', \\ \mathcal{Q}, \mathcal{Q}', \mathsf{pf}, \mathsf{pf}' \end{pmatrix} \leftarrow A_{\mathsf{VC}}(\mathsf{pp}) \end{array}\right] \\
&\leq \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}_{\mathsf{max}}, \mathsf{q}_{\mathsf{max}}, t_{\mathsf{VC}}) \ .
\end{aligned}
$$

Missing positions in $\widetilde{\Pi}_i$. We are left to show

$$
\Pr\left[\begin{array}{l} \left(\exists\, i \in [\mathsf{k}] : \mathcal{Q}_i \setminus \widetilde{\mathcal{Q}}_i \neq \emptyset\right) \\ \wedge \left(\bigwedge_{i \in [\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\right) = 1 \end{array}\right] \leq \epsilon \ .
$$

To this end, fix a partial sequence of prover inputs $(\mathsf{pp}, \mathsf{ai}, (\rho_j)_{j<i})$ of Eq. 4 and define the *weight* of a coordinate $q \in [\mathsf{l}_i]$ with respect to $(\mathsf{pp}, \mathsf{ai}, (\rho_j)_{j<i})$ as

$$
\begin{aligned}
&\delta_{(\mathsf{pp},\mathsf{ai},(\rho_j)_{j<i})}(q) \\
&:= \Pr\left[\begin{array}{l} q \in \mathcal{Q}_i \\ \wedge \left(\bigwedge_{j \in [\mathsf{k}]} \mathsf{VC.Check}\begin{pmatrix} \mathsf{pp}, \mathsf{cm}_j, \\ \mathcal{Q}_j, \mathsf{ans}_j, \mathsf{pf}_j \end{pmatrix}\right) = 1 \end{array} \middle| \begin{array}{l} (\rho_j)_{i \leq j \leq \mathsf{k}} \leftarrow \{0,1\}^{\mathsf{r}_i + \cdots + \mathsf{r}_\mathsf{k}} \\ \begin{pmatrix} \mathbb{x}, \mathsf{cm}_1, \ldots, \mathsf{cm}_\mathsf{k}, \\ \big((\mathcal{Q}_j, \mathsf{ans}_j, \mathsf{pf}_j)\big)_{j \in [\mathsf{k}]} \end{pmatrix} \leftarrow \widetilde{\mathcal{P}}\begin{pmatrix} \mathsf{pp}, \mathsf{ai}, \\ (\rho_1, \ldots, \rho_\mathsf{k}) \end{pmatrix} \end{array}\right] \ .
\end{aligned}
$$

(We perform a partial execution of $\widetilde{\mathcal{P}}$ as in Lemma 4.6, omiting intermediate auxiliary states.) A union bound over all $q \in [\mathsf{l}_i]$ yields

$$
\begin{aligned}
&\Pr\left[\mathcal{Q}_i \setminus \widetilde{\mathcal{Q}}_i \neq \emptyset\right] \\
&= \Pr\left[\exists\, q \in [\mathsf{l}_i] : q \in \mathcal{Q}_i \ \wedge\ q \notin \widetilde{\mathcal{Q}}_i\right] \\
&\leq \max_{(\mathsf{pp},\mathsf{ai},(\rho_j)_{j<i})} \left\{ \Pr\left[\begin{array}{l} \exists\, q \in [\mathsf{l}_i] : q \in \mathcal{Q}_i \\ \wedge\, q \notin \widetilde{\mathcal{Q}}_i \end{array} \middle| \begin{array}{l} (\rho_j)_{i \leq j \leq \mathsf{k}} \leftarrow \{0,1\}^{\mathsf{r}_i + \cdots + \mathsf{r}_\mathsf{k}} \\ \begin{pmatrix} \mathbb{x}, \mathsf{cm}_1, \ldots, \mathsf{cm}_\mathsf{k}, \\ \big((\mathcal{Q}_j, \mathsf{ans}_j, \mathsf{pf}_j)\big)_{j \in [\mathsf{k}]} \end{pmatrix} \leftarrow \widetilde{\mathcal{P}}\begin{pmatrix} \mathsf{pp}, \mathsf{ai}, \\ (\rho_1, \ldots, \rho_\mathsf{k}) \end{pmatrix} \end{array}\right] \right\} \\
&\leq \max_{(\mathsf{pp},\mathsf{ai},(\rho_j)_{j<i})} \left\{ \sum_{q \in [\mathsf{l}_i]} \delta_{(\mathsf{pp},\mathsf{ai},(\rho_j)_{j<i})}(q) \cdot \big(1 - \delta_{(\mathsf{pp},\mathsf{ai},(\rho_j)_{j<i})}(q)\big)^{\mathsf{N}} \right\} \\
&\leq \frac{\mathsf{l}_i}{\mathsf{N}} \ ,
\end{aligned}
$$

[30]Note that A_{VC}, if successful, outputs a tuple in one of the $\mathcal{K}_i$. Denoting by E_i the event that it is found in $\mathcal{K}_i$, the right-hand side of the second inequality may be replaced by $\sum_i \Pr[E_i] \cdot \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}_i, \mathsf{q}_i, t_{\mathsf{VC}})$, and the last follows from monotonicity of ϵ_{VC} (see Remark 3.9).

and another union bound over $i \in [\mathsf{k}]$ gives

$$\Pr\left[\begin{array}{l}\left(\exists\, i \in [\mathsf{k}] : \mathcal{Q}_i \setminus \widetilde{\mathcal{Q}}_i \neq \emptyset\right) \\ \wedge \left(\bigwedge_{i\in[\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\right) = 1\end{array}\right] \leq \sum_{i\in[\mathsf{k}]} \frac{\mathsf{l}_i}{\mathsf{N}} = \frac{\mathsf{l}}{\mathsf{N}} \enspace .$$

Since $\mathsf{N} = \mathsf{l}/\epsilon$, we have

$$\begin{aligned}
&\Pr\left[\begin{array}{l}\mathbf{V}^{([\widetilde{\mathcal{Q}}_i,\widetilde{\Pi}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big) \neq 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i,\mathsf{ans}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big) = 1 \\ \wedge \left(\bigwedge_{i\in[\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\right) = 1\end{array}\right] \\
&\leq \Pr\left[\begin{array}{l}\exists\, i \in [\mathsf{k}], q \in \mathcal{Q}_i \cap \widetilde{\mathcal{Q}}_i : \mathsf{ans}_i[q] \neq \widetilde{\Pi}_i[q] \\ \wedge \left(\bigwedge_{i\in[\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\right) = 1\end{array}\right] \\
&\quad + \Pr\left[\begin{array}{l}\exists\, i \in [\mathsf{k}], q \in \mathcal{Q}_i : q \notin \widetilde{\mathcal{Q}}_i \\ \wedge \left(\bigwedge_{i\in[\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\right) = 1\end{array}\right] \\
&\leq \epsilon_{\mathsf{VC}}\left(\lambda, \mathsf{l}_{\mathsf{max}}, \mathsf{q}_{\mathsf{max}}, t_{\mathsf{VC}}\right) + \epsilon \enspace ,
\end{aligned}$$

which concludes the proof. □

5.3 Adaptive soundness

Lemma 5.7. *For every security parameter $\lambda \in \mathbb{N}$, circuit size bound $t_{\mathsf{ARG}} \geq t_{\mathbf{V}} + t_{\mathsf{VC.Check}} + \log|\Sigma| + \log \mathsf{l}_{\mathsf{max}}$, circuit $\widetilde{\mathcal{P}}$ of size t_{ARG}, instance size bound $n \in \mathbb{N}$ and $\epsilon > 0$, the soundness error of the argument system in Construction 5.3 satisfies*

$$\epsilon_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) \leq \epsilon_{\mathsf{IOP}}(n) + \epsilon_{\mathsf{VC}}\left(\lambda, \mathsf{l}_{\mathsf{max}}, \mathsf{q}_{\mathsf{max}}, t_{\mathsf{VC}}\right) + \epsilon \enspace ,$$

where $t_{\mathsf{VC}} = O(\frac{\mathsf{k}\cdot\mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}})$.

Proof. From Definition 3.3 and Construction 5.3, our goal is to upper bound

$$\begin{aligned}
&\Pr\left[\begin{array}{l|l}|\mathbb{x}| \leq n & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \wedge\, \mathbb{x} \notin L(R) & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, b = 1 & (\mathbb{x}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ & b \leftarrow \big\langle \widetilde{\mathcal{P}}(\mathsf{aux}), \mathcal{V}(\mathsf{pp}, \mathbb{x})\big\rangle\end{array}\right] \\
&= \Pr\left[\begin{array}{l|l} & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ & \mathsf{ai} \leftarrow \mathcal{D} \\ |\mathbb{x}| \leq n & (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \wedge\, \mathbb{x} \notin L(R) & \rho_0 := \bot \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i,\mathsf{ans}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big) = 1 & \text{For } i \in [\mathsf{k}] : \\ \wedge \left(\bigwedge_{i\in[\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\right) = 1 & \quad (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \rho_{i-1}) \\ & \quad \rho_i \leftarrow \{0,1\}^{\mathsf{r}_i} \\ & \big((\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i)\big)_{i\in[\mathsf{k}]} \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{\mathsf{k}}, \rho_{\mathsf{k}})\end{array}\right] \enspace .
\end{aligned}$$

As in Lemma 5.4, we consider the following experiment, which augments the above by executing $\mathcal{R}$ (multiple times) and leaves the probability unchanged.

$$\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \rho_0 := \bot \\ \text{For } i \in [\mathsf{k}] : \\ \quad (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \rho_{i-1}) \\ \quad (\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i, \cdot)}\big(\mathsf{pp}, (\mathsf{cm}_j)_{j \in [i]}, (\rho_j)_{j<i}, \epsilon\big) \\ \quad \rho_i \leftarrow \{0,1\}^{\mathsf{r}_i} \\ \big((\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i)\big)_{i \in [\mathsf{k}]} \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_\mathsf{k}, \rho_\mathsf{k}) \end{array}\right].$$

By total probability,

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i\in[\mathsf{k}]}}(\mathbb{x}; \rho) = 1 \\ \wedge\, \Big(\bigwedge_{i \in [\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\Big) = 1 \end{array}\right]$$
$$= \Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{([\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i])_{i\in[\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}) = 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big) = 1 \\ \wedge\, \Big(\bigwedge_{i \in [\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\Big) = 1 \end{array}\right]$$
$$+ \Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{([\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i])_{i\in[\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}) \neq 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big) = 1 \\ \wedge\, \Big(\bigwedge_{i \in [\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\Big) = 1 \end{array}\right].$$

We first bound the probability on the left-hand side by using the soundness error definition of IOP in Definition 3.14.

Construction 5.8. We will construct an IOP prover $\widetilde{\mathbf{P}}$, and first define its auxiliary input distribution $\mathbf{D}$ as follows:

$\mathbf{D}$:

1. Sample $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n)$ followed by $\mathsf{ai} \leftarrow \mathcal{D}$.
2. Sample $\boldsymbol{\rho} := (\boldsymbol{\rho}_i)_{i\in[\mathsf{k}]} \leftarrow \big(\{0,1\}^{\mathsf{r}_1+\cdots+\mathsf{r}_\mathsf{k}}\big)^{\mathsf{N}} \times \big(\{0,1\}^{\mathsf{r}_2+\cdots+\mathsf{r}_\mathsf{k}}\big)^{\mathsf{N}} \times \cdots \times \big(\{0,1\}^{\mathsf{r}_\mathsf{k}}\big)^{\mathsf{N}}$.
3. Output $\mathbf{ai} := \big(\mathsf{pp}, \mathsf{ai}, \boldsymbol{\rho}\big)$.

The IOP prover is then defined as follows.

- $\widetilde{\mathbf{P}}(\mathbf{ai})$:
  1. Parse $\mathbf{ai}$ as $\big(\mathsf{pp}, \mathsf{ai}, \boldsymbol{\rho}\big)$.

2. Run $(\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai})$.
3. Set $\mathbf{aux} := \big(\mathsf{pp}, \mathsf{aux}_0, \boldsymbol{\rho}\big)$.
4. Output $(\mathbb{x}, \mathbf{aux})$.

