Untangling the Security of Kilian's Protocol: Upper and Lower Bounds *

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

Eylon Yogev eylon.yogev@biu.ac.il Bar-Ilan University

September 13, 2024

Abstract

Sigma protocols are elegant cryptographic proofs that have become a cornerstone of modern cryptography. A notable example is Schnorr's protocol, a zero-knowledge proof-of-knowledge of a discrete logarithm. Despite extensive research, the security of Schnorr's protocol in the standard model is not fully understood.

In this paper we study *Kilian's protocol*, an influential public-coin interactive protocol that, while not a sigma protocol, shares striking similarities with sigma protocols. The first example of a succinct argument, Kilian's protocol is proved secure via *rewinding*, the same idea used to prove sigma protocols secure. In this paper we show how, similar to Schnorr's protocol, a precise understanding of the security of Kilian's protocol remains elusive. We contribute new insights via upper bounds and lower bounds.

- *Upper bounds.* We establish the tightest known bounds on the security of Kilian's protocol in the standard model, via strict-time reductions and via expected-time reductions. Prior analyses are strict-time reductions that incur large overheads or assume restrictive properties of the PCP underlying Kilian's protocol.
- *Lower bounds.* We prove that significantly improving on the bounds that we establish for Kilian's protocol would imply improving the security analysis of Schnorr's protocol beyond the current state-of-the-art (an open problem). This partly explains the difficulties in obtaining tight bounds for Kilian's protocol.

Keywords: succinct interactive arguments; vector commitment schemes

^{*}This paper extends a subset of the material presented in <https://ia.cr/2023/1737>; for that material, this paper should be used as the most up-to-date reference.

Contents

1 Introduction

Sigma protocols are a fundamental class of cryptographic proofs with notable applications in cryptography (see [\[KO21\]](#page-49-0) and references therein). A sigma protocol is a public-coin interactive protocol that satisfies strong zero knowledge and soundness properties, and enjoys a simple structure. The prover sends a commitment, then the verifier responds with a random challenge, and finally, the prover sends an opening; the verifier computes a decision bit based on the instance and the interaction transcript. Perhaps the most prominent example of a sigma protocol is Schnorr's protocol [\[Sch89;](#page-50-0) [Sch91\]](#page-50-1), which proves, in zero knowledge, the knowledge of the discrete logarithm of a given group element (for a given cyclic group and base group element). Numerous works study in detail Schnorr's protocol (and its derivates), establishing upper and lower bounds on its security in different settings [\[Sho97;](#page-50-2) [PS00;](#page-49-1) [BP02;](#page-48-2) [FPS20;](#page-49-2) [BD20;](#page-48-3) [RS21;](#page-49-3) [SSY23\]](#page-50-3). Remarkably, gaps remain in our understanding of the security of Schnorr's protocol, and closing these gaps remains a challenging open problem.

In this paper we study *Kilian's protocol* [\[Kil92\]](#page-49-4), a public-coin interactive protocol that, while not a sigma protocol, shares striking similarities with sigma protocols. This protocol is historically significant as the first example of a *succinct argument*, a computationally-sound interactive proof for nondeterministic relations where the communication complexity is much smaller than the size of the relation's witness. Kilian's protocol is also the simplest example of a succinct interactive argument obtained via the VC-based approach, a fundamental paradigm for constructing succinct arguments from a probabilistic proof and a vector commitment (VC) scheme.

The shared structure with a sigma protocol is evident. The argument prover commits to a probabilistically checkable proof (PCP) string via a VC scheme (Kilian's presentation uses a Merkle commitment scheme, a VC scheme obtained 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.

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 remarkable 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.

While Kilian [\[Kil92\]](#page-49-4) gives only an informal analysis, the security of Kilian's protocol via rewinding is studied in later works. Barak and Goldreich [\[BG08\]](#page-48-4) give a detailed analysis, but with limitations: their analysis incurs overheads and applies only to PCPs that satisfy restrictive properties. Other works [\[IMSX15;](#page-49-5) [LM19;](#page-49-6) [CMSZ21;](#page-49-7) [LMS22\]](#page-49-8) provide brief analyses in the setting of negligible errors, without quantifying security bounds in terms of the underlying ingredients. We further elaborate on prior work in Section [1.3.](#page-6-0)

Motivated by the surge of interest in succinct arguments (e.g., in the context of blockchains [\[SSV19;](#page-49-9) [SSV21\]](#page-49-10)), we revisit the security of Kilian's protocol. As we discuss shortly, we expose a fine structure and open problems, much alike to the state of affairs for arguably simpler protocols such as Schnorr's protocol. This challenges the commonly-held belief that Kilian's protocol is "understood". We now turn to discuss our results.

1.1 Our results

Kilian's protocol [\[Kil92\]](#page-49-4) combines a PCP system PCP and a vector commitment scheme VC to obtain a succinct (public-coin) interactive argument. Throughout this section, we fix these ingredients (unless otherwise specified).

- PCP is a PCP system for a relation R with proof length ℓ , query complexity q, and soundness error ϵ_{PCP} . (These may depend on the given instance x.)
- VC is a vector commitment scheme VC and we denote by ϵ_{VC} its *position binding error*, which bounds the probability that an adversary outputs valid openings for the same commitment that disagree in at least one position. In general, $\epsilon_{\vee c}$ 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 running time.

Soundness. We provide the tightest known bounds for the soundness error of Kilian's protocol.

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

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

The above bound for Kilian's protocol has an intuitive explanation. An adversary that commits to the PCP string Π with maximal acceptance probability (and opens accordingly) convinces the argument verifier with probability at least ϵ_{PCP} . Moreover, an adversary that then tries to find a collision when Π is rejected achieves (under some mild conditions) a convincing probability of ϵ_{PCP} + $(1 - \epsilon_{PCP}) \cdot \epsilon_{VC}$. The $\frac{\ell}{\epsilon}$ multiplicative loss in t_{VC} compared to t_{ARG} expresses the *price of rewinding*: to reconstruct an almost full PCP string from small fragments revealed in each (valid) opening, we rewind the malicious argument prover sufficiently many times. Improving this multiplicative factor remains an open problem. Nevertheless, we show an exponential improvement when VC satisfies *expected*-time position binding, via an expected-time reduction that we discuss next.

Expected-time adversaries. We use ϵ_{VC}^* to denote the *expected-time* position binding error of VC, in which case we use t_{VC}^* to denote a bound on the *expected* running time of the adversary. Namely, ϵ_{VC}^* is the error probability given an adversary that runs in *expected-time* t_{vc}^* .

We provide the first soundness analysis of Kilian's protocol against adversaries with bounded expected running time. Since the (strict-time) soundness error of Kilian's protocol is upper-bounded by its expected-time soundness error, the following theorem gives an alternative upper bound on the (strict-time) soundness error in terms of the expected-time position binding error $\epsilon_{\vee c}^*$ of VC.

Theorem 2 (informal). *If* PCP *has a non-adaptive verifier with running time* t_v *, the expected-time soundness error* ϵ_{ARG}^* *of* Kilian[PCP, VC] *satisfies the following for every security parameter* λ *, instance* $x \notin L(R)$ *,* adversary expected time bound t^{\star}_{ARG} , and error tolerance $\epsilon > 0$:

$$
\epsilon^\star_{\text{ARG}}(\lambda, \mathbf{x}, t^\star_{\text{ARG}}) \leq \epsilon_{\text{PCP}}(\mathbf{x}) + \text{q} \cdot \epsilon^\star_{\text{VC}}\big(\lambda, \ell, \text{q}, t^\star_{\text{VC}}\big) + \epsilon, \text{ where } t^\star_{\text{VC}} = O\left(\log \frac{\text{q}}{\epsilon} \cdot (t^\star_{\text{ARG}} + \ell \cdot t_{\text{V}}) \right) \enspace .
$$

This is an *exponential improvement* in the dependency of ϵ compared to the strict-time setting (Theorem [1\)](#page-3-0). Note that Theorem [2](#page-3-1) assumes that the PCP underlying Kilian's protocol has a *non-adaptive verifier* (its queries are determined by the instance x and the PCP verifier randomness). For PCPs with an adaptive verifier, we prove an alternative statement that achieves the same bound except with $t_{\text{VC}}^* = O\left(\log \frac{q}{\epsilon} \cdot \ell \cdot t_{\text{ARG}}^*\right)$.