- $\widetilde{\mathbf{P}}(\mathbf{aux}, \rho)$:
  1. Parse $\mathbf{aux}$ as $\big(\mathsf{pp}, \mathsf{aux}_{i-1}, (\mathsf{cm}_j)_{j<i}, \boldsymbol{\rho}\big)$.[31]
  2. Set $\rho_{i-1} := \rho$ and run $(\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \rho_{i-1})$
  3. Run $(\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i, \cdot)}\big(\mathsf{pp}, (\mathsf{cm}_j)_{j\in[i]}, (\rho_j)_{j<i}, \epsilon; \boldsymbol{\rho}_i\big)$.
  4. Set $\mathbf{aux}_i := \big(\mathsf{pp}, \mathsf{aux}_i, (\mathsf{cm}_j)_{j\in[i]}, (\rho_j)_{j<i}, \boldsymbol{\rho}\big)$.[32]
  5. Output $(\widetilde{\Pi}_i, \mathbf{aux}_i)$.

Using Definition 3.14 (with Experiment 1, the explicit public-coin IOP), we obtain

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathbf{V}^{([\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i])_{i\in[\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}) = 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big) = 1 \\ \wedge\, \Big(\bigwedge_{i\in[\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\Big) = 1 \end{array}\right]$$
$$\le \Pr\left[\begin{array}{l|l} & \mathbf{ai} \leftarrow \mathbf{D} \\ |\mathbb{x}| \le n & (\mathbb{x}, \mathbf{aux}_0) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \\ \wedge\, \mathbb{x} \notin L(R) & \rho_0 := \bot \\ \wedge\, \mathbf{V}^{(\widetilde{\Pi}_i)_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big) = 1 & \text{For } i \in [\mathsf{k}]: \\ & \quad (\widetilde{\Pi}_i, \mathbf{aux}_i) \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux}_{i-1}, \rho_{i-1}) \\ & \quad \rho_i \leftarrow \{0,1\}^{\mathsf{r}_i} \end{array}\right]$$
$$\le \epsilon_{\mathsf{IOP}}(n) \;\; .$$

Lastly, an application of Lemma 5.4 yields

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathbf{V}^{([\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i])_{i\in[\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}) \neq 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big) = 1 \\ \wedge\, \Big(\bigwedge_{i\in[\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\Big) = 1 \end{array}\right]$$
$$\le \Pr\left[\begin{array}{l} \mathbf{V}^{([\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i])_{i\in[\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}) = 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big) = 1 \\ \wedge\, \Big(\bigwedge_{i\in[\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\Big) = 1 \end{array}\right]$$
$$\le \epsilon_{\mathsf{VC}}\left(\lambda, \mathsf{l}_{\max}, \mathsf{q}_{\max}, t_{\mathsf{VC}}\right) + \epsilon \;\; ,$$

which concludes the proof. □

[31] Note that the round $i < \mathsf{k}$ is determined by the contents of $\mathbf{aux}$.

[32] We may assume $\mathbf{aux} = \bot$ in the case $i = \mathsf{k}$, and the following step outputs $(\widetilde{\Pi}_{\mathsf{k}}, \mathbf{aux}_{\mathsf{k}}) = \widetilde{\Pi}_{\mathsf{k}}$.

5.4 Adaptive knowledge soundness

Lemma 5.9. *For every security parameter $\lambda \in \mathbb{N}$, circuit size bound $t_{\mathsf{ARG}} \geq t_{\mathbf{V}} + t_{\mathsf{VC.Check}} + \log|\Sigma| + \log \mathsf{l}_{\max}$, circuit $\widetilde{\mathcal{P}}$ of size t_{ARG}, instance size bound $n \in \mathbb{N}$ and $\epsilon > 0$, the knowledge soundness error of the argument system obtained by Construction 5.3 satisfies*

$$\kappa_{\mathsf{ARG}}(\lambda, n, t_{\mathsf{ARG}}) \leq \kappa_{\mathsf{IOP}}(n) + \epsilon_{\mathsf{IOP}}(n) + \epsilon_{\mathsf{VC}}\left(\lambda, \mathsf{l}_{\max}, \mathsf{q}_{\max}, t_{\mathsf{VC}}\right) + \epsilon \ ,$$

where $t_{\mathsf{VC}} = O\left(\frac{\mathsf{k} \cdot \mathsf{l}}{\epsilon} \cdot t_{\mathsf{ARG}}\right)$. If the IOP extractor's runtime is $t_{\mathbf{E}}$, the argument system's extractor is $t_{\mathcal{E}} = t_{\mathbf{E}} + O(t_{\mathsf{VC}})$.

Construction 5.10. Let $\mathbf{E}$ be the extractor for IOP. We use $\widetilde{\mathbf{P}}$ (Construction 5.8) and $\mathbf{E}$ to construct the knowledge extractor $\mathcal{E}$ for ARG:

$\mathcal{E}^{\widetilde{\mathcal{P}}(\mathsf{aux})}(\mathsf{pp}, \mathbb{x}, \mathsf{tr})$:

1. Parse tr as $\left((\mathsf{cm}_i, \rho_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i)\right)_{i \in [\mathsf{k}]}$.
2. Sample $\boldsymbol{\rho} := (\boldsymbol{\rho}_i)_{i \in [\mathsf{k}]} \leftarrow \left(\{0,1\}^{\mathsf{r}_1 + \cdots + \mathsf{r}_\mathsf{k}}\right)^{\mathsf{N}} \times \left(\{0,1\}^{\mathsf{r}_2 + \cdots + \mathsf{r}_\mathsf{k}}\right)^{\mathsf{N}} \times \cdots \times \left(\{0,1\}^{\mathsf{r}_\mathsf{k}}\right)^{\mathsf{N}}$.
3. Set $\mathbf{aux} := \left(\mathsf{pp}, \mathsf{aux}, \boldsymbol{\rho}\right)$.
4. Run $(\widetilde{\Pi}_i)_{i \in [\mathsf{k}]} \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux})$ and set $\mathbf{tr} := \left((\widetilde{\Pi}_i, \rho_i)_{i < \mathsf{k}}, \widetilde{\Pi}_\mathsf{k}\right)$.
5. Run $\mathbb{w} \leftarrow \mathbf{E}^{\widetilde{\mathbf{P}}(\mathbf{aux})}(\mathbb{x}, \mathbf{tr})$.
6. Output $\mathbb{w}$.

Proof. From Definition 3.4 and Construction 5.3, out goal is to upper bound

$$\Pr\left[\begin{array}{l|l} & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ |\mathbb{x}| \leq n & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R & (\mathbb{x}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \wedge\, b = 1 & b \overset{\mathsf{tr}}{\leftarrow} \left\langle \widetilde{\mathcal{P}}(\mathsf{aux}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \right\rangle \\ & \mathbb{w} \leftarrow \mathcal{E}^{\widetilde{\mathcal{P}}(\mathsf{aux})}(\mathsf{pp}, \mathbb{x}, \mathsf{tr}) \end{array}\right]$$

$$= \Pr\left[\begin{array}{l|l} & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ & \mathsf{ai} \leftarrow \mathcal{D} \\ & (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ |\mathbb{x}| \leq n & \rho_0 := \bot \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R & \text{For } i \in [\mathsf{k}] : \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i \in [\mathsf{k}]}}\left(\mathbb{x}; (\rho_i)_{i \in [\mathsf{k}]}\right) = 1 & \quad (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \rho_{i-1}) \\ \wedge\, \left(\bigwedge_{i \in [\mathsf{k}]} \mathsf{VC.Check}\left(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\right)\right) = 1 & \quad \rho_i \leftarrow \{0,1\}^{\mathsf{r}_i} \\ & \left((\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i)\right)_{i \in [\mathsf{k}]} \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_\mathsf{k}, \rho_\mathsf{k}) \\ & \mathsf{tr} := \left((\mathsf{cm}_i, \rho_i, (\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i))\right)_{i \in [\mathsf{k}]} \\ & \mathbb{w} \leftarrow \mathcal{E}^{\widetilde{\mathcal{P}}(\mathsf{aux})}(\mathsf{pp}, \mathbb{x}, \mathsf{tr}) \end{array}\right]$$

We consider the following experiment, which is equivalent to the above and augments that of Lemma 5.4 by appending an execution of $\mathbf{E}$, for the rest of the proof.

$$\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ \rho_0 \coloneqq \bot \\ (\boldsymbol{\rho}_i)_{i \in \mathsf{k}} \leftarrow \left(\{0,1\}^{\sum_{i=i}^{\mathsf{k}} \mathsf{r}_j}\right)^{\mathsf{N}} \times \cdots \times \left(\{0,1\}^{\mathsf{r}_\mathsf{k}}\right)^{\mathsf{N}} \\ \text{For } i \in [\mathsf{k}] : \\ \quad (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \rho_{i-1}) \\ \quad (\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i, \cdot)}\left(\mathsf{pp}, (\mathsf{cm}_\ell)_{\ell \in [i]}, (\rho_\ell)_{\ell < i}, \epsilon; \boldsymbol{\rho}_i\right) \\ \quad \rho_i \leftarrow \{0,1\}^{\mathsf{r}_i} \\ \left((\mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i)\right)_{i \in [\mathsf{k}]} \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_\mathsf{k}, \rho_\mathsf{k}) \\ \mathbf{aux} \coloneqq \left(\mathsf{pp}, \mathsf{aux}, \boldsymbol{\rho}\right) \\ \mathbf{tr} \coloneqq \left((\widetilde{\Pi}_i, \rho_i)_{i \in [\mathsf{k}-1]}, \widetilde{\Pi}_\mathsf{k}\right) \\ \mathbb{w} \leftarrow \mathbf{E}^{\widetilde{\mathbf{P}}(\mathbf{aux})}(\mathbb{x}, \mathbf{tr}) \end{array}\right] .$$

By total probability,

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i \in [\mathsf{k}]}}\left(\mathbb{x}; (\rho_i)_{i \in [\mathsf{k}]}\right) = 1 \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 \end{array}\right]$$
$$= \Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{([\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i])_{i \in [\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i \in [\mathsf{k}]}) = 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i \in [\mathsf{k}]}}\left(\mathbb{x}; (\rho_i)_{i \in [\mathsf{k}]}\right) = 1 \\ \wedge\, \left(\bigwedge_{i \in [\mathsf{k}]} \mathsf{VC.Check}\left(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\right)\right) = 1 \end{array}\right]$$
$$+ \Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{([\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i])_{i \in [\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i \in [\mathsf{k}]}) \neq 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i \in [\mathsf{k}]}}\left(\mathbb{x}; (\rho_i)_{i \in [\mathsf{k}]}\right) = 1 \\ \wedge\, \left(\bigwedge_{i \in [\mathsf{k}]} \mathsf{VC.Check}\left(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\right)\right) = 1 \end{array}\right] .$$

By the construction of $\widetilde{\mathbf{P}}$ (Construction 5.8) and Definition 3.15, we have

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{([\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i])_{i \in [\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i \in [\mathsf{k}]}) = 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i \in [\mathsf{k}]}}\left(\mathbb{x}; (\rho_i)_{i \in [\mathsf{k}]}\right) = 1 \\ \wedge\, \left(\bigwedge_{i \in [\mathsf{k}]} \mathsf{VC.Check}\left(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\right)\right) = 1 \end{array}\right]$$
$$\le \Pr\left[\begin{array}{l|l} |\mathbb{x}| \le n & \mathbf{ai} \leftarrow \mathbf{D} \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R & (\mathbb{x}, \mathbf{aux}) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \\ \wedge\, b = 1 & b \xleftarrow{\mathbf{tr}} \langle \widetilde{\mathbf{P}}(\mathbf{aux}), \mathbf{V}(\mathbb{x}) \rangle \\ & \mathbb{w} \leftarrow \mathbf{E}^{\widetilde{\mathbf{P}}(\mathbf{aux})}(\mathbb{x}, \mathbf{tr}) \end{array}\right]$$

$$\leq \kappa_{\mathsf{IOP}}(n) \ \ .$$

Finally, Lemma 5.4 implies

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{([\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i])_{i\in[\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}) \neq 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big) = 1 \\ \wedge\, \Big(\bigwedge_{i\in[\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\Big) = 1 \end{array}\right]$$

$$\leq \Pr\left[\begin{array}{l} \mathbf{V}^{([\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i])_{i\in[\mathsf{k}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}) \neq 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i\in[\mathsf{k}]}}\big(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}]}\big) = 1 \\ \wedge\, \Big(\bigwedge_{i\in[\mathsf{k}]} \mathsf{VC.Check}\big(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \mathsf{ans}_i, \mathsf{pf}_i\big)\Big) = 1 \end{array}\right]$$

$$\leq \epsilon_{\mathsf{VC}}\left(\lambda, \mathsf{l}_{\mathsf{max}}, \mathsf{q}_{\mathsf{max}}, t_{\mathsf{VC}}\right) + \epsilon \ \ ,$$

which concludes the proof. □

6 Interactive arguments based on public-query IOPs

We argue that the security analysis for interactive arguments based on public-coin IOPs generalizes to interactive arguments based on public-query IOPs with an additional property: the existence of a *random continuation sampler (RCS)*. More precisely, we show that the existence of an RCS for $\mathsf{Finale}[\mathsf{IOP},\mathsf{VC}]$ is equivalent to its existence for the underlying IOP; and that having an RCS for $\mathsf{Finale}[\mathsf{IOP},\mathsf{VC}]$ is a sufficient condition for the (natural extension of the) public-coin security reduction to hold.

Theorem 6.1. *Consider these three ingredients:*

- $\mathsf{IOP} = (\mathbf{P},\mathbf{V})$*, a public-query IOP system for a relation* R *with round complexity* k*, alphabet* Σ*, proof length* l*, and query complexity* q*; moreover,* IOP *has an IOP random continuation sampler* $\mathbf{S}$ *with error* α *and running time* $t_{\mathbf{S}}$*; and*
- $\mathsf{VC} = (\mathsf{Gen},\mathsf{Commit},\mathsf{Open},\mathsf{Check})$*, a vector commitment scheme over alphabet* Σ*.*

The $\mathsf{ARG} = (\mathcal{G},\mathcal{P},\mathcal{V}) := \mathsf{Finale}[\mathsf{IOP},\mathsf{VC}]$ *(Construction 6.2) is a* $4\mathsf{k}$*-message interactive argument system for* R *which, for every* $\epsilon > 0$ *and* $t_{\mathsf{ARG}} \geq t_{\mathbf{V}} + t_{\mathsf{VC.Check}} + \log|\Sigma| + \log \mathsf{l}_{\mathsf{max}}$*, has soundness error* ϵ_{ARG} *and knowledge soundness error* κ_{ARG} *satisfying the following:*

$$\epsilon_{\mathsf{ARG}}(\lambda,n,t_{\mathsf{ARG}}) \leq \epsilon_{\mathsf{IOP}}(n) + \epsilon_{\mathsf{VC}}\left(\lambda,\mathsf{l}_{\mathsf{max}},\mathsf{q}_{\mathsf{max}},t_{\mathsf{VC}}\right) + \epsilon + \frac{\mathsf{l}\cdot\alpha}{\epsilon} \quad \textit{and}$$
$$\kappa_{\mathsf{ARG}}(\lambda,n,t_{\mathsf{ARG}}) \leq \kappa_{\mathsf{IOP}}(n) + \epsilon_{\mathsf{VC}}\left(\lambda,\mathsf{l}_{\mathsf{max}},\mathsf{q}_{\mathsf{max}},t_{\mathsf{VC}}\right) + \epsilon + \frac{\mathsf{l}\cdot\alpha}{\epsilon} \quad ,$$

where $t_{\mathsf{VC}} = O\left(\frac{\mathsf{k}\cdot\mathsf{l}}{\epsilon}\cdot(t_{\mathsf{ARG}}+t_{\mathbf{S}})\right)$*. Moreover, the knowledge extractor runs in time* $t_{\mathcal{E}} = t_{\mathbf{E}} + O(t_{\mathsf{VC}})$*.*

Observe that if IOP admits an IOP random continuation sampler with error $\alpha = 0$, we obtain the same security bounds as in Theorem 5.1; conversely, if $\alpha > 0$, then setting $\epsilon = \sqrt{\alpha\cdot\mathsf{l}}$ achieves the minimum additional error of 2ϵ. Note, moreover, that ARG is public-coin if IOP is public-coin.

6.1 Construction

Below, we describe the construction of the interactive argument we denote $(\mathcal{P},\mathcal{V}) := \mathsf{Finale}[\mathsf{IOP},\mathsf{VC}]$.

Construction 6.2 (Arguments from public-query IOPs)**.** The argument generator $\mathcal{G}$ receives as input a security parameter $\lambda \in \mathbb{N}$ and an instance size bound $n \in \mathbb{N}$, and works as follows.

$\mathcal{G}(\lambda,n)$:
1. Sample public parameter for the VC scheme: $\mathsf{pp}_{\mathsf{VC}} \leftarrow \mathsf{VC.Gen}\left(1^{\lambda},\mathsf{l}_{\mathsf{max}}(n)\right)$.
2. Set public parameter for the interactive argument: $\mathsf{pp} := \mathsf{pp}_{\mathsf{VC}}$.
3. Output pp.

The argument prover $\mathcal{P}$ receives as input an instance $\mathbb{x}$ and a witness $\mathbb{w}$, and the argument verifier $\mathcal{V}$ receives as input the instance $\mathbb{x}$. Then $\mathcal{P}$ and $\mathcal{V}$ interact as follows.

1. $\mathcal{V}$'s randomness: Sample $\rho \leftarrow \{0,1\}^{\mathsf{r}}$.
2. $\mathcal{P}$'s commitments.
   For $i \in [\mathsf{k}]$:
   (a) $\mathcal{P}$'s i-th commitment.

i. Compute the i-th IOP string and auxiliary state:

$$(\Pi_i, \mathbf{aux}_i) \leftarrow \begin{cases} \mathbf{P}(\mathbb{x}, \mathbb{w}) & \text{if } i = 1 \\ \mathbf{P}(\mathbf{aux}_{i-1}, m_{i-1}) & \text{if } i > 1 \end{cases} .$$

ii. Compute a VC commitment to the IOP string: $(\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \mathsf{VC.Commit}(\mathsf{pp}, \Pi_i)$.

iii. Send $(\mathsf{cm}_i, \mathsf{aux}_i)$ to $\mathcal{V}$.

(b) $\mathcal{V}$'s i-th query.

i. Compute the i-th IOP verifier query set $\mathbf{Q}_i := \mathbf{V}_{\mathsf{q}}\big(\mathbb{x}, \rho, (\mathbf{ans}_j)_{j<i}\big)$.

ii. Send $\mathbf{Q}_i$ to $\mathcal{P}$.

(c) $\mathcal{P}$'s i-th response.

i. Parse $\mathbf{Q}_i$ as $(\mathcal{Q}_{i,1}, \dots, \mathcal{Q}_{i,i})$, where $\mathcal{Q}_i \subseteq [\mathsf{l}_i]$ is the query set of $\mathbf{V}$ to Π_i.

ii. For every $j \in [i]$, compute $\mathsf{pf}_{i,j} \leftarrow \mathsf{VC.Open}(\mathsf{pp}, \mathsf{aux}_j, \mathcal{Q}_{i,j})$ and set $\mathsf{ans}_{i,j} := \Pi_j[\mathcal{Q}_{i,j}]$.

iii. Set $\mathbf{ans}_i := (\mathsf{ans}_{i,j})_{j\in[i]}$ and $\mathbf{pf}_i := (\mathsf{pf}_{i,j})_{j\in[i]}$.

iv. Send $(\mathbf{ans}_i, \mathbf{pf}_i)$ to $\mathcal{V}$.

(d) $\mathcal{V}$'s i-th message.

i. Compute the i-th IOP verifier message $m_i := \mathbf{V}_{\mathsf{m}}\big(\mathbb{x}, \rho, (\mathbf{ans}_j)_{j\in[i]}\big)$.

ii. Send m_i to $\mathcal{P}$.

3. $\mathcal{V}$'s decision: Check if the following hold:

(a) $\mathbf{V}_{\mathsf{d}}\big(\mathbb{x}, \rho, (\mathbf{ans}_i)_{i\in[\mathsf{k}]}\big) = 1$.

(b) For every $i \in [\mathsf{k}]$, $\bigwedge_{j\le i} \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}_j, \mathcal{Q}_{i,j}, \mathsf{ans}_{i,j}, \mathsf{pf}_{i,j})$.

Above, we use $\mathbf{V}_{\mathsf{q}}$, $\mathbf{V}_{\mathsf{m}}$ and $\mathbf{V}_{\mathsf{d}}$ to denote the query, message and decision algorithms, respectively, for the IOP verifier $\mathbf{V}$.

The protocol has $2\mathsf{k}$ rounds: Each round of the IOP needs two rounds to simulate because $\mathbf{V}$'s message in the i-th round depends on the answers to the queries in the i-th round. We comment on the protocol's efficiency measures:

- the generator communication to prover and verifier consists of $|\mathsf{pp}_{\mathsf{VC}}|$ bits;
- the prover-to-verifier communication consists of $\sum_{i\in[\mathsf{k}]}(|\mathsf{cm}_i| + \mathsf{q} \cdot (\log \mathsf{l}_i + \log|\Sigma|) + \sum_{j\le i}|\mathsf{pf}_{i,j}|)$ bits;
- the verifier-to-prover communication consists of $\sum_{i\in[\mathsf{k}]}|m_i| + \mathsf{q} \cdot \log \mathsf{l}_i$ bits;
- the time complexity of the argument generator is $t_{\mathsf{VC.Gen}}$.
- the time complexity of the argument prover is $t_{\mathbf{P}} + \mathsf{k} \cdot t_{\mathsf{VC.Commit}} + \frac{\mathsf{k}\cdot(\mathsf{k}+1)}{2} \cdot t_{\mathsf{VC.Open}}$;
- the time complexity of the argument verifier is $t_{\mathbf{V}} + \frac{\mathsf{k}\cdot(\mathsf{k}+1)}{2} \cdot t_{\mathsf{VC.Check}}$.

6.2 Random continuation samplers

Definition 6.3 (IOP transcript and partial execution). *An* **IOP (interaction) transcript $\mathbf{tr}$ for a public-query IOP** *has the following form:*

$$\mathbf{tr} := \big((\mathbf{Q}_i, \mathbf{ans}_i, m_i)\big)_{i\in[\mathsf{k}]} .$$

For every $i \in [\mathsf{k}]$, a **partial IOP transcript $\mathbf{tr}_i$ for a public-query IOP** *has one of the following forms:*

- $\mathbf{tr}_i := \big(((\mathbf{Q}_j, \mathbf{ans}_j, m_j))_{j<i}, (\mathbf{Q}_i, \mathbf{ans}_i)\big)$*;*
- $\mathbf{tr}_i := \big(((\mathbf{Q}_j, \mathbf{ans}_j, m_j))_{j\le i}\big)$ *for $i < \mathsf{k}$.*

For every IOP prover $\widetilde{\mathbf{P}}$*, auxiliary input* $\mathbf{ai}$*, verifier randomness* $\rho \in \{0,1\}^{\mathsf{r}}$*, and integer* $i \in [\mathsf{k}]$*, the* i**-round partial execution of the IOP with respect to** $(\widetilde{\mathbf{P}}(\mathbf{ai}), \rho)$*, denoted by* $\langle \widetilde{\mathbf{P}}(\mathbf{ai}), \mathbf{V}(\rho) \rangle_i$*, is*

$$\langle \widetilde{\mathbf{P}}(\mathbf{ai}), \mathbf{V}(\rho) \rangle_i := (\mathbb{x}, \mathbf{tr}_i) \ ,$$

where $\mathbb{x}$ *and the partial IOP transcript* $\mathbf{tr}_i = \big(((\mathbf{Q}_j, \mathbf{ans}_j, m_j))_{j<i}, (\mathbf{Q}_i, \mathbf{ans}_i) \big)$ *are obtained via the experiment*

$$\left[\begin{array}{l} (\mathbb{x}, \mathbf{aux}_0) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \\ (\mathbf{Q}_0, m_0) := (\bot, \bot) \\ \textit{For } 1 \leq j < i : \\ \quad (\widetilde{\Pi}_j, \mathbf{aux}_j) \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux}_{j-1}, \mathbf{Q}_{j-1}, m_{j-1}) \\ \quad \mathbf{Q}_j \leftarrow \mathbf{V}\big(\mathbb{x}, \rho, (\mathbf{ans}_\ell)_{\ell<j}\big) \\ \quad \mathbf{ans}_j := \mathsf{GetAnswers}\big((\widetilde{\Pi}_\ell)_{\ell \leq j}, \mathbf{Q}_j\big) \\ \quad m_j \leftarrow \mathbf{V}\big(\mathbb{x}, \rho, (\mathbf{ans}_\ell)_{\ell \leq j}\big) \\ (\widetilde{\Pi}_i, \mathbf{aux}_i) \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux}_{i-1}, \mathbf{Q}_{i-1}, m_{i-1}) \\ \mathbf{Q}_i \leftarrow \mathbf{V}\big(\mathbb{x}, \rho, (\mathbf{ans}_j)_{j<i}\big) \\ \mathbf{ans}_i := \mathsf{GetAnswers}\big((\widetilde{\Pi}_j)_{j \leq i}, \mathbf{Q}_i\big) \end{array} \right] ,$$

and $\mathsf{GetAnswers}\big((\Pi_j)_{j\in[i]}, \mathbf{Q}_i\big)$ *denotes* $\big((\Pi_j[\mathcal{Q}_{i,j}])\big)_{j\in[i]}$ *where* $\big((\mathcal{Q}_{i,j})\big)_{j\in[i]} = \mathbf{Q}_i$*.*

Definition 6.4 (ARG transcript and partial execution)**.** *An* **ARG (interaction) transcript** tr **for a public-query IOP based argument system** *has the following form:*

$$\mathsf{tr} := \big((\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i, m_i)\big)_{i\in[\mathsf{k}]} \ .$$

Similarly, for every $i \in [\mathsf{k}]$*, a* **partial ARG transcript** tr_i *has one of the following forms:*

- $\mathsf{tr}_i := \big(((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j))_{j<i}, (\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i) \big)$*;*
- $\mathsf{tr}_i := \big(((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j))_{j\leq i} \big)$ *for* $i < \mathsf{k}$.