Lower bounds on soundness. Kilian's protocol shares certain structural resemblances to a sigma protocol: both start with a prover's commitment, followed by a verifier's challenge, and end with an opening to the commitment. However, sigma protocols are special sound while Kilian's protocol is not (see Section [1.3\)](#page-6-0). Hence, it is unclear whether there is any formal connection between Kilian's protocol and sigma protocols.

We obtain the first lower bound on the soundness error of Kilian's protocol by showing that bounding the soundness of Kilian's protocol is *as hard as that of the Schnorr identification scheme*, a sigma protocol obtained from Schorr's protocol whose security, despite significant research efforts, remains only partially understood.

Theorem 3 (informal). *There exists* PCP *for a relation* R and VC *such that, for every security parameter* λ , ϕ instance $x \notin L(R)$, Schnorr adversary time bound $t_{ID} \in \mathbb{N}$, and Schnorr adversary expected time bound $t_{ID}^{\star} \in \mathbb{N}$,

$$
\epsilon_{\text{Schnor}}(\lambda, t_{\text{ID}}) \leq \epsilon_{\text{ARG}}(\lambda, \mathbf{x}, t_{\text{ARG}}) , \text{ and}
$$

$$
\epsilon_{\text{Schnor}}^{\star}(\lambda, t_{\text{ID}}^{\star}) \leq \epsilon_{\text{ARG}}^{\star}(\lambda, \mathbf{x}, t_{\text{ARG}}^{\star}) .
$$

Above, ϵ_{ARG} and ϵ_{ARG}^* are the soundness error and the expected-time soundness error of ARG $:=$ Kilian[PCP, VC], *respectively;* ϵ_{Schnor} and $\epsilon_{\text{Schnor}}^*$ are the security against passive impersonation attacks of the Schnorr identification scheme and its expected-time analogue, respectively. Moreover, $t_{\text{ARG}} = O(t_{\text{ID}})$, and $t_{\text{ARG}}^{\star} = O(t_{\text{ID}}^{\star})$.

In Section [1.2](#page-4-0) we discuss how Theorem [3](#page-3-2) tells us that in the strict-time setting there is a polynomial gap between upper and lower bounds, whereas in the expected-time setting there is essentially no gap.

Knowledge soundness. The above discussion focuses only on the soundness error of Kilian's protocol. Recall that the soundness error is an upper bound on the probability that a time-bounded adversary convinces the argument verifier to accept an instance not in the language. Another important security notion is the *knowledge soundness error*, which bounds the probability that a time-bounded adversary convinces the verifier but a corresponding extractor, given that adversary, fails to find a valid witness for the instance. The knowledge soundness error is an upper bound on the soundness error because any extractor cannot find a valid witness for an instance not in the language (there are no valid witnesses).

We construct an extractor for Kilian's protocol that runs in time $O\left(\frac{\ell}{\epsilon}\right)$ $\frac{\ell}{\epsilon} \cdot t_{\text{ARG}}$, and similarly to Theorem [1,](#page-3-0) prove that its knowledge soundness error satisfies

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

We prove this bound by making explicit, in the proof of Theorem [1,](#page-3-0) a subroutine with running time $O\left(\frac{\ell}{\epsilon}\right)$ $\frac{\ell}{\epsilon}\cdot t_{\sf ARG}\big)$ that outputs a "good" PCP (after which we rely on the PCP knowledge extractor to obtain a witness).

What can we say about the setting of expected-time adversaries?

A similar bound as the above can straightforwardly be proved, but this would not take advantage of an expected-time reduction to achieve a smaller upper bound. Ideally, we would convert the bound on soundness error in Theorem [2](#page-3-1) into a similar bound on knowledge soundness error; however, this does *not* work. Indeed, while the proofs behind Theorems [1](#page-3-0) and [2](#page-3-1) both use rewinding arguments, they are qualitatively different (more details in Sections [2.2](#page-9-0) and [2.3\)](#page-11-0). Obtaining a knowledge soundness bound from the proof of Theorem [1](#page-3-0) is straightforward because the extractor for the PCP and the collision finder for the VC are similar algorithms. However, the proof of Theorem [2](#page-3-1) leverages extra efficiency by breaking this symmetry: only the VC collision finder is efficient, while the extractor that constructs a PCP is not. Hence, obtaining better bounds for the expected-time knowledge soundness of Kilian's protocol remains open.

We conclude by noting that similar considerations apply for proving the security of Kilian's protocol when based on a probabilistically checkable *argument* (PCA) [\[KR09;](#page-49-11) [BR22;](#page-48-5) [Ben24\]](#page-48-6) rather than a probabilistically checkable *proof*. Since a PCA is computationally sound, the running time to generate the PCA string is essential. Hence, our work yields a strict-time reduction that is compatible with PCAs, while our expected-time reduction is not compatible with PCAs.

Remark [1](#page-3-0) (adaptive choice of ∞). The results of Theorems 1 to [3](#page-3-2) 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* x *after learning* pp. Since in these stronger theorems there is no pre-set instance x, the analogous statements (for corresponding security properties) in the common reference model replace x with a size bound n (and hold for all instances such that $|x| \le n$). The plain model variants are straightforwardly implied (see Section [2.6](#page-19-0) and Remark [3.7\)](#page-21-1).

1.2 Discussion

How tight are the soundness bounds? We discuss the tightness of the soundness upper bounds in Theorems [1](#page-3-0) and [2.](#page-3-1) The takeaway is that, for the setting in Theorem [3:](#page-3-2) (i) there is a polynomial gap between Theorem [1](#page-3-0) and the best strict-time analysis of the security of the Schnorr identification scheme; and (ii) there is essentially no gap between Theorem [2](#page-3-1) and the best expected-time analysis of the security of the Schnorr identification scheme. This mirrors the state of the affairs for the Schnorr identification scheme, as we now elaborate.

The security of the Schnorr identification scheme relies on the hardness of the discrete logarithm problem. In the strict-time setting, the best analysis shows (roughly) a square-root loss in the error:

$$
\epsilon_{\text{Schnor}}(\lambda,\mathsf{t}_{\text{ID}}) \leq \sqrt{\epsilon_{\text{DLOG}}(\lambda,O(\mathsf{t}_{\text{ID}}))} \enspace,
$$

On the other hand, in the expected-time setting, it is straightforward to show that there is essentially no loss:

$$
\epsilon_\text{Schnor}^\star(\lambda,t_\text{ID}^\star) \leq \epsilon_\text{DLOG}^\star(\lambda,O(t_\text{ID}^\star))\enspace.
$$

Above $\epsilon_{\text{DLOG}} = \epsilon_{\text{DLOG}}(\lambda, t_{\text{DLOG}})$ and $\epsilon_{\text{DLOG}}^* = \epsilon_{\text{DLOG}}^*(\lambda, t_{\text{DLOG}}^*)$ are the discrete logarithm error and the expected-time discrete logarithm error, respectively for a given group. That is, for every $t_{DLOG} \in \mathbb{N}$ and t_{DLOG} -time adversary, given random y, the probability of finding x such that $y = g^x$ is bounded by $\epsilon_{DLOG}(\lambda, t_{DLOG})$. Similarly, the success probability of any adversary that has expected running time t_{DLOG}^{\star} is bounded by $\epsilon_{\text{DLOG}}^{\star}(\lambda, t_{\text{DLOG}}^{\star})$.