For all public parameters pp*, argument prover* $\widetilde{\mathcal{P}}$*, auxiliary input* $\mathbf{ai}$*, verifier randomness* $\rho \in \{0,1\}^{\mathsf{r}}$*, and integer* $i \in [\mathsf{k}]$*, the* i**-round partial execution of** ARG **with respect to** $(\mathsf{pp}, \widetilde{\mathcal{P}}(\mathsf{ai}), \rho)$*, denoted by* $\langle \widetilde{\mathcal{P}}(\mathsf{ai}), \mathcal{V}(\rho) \rangle_i$*, is*

$$\langle \widetilde{\mathcal{P}}(\mathsf{ai}), \mathcal{V}(\rho) \rangle_i := (\mathbb{x}, \mathsf{tr}_i) \ ,$$

where $\mathbb{x}$ *and the partial ARG transcript* $\mathsf{tr}_i = \big(((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j))_{j<i}, (\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i) \big)$ *are obtained via the experiment*

$$\left[\begin{array}{l} (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{aux}) \\ (\mathbf{Q}_0, m_0) := (\bot, \bot) \\ \textit{For } 1 \leq j < i : \\ \quad (\mathsf{cm}_j, \mathsf{aux}_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{j-1}, \mathbf{Q}_{j-1}, m_{j-1}) \\ \quad \mathbf{Q}_j \leftarrow \mathcal{V}(\mathbb{x}, \rho, (\mathbf{ans}_\ell)_{\ell<j}) \\ \quad (\mathbf{ans}_j, \mathbf{pf}_j, \mathsf{aux}_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_j, \mathbf{Q}_j) \\ \quad m_j \leftarrow \mathcal{V}(\mathbb{x}, \rho, (\mathbf{ans}_\ell)_{\ell\in[j]}) \\ (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \mathbf{Q}_{i-1}, m_{i-1}) \\ \mathbf{Q}_i \leftarrow \mathcal{V}(\mathbb{x}, \rho, (\mathbf{ans}_j)_{j<i}) \\ (\mathbf{ans}_i, \mathbf{pf}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \mathbf{Q}_i) \end{array} \right] .$$

Definition 6.5. *For every IOP prover $\widetilde{\mathbf{P}}$ and IOP prover auxiliary input* $\mathbf{ai}$*, a partial IOP execution* $(\mathbb{x}, \mathbf{tr}_i)$ *has* **non-zero measure with respect to $\widetilde{\mathbf{P}}(\mathbf{ai})$** *if*

$$\Pr\left[(\mathbb{x}, \mathbf{tr}_i) = \left\langle \widetilde{\mathbf{P}}(\mathbf{ai}), \mathbf{V}(\rho) \right\rangle_i \;\middle|\; \rho \leftarrow \{0,1\}^{\mathsf{r}} \right] > 0 \ .$$

A partial execution for ARG *with non-zero measure is defined similarly.*

Definition 6.6. *Consider* $\mathsf{IOP} = (\mathbf{P}, \mathbf{V})$ *with randomness complexity* r*. For any partial IOP execution* $(\mathbb{x}, \mathbf{tr}_i)$*, we define* $\mathbf{R}_{(\mathbb{x}, \mathbf{tr}_i)}$*,* **the set of random strings consistent with $(\mathbb{x}, \mathbf{tr})$***, as follows:*

$$\mathbf{R}_{(\mathbb{x}, \mathbf{tr}_i)} := \left\{ \rho \in \{0,1\}^{\mathsf{r}} : \left\langle \widetilde{\mathbf{P}}(\mathbf{ai}), \mathbf{V}(\rho) \right\rangle_i = \mathbf{tr}_i \right\} \ .$$

For a partial argument execution tr_i*, we define* $\mathbf{R}_{(\mathbb{x}, \mathsf{tr}_i)}$ *analogously.*

For simplicity, we often drop the instance $\mathbb{x}$ (which is fixed by $\widetilde{\mathbf{P}}(\mathbf{aux})$) and refer to a partial transcript $\mathbf{tr}_i$ with non-zero measure, or a set $\mathbf{R}_{\mathbf{tr}_i}$ of random strings consistent with $\mathbf{tr}_i$.

Definition 6.7 (IOP random continuation sampler)**.** *Let* IOP *be a public-query IOP with round complexity* k*, and let* $\mathbf{S}$ *be a probabilistic algorithm with the following syntax:*

- *On input* $(\mathbb{x}, \mathbf{tr}_i)$ *where* $\mathbf{tr}_i = \left(\left((\mathbf{Q}_j, \mathbf{ans}_j, m_j) \right)_{j<i} \right)$ *for* $i \in [\mathsf{k}]$*,* $\mathbf{S}$ *outputs a query set* $\mathbf{Q}_i$*;*
- *On input* $(\mathbb{x}, \mathbf{tr}_i)$ *where* $\mathbf{tr}_i = \left(\left((\mathbf{Q}_j, \mathbf{ans}_j, m_j) \right)_{j<i}, (\mathbf{Q}_i, \mathbf{ans}_i) \right)$*,* $\mathbf{S}$ *outputs a message* m_i*.*

Given $\alpha \geq 0$ *and* $t_{\mathbf{S}} \in \mathbb{N}$*, we say* $\mathbf{S}$ *is an* **IOP random continuation sampler (IOP-RCS) with error α and running time $t_{\mathbf{S}}$** *if it satisfies the following guarantee: for every size bound* t_{IOP}*,* t_{IOP}*-size circuit* $\widetilde{\mathbf{P}}$*, IOP prover auxiliary input* $\mathbf{ai}$*,* $i \in [\mathsf{k}]$*, and partial IOP execution* $(\mathbb{x}, \mathbf{tr}_i)$ *that has non-zero measure with respect to* $(\widetilde{\mathbf{P}}, \mathbf{ai})$*, we have*

$$\Delta\left(\mathcal{D}\left[\widetilde{\mathbf{P}}(\mathbf{ai}), \mathbf{S}(\mathbb{x}, \mathbf{tr}_i) \right], \mathcal{U}\left[\widetilde{\mathbf{P}}(\mathbf{ai}), \mathbf{R}_{(\mathbb{x}, \mathbf{tr}_i)} \right] \right) = \alpha \ ,$$

where

- $\Delta(\cdot, \cdot)$ *is the statistical (total variation) distance between two distributions;*
- $\mathcal{D}\left[\widetilde{\mathbf{P}}(\mathbf{ai}), \mathbf{S}(\mathbb{x}, \mathbf{tr}_i) \right]$ *is the distribution of IOP transcripts produced by* $\mathbf{S}$*:*

$$\mathcal{D}\left[\widetilde{\mathbf{P}}(\mathbf{ai}), \mathbf{S}(\mathbb{x}, \mathbf{tr}_i) \right] := \left[\left((\mathbf{Q}_j, \mathbf{ans}_j, m_j) \right)_{j \in [\mathsf{k}]} \;\middle|\; \begin{array}{l} (\mathbb{x}, \mathbf{aux}_0) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \\ (\mathbf{Q}_0, m_0) := (\bot, \bot) \\ \textit{For } 1 \leq j \leq i : \\ \quad (\widetilde{\Pi}_j, \mathbf{aux}_j) \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux}_{j-1}, \mathbf{Q}_{j-1}, m_{j-1}) \\ m_i \leftarrow \mathbf{S}(\mathbb{x}, \mathbf{tr}_i) \\ \textit{For } i < j \leq \mathsf{k} : \\ \quad (\widetilde{\Pi}_j, \mathbf{aux}_j) \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux}_{j-1}, \mathbf{Q}_{j-1}, m_{j-1}) \\ \quad \mathbf{Q}_j \leftarrow \mathbf{S}\big(\mathbb{x}, \big(((\mathbf{Q}_\ell, \mathbf{ans}_\ell, m_\ell))_{\ell<j} \big)\big) \\ \quad \mathbf{ans}_j := \mathsf{GetAnswers}\big((\widetilde{\Pi}_\ell)_{\ell \leq j}, \mathbf{Q}_j\big) \\ \quad m_j \leftarrow \mathbf{S}\big(\mathbb{x}, \big(((\mathbf{Q}_\ell, \mathbf{ans}_\ell, m_\ell))_{\ell<j}, (\mathbf{Q}_j, \mathbf{ans}_j) \big)\big) \end{array} \right] ;$$

- $\mathcal{U}\Big[\widetilde{\mathbf{P}}(\mathbf{ai}), \mathbf{R}_{(\mathbb{x},\mathbf{tr}_i)}\Big]$ *is the uniform distribution over all complete IOP transcripts consistent with* $(\mathbb{x}, \mathbf{tr}_i)$*:*[33]

$$\mathcal{U}\Big[\widetilde{\mathbf{P}}(\mathbf{ai}), \mathbf{R}_{(\mathbb{x},\mathbf{tr}_i)}\Big] := \left[\big((\mathbf{Q}_j, \mathbf{ans}_j, m_j)\big)_{j\in[\mathsf{k}]} \;\middle|\; \begin{array}{l} (\mathbb{x}, \mathbf{aux}_0) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \\ \rho \leftarrow \mathbf{R}_{(\mathbb{x},\mathbf{tr}_i)} \\ (\mathbf{Q}_0, m_0) := (\bot, \bot) \\ \textit{For } j \in [\mathsf{k}] : \\ \quad (\widetilde{\Pi}_j, \mathbf{aux}_j) \leftarrow \widetilde{\mathbf{P}}(\mathbf{aux}_{j-1}, \mathbf{Q}_{j-1}, m_{j-1}) \\ \quad \mathbf{Q}_j \leftarrow \mathbf{V}_{\mathsf{q}}\big(\mathbb{x}, \rho, (\mathbf{ans}_\ell)_{\ell<j}\big) \\ \quad \mathbf{ans}_j := \mathsf{GetAnswers}\big((\widetilde{\Pi}_\ell)_{\ell\in[j]}, \mathbf{Q}_j\big) \\ \quad m_j \leftarrow \mathbf{V}_{\mathsf{m}}\big(\mathbb{x}, \rho, (\mathbf{ans}_\ell)_{\ell\in[j]}\big) \end{array} \right] .$$

Definition 6.8 (ARG random continuation sampler)**.** *Let* ARG *be an argument system with round complexity* $2\mathsf{k}$ *that follows the same message structure as defined in Construction 6.2, and let* $\mathcal{S}$ *be a probabilistic algorithm with the following syntax.*

- *On input* $(\mathsf{pp}, \mathbb{x}, \mathsf{tr}_i)$ *where* $\mathsf{tr}_i = \big(\big((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j<i}, \mathsf{cm}_i\big)$ *for* $i \in [\mathsf{k}]$*,* $\mathcal{S}$ *outputs* $\mathbf{Q}_{i+1}$*;*
- *On input* $(\mathsf{pp}, \mathbb{x}, \mathsf{tr}_i)$ *where* $\mathsf{tr}_i := \big(\big((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j<i}, (\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i)\big)$*,* $\mathcal{S}$ *outputs* m_i*.*

Given $\alpha \geq 0$ *and* $t_{\mathcal{S}} \in \mathbb{N}$*,* $\mathcal{S}$ *is an* **ARG random continuation sampler (ARG-RCS) with error** α **and running time** $t_{\mathcal{S}}$ *for* ARG *if it satisfies the following guarantee: for every size bound* t_{ARG}*,* t_{ARG}*-size circuit* $\widetilde{\mathcal{P}}$*, ARG prover auxiliary input* ai*, public parameter* pp*,* $i \in [\mathsf{k}]$*, and partial ARG execution* $(\mathbb{x}, \mathsf{tr}_i)$ *that has non-zero measure with respect to* $\widetilde{\mathcal{P}}(\mathsf{ai})$*, we have*

$$\Delta\left(\mathcal{D}\Big[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathcal{S}(\mathsf{pp}, \mathbb{x}, \mathsf{tr}_i)\Big], \mathcal{U}\Big[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathbf{R}_{(\mathbb{x},\mathsf{tr}_i)}\Big]\right) = \alpha \;,$$

where

- $\mathcal{D}\Big[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathcal{S}(\mathsf{pp}, \mathbb{x}, \mathsf{tr}_i)\Big]$ *is a distribution of ARG transcripts produced by* $\mathcal{S}$*:*

$$\mathcal{D}\Big[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathcal{S}(\mathsf{pp}, \mathbb{x}, \mathsf{tr}_i)\Big]$$

$$:= \left[\big((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j\in[\mathsf{k}]} \;\middle|\; \begin{array}{l} (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{aux}) \\ (\mathbf{Q}_0, m_0) := (\bot, \bot) \\ \textit{For } 1 \leq j \leq i : \\ \quad (\mathsf{cm}_j, \mathsf{aux}_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{j-1}, \mathbf{Q}_{j-1}, m_{j-1}) \\ m_i \leftarrow \mathcal{S}(\mathsf{pp}, \mathbb{x}, \mathsf{tr}_i) \\ \textit{For } i < j \leq \mathsf{k} : \\ \quad (\mathsf{cm}_j, \mathsf{aux}_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{j-1}, \mathbf{Q}_{j-1}, m_{j-1}) \\ \quad \mathbf{Q}_j \leftarrow \mathcal{S}\big(\mathsf{pp}, \mathbb{x}, \big(((\mathsf{cm}_\ell, \mathbf{Q}_\ell, \mathbf{ans}_\ell, \mathbf{pf}_\ell, m_\ell))_{\ell<j}\big)\big) \\ \quad (\mathbf{ans}_j, \mathbf{pf}_j, \mathsf{aux}_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_j, \mathbf{Q}_j) \\ \quad m_j \leftarrow \mathcal{S}\big(\mathsf{pp}, \big(((\mathsf{cm}_\ell, \mathbf{Q}_\ell, \mathbf{ans}_\ell, \mathbf{pf}_\ell, m_\ell))_{\ell<j}, (\mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j)\big)\big) \end{array} \right] ;$$

[33]Note that we run the malicious IOP prover and the IOP verifier from scratch in the experiment. Alternatively, one may break it into two parts as in $\mathcal{D}\Big[\widetilde{\mathbf{P}}(\mathbf{ai}), \mathbf{S}(\mathbb{x}, \mathbf{tr}_i)\Big]$: rounds $1 \leq j \leq i$ and $i + 1 \leq j \leq \mathsf{k}$. We opt for the more concise notation.

- $\mathcal{U}\left[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathbf{R}_{(\mathbb{x},\mathsf{tr}_i)}\right]$ *is the uniform distribution over all complete ARG transcripts consistent with* tr_i*:*

$$\mathcal{U}\left[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathbf{R}_{(\mathbb{x},\mathsf{tr}_i)}\right] := \left[\big((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j\in[\mathsf{k}]} \;\middle|\; \begin{array}{l} (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{aux}) \\ \rho \leftarrow \mathbf{R}_{(\mathbb{x},\mathsf{tr}_i)} \\ (\mathbf{Q}_0, m_0) := (\bot, \bot) \\ \textit{For } j \in [\mathsf{k}]: \\ \quad (\mathsf{cm}_j, \mathsf{aux}_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{j-1}, \mathbf{Q}_{j-1}, m_{j-1}) \\ \quad \mathbf{Q}_j \leftarrow \mathcal{V}(\mathbb{x}, \rho, (\mathbf{ans}_\ell)_{\ell<j}) \\ \quad (\mathbf{ans}_j, \mathbf{pf}_j, \mathsf{aux}_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_j, \mathbf{Q}_j) \\ \quad m_j \leftarrow \mathcal{V}(\mathbb{x}, \rho, (\mathbf{ans}_\ell)_{\ell\in[j]}) \end{array} \right].$$

6.3 Equivalence of IOP-RCS and ARG-RCS

We now show that an IOP-RCS exists if and only if a corresponding ARG-RCS exists.

Lemma 6.9 (IOP sampler to ARG sampler)**.** *For every $\alpha \geq 0$ and $t_{\mathbf{S}} \in \mathbb{N}$, if* IOP *has an IOP-RCS with error α and running time $t_{\mathbf{S}}$, then* $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ *admits an ARG-RCS $\mathcal{S}$ with error α and running time $t_{\mathcal{S}} = O(t_{\mathbf{S}})$.*

Proof. We first define the ARG random continuation sampler $\mathcal{S}$ from its IOP analogue $\mathbf{S}$ the natural way:

$\mathcal{S}(\mathsf{pp}, \mathbb{x}, \mathsf{tr}_i)$:
1. Set $\mathbf{tr}_i := \big(((\mathbf{Q}_j, \mathbf{ans}_j, m_j))_{j<i}\big)$ if $\mathsf{tr}_i = \big(((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j))_{j<i}, \mathsf{cm}_i\big)$.
2. Set $\mathbf{tr}_i := \big(((\mathbf{Q}_j, \mathbf{ans}_j, m_j))_{j<i}, (\mathbf{Q}_i, \mathbf{ans}_i)\big)$ if $\mathsf{tr}_i = \big(((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j))_{j<i}, (\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i)\big)$.
3. Output $\mathbf{S}(\mathbb{x}, \mathbf{tr}_i)$.

It is clear from construction that $\mathcal{S}$ has running time $O(t_{\mathbf{S}})$. Moreover, since the argument verifier's messages in Construction 6.2 *only depend on the query sets and IOP prover answers* ($\mathcal{V}$ replies with the messages of $\mathbf{V}$; an invalid opening interferes with the decision, but not the communication), the error of $\mathcal{S}$ is exactly α, the error of $\mathbf{S}$. □

We now show that given an ARG random continuation sampler, we can construct an IOP random continuation sampler.

Lemma 6.10 (ARG sampler to IOP sampler)**.** *For every $\alpha \geq 0$ and $t_{\mathcal{S}} \in \mathbb{N}$, if* $\mathsf{Finale}[\mathsf{IOP}, \mathsf{VC}]$ *has an ARG-RCS with error α and running time $t_{\mathcal{S}}$, then* IOP *admits an IOP-RCS* $\mathbf{S}$ *with error α and running time $t_{\mathbf{S}} = O(t_{\mathcal{S}})$.*

Proof. Note that the apparent difficulty is caused by the fact that IOP transcripts do not contain vector commitments nor VC openings, but to run the ARG-RCS, one needs to supply the commitments and openings as inputs.

However, we observe (as we did above) that the argument verifier's messages in Construction 6.2 are independent of the commitments and VC openings. Therefore, these are only syntactic requirements that an arbitrary sequence of strings can fulfil; in particular, it suffices for the ARG-RCS to receive empty strings as commitments and VC openings:

$\mathbf{S}(\mathbb{x}, \mathbf{tr}_i)$:
1. Set $\mathsf{tr}_i := \big(((\bot, \mathbf{Q}_j, \mathbf{ans}_j, \bot, m_j))_{j<i}, \bot\big)$ if $\mathbf{tr}_i = \big(((\mathbf{Q}_j, \mathbf{ans}_j, m_j))_{j<i}\big)$.

2. Set $\mathsf{tr}_i := \left(\left((\bot, \mathbf{Q}_j, \mathbf{ans}_j, \bot, m_j)\right)_{j<i}, (\bot, \mathbf{Q}_i, \mathbf{ans}_i, \bot)\right)$ if $\mathbf{tr}_i = \left(\left((\mathbf{Q}_j, \mathbf{ans}_j, m_j)\right)_{j<i}, (\mathbf{Q}_i, \mathbf{ans}_i)\right)$.
3. Output $\mathcal{S}(\bot, \mathbb{x}, \mathsf{tr}_i)$.

It is again clear from construction and definition of $\mathcal{S}$ that $\mathbf{S}$ has running time $O(t_{\mathcal{S}})$ and error α. $\square$

6.4 Security reduction

We show that given a random continuation sampler for IOP (or, equivalently, a random continuation sampler for ARG), a security reduction claim similar to Lemma 5.4 can be shown.

Lemma 6.11. *Assume there is an ARG-RCS $\mathcal{S}$ with error α and running time $t_{\mathcal{S}}$. There exist a probabilistic algorithms $\mathcal{R}$ which, for every $\epsilon > 0$, size bound $t_{\mathsf{ARG}} \geq t_{\mathbf{V}} + t_{\mathsf{VC.Check}} + \log|\Sigma| + \log \mathsf{l}_{\mathsf{max}}$, and t_{ARG}-size circuit $\widetilde{\mathcal{P}}$, satisfy*

$$\Pr\left[\begin{array}{l} \mathbf{V}^{([\widetilde{\mathcal{Q}}_i,\widetilde{\Pi}_i])_{i\in[\mathsf{k}]}}(\mathbb{x};\rho) \neq 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i,\mathsf{ans}_i])_{i\in[\mathsf{k}]}}(\mathbb{x};\rho) = 1 \\ \wedge\, \left(\bigwedge_{i\in[\mathsf{k}]}\left(\bigwedge_{j\leq i}\mathsf{VC.Check}(\mathsf{pp},\mathsf{cm}_j,\mathcal{Q}_{i,j},\mathsf{ans}_{i,j},\mathsf{pf}_{i,j})\right)\right) \end{array}\right]$$
$$\leq \epsilon_{\mathsf{VC}}\left(\lambda, \mathsf{l}_{\mathsf{max}}, \mathsf{q}_{\mathsf{max}}, t_{\mathsf{VC}}\right) + \epsilon + \frac{\mathsf{l}\cdot\alpha}{\epsilon} \ ,$$

with respect to

$$\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ \left((\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i, m_i)\right)_{i\in[\mathsf{k}]} \leftarrow \left\langle \widetilde{\mathcal{P}}(\mathsf{aux}_0), \mathcal{V}(\mathbb{x};\rho)\right\rangle \\ \textit{For } i \in [\mathsf{k}]: \\ \quad (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathbb{x}, \mathsf{aux}_{i-1}, \mathbf{Q}_{i-1}, m_{i-1}) \\ \quad (\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)}\left(\mathsf{pp}, \mathbb{x}, (\mathsf{cm}_j)_{j\leq i}, \left((\mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\right)_{j<i}, \epsilon\right) \end{array}\right] \tag{5}$$

where $t_{\mathsf{VC}} = O\left(\frac{\mathsf{k}\cdot\mathsf{l}}{\epsilon}\cdot(t_{\mathsf{ARG}} + t_{\mathbf{S}})\right)$.