Below we only state the bounds, the detailed calculation can be found in Sections [8.3](#page-46-0) and [8.4.](#page-47-0)

• *Tightness of Theorem [1:](#page-3-0)* Consider the PCP and VC from Theorem [3.](#page-3-2) Let ARG := Kilian [PCP, VC]. Theorem [1](#page-3-0) implies that

$$
\epsilon_{\text{ARG}}(\lambda, \mathbf{x}, t_{\text{ARG}}) \leq 2^{-\lambda} + \epsilon_{\text{DLOG}} \left(\lambda, \frac{\ell}{\epsilon} \cdot t_{\text{ARG}} \right) + \epsilon \enspace .
$$

For a natural setting of parameters, we can instantiate the bounds of ϵ_{Schnor} and ϵ_{ARG} as follows:

$$
\epsilon_{\text{Schnorr}}(\lambda, t_{\text{ID}}) \le O\left(\sqrt{\frac{t_{\text{ID}}^2}{2^{\lambda}}}\right) , \text{ and}
$$

$$
\epsilon_{\text{ARG}}(\lambda, x, t_{\text{ARG}}) \le 2^{-\lambda} + \ell^{2/3} \cdot \Theta\left(\sqrt[3]{\frac{t_{\text{ARG}}^2}{2^{\lambda}}}\right)
$$

.

This shows a polynomial gap between the best analysis of the Schnorr identification scheme and our analysis of Kilian's protocol. Closing this gap remains an open problem.

• *Tightness of Theorem [2:](#page-3-1)* From Theorem [2,](#page-3-1) ARG := Kilian [PCP, VC] has expected-time soundness error ϵ_{ARG}^* where

$$
\epsilon^\star_{\text{ARG}}(\lambda, \mathbf{x}, t^\star_{\text{ARG}}) \leq 2^{-\lambda} + \epsilon^\star_{\text{DLOG}}\left(\lambda, O\left(\log \frac{\mathsf{q}}{\epsilon} \cdot t^\star_{\text{ARG}}\right)\right) + \epsilon \enspace.
$$

This upper bound almost matches with the best known expected-time upper bound for the security of the Schnorr identification scheme, except for a polylogarithmic loss in the adversary running time.

Why not use a random oracle? One method of analyzing Kilian's protocol is relying on idealized models such as the random oracle model. Here, the need to rewind the adversary is obviated as the PCP can be extracted directly by observing the queries performed by the adversary to the random oracle. This approach yields an analysis with tight bounds (see e.g., $[CY24]$) but is not applicable in the standard model.

In applications, practitioners replace the random oracle with a specific hash function, choosing parameters based on the idealized model's analysis. However, this limits the choice of hash functions to those presumed to sufficiently mimic a random oracle, excluding hash functions that offer notable benefits but cannot replace a random oracle. This includes, for example, hash functions with an algebraic structure (e.g., Pedersen hash), which can be fast to compute or friendly for recursive composition. Understanding the trade-offs in security bounds when using a rewinding-based analysis instead of the random oracle model is meaningful and valuable.

On the price of rewinding. We compare the soundness of Kilian's protocol when analyzed via: (i) a rewinding extractor when VC is based on a collision resistant hash function; or (ii) a straightline extractor when VC is based on an ideal hash function (a random oracle). This highlights the "price of rewinding": 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 a Merkle commitment scheme obtained from a collision-resistant hash function with security $\epsilon_{\text{CRH}}(\lambda, t_{\text{CRH}})$. By Remark [2,](#page-6-1)

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

Suppose that $\epsilon_{CRH}(\lambda, t_{CRH}) \leq t_{CRH}^2/2^{\lambda}$, which is what would be achieved by an ideal hash function. In this case, Theorem [1](#page-3-0) gives the following upper bound on the soundness error for Kilian $[PCP, VC]$:

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

Setting $\epsilon = \Theta((\ell \cdot t_{\text{ARG}})^{2/3} \cdot 2^{-\lambda/3})$ minimizes the right-hand side at $\epsilon_{\text{PCP}}(\mathbf{x}) + \Theta(\ell^{2/3} \cdot (t_{\text{ARG}}^2 \cdot 2^{-\lambda})^{1/3})$ $\epsilon_{\text{PCP}}(\mathbf{x}) + \Theta(\ell^{2/3} \cdot (t_{\text{ARG}}^2 \cdot 2^{-\lambda})^{1/3})$ $\epsilon_{\text{PCP}}(\mathbf{x}) + \Theta(\ell^{2/3} \cdot (t_{\text{ARG}}^2 \cdot 2^{-\lambda})^{1/3})$.¹

(ii) *Straightline extractor.* Suppose that we model the collision-resistant hash function as an ideal hash function, and analyze Kilian[PCP, VC] in the random oracle model. Then [\[CY24\]](#page-49-12) shows that:

$$
\epsilon_{\text{ARG}}(\lambda, \mathbf{x}, t_{\text{ARG}}) \leq \epsilon_{\text{PCP}}(\mathbf{x}) + \Theta(t_{\text{ARG}}^2 \cdot 2^{-\lambda}) \ .
$$

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

Remark 2 (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 $\epsilon_{\rm VC}$, $\epsilon_{\rm PCP}$, $\kappa_{\rm PCP}$ as given. While statistical soundness bounds on PCPs 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 commitment scheme obtained from a collision-resistant hash function h_{λ} : $\{0,1\}^2$ \rightarrow $\{0,1\}^{\lambda}$ computable in time $t_{h_{\lambda}}$ whose collision probability against t_{CRH} -size adversaries is bounded by $\epsilon_{\text{CRH}}(\lambda, t_{\text{CRH}})$ then VC has binding error $\epsilon_{\text{VC}}(\lambda, \ell, s, t_{\text{VC}}) \leq \epsilon_{\text{CRH}}(\lambda, t_{\text{CRH}})$ where $t_{\text{CRH}} = t_{\text{VC}} + O(t_{h_{\lambda}} \cdot t_{\text{CRH}})$ $q \cdot \log \ell$) for a small hidden constant that can be derived from the security reduction. (The reduction transforms a t_{VC}-size adversary $A_{\rm VC}$ against the Merkle commitment scheme into a t_{CRH}-size adversary $A_{\rm CRH}$ 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 $O(t_{h_{\lambda}} \cdot \mathsf{q} \cdot \log \ell)$ in size.)

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 Kilian's protocol, which is a succinct *interactive* argument. Below we summarize only the most relevant prior work.

Succinct arguments from collision-resistant functions. The first construction of a succinct argument is due to Kilian [\[Kil92\]](#page-49-4), and follows the VC-based approach (the underlying vector commitment is a Merkle commitment scheme based on a collision-resistant hash function). The security reduction in [\[Kil92\]](#page-49-4) is informal, and does not provide any asymptotic (nor explicit) security bounds.

Barak and Goldreich [\[BG08\]](#page-48-4) 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 [\[KPT97\]](#page-49-13)), the latter restriction is a non-standard strong property of the PCP query algorithm, which has not been shown to hold for a number of PCP constructions of interest (e.g., the short PCPs in [\[BS06;](#page-48-7) [BKKMS13\]](#page-48-8)). Under these conditions, they

¹Ignoring the lower-order term $t_{h_{\lambda}} \cdot \mathsf{q} \cdot \log \ell$.

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 work applies to *all* PCPs (including adaptive PCPs) and establishes the tightest known bound for adaptive knowledge soundness.^{[2](#page-0-0)}

Ishai, Mahmoody, Sahai, and Xiao [\[IMSX15\]](#page-49-5) provide a soundness analysis for Kilian's protocol instantiated with a PCP with negligible soundness error and a Merkle commitment scheme with negligible position binding error; they do not quantify the security of the succinct argument in terms of the security of the underlying cryptography. Lai and Malavolta [\[LM19,](#page-49-6) Appendix C] prove secure a variant of Kilian's protocol, realized with any *linear PCP* and *linear map commitment*; this generality can lead to shorter proofs.

Chiesa, Ma, Spooner, and Zhandry [\[CMSZ21\]](#page-49-7) 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.