The reductor is defined as follows:

- $\mathcal{R}_{\mathrm{post}}(\mathcal{K}, \mathsf{l})$:
  1. Initialize $\widetilde{\Pi} := (\sigma)^{\mathsf{l}}$, where σ is an arbitrary symbol in Σ.
  2. For all $(\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \in \mathcal{K}$ and $q \in \mathcal{Q}$: Set $\widetilde{\Pi}[q] := \mathsf{ans}[q]$.
  3. Set $\widetilde{\mathcal{Q}} := \bigcup_{(\mathcal{Q},\mathsf{ans})\in\mathcal{K}} \mathcal{Q}$.
  4. Output $(\widetilde{\mathcal{Q}}, \widetilde{\Pi})$.
- $\mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)}\left(\mathsf{pp}, \mathbb{x}, (\mathsf{cm}_j)_{j\leq i}, \left((\mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\right)_{j<i}, \epsilon\right)$:[34]
  1. Initialize $\mathcal{K} = \emptyset$.
  2. Set $\mathsf{N} := \frac{\mathsf{l}}{\epsilon}$ and $\mathsf{cm}'_i := \mathsf{cm}_i$.

[34]Note that, when $i = 1$ (i.e., when the second argument is $\bot$) the reductor simply samples ρ and applies the functions $\mathbf{V}_{\mathsf{q}}$ and $\mathbf{V}_{\mathsf{m}}$ as appropriate; there is no need for a sampler in this case. Equivalently, we assume $\mathcal{S}$ performs this trivial sampling when $i = 1$ and is only nontrivial when $i > 1$.

```
3. Repeat the following N times:
   (a) Run $\mathbf{Q}'_i \leftarrow \mathcal{S}\big(\mathsf{pp}, \mathbb{x}, \big(\big((\mathsf{cm}_\ell, \mathbf{Q}_\ell, \mathbf{ans}_\ell, \mathbf{pf}_\ell, m_\ell)\big)_{\ell<i}, \mathsf{cm}_i\big)\big)$.
   (b) Run $(\mathbf{ans}'_i, \mathbf{pf}'_i, \mathsf{aux}'_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_i, \mathbf{Q}_i)$.
   (c) Run $m'_i \leftarrow \mathcal{S}\big(\mathsf{pp}, \mathbb{x}, \big(\big((\mathsf{cm}_\ell, \mathbf{Q}_\ell, \mathbf{ans}_\ell, \mathbf{pf}_\ell, m_\ell)\big)_{\ell<i}, (\mathbf{Q}'_i, \mathbf{ans}'_i, \mathbf{pf}'_i)\big)\big)$.
   (d) For $i < j \le \mathsf{k}$:
       i. Run $(\mathsf{cm}'_j, \mathsf{aux}'_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}'_{j-1}, \mathbf{Q}'_{j-1}, m'_{j-1})$.
       ii. Run $\mathbf{Q}'_j \leftarrow \mathcal{S}\big(\mathsf{pp}, \mathbb{x}, \big(\big((\mathsf{cm}_\ell, \mathbf{Q}_\ell, \mathbf{ans}_\ell, \mathbf{pf}_\ell, m_\ell)\big)_{\ell<i}, \big((\mathsf{cm}'_\ell, \mathbf{Q}'_\ell, \mathbf{ans}'_\ell, \mathbf{pf}'_\ell, m'_\ell)\big)_{i\le\ell<j}\big)\big)$.
       iii. Run $(\mathbf{ans}'_j, \mathbf{pf}'_j, \mathsf{aux}'_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}'_j, \mathbf{Q}'_j)$.
       iv. Run $m'_j \leftarrow \mathcal{S}\big(\mathsf{pp}, \mathbb{x}, \big(\big((\mathsf{cm}_\ell, \mathbf{Q}_\ell, \mathbf{ans}_\ell, \mathbf{pf}_\ell, m_\ell)\big)_{\ell<i}, \big((\mathsf{cm}'_\ell, \mathbf{Q}'_\ell, \mathbf{ans}'_\ell, \mathbf{pf}'_\ell, m'_\ell)\big)_{i\le\ell<j}, (\mathbf{Q}'_j, \mathsf{ans}'_j, \mathbf{pf}'_j)\big)\big)$.
       v. If $\mathsf{VC.Check}(\mathsf{cm}_i, \mathcal{Q}'_{j,i}, \mathsf{ans}'_{j,i}, \mathsf{pf}'_{j,i}) = 1$, add $(\mathcal{Q}'_{j,i}, \mathsf{ans}'_{j,i}, \mathsf{pf}'_{j,i})$ to $\mathcal{K}$.[35]
4. Output $(\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i) \leftarrow \mathcal{R}_{\text{post}}(\mathcal{K}, \mathsf{l}_i)$.
```

Proof. Throughout the proof, probabilistic expressions are with respect to the experiment in Eq. 5.

Our goal is to upper bound the following expression:

$$\Pr\left[\begin{array}{l} \mathbf{V}^{([\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i])_{i\in[\mathsf{k}]}}(\mathbb{x};\rho) \neq 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \mathsf{ans}_i])_{i\in[\mathsf{k}]}}(\mathbb{x};\rho) = 1 \\ \wedge\, \Big(\bigwedge_{i\in[\mathsf{k}]}\Big(\bigwedge_{j\le i}\mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}_j, \mathcal{Q}_{i,j}, \mathsf{ans}_{i,j}, \mathsf{pf}_{i,j})\Big)\Big) \end{array}\right] .$$

Similar to the proof of Lemma 5.9, the above even implies either there are valid openings with disagreeing answers or there is a missing query in one of the IOP strings. We analyze them separately.

Valid openings with disagreeing answers. We show that

$$\Pr\left[\begin{array}{l} \exists\, i \in [\mathsf{k}], j \in [i,\mathsf{k}], q \in \mathcal{Q}_{j,i} \cap \widetilde{\mathcal{Q}}_i \\ \wedge\, \widetilde{\Pi}_i[q] \neq \mathsf{ans}_{j,i}[q] \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_{j,i}, \mathsf{ans}_{j,i}, \mathsf{pf}_{j,i}) = 1 \end{array}\right] \le \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}_{\mathsf{max}}, \mathsf{q}_{\mathsf{max}}, t_{\mathsf{VC}})$$

where $t_{\mathsf{VC}} = O\left(\frac{\mathsf{k}\cdot\mathsf{l}}{\epsilon}\cdot(t_{\mathcal{S}} + t_{\mathsf{ARG}})\right)$.

We define a VC adversary A_{VC} as before:

```
$A_{\mathsf{VC}}(\mathsf{pp}, \mathsf{ai})$:
1. Run $(\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai})$.
2. Sample $\rho \leftarrow \{0,1\}^{\mathsf{r}}$.
3. Run $((\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i, m_i))_{i\in[\mathsf{k}]} \leftarrow \langle \widetilde{\mathcal{P}}(\mathsf{aux}_0), \mathcal{V}(\mathsf{pp}, \mathbb{x}, \rho)\rangle$.
4. For every $i \in [\mathsf{k}]$:
   (a) Run $(\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \mathbf{Q}_{i-1}, \mathbf{ans}_{i-1})$.
   (b) Run $\mathcal{K}_i \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)}\big(\mathsf{pp}, \mathbb{x}, (\mathsf{cm}_j)_{j\le i}, \big((\mathbf{Q}_j, \mathbf{ans}_j, m_j)\big)_{j<i}, \epsilon\big)$.
5. If there exists $i \in [\mathsf{k}]$, $j \in [i,\mathsf{k}]$, and $(\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \in \mathcal{K}_i$ with $q \in \mathcal{Q}_{j,i} \cap \mathcal{Q}$ and $\mathsf{ans}_{j,i}[q] \neq \mathsf{ans}[q]$, output
   $(\mathsf{cm}_i, \mathsf{ans}_{j,i}, \mathsf{ans}, \mathcal{Q}_{j,i}, \mathcal{Q}, \mathsf{pf}_{j,i}, \mathsf{pf})$.
6. Otherwise, output (the "dummy" tuple) $(\mathsf{cm}_1, \mathsf{ans}_1, \mathsf{ans}_1, \mathcal{Q}_1, \mathcal{Q}_1, \mathsf{pf}_1, \mathsf{pf}_1)$.
```

Note that in A_{VC}, we output $\mathcal{K}$ when invoking $\mathcal{R}$. This is a notational overload where we partially execute $\mathcal{R}$ without calling $\mathcal{R}_{\text{post}}$. The time complexity of A_{VC} is $O\left(\frac{\mathsf{k}\cdot\mathsf{l}}{\epsilon}\cdot(t_{\mathcal{S}} + t_{\mathsf{ARG}})\right)$. Therefore, according to

[35]Note the order of indices: in order to fill in the i-th IOP proof, the reductor uses the i-th query sets and answers of all rounds $j > i$ (i.e., all $\mathcal{Q}_{j,i}$ and $\mathsf{ans}_{j,i}$).

Definition 3.8,

$$\Pr\left[\begin{array}{l}\exists\, i \in [\mathsf{k}], j \in [i, \mathsf{k}], q \in \mathcal{Q}_{j,i} \cap \widetilde{\mathcal{Q}}_i \\ \wedge\, \widetilde{\Pi}_i[q] \neq \mathsf{ans}_{j,i}[q] \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_{j,i}, \mathsf{ans}_{j,i}, \mathsf{pf}_{j,i}) = 1\end{array}\right]$$

$$\leq \Pr\left[\begin{array}{l|l}|\mathcal{Q}| = |\mathcal{Q}'| \leq \mathsf{q} & \mathsf{pp} \leftarrow \mathsf{VC.Gen}(1^\lambda, \mathsf{l}_{\max}) \\ \wedge\, \exists\, q \in \mathcal{Q} \cap \mathcal{Q}' : \mathsf{ans}[q] \neq \mathsf{ans}'[q] & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1 & \begin{pmatrix}\mathsf{cm}, \mathsf{ans}, \mathsf{ans}', \\ \mathcal{Q}, \mathcal{Q}', \mathsf{pf}, \mathsf{pf}'\end{pmatrix} \leftarrow A_{\mathsf{VC}}(\mathsf{pp}) \\ \wedge\, \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}, \mathcal{Q}', \mathsf{ans}', \mathsf{pf}') = 1 & \end{array}\right]$$

$$\leq \epsilon_{\mathsf{VC}}(\lambda, \mathsf{l}_{\max}, \mathsf{q}_{\max}, t_{\mathsf{VC}}) \ .$$

Missing positions in $\widetilde{\Pi}_i$. We show that

$$\Pr\left[\exists\, i \in [\mathsf{k}], j \in [i, \mathsf{k}], q \in \mathcal{Q}_{j,i} \setminus \widetilde{\mathcal{Q}}_i\right] \leq \epsilon + 2\mathsf{l} \cdot \frac{\alpha}{\epsilon} \ .$$

For this case, we rely on the property of ARG-RCS. In particular, according to Definition 6.8, the ARG verifier's next message in tr_i sampled by $\mathcal{S}$ is statistically close to the "ideal" distribution of ARG verifier's next message, namely the ARG execution with respect to verifier randomness in $\mathbf{R}_{(\mathbb{x},\mathsf{tr}_i)}$.