Succinct arguments from ideal hash functions. A line of work studies security reductions for succinct *non-interactive* arguments in the random oracle model (ROM) [\[Mic00;](#page-49-14) [Val08;](#page-50-4) [BCS16;](#page-48-9) [CMS19;](#page-49-15) [CY21a;](#page-49-16) [CY21b;](#page-49-17) [BGTZ23;](#page-48-10) [CY24\]](#page-49-12). They take advantage of the ROM in two key ways. First, they use the observability of oracle queries to construct a vector commitment scheme with a *straightline* (i.e., non-rewinding) extractor: a Merkle commitment scheme in the ROM. As noted in Section [1.2,](#page-4-0) 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 sigma protocol setting (three-message public-coin protocols), 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 [\[AC20;](#page-48-11) [ACK22;](#page-48-12) [AF22\]](#page-48-13). The concrete security of general special sound protocols is relatively well-understood.

As noted in [\[CMSZ21\]](#page-49-7), 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).^{[3](#page-0-0)} We are therefore not able to apply results about special soundness directly.

 2 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.

³Towards a tighter security proof for Kilian in the post-quantum setting, Lombardi, Ma, and Spooner [\[LMS22\]](#page-49-8) introduce the 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](#page-8-1) we review Kilian's protocol. In Section [2.2](#page-9-0) we sketch our proof of Theorem [1.](#page-3-0) In Section [2.3](#page-11-0) we sketch our proof of Theorem [2.](#page-3-1) In Section [2.4](#page-15-0) we sketch our proof of Theorem [3.](#page-3-2) In Section [2.5](#page-18-0) we explain how to show the strict-time knowledge soundness of Kilian's protocol. In Section [2.6](#page-19-0) 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](#page-21-0) for formal definitions. Here we omit the algorithm that samples public parameters (and suppress these parameters in the interfaces of VC).^{[4](#page-0-0)}

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

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

> $\exists i \in \mathcal{Q} \cap \mathcal{Q}'$: ans $[i] \neq \mathsf{ans}'[i]$ \land VC. Check(cm, Q , ans, pf) = 1 \land VC. Check $(\textsf{cm}, \mathcal{Q}', \textsf{ans}', \textsf{pf}') = 1$

.

In other words, position binding makes it hard to produce two incompatible openings to the same commitment. Moreover, we also consider *expected-time position binding*: the expected-time position binding error $\epsilon_{\text{VC}}^* =$ $\epsilon_{\rm VC}^{\rm x}(\lambda,\ell,s,t_{\rm VC}^{\rm x})$ is the position binding property against adversaries whose expected running time is at most $t_{\rm VC}^{\rm x}$. **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 Kilian's protocol

We review Kilian's protocol that compiles a PCP and a VC scheme to a succinct interactive argument.

Kilian's protocol [\[Kil92\]](#page-49-4) obtains a succinct interactive argument by combining two ingredients: a probabilistically checkable proof (PCP) and a vector commitment scheme VC (fixed above). Let $PCP = (\mathbf{P}, \mathbf{V})$ be a PCP system for a relation R with alphabet Σ , proof length ℓ , query complexity q, and verifier randomness complexity r. Kilian [PCP, VC] is an interactive argument ARG = $(\mathcal{P}, \mathcal{V})$ in which the argument prover $\mathcal P$ receives an instance x and a witness w, and the argument verifier V receives the instance x. Then $\mathcal P$ and V interact, exchanging 3 messages, as follows.

- 1. P computes the PCP string $\Pi \leftarrow P(x, w)$, computes the commitment (cm, aux) $\leftarrow VC$. Commit(Π), and sends cm to V .
- 2. V samples PCP verifier randomness $\rho \leftarrow \{0, 1\}^r$ and sends it to \mathcal{P} .
- 3. P deduces the set Q of queries that $V(x; \rho)$ makes to Π , sets the query answers ans := $\Pi[Q]$, generates an opening proof pf \leftarrow VC.Open(aux, Q), and sends the tuple (Q, ans, pf) to V.
- 4. ν performs the following checks.

(a) VC.Check(pp, cm, Q , ans, pf) = 1 (i.e., ans are valid answers for positions Q relative to cm);

⁴ For example, if VC is based on a Merkle commitment scheme, the public parameters are the (randomly sampled) collision-resistant function to be used for hashing the given message down to the Merkle root.

(b) $V^{[Q,ans]}(x; \rho) = 1$ (i.e., the PCP verifier $V(x; \rho)$ accepts the answers ans on Q).

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

2.2 Soundness analysis of Kilian's protocol

We discuss the proof idea for Theorem [1.](#page-3-0)

2.2.1 Security reduction

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

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

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

where $t_{\text{VC}} = O\left(\frac{\ell}{\epsilon}\right)$ $\frac{\ell}{\epsilon} \cdot t_{\sf ARG}$).

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

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 \tilde{P} but the PCP verifier V rejects $\tilde{\Pi}$ under the same randomness. Intuitively, this allows us to approximately equate the probability that \overline{P} convinces the argument verifier V to the probability that Π convinces the PCP verifier V.

First we discuss how to use Lemma [1](#page-9-1) to establish soundness error of Kilian [PCP, VC] in Section [2.2.2.](#page-9-2) Then in Section [2.2.3](#page-10-0) we sketch the proof of Lemma [1.](#page-9-1) For simplicity, all probability statements in this section are with respect to the experiment in Lemma [1](#page-9-1) unless otherwise specified.

2.2.2 Soundness analysis

We wish to upper bound the soundness error of Kilian [PCP, VC]. As claimed in Theorem [1,](#page-3-0) we argue that for every instance $x \notin L(R)$, time bound $t_{\text{ARG}} \in \mathbb{N}$, and t_{ARG} -size adversary \mathcal{P} ,

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

The above probability can be bounded with the following by the law of total probability:

$$
\Pr\left[\begin{array}{l}\n\mathbf{V}^{[\mathcal{Q},\Pi]}(\mathbf{x};\rho) = 1 \\
\wedge \mathbf{V}^{[\mathcal{Q},\text{ans}]}(\mathbf{x};\rho) = 1 \\
\wedge \mathbf{V} \text{C}.\text{Check}(cm, \mathcal{Q}, \text{ans}, \text{pf}) = 1\n\end{array}\right] + \Pr\left[\begin{array}{l}\n\mathbf{V}^{[\mathcal{Q},\Pi]}(\mathbf{x};\rho) \neq 1 \\
\wedge \mathbf{V}^{[\mathcal{Q},\text{ans}]}(\mathbf{x};\rho) = 1 \\
\wedge \mathbf{V} \text{C}.\text{Check}(cm, \mathcal{Q}, \text{ans}, \text{pf}) = 1\n\end{array}\right]\ .
$$

The term on the right is bounded from above by $\epsilon_{\text{VC}}(\lambda, \ell, q, t_{\text{VC}}) + \epsilon$, due to Lemma [1.](#page-9-1)

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

$$
\Pr\left[\begin{array}{l} \mathbf{V}^{[\mathcal{Q},\Pi]}(\mathbb{x};\rho)=1\\ \wedge\mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbb{x};\rho)=1\\ \wedge\mathbf{V} \text{C.Check}(cm,\mathcal{Q},\mathsf{ans},\mathsf{p}\mathsf{f})=1 \end{array}\right]\leq \Pr\left[\mathbf{V}^{\widetilde{\Pi}}(\mathbb{x})=1\right]\leq \epsilon_{\mathsf{PCP}}(\mathbb{x})\enspace.
$$

2.2.3 Proof sketch of Lemma [1](#page-9-1)

We are left to sketch the proof of Lemma [1.](#page-9-1) To do so, we present a reductor algorithm \mathcal{R} .

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

 $\mathcal{R}^{\mathcal{P}(\mathsf{aux},\cdot)}(\mathsf{cm},\epsilon)$:

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

4. Output (Q, Π) .

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

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

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

As stated in Lemma [1,](#page-9-1) with the above notation we wish to bound the following probability:

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

If VC.Check(cm, Q, ans, pf) = 1 then $\mathbf{V}^{\Pi}(\mathbf{x};\rho) \neq 1 \wedge \mathbf{V}^{[Q,\text{ans}]}(\mathbf{x};\rho) = 1$ implies either: (i) $\widetilde{\Pi}$ and ans disagree at a position $q \in \mathcal{Q} \cap \tilde{\mathcal{Q}}$; or (ii) there is query q in \mathcal{Q} but not in $\tilde{\mathcal{Q}}$. We analyze the two events separately, which bounds the probability above by a union bound. We suppress the probability experiment in the derivations below. (i) Valid openings with disagreeing answers. We informally argue that

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

The reductor R checks the validity of the opening for each position it fills into $\tilde{\Pi}$. Hence the event above implies that there are valid openings to two different values at the same query position; equivalently, one can construct an adversary $A_{\nu c}$ that runs the reductor R and executes $\langle \mathcal{P}, \mathcal{V}(x, \rho) \rangle$ for some verifier randomness ρ that breaks VC's position binding. Since $A_{\nu C}$ has running time $t_{\nu C} = O(N \cdot t_{\text{ARG}})$ (its running time is dominated by the running time of R), the target probability is at most $\epsilon_{\rm VC}(\lambda, \ell, q, t_{\rm VC})$ by the position binding property of the VC. (ii) Missing position in Π . We show that

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

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 $q \in [\ell]$ is queried. The *weight* $\delta(q)$ of a query $q \in [\ell]$ is the probability that it is queried by the argument verifier with uniformly sampled randomness. We can write:

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

where the inequality follows from the fact that Q and all query sets used to generate \tilde{Q} correspond to independently sampled verifier randomness. Note that, for every $\delta \in [0,1]$, $\delta \cdot (1-\delta)^N \le 1/N$.^{[5](#page-0-0)} Hence, the target probability is upper bounded by $\frac{\ell}{N}$.

In fact, the proof for this case is more delicate than sketched above. If a position $q \in [\ell]$ has weight $\delta(q)$, we cannot conclude that $q \notin \tilde{Q}$ with probability at most $(1 - \delta(q))^N$, because \tilde{P} may often output invalid openings for q while 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* P *outputs a valid VC opening for the query set*.

Setting parameters. By an union bound, the desired probability can be upper bounded by $\epsilon_{\text{VC}}(\lambda, \ell, q, t_{\text{VC}}) + \frac{\ell}{N}$. Setting $N := \frac{\ell}{\epsilon}$ $\frac{\ell}{\epsilon}$, we get $t_{\text{VC}} = O(\textsf{N}\cdot t_{\text{ARG}}) = O\left(\frac{\ell}{\epsilon}\right)$ $\frac{\ell}{\epsilon} \cdot t_{\text{ARG}}$ and $\frac{\ell}{N} = \epsilon$, yielding the bound stated in Lemma [1.](#page-9-1)

Remark 3. Superficially one might hope for an improved analysis showing that one only needs $\frac{\ell}{q \cdot \epsilon}$ rewindings rather than $\frac{\ell}{\epsilon}$. 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 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 $\frac{\ell}{\epsilon}$ can be improved, for example, our Theorem [2](#page-3-1) considers VC schemes with expected-time position binding and avoids this multiplicative factor.

2.3 Expected-time soundness analysis of Kilian's protocol

We provide an alternative analysis for the expected-time soundness of Kilian's protocol (Theorem [2\)](#page-3-1) to avoid the blowup of $\frac{\ell}{\epsilon}$ in the VC adversary running time.

Recall that in the previous analysis, we "coupled" the reductor R and the VC adversary $A_{\nu c}$: they are essentially the same algorithm. However, notice the running time of $A_{\rm VC}$ affects the soundness error, while the running time of R does not. This leads us to the following new security reduction lemma, which "decouples" the two algorithms:

Lemma 2 (informal). *There exists a probabilistic algorithm* R *(the reductor) and algorithms* $A_{\text{VC}}^{(i)}$ *<i>(the* VC *adversaries)* for each i ∈ [q] *that, for every instance* x, *error tolerance* ϵ > 0, *adversary time bound* t_{ARG} ∈ N,

⁵A simple derivation of the inequality is the following: with $f(x) = x \cdot (1-x)^N$, we have $\frac{d}{dx} f(\delta) = 0 \iff \delta = \frac{1}{N+1}$. As $f(0) = f(1) = 0$ and δ is the only critical point in [0, 1], it achieves the maximum: $\max_{x \in [0,1]} \{f(x)\} = f(\delta) \le 1/N$.

and t_{ABC} -time adversary \widetilde{P} , satisfies

$$
\Pr\left[\begin{array}{c} \mathbf{V}^{\widetilde{\Pi}^{\star} }(\mathbf{x} ; \rho) \neq 1 \\ \wedge \mathbf{V}^{[\mathcal{Q}, \text{ans}]}(\mathbf{x} ; \rho) = 1 \\ \wedge \mathbf{V} \mathbf{C}.\text{Check}(\text{cm}, \mathcal{Q}, \text{ans}, \text{pf}) = 1 \\ \wedge \forall i \in [\mathsf{q}], \text{ans}[\mathcal{Q}[i]] = \text{ans}^{(i)}[\mathcal{Q}[i]] \end{array}\right] \left[\begin{array}{c} \mathsf{cm} \leftarrow \widetilde{\mathcal{P}} \\ \widetilde{\Pi}^{\star} \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}}(\mathsf{cm}, \epsilon) \\ \rho \leftarrow \{0, 1\}^{\mathsf{r}} \\ (\mathcal{Q}, \text{ans}, \text{pf}) \leftarrow \widetilde{\mathcal{P}}(\rho) \\ \text{for } i \in [\mathsf{q}] : \\ (\mathsf{cm}, \mathcal{Q}, \text{ans}, \text{pf}, \mathcal{Q}^{(i)}, \text{ans}^{(i)}, \text{pf}^{(i)}) \leftarrow A_{\text{VC}}^{(i)}(\rho) \end{array}\right] \leq \epsilon \enspace ,
$$

where the expected running time of $A_{\text{VC}}^{(i)}$ is $t_{\text{VC}}^{\star} = O\left(\log \frac{q}{\epsilon} \cdot (t_{\text{ARG}} + \ell \cdot t_{\text{V}})\right)$ for every $i \in [q]$.

For simplicity, all probability statements in the rest of this section are with respect to the experiment in Lemma [2](#page-11-1) unless otherwise specified.

Construction of the reductor. Similar to the reductor in Section [2.2.3,](#page-10-0) our new reductor \mathcal{R} rewinds to extract the PCP string committed by the adversary \tilde{P} . In fact, R rewinds over all possible verifier randomness to extract the "best" PCP string.

$$
\mathcal{R}^{\mathcal{P}}(\mathsf{cm})
$$
:

- 1. Initialize a proof string: $\tilde{\Pi}^* := (\perp)^{\ell}$.
- 2. For every PCP verifier randomness $\rho \in \{0, 1\}^r$:
	- (a) Run $(Q, \text{ans}, \text{pf}) \leftarrow \widetilde{\mathcal{P}}(\rho)$.
	- (b) If VC. Check(cm, Q , ans, pf) = 0, skip to next iteration.
	- (c) Record the answer for each location for later.
- 3. For every location $i \in [\ell]$, set $\tilde{\Pi}^*[i]$ to be the most frequently appeared answer in the loop (break ties with the lexicographic order).
- 4. Output $\widetilde{\Pi}^*$.

Constructions of the VC adversaries. Let C be some constant to be specified later. For every $q \in [\ell]$, we define S_q to be the following set:

 $\mathcal{S}_q := \{ \rho \in \{0,1\}^r : q \in \mathcal{Q} \text{ where } \mathcal{Q} \text{ is the set of queries make by } \mathbf{V}(\mathbf{x}; \rho) \}$.