We first define the following *ideal reductor* $\hat{\mathcal{R}}$ with respect to $\mathcal{U}\left[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathbf{R}_{(\mathbb{x},\mathsf{tr}_i)}\right]$. In particular, $\hat{\mathcal{R}}$ works as if there were a perfect ARG-RCS as in the public-coin case:

```
$\hat{\mathcal{R}}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)}\big(\mathsf{pp}, \mathbb{x}, (\mathsf{cm}_j)_{j\leq i}, \big((\mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j<i}, \epsilon\big)$:
1. Initialize $\mathcal{K} := \emptyset$ and set $\mathsf{N} := \frac{\mathsf{l}}{\epsilon}$.
2. Repeat the following $\mathsf{N}$ times:
   (a) Sample $\rho \leftarrow \mathbf{R}_{((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j))_{j<i}}$.
   (b) Run $\mathbf{Q}_i \leftarrow \mathcal{V}(\mathbb{x}, \rho, (\mathbf{ans}_j)_{j<i})$.
   (c) Run $(\mathbf{ans}_i, \mathbf{pf}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_i, \mathbf{Q}_i)$.
   (d) Run $m_i \leftarrow \mathcal{V}(\mathbb{x}, \rho, (\mathbf{ans}_j)_{j\in[i]})$.
   (e) For $i < j \leq \mathsf{k}$:
       i. Run $(\mathsf{cm}, \mathsf{aux}'_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{j-1}, m_{j-1})$.
       ii. Run $\mathbf{Q}_j \leftarrow \mathcal{V}(\mathbb{x}, \rho, (\mathbf{ans}_\ell)_{\ell<j})$.
       iii. Run $(\mathbf{ans}_j, \mathbf{pf}_j, \mathsf{aux}_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}'_j, \mathbf{Q}_j)$.
       iv. Run $m_j \leftarrow \mathcal{V}(\mathbb{x}, \rho, (\mathbf{ans}_\ell)_{\ell\in[j]})$.
   (f) If $\mathsf{VC.Check}(\mathsf{cm}_i, \mathcal{Q}_{j,i}, \mathsf{ans}_{j,i}, \mathsf{pf}_{j,i}) = 1$, add $(\mathcal{Q}_{j,i}, \mathsf{ans}_{j,i}, \mathsf{pf}_{j,i})$ to $\mathcal{K}$.
3. Output $(\hat{\mathcal{Q}}_i, \hat{\Pi}_i) \leftarrow \mathcal{R}_{\mathrm{post}}(\mathcal{K}, \mathsf{l}_i)$.
```

We can adapt the proof of Lemma 5.4 to bound the following probability:

$$\Pr\left[\begin{array}{l|l} & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ & \mathsf{ai} \leftarrow \mathcal{D} \\ & (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ & \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ \exists\, i \in [\mathsf{k}], j \in [i, \mathsf{k}], q \in \mathcal{Q}_{j,i} \setminus \hat{\mathcal{Q}}_i & ((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j))_{j\in[\mathsf{k}]} \leftarrow \left\langle \widetilde{\mathcal{P}}(\mathsf{aux}_0), \mathcal{V}(\mathsf{pp}, \mathbb{x}, \rho)\right\rangle \\ & \text{For } i \in [\mathsf{k}] : \\ & \quad (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{i-1}, \mathbf{Q}_{i-1}, m_{i-1}) \\ & \quad (\hat{\mathcal{Q}}_i, \hat{\Pi}_i) \leftarrow \hat{\mathcal{R}}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)}\big(\mathsf{pp}, \mathbb{x}, (\mathsf{cm}_j)_{j\leq i}, \big((\mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j<i}, \epsilon\big)\end{array}\right] . \quad (6)$$

Fix prover auxiliary input ai, public parameters pp and a partial transcript $\mathsf{tr}_i = \big((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j<i}$ that has non-zero measure with respect to $\widetilde{\mathcal{P}}(\mathsf{ai})$, we define the *weight* of a coordinate $q \in [\mathsf{l}_i]$ with respect to $(\mathsf{pp}, \mathsf{ai}, \mathsf{tr}_i)$ as

$$\delta_{(\mathsf{pp},\mathsf{ai},\mathsf{tr}_i)}(q) := \Pr\left[\begin{array}{l} \exists\, j \in [i, \mathsf{k}] : q \in \mathcal{Q}_{j,i} \\ \wedge \bigwedge_{j\in[\mathsf{k}]} \Big(\bigwedge_{\ell\in[j]} \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}_\ell, \mathcal{Q}_{j,\ell}, \mathsf{ans}_{j,\ell}, \mathsf{pf}_{j,\ell})\Big) \end{array} \middle| \begin{array}{l} \rho \leftarrow \mathbf{R}_{(\mathbb{x},\mathsf{tr}_i)} \\ b \xleftarrow{\mathsf{tr}} \langle \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}), \mathcal{V}(\mathsf{pp}, \mathbb{x}, \rho)\rangle \\ \big((\mathsf{cm}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j\in[\mathsf{k}]} := \mathsf{tr} \end{array}\right] .$$

A union bound over all $q \in [\mathsf{l}_i]$ yields

$$\Pr\left[\exists\, j \in [i, \mathsf{k}], q \in \mathcal{Q}_{j,i} : q \notin \hat{\mathcal{Q}}_i \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ ((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j))_{j\in[\mathsf{k}]} \leftarrow \langle \widetilde{\mathcal{P}}(\mathsf{aux}_0), \mathcal{V}(\mathsf{pp}, \mathbb{x}, \rho)\rangle \\ \text{For } j \in [\mathsf{k}] : \\ \quad (\mathsf{cm}_j, \mathsf{aux}_j) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_{j-1}, \mathbf{Q}_{j-1}, m_{j-1}) \\ \quad (\hat{\mathcal{Q}}_j, \hat{\Pi}_j) \leftarrow \hat{\mathcal{R}}^{\widetilde{\mathcal{P}}(\mathsf{aux}_j,\cdot)}\big(\mathsf{pp}, \mathbb{x}, (\mathsf{cm}_\ell)_{\ell\le j}, \big((\mathbf{Q}_\ell, \mathbf{ans}_\ell, \mathbf{pf}_\ell, m_\ell)\big)_{\ell<j}, \epsilon\big) \end{array}\right]$$

$$\le \max_{(\mathsf{pp},\mathsf{ai},\mathsf{tr}_i)} \left\{ \Pr\left[\exists\, q \in [\mathsf{l}_i], j \in [i, k] : q \in \mathcal{Q}_{j,i} \wedge q \notin \widetilde{\mathcal{Q}}_i \middle| \begin{array}{l} \rho \leftarrow \mathbf{R}_{(\mathbb{x},\mathsf{tr}_i)} \\ b \xleftarrow{\mathsf{tr}} \langle \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}), \mathcal{V}(\mathsf{pp}, \rho)\rangle \\ \big((\mathsf{cm}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j\in[\mathsf{k}]} := \mathsf{tr} \end{array}\right]\right\}$$

$$\le \max_{(\mathsf{pp},\mathsf{ai},\mathsf{tr}_i)} \left\{ \sum_{q\in[\mathsf{l}_i]} \delta_{(\mathsf{pp},\mathsf{ai},\mathsf{tr}_i)}(q) \cdot (1 - \delta_{(\mathsf{pp},\mathsf{ai},\mathsf{tr}_i)}(q))^{\mathsf{N}} \right\}$$

$$\le \frac{\mathsf{l}_i}{\mathsf{N}} \;,$$

and another union bound over $i \in [\mathsf{k}]$ gives

$$\Pr\left[\exists\, i \in [\mathsf{k}], j \in [i, \mathsf{k}], q \in \mathcal{Q}_{j,i} : q \notin \hat{\mathcal{Q}}_i \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ ((\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i, m_i))_{i\in[\mathsf{k}]} \leftarrow \langle \widetilde{\mathcal{P}}(\mathsf{aux}_0), \mathcal{V}(\mathsf{pp}, \mathbb{x}; \rho)\rangle \\ \text{For } i \in [\mathsf{k}] : \\ \quad (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathbb{x}, \mathsf{aux}_{i-1}, \mathbf{Q}_{i-1}, m_{i-1}) \\ \quad (\hat{\mathcal{Q}}_i, \hat{\Pi}_i) \leftarrow \hat{\mathcal{R}}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i,\cdot)}\big(\mathsf{pp}, \mathbb{x}, (\mathsf{cm}_j)_{j\le i}, \big((\mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j<i}, \epsilon\big) \end{array}\right]$$

$$\le \sum_{i\in[\mathsf{k}]} \frac{\mathsf{l}_i}{\mathsf{N}} = \frac{\mathsf{l}}{\mathsf{N}} \;.$$

It remains to show the connection between Eq. 6 and $\Pr\Big[\exists\, i \in [\mathsf{k}], j \in [i, \mathsf{k}], q \in \mathcal{Q}_{j,i} : q \notin \widetilde{\mathcal{Q}}_i\Big]$. According to Definition 6.8,

$$\Delta\left(\mathcal{D}\Big[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathcal{S}(\mathsf{pp}, \mathbb{x}, \mathsf{tr}_i)\Big], \mathcal{U}\Big[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathbf{R}_{(\mathbb{x},\mathsf{tr}_i)}\Big]\right) = \alpha \;.$$

Therefore, for any $\mathsf{N} \in \mathbb{N}$,

$$\Delta\left(\left(\mathcal{D}\left[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathcal{S}(\mathsf{pp}, \mathbb{x}, \mathsf{tr}_i)\right]\right)^{\mathsf{N}}, \left(\mathcal{U}\left[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathbf{R}_{(\mathbb{x}, \mathsf{tr}_i)}\right]\right)^{\mathsf{N}}\right) = \mathsf{N} \cdot \alpha \ .$$

Consider the following distribution:

$$\left[((\mathbf{Q}_i, \hat{\mathcal{Q}}_i, \hat{\Pi}_i))_{i \in [\mathsf{k}]} \ \middle| \ \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ ((\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i, m_i))_{i \in [\mathsf{k}]} \leftarrow \langle \widetilde{\mathcal{P}}(\mathsf{aux}_0), \mathcal{V}(\mathsf{pp}, \mathbb{x}, \rho) \rangle \\ \text{For } i \in [\mathsf{k}] : \\ \quad (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathbb{x}, \mathsf{aux}_{i-1}, \mathbf{Q}_{i-1}, m_{i-1}) \\ \quad (\hat{\mathcal{Q}}_i, \hat{\Pi}_i) \leftarrow \hat{\mathcal{R}}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i, \cdot)}(\mathsf{pp}, \mathbb{x}, (\mathsf{cm}_j)_{j \le i}, ((\mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j))_{j<i}, \epsilon) \end{array} \right] .$$

We can rewrite it equivalently in terms of $\left(\mathcal{U}\left[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathbf{R}_{(\mathbb{x}, \cdot)}\right]\right)^{\mathsf{N}}$:

$$\left[((\mathbf{Q}_i, \hat{\mathcal{Q}}_i, \hat{\Pi}_i))_{i \in [\mathsf{k}]} \ \middle| \ \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ ((\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i, m_i))_{i \in [\mathsf{k}]} \leftarrow \langle \widetilde{\mathcal{P}}(\mathsf{aux}_0), \mathcal{V}(\mathsf{pp}, \mathbb{x}, \rho) \rangle \\ \text{For } i \in [\mathsf{k}] : \\ \quad \text{For } j \in [\mathsf{N}] : \\ \quad \quad ((\mathsf{cm}_{j,\ell}, \mathbf{Q}_{j,\ell}, \mathbf{ans}_{j,\ell}, \mathbf{pf}_{j,\ell}, m_{j,\ell}))_{\ell \in [\mathsf{k}]} \leftarrow \mathcal{U}\left[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathbf{R}_{(((\mathsf{cm}_\ell, \mathbf{Q}_\ell, \mathbf{ans}_\ell, \mathbf{pf}_\ell, m_\ell))_{\ell<i}, \mathsf{cm}_i)}\right] \\ \quad \mathcal{K}_i := \left\{ \begin{array}{l} (\mathcal{Q}_{j,\ell,i}, \mathsf{ans}_{j,\ell,i}, \mathsf{pf}_{j,\ell,i}) : j \in [\mathsf{N}], i \le \ell \le \mathsf{k}, \\ \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}_{j,i}, \mathcal{Q}_{j,\ell,i}, \mathsf{ans}_{j,\ell,i}, \mathsf{pf}_{j,\ell,i}) = 1 \end{array} \right\} \\ \quad (\hat{\mathcal{Q}}_i, \hat{\Pi}_i) \leftarrow \mathcal{R}_{\text{post}}(\mathcal{K}_i, \mathsf{I}_i) \end{array} \right]$$

$$= \left[((\mathbf{Q}_i, \hat{\mathcal{Q}}_i, \hat{\Pi}_i))_{i \in [\mathsf{k}]} \ \middle| \ \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ ((\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i, m_i))_{i \in [\mathsf{k}]} \leftarrow \langle \widetilde{\mathcal{P}}(\mathsf{aux}_0), \mathcal{V}(\mathsf{pp}, \mathbb{x}, \rho) \rangle \\ \text{For } i \in [\mathsf{k}] : \\ \quad ((\mathsf{cm}_{j,\ell}, \mathbf{Q}_{j,\ell}, \mathbf{ans}_{j,\ell}, \mathbf{pf}_{j,\ell}, m_{j,\ell}))_{j \in [\mathsf{N}], \ell \in [\mathsf{k}]} \\ \quad \quad \leftarrow \left(\mathcal{U}\left[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathbf{R}_{(((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j))_{j<i}, \mathsf{cm}_i)}\right]\right)^{\mathsf{N}} \\ \quad \mathcal{K}_i := \left\{ \begin{array}{l} (\mathcal{Q}_{j,\ell,i}, \mathsf{ans}_{j,\ell,i}, \mathsf{pf}_{j,\ell,i}) : j \in [\mathsf{N}], i \le \ell \le \mathsf{k}, \\ \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}_{j,i}, \mathcal{Q}_{j,\ell,i}, \mathsf{ans}_{j,\ell,i}, \mathsf{pf}_{j,\ell,i}) = 1 \end{array} \right\} \\ \quad (\hat{\mathcal{Q}}_i, \hat{\Pi}_i) \leftarrow \mathcal{R}_{\text{post}}(\mathcal{K}_i, \mathsf{I}_i) \end{array} \right] .$$