We first introduce a subroutine of the VC adversaries, the reverse sampler Samp. On input a query $q \in [\ell]$, Samp outputs a randomness ρ sampled uniformly from S_q . We can implement Samp as follows.

 $Samp(q)$:

- 1. Repeat the following:
	- (a) Sample $\rho \leftarrow \{0, 1\}^r$.
	- (b) Compute the query set Q corresponding to ρ by running the PCP verifier $V(x; \rho)$.
	- (c) If $q \in \mathcal{Q}$, output ρ .

For every $i \in [q]$, we construct the VC adversary $A_{\nu c}^{(i)}$. In particular, given a randomness ρ with corresponding query set is $\mathcal{Q}, A_{\text{VC}}^{(i)}$ tries to find inconsistent answers for the *i*-th query in \mathcal{Q} .

 $A_{\rm \scriptscriptstyle VC}^{(i)}(\rho)$:

- 1. Run cm $\leftarrow \widetilde{\mathcal{P}}(\mathbf{x})$ and $(\mathcal{Q}, \text{ans}, \text{pf}) \leftarrow \widetilde{\mathcal{P}}(\rho)$.
- 2. Check that VC.Check(cm, Q , ans, pf) = 1. If not, output (cm, ans, ans, Q , Q , pf, pf).
- 3. Define $q := \mathcal{Q}(i)$ and set $j := 0$.
- 4. Repeat the following:
	- (a) Run $\rho' \leftarrow$ Samp (q) .
- (b) Run $(Q', \text{ans}', \text{pf}') \leftarrow \widetilde{\mathcal{P}}(\rho').$
- (c) If VC. Check $(cm, Q', ans', pf') = 1$:
	- i. If ans $[q]\neq$ ans $'[q],$ output $(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}, \mathcal{Q}', \mathsf{ans}', \mathsf{pf}').$
	- ii. If ans $[q] = \text{ans}'[q]$, set $j := j + 1$. Further, if $j = C$, output (cm, ans, ans, $\mathcal{Q}, \mathcal{Q}, \mathsf{pf}, \mathsf{pf}$).

We compute the expected running time of $A_{\nu c}^{(i)}$. For every $q \in [\ell]$, let $p_{i,q}$ be the probability that the *i*-th query is q for a uniformly sampled randomness:

$$
p_{i,q} \coloneqq \Pr\left[\mathcal{Q}(i) = q \middle| \begin{array}{c} \mathsf{cm} \leftarrow \widetilde{\mathcal{P}} \\ \rho \leftarrow \{0,1\}^r \\ (\mathcal{Q}, \mathsf{ans}, \mathsf{p} \mathsf{f}) \leftarrow \widetilde{\mathcal{P}}(\rho) \end{array} \right] .
$$

Let X be the running time of the reverse sampler Samp. Then,

$$
\mathbb{E}[X] \leq \sum_{q \in [\ell]} p_{i,q} \cdot \frac{1}{p_{i,q}} \cdot t_{\mathbf{v}} = \ell \cdot t_{\mathbf{v}}.
$$

For every $q \in [\ell]$, let ξ_q be the probability that $\widetilde{\mathcal{P}}$ gives a valid opening to a query set given that the *i*-th query is q:

$$
\xi_q := \Pr\left[\begin{array}{c} \mathsf{VC}. \mathsf{Check}(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{p}\mathsf{f}) = 1 \\ \text{conditioned on} \\ \mathcal{Q}(i) = q \end{array} \middle| \begin{array}{c} \mathsf{cm} \leftarrow \mathcal{P} \\ \rho \leftarrow \{0, 1\}^r \\ (\mathcal{Q}, \mathsf{ans}, \mathsf{p}\mathsf{f}) \leftarrow \widetilde{\mathcal{P}}(\rho) \end{array} \right] \right].
$$

Let I be the random variable that equals to 1 if the check in Step [2](#page-15-1) passes and equals to 0 otherwise. Let Y be the random variable for the running time of Step [4.](#page-15-2) The expected running time of $A_{VC}^{(i)}$ can be computed as follows:

$$
t_{\rm VC}^{\star} = t_{\rm ARG}^{\star} + \mathbb{E}[Y]
$$

= $t_{\rm ARG}^{\star} + 0 \cdot \mathbb{E}[Y | I = 0] \cdot \Pr[I = 0] + \mathbb{E}[Y | I = 1] \cdot \Pr[I = 1]$

$$
\leq t_{\rm ARG}^{\star} + C \cdot \mathbb{E}[X] + \sum_{q \in [\ell]} C \cdot p_{i,q} \cdot \frac{1}{\xi_q} \cdot t_{\rm ARG}^{\star} \cdot \xi_q
$$

= $t_{\rm ARG}^{\star} + C \cdot (t_{\rm ARG}^{\star} + \ell \cdot t_{\rm V})$.

Proof sketch of security reduction lemma. We wish to bound the following probability:

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

.

Similar to Section [2.2.3,](#page-10-0) if VC.Check(cm, Q, ans, pf) = 1, then $V^{\tilde{\Pi}^*}(\mathbf{x};\rho) \neq 1$ and $V^{[Q,\text{ans}]}(\mathbf{x};\rho) = 1$ implies that $\overline{\Pi}^*$ and ans disagree at a position $q \in \mathcal{Q}$. Unlike before, here there is no case of missing queries, because the reductor R , by construction, exhausts all verifier randomness. On the other hand, we have a new condition, $\forall i \in [q], \text{ans}[\mathcal{Q}[i]] = \text{ans}^{(i)}[\mathcal{Q}[i]]$, which means that none of the VC adversaries successfully find inconsistent openings to the same location. Hence, we focus on the following event:

$$
\exists i \in [q], \text{ans}[\mathcal{Q}[i]] = \text{ans}^{(i)}[\mathcal{Q}[i]] \neq \widetilde{\Pi}^{\star}[\mathcal{Q}[i]] .
$$

For every $q \in [\ell], \Pi^* [q]$ consists of the symbol that $\tilde{\mathcal{P}}$ opens to with highest probability. In other words, let $p(q, \sigma)$ be defined as follows:

$$
p(q, \sigma) \coloneqq \Pr \left[\begin{array}{l} q \in \mathcal{Q} \\ \wedge \operatorname{ans}[q] = \sigma \\ \wedge \operatorname{VC.Check}(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{p}\mathsf{f}) = 1 \end{array} \middle| \begin{array}{l} \mathsf{cm} \leftarrow \widetilde{\mathcal{P}} \\ \rho \leftarrow \{0, 1\}^r \\ (\mathcal{Q}, \mathsf{ans}, \mathsf{p}\mathsf{f}) \leftarrow \widetilde{\mathcal{P}}(\rho) \end{array} \right] \right].
$$

Then, by construction of \mathcal{R} ,

$$
\widetilde{\Pi}^{\star}[q] = \underset{\sigma \in \Sigma}{\arg \max} \{p(q, \sigma)\},
$$

with ties broken lexicographically.

Therefore, let $q \in \mathcal{Q}$ be the location such that $\tilde{\Pi}^{\star}[q] \neq \text{ans}[q]$, and $p(q, \text{ans}[q]) \leq \frac{1}{2}$ $\frac{1}{2}$, as otherwise, $p(q,\widetilde{\Pi}^{\star}[q]) \geq p(q,\mathsf{ans}[q]) > \frac{1}{2}$ $\frac{1}{2}$, which implies that $p(q, \tilde{\Pi}^{\star}[q]) + p(q, \text{ans}[q]) > 1$, a contradiction.

Since $A_{\text{VC}}^{(i)}$ samples C randomness ρ' such that for $(Q', \text{ans}', \text{pf}') \leftarrow \widetilde{\mathcal{P}}(\rho')$ (i) $\mathcal{Q}[i] \in \mathcal{Q}'$, (ii) ans $[\mathcal{Q}[i]] =$ ans'[Q[i]], and (iii) VC.Check(cm, Q', ans', pf') = 1, we can conclude that for every $i \in [q]$,

$$
\Pr\left[\textsf{ans}[\mathcal{Q}[i]] = \textsf{ans}^{(i)}[\mathcal{Q}[i]] \neq \widetilde{\Pi}^{\star}[\mathcal{Q}[i]] \right] \leq 2^{-C} \enspace .
$$

Hence,

$$
\Pr\left[\exists i \in [\mathsf{q}], \mathsf{ans}[\mathcal{Q}[i]] = \mathsf{ans}^{(i)}[\mathcal{Q}[i]] \neq \widetilde{\Pi}^{\star}[\mathcal{Q}[i]] \right] \leq \mathsf{q} \cdot 2^{-C} \enspace .
$$

Setting $C = \log \frac{q}{\epsilon}$ gives us the desired bound.

Soundness analysis from Lemma [2.](#page-11-1) Similar to Section [2.2.2,](#page-9-2) we wish to upper bound the soundness error of Kilian[PCP, VC]. As claimed in Theorem [2,](#page-3-1) we argue that for every instance $x \notin L(R)$, time bound $t_{\text{ARG}} \in \mathbb{N}$, and t_{ARG} -size adversary P ,

$$
\Pr\left[\big\langle\widetilde{\mathcal{P}},\mathcal{V}(\textbf{x})\big\rangle=1\right]\leq \epsilon_{\text{PCP}}(\textbf{x})+\epsilon_{\text{VC}}^{\star}(\lambda,\ell,\textbf{q},t_{\text{VC}}^{\star})+\epsilon\enspace.
$$

Using the law of total probability, the above probability can be bounded by

$$
\Pr\left[\begin{array}{l} {\bf V}^{\widetilde\Pi^\star}({\mathbbm x};\rho)=1\\ \wedge {\bf V}^{[{\mathcal{ Q}},\text{ans}]}({\mathbbm x};\rho)=1\\ \wedge {\bf V} {\bf C}.\text{Check}(\text{cm}, {\mathcal{ Q}}, \text{ans}, \text{pf})=1 \end{array}\right]+\Pr\left[\begin{array}{l} {\bf V}^{\widetilde\Pi^\star}({\mathbbm x};\rho)\neq 1\\ \wedge {\bf V}^{[{\mathcal{ Q}},\text{ans}]}({\mathbbm x};\rho)=1\\ \wedge {\bf V} {\bf C}.\text{Check}(\text{cm}, {\mathcal{ Q}}, \text{ans}, \text{pf})=1 \end{array}\right]\enspace.
$$

The term on the left is bounded by $\epsilon_{PCP}(x)$ (the soundness error of PCP) using similar reasoning as in Section [2.2.2.](#page-9-2)

The term on the right can be bounded by Lemma [2](#page-11-1) and the expected-time position binding error of the VC. Another application of the law of total probability gives

$$
\begin{array}{l} \Pr\left[\begin{array}{l} \mathbf{V}^{\widetilde{\Pi}^{\star} }(\mathbbm{x} ; \rho) \neq 1 \\[1mm] \wedge \mathbf{V}^{[\mathcal{Q}, \text{ans}]}(\mathbbm{x} ; \rho) = 1 \\[1mm] \wedge \mathbf{V} \mathsf{C}.\text{Check}(\text{cm}, \mathcal{Q}, \text{ans}, \text{pf}) = 1 \end{array}\right] \\[1mm] = \Pr\left[\begin{array}{l} \mathbf{V}^{\widetilde{\Pi}^{\star}}(\mathbbm{x} ; \rho) \neq 1 \\[1mm] \wedge \mathbf{V}^{[\mathcal{Q}, \text{ans}]}(\mathbbm{x} ; \rho) = 1 \\[1mm] \wedge \mathbf{V} \mathsf{C}.\text{Check}(\text{cm}, \mathcal{Q}, \text{ans}, \text{pf}) = 1 \\[1mm] \wedge \forall i \in [\mathsf{q}], \text{ans}[\mathcal{Q}[i]] = \text{ans}^{(i)}[\mathcal{Q}[i]] \end{array}\right] + \Pr\left[\begin{array}{l} \mathbf{V}^{\widetilde{\Pi}^{\star}}(\mathbbm{x} ; \rho) \neq 1 \\[1mm] \wedge \mathbf{V} \mathsf{C}.\text{Check}(\text{cm}, \mathcal{Q}, \text{ans}, \text{pf}) = 1 \\[1mm] \wedge \exists\ i \in [\mathsf{q}], \text{ans}[\mathcal{Q}[i]] \neq \text{ans}^{(i)}[\mathcal{Q}[i]] \end{array}\right] \\[1mm] \leq \epsilon + \sum_{i \in [\mathsf{q}]} \Pr\left[\text{ans}[\mathcal{Q}[i]] \neq \text{ans}^{(i)}[\mathcal{Q}[i]] \right] \ . \end{array}\right.
$$

The term on the right can be bounded by $\mathsf{q} \cdot \epsilon_{\text{VC}}^{\star}(\lambda, \ell, \mathsf{q}, t_{\text{VC}}^{\star} = O(\log \frac{\mathsf{q}}{\epsilon} \cdot (t_{\text{ARG}}^{\star} + \ell \cdot t_{\text{V}})))$ from the expected-time position binding property of the VC.

Extension to PCPs with adaptive verifiers. The above $A_{VC}^{(i)}$ construction only works for PCPs with nonadaptive verifiers because we cannot compute the query set as in Item [4a](#page-15-3) for adaptive PCP verifiers. However, we can adapt the construction of $A_{VC}^{(i)}$ to work for adaptive PCP verifiers as follows.

$$
A_{\rm VC}^{(i)}(\rho)
$$
:

- 1. Run cm $\leftarrow \widetilde{\mathcal{P}}(\mathbf{x})$ and $(\mathcal{Q}, \text{ans}, \text{pf}) \leftarrow \widetilde{\mathcal{P}}(\rho)$.
- 2. Check that VC.Check(cm, Q , ans, pf) = 1. If not, output (cm, ans, ans, Q , Q , pf, pf).
- 3. Define $q := \mathcal{Q}(i)$ and set $j := 0$.
- 4. Repeat the following:
	- (a) Repeatedly sample $\rho' \leftarrow \{0, 1\}^r$ and run $(Q', \text{ans}', pf') \leftarrow \tilde{\mathcal{P}}(\rho')$ until the following holds: i. $q \in \mathcal{Q}'$, and
		- ii. $VC.Check(cm, Q', ans', pf') = 1.$
	- (b) Run $(Q', \text{ans}', pf') \leftarrow \widetilde{\mathcal{P}}(\rho').$
	- (c) If $\textsf{ans}[q] \neq \textsf{ans}'[q]$, output (cm, Q, ans, pf, Q', ans', pf').
	- (d) If ans $[q] = \text{ans}'[q]$, set $j \coloneqq j + 1$. Further, if $j = C$, output (cm, ans, ans, $\mathcal{Q}, \mathcal{Q}, \text{pf}, \text{pf}$).

By a similar analysis (details in Section [6.1\)](#page-33-1), we can conclude that, for the above $A_{\rm VC}^{(i)}$,

$$
t_{\rm VC}^\star = C \cdot \ell \cdot t_{\rm ARG}^\star \ .
$$

The rest of the analysis can be directly applied to the new construction of $A_{\text{VC}}^{(i)}$.

Remark 4. When the underlying PCP has an adaptive verifier, the expected running time of $A_{VC}^{(i)}$ cannot be better than $C \cdot \ell \cdot t_{\text{ARC}}^*$. Consider a PCP with proof length ℓ and query complexity q. Assume that the PCP verifier is adaptive and the query distribution is almost uniform. Then, in order to sample a randomness that queries a fix location q as in Item [4a,](#page-15-3) the expected number of iterations is roughly ℓ . Since each iteration runs the argument adversary $\widetilde{\mathcal{P}}$, $A_{\text{VC}}^{(i)}$ has expected running time $C \cdot \ell \cdot t_{\text{ARG}}^{\star}$ for this PCP.