Similarly, we can write

$$\left[((\mathbf{Q}_i, \widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i))_{i\in[\mathsf{k}]} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ \big((\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i, m_i)\big)_{i\in[\mathsf{k}]} \leftarrow \big\langle \widetilde{\mathcal{P}}(\mathsf{aux}_0), \mathcal{V}(\mathsf{pp}, \mathbb{x}, \rho) \big\rangle \\ \text{For } i \in [\mathsf{k}] : \\ \quad (\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \widetilde{\mathcal{P}}(\mathbb{x}, \mathsf{aux}_{i-1}, \mathbf{Q}_{i-1}, m_{i-1}) \\ \quad (\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}_i, \cdot)}\big(\mathsf{pp}, \mathbb{x}, (\mathsf{cm}_j)_{j\leq i}, \big((\mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j<i}, \epsilon\big) \end{array} \right]$$

in terms of $\left(\mathcal{D}\left[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathcal{S}(\mathsf{pp}, \mathbb{x}, \cdot)\right]\right)^{\mathsf{N}}$:

$$\left[((\mathbf{Q}_i, \widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i))_{i\in[\mathsf{k}]} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \mathsf{ai}) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ \big((\mathsf{cm}_i, \mathbf{Q}_i, \mathbf{ans}_i, \mathbf{pf}_i, m_i)\big)_{i\in[\mathsf{k}]} \leftarrow \big\langle \widetilde{\mathcal{P}}(\mathsf{aux}_0), \mathcal{V}(\mathsf{pp}, \mathbb{x}, \rho) \big\rangle \\ ((\mathsf{cm}_{i,j}, \mathbf{Q}_{i,j}, \mathbf{ans}_{i,j}, m_{i,j}))_{i\in[\mathsf{N}], j\in[\mathsf{k}]} \leftarrow \left(\mathcal{U}\left[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathbf{R}_{\mathsf{cm}_1}\right]\right)^{\mathsf{N}} \\ \mathcal{K}_1 := \left\{ \begin{array}{l} (\mathcal{Q}_{i,j,1}, \mathsf{ans}_{i,j,1}, \mathsf{pf}_{i,j,1}) : i \in [\mathsf{N}], 1 \leq j \leq \mathsf{k}, \\ \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}_{i,1}, \mathcal{Q}_{i,j,1}, \mathsf{ans}_{i,j,1}, \mathsf{pf}_{i,j,1}) = 1 \end{array} \right\} \\ (\widetilde{\mathcal{Q}}_1, \widetilde{\Pi}_1) \leftarrow \mathcal{R}_{\text{post}}(\mathcal{K}_1, \mathsf{l}_1) \\ \text{For } 1 < i \leq \mathsf{k} : \\ \quad ((\mathsf{cm}_{j,\ell}, \mathbf{Q}_{j,\ell}, \mathbf{ans}_{j,\ell}, \mathbf{pf}_{j,\ell}, m_{j,\ell}))_{j\in[\mathsf{N}], \ell\in[\mathsf{k}]} \\ \qquad \leftarrow \left(\mathcal{D}\left[\widetilde{\mathcal{P}}(\mathsf{ai}), \mathcal{S}\big(\mathsf{pp}, \mathbb{x}, \big(\big((\mathsf{cm}_j, \mathbf{Q}_j, \mathbf{ans}_j, \mathbf{pf}_j, m_j)\big)_{j<i}, \mathsf{cm}_i\big)\big)\right]\right)^{\mathsf{N}} \\ \quad \mathcal{K}_i := \left\{ \begin{array}{l} (\mathcal{Q}_{j,\ell,i}, \mathsf{ans}_{j,\ell,i}, \mathsf{pf}_{j,\ell,i}) : j \in [\mathsf{N}], i \leq \ell \leq \mathsf{k}, \\ \mathsf{VC.Check}(\mathsf{pp}, \mathsf{cm}_{j,i}, \mathcal{Q}_{j,\ell,i}, \mathsf{ans}_{j,\ell,i}, \mathsf{pf}_{j,\ell,i}) = 1 \end{array} \right\} \\ \quad (\widetilde{\mathcal{Q}}_i, \widetilde{\Pi}_i) \leftarrow \mathcal{R}_{\text{post}}(\mathcal{K}_i, \mathsf{l}_i) \end{array} \right] .$$

Therefore, we conclude that

$$\begin{aligned} \Pr\left[\exists\, i \in [\mathsf{k}], j \in [i, \mathsf{k}], q \in \mathcal{Q}_{j,i} \setminus \widetilde{\mathcal{Q}}_i\right] &\leq \Pr\left[\exists\, i \in [\mathsf{k}], j \in [i, \mathsf{k}], q \in \mathcal{Q}_{j,i} \setminus \hat{\mathcal{Q}}_i\right] + \mathsf{N} \cdot \alpha \\ &\leq \frac{\mathsf{l}}{\mathsf{N}} + \mathsf{N} \cdot \alpha \ , \end{aligned}$$

where $\Pr\left[\exists\, i \in [\mathsf{k}], j \in [i, \mathsf{k}], q \in \mathcal{Q}_{j,i} \setminus \hat{\mathcal{Q}}_i\right]$ is with respect to the same experiment as in Eq. 6. □

Acknowledgments

Alessandro Chiesa and Ziyi Guan are partially supported by the Ethereum Foundation. We thank Fermi Ma and Julius Vering for valuable discussions and participating in early stages of this work. We thank Zijing Di for valuable feedback and comments on earlier drafts of this paper.

References

[AC20] Thomas Attema and Ronald Cramer. "Compressed Σ-Protocol Theory and Practical Application to Plug & Play Secure Algorithmics". In: *Proceedings of the 40th Annual International Cryptology Conference*. CRYPTO '20. 2020, pp. 513–543.

[ACK22] Thomas Attema, Ronald Cramer, and Lisa Kohl. "A Compressed Σ-Protocol Theory for Lattices". In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO '21. 2022, pp. 549–579.

[AF22] Thomas Attema and Serge Fehr. "Parallel Repetition of $(k_1, \ldots, k_\mu)$-Special-Sound Multi-Round Interactive Proofs". In: *Proceedings of the 42nd Annual International Cryptology Conference*. CRYPTO '22. 2022, pp. 415–443.

[AFR23] Thomas Attema, Serge Fehr, and Nicolas Resch. *A Generalized Special-Soundness Notion and its Knowledge Extractors*. IACR Cryptology ePrint Archive, Report 2023/818. 2023.

[BBHMR19] James Bartusek, Liron Bronfman, Justin Holmgren, Fermi Ma, and Ron D. Rothblum. "On the (In)security of Kilian-Based SNARGs". In: *Proceedings of the 17th Theory of Cryptography Conference*. TCC '19. 2019, pp. 522–551.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Fast Reed–Solomon Interactive Oracle Proofs of Proximity". In: *Proceedings of the 45th International Colloquium on Automata, Languages and Programming*. ICALP '18. 2018, 14:1–14:17.

[BCG20] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. "Linear-Time Arguments with Sublinear Verification from Tensor Codes". In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC '20. 2020, pp. 19–46.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *Proceedings of the 14th Theory of Cryptography Conference*. TCC '16-B. 2016, pp. 31–60.

[BG08] Boaz Barak and Oded Goldreich. "Universal Arguments and their Applications". In: *SIAM Journal on Computing* 38.5 (2008). Preliminary version appeared in CCC '02., pp. 1661–1694.

[BGTZ23] Alexander R. Block, Albert Garreta, Pratyush Ranjan Tiwari, and Michal Zajac. "On Soundness Notions for Interactive Oracle Proofs". In: (2023), p. 1256.

[BIN97] Mihir Bellare, Russell Impagliazzo, and Moni Naor. "Does Parallel Repetition Lower the Error in Computationally Sound Protocols?" In: *FOCS '97*. 1997, pp. 374–383.

[BKKMS13] Eli Ben-Sasson, Yohay Kaplan, Swastik Kopparty, Or Meir, and Henning Stichtenoth. "Constant Rate PCPs for Circuit-SAT with Sublinear Query Complexity". In: *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '13. 2013, pp. 320–329.

[BS06] Eli Ben-Sasson and Madhu Sudan. "Robust locally testable codes and products of codes". In: *Random Structures and Algorithms* 28.4 (2006), pp. 387–402.

[CF13] Dario Catalano and Dario Fiore. "Vector Commitments and Their Applications". In: *Proceedings of the 16th International Conference on Practice and Theory in Public Key Cryptography*. PKC '13. 2013, pp. 55–72.

[CHMMVW20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020, pp. 738–768.

[CL10] Kai-Min Chung and Feng-Hao Liu. "Parallel Repetition Theorems for Interactive Arguments". In: *Proceedings of the 7th Theory of Cryptography Conference*. TCC '10. 2010, pp. 19–36.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. "Succinct Arguments in the Quantum Random Oracle Model". In: *Proceedings of the 17th Theory of Cryptography Conference*. TCC '19. Available as Cryptology ePrint Archive, Report 2019/834. 2019, pp. 1–29.

[CMSZ21] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. "Post-Quantum Succinct Arguments: Breaking the Quantum Rewinding Barrier". In: *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science*. FOCS '21. 2021, pp. 49–58.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. "Fractal: Post-Quantum and Transparent Recursive Proofs from Holography". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020, pp. 769–793.

[CY21a] Alessandro Chiesa and Eylon Yogev. In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO '21. 2021, pp. 711–741.

[CY21b] Alessandro Chiesa and Eylon Yogev. "Tight Security Bounds for Micali's SNARGs". In: *Proceedings of the 19th Theory of Cryptography Conference*. TCC '21. 2021, pp. 401–434.

[GK96] Oded Goldreich and Ariel Kahan. "How to construct constant-round zero-knowledge proof systems for NP". In: *Journal of Cryptology* 9.3 (1996), pp. 167–189.

[GW11] Craig Gentry and Daniel Wichs. "Separating Succinct Non-Interactive Arguments From All Falsifiable Assumptions". In: *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*. STOC '11. 2011, pp. 99–108.

[HPWP10] Johan Håstad, Rafael Pass, Douglas Wikström, and Krzysztof Pietrzak. "An Efficient Parallel Repetition Theorem". In: *Proceedings of the 7th Theory of Cryptography Conference*. TCC '10. 2010, pp. 1–18.

[HR22] Justin Holmgren and Ron D. Rothblum. "Faster Sounder Succinct Arguments and IOPs". In: *Proceedings of the 42rd Annual International Cryptology Conference*. CRYPTO '22. 2022, pp. 474–503.

[KPT97] Joe Kilian, Erez Petrank, and Gábor Tardos. "Probabilistically checkable proofs with zero knowledge". In: *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*. STOC '97. 1997, pp. 496–505.

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

[LM19] Russell W. F. Lai and Giulio Malavolta. "Subvector Commitments with Application to Succinct Arguments". In: *Proceedings of the 39th Annual International Cryptology Conference*. CRYPTO '19. 2019, pp. 530–560.

[LMS22] Alex Lombardi, Fermi Ma, and Nicholas Spooner. "Post-Quantum Zero Knowledge, Revisited or: How to Do Quantum Rewinding Undetectably". In: *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science*. FOCS '22. 2022, pp. 851–859.

[Mic00] Silvio Micali. "Computationally Sound Proofs". In: *SIAM Journal on Computing* 30.4 (2000). Preliminary version appeared in FOCS '94., pp. 1253–1298.

[RR22] Noga Ron-Zewi and Ron D. Rothblum. "Proving as fast as computing: succinct arguments with constant prover overhead". In: *Proceedings of the 54th Annual ACM Symposium on Theory of Computing*. STOC '22. 2022, pp. 1353–1363.

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

[Val08] Paul Valiant. “Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency”. In: *Proceedings of the 5th Theory of Cryptography Conference*. TCC ’08. 2008, pp. 1–18.