Remark 5 (Comparison with [\[BG08\]](#page-48-4)). [\[BG08\]](#page-48-4) gives a formal analysis of a variant of Kilian's protocol based on a *non-adaptive* and *reverse-samplable* PCP and a collision-resistance hash function. Their analysis shares similarities with our analysis in this section; both analyses construct the adversaries to the vector commitment scheme and to the PCP separately, contrary to the approach in Section [2.2.](#page-9-0) Nevertheless, our analysis in this section deviates from the analysis in [\[BG08\]](#page-48-4) (and other previous analyses) due to the differences below.

- Our analysis considers VC adversaries that run in expected running time while [\[BG08\]](#page-48-4) considers strict running time. As a result, [\[BG08\]](#page-48-4) crucially relies on the PCP's non-adaptivity and reverse sampler, as they cannot construct an efficient strict-time collision finder without these. Instead, our analysis works for all PCPs.
- The rewinding algorithm in [\[BG08\]](#page-48-4) uses the PCP reverse sampler. For every location $q \in [\ell]$, they reverse sample several PCP randomness strings that query q and record an answer only if \tilde{P} opens to it sufficiently often. This construction has a tradeoff between the running time and error probability similar to our reductor in Section [2.2](#page-9-0) (the $\frac{1}{\epsilon}$ blowup in the VC adversary time versus the additive error ϵ). In contrast, our reductor in this section searches over all PCP randomness strings to find the "best" PCP string according to \tilde{P} 's answers. This difference allows us to significantly mediate this tradeoff: to achieve an additive error of ϵ , the expected running time of our VC adversary only has a blowup of $\log \frac{1}{\epsilon}$. Unfortunately, while [\[BG08\]](#page-48-4)'s analysis gives knowledge soundness guarantee, ours in this section does not. We discuss in Section [2.5](#page-18-0) how to extend our strict-time soundness analysis in Section [2.2](#page-9-0) to prove knowledge soundness.

2.4 Lower bounds from the Schnorr identification scheme

We discuss how to prove Theorem [3,](#page-3-2) and the connection to Schnorr's protocol

2.4.1 Review: the Schnorr identification scheme

Let GroupGen be a group generation algorithm that, given a security parameter λ , samples a tuple (\mathbb{G}, p, q) where G is a group of prime order $p \geq 2^{\lambda}$ and g is a generator of the group. The Schnorr identification scheme

[\[Sch89;](#page-50-0) [Sch91\]](#page-50-1) is of a tuple of algorithms $ID_{Schner} = (P, V)$ where, for a random w in \mathbb{Z}_p , the prover P receives the instance $\mathbf{x} = ((\mathbb{G}, p, q), h = q^w \in \mathbb{G})$ and witness w, and the verifier V receives the instance x. Then P and V interact as follows.

- 1. P samples a random element $r \leftarrow \mathbb{Z}_p$, computes its first message $\alpha := g^r \in \mathbb{G}$, and sends α to V.
- 2. V samples a random challenge $\beta \leftarrow \mathbb{Z}_p$ and sends it to P.
- 3. P computes its second message $\gamma := w \cdot \beta + r \mod p$ and sends it to V.
- 4. V checks that $g^{\gamma} = \alpha \cdot h^{\beta}$.

We say that ID_{Schnorr} has error ϵ_{Schner} if for every time bound $t_{ID} \in \mathbb{N}$ and t_{ID} -time adversary \widetilde{P} ,

$$
\Pr\left[\langle \widetilde{\mathsf{P}}(\mathbf{x}), \mathsf{V}(\mathbf{x})\rangle = 1 \left| \begin{array}{c} (\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^{\lambda}) \\ \mathbb{w} \leftarrow \mathbb{Z}_p \\ h := g^{\mathbb{w}} \\ \mathbf{x} := ((\mathbb{G}, p, g), h) \end{array} \right. \right] \leq \epsilon_{\mathsf{Schnorr}}(\lambda, \mathsf{t}_{\mathsf{ID}}) \enspace.
$$

The security of the Schnorr identification scheme is based on the hardness of the *discrete logarithm problem* (it is hard for any time-bounded adversary, given a random $y \in \mathbb{G}$, to find $x \in \mathbb{Z}_p$ such that $y = g^x$). The protocol has special soundness meaning that one can efficiently compute the discrete logarithm when given two valid interaction transcripts. Thus, given a transcript of the protocol, the security reduction rewinds the adversary in order to obtain an additional accepting transcript and then extracts a witness. The analysis uses the *forking lemma* [\[PS00\]](#page-49-1) to bound the success probability of the second invocation of the adversary (conditioned on a successful first invocation).

2.4.2 VC scheme from the Schnorr identification scheme

While Kilian's protocol shares a similar structure with sigma protocols like the Schnorr identification scheme (prover's commitment, verifier's challenge, and prover's opening), Kilian's protocol is *not* a sigma protocol. Nevertheless, we show how to construct a VC scheme whose security is based on that of the Schnorr identification scheme, and later we will see how to connect this to the security of Kilian's protocol.

Recall that the position binding property ensures that the probability for any time-bounded adversary to find two inconsistent openings for the same location is bounded. On the other hand, the security of the Schnorr identification scheme relies on the fact that the it is hard for any time-bounded adversary to find two accepting transcripts of the protocol.

Therefore, the VC scheme we construct reduces finding inconsistent answers to finding accepting Schnorr transcripts, which ensures position binding from the hardness of discrete logarithm.

We construct $VC = (VC.Commit, VC.Open, VC.Check)$ as follows. (Our VC only supports messages of length 1.) Recall that in this section, we omit the algorithm for VC that samples the public parameters. For this construction, the public parameter consists of a description (\mathbb{G}, p, q) of a group generated by GroupGen given the security parameter λ , and a random group element $h \in \mathbb{G}$.

- VC.Commit (m) :
	- 1. Sample $r \leftarrow \mathbb{Z}_p$.
	- 2. Set cm $:= g^r$.
	- 3. Set aux $:= (r, m)$.
	- 4. Output (cm, aux).
- VC.Open(aux = (r, m) , {1}): Output pf := $r + m$.
- VC.Check(cm, $\{1\}$, ans, pf): Check that $g^{pf} = \text{cm} \cdot h^{ans}$.

Consider a VC adversary $A_{\nu c}$ that outputs (cm, $\mathcal{Q} = \{1\}$, $\mathcal{Q}' = \{1\}$, ans, ans', pf, pf') such that

- ans \neq ans['],
- $g^{\text{pf}} = \text{cm} \cdot h^{\text{ans}},$ and
- $g^{\mathsf{p} \mathsf{f}'} = \mathsf{cm} \cdot h^{\mathsf{ans}'}$.

Then, one can recover $x \in \mathbb{Z}_p$ such that $h = g^x$:

$$
x := (\mathsf{p}\mathsf{f}' - \mathsf{p}\mathsf{f}) \cdot (\mathsf{ans}' - \mathsf{ans})^{-1} \enspace .
$$

We can conclude that VC has position binding error ϵ_{VC} such that

$$
\epsilon_{\text{VC}}(\lambda, 1, 1, t_{\text{VC}}) \leq \epsilon_{\text{DLOG}}(\lambda, O(t_{\text{VC}})) .
$$

2.4.3 Security reduction from Kilian to Schnorr

We explain how to connect the security of Kilian's protocol to the security of the Schnorr identification scheme. The VC scheme that we consider is described above. We are left to fix a PCP.

Since the VC scheme works for messages of length 1, the PCP we consider has proof length 1. Moreover, we ensure that the PCP has very small soundness error, so that the dominant term will come from the VC scheme. In more detail, we consider a PCP system PCP for the empty relation $R = \emptyset$ with alphabet $\Sigma = \{0, 1\}^{\lambda}$, proof length $\ell = 1$, query complexity q = 1, and verifier randomness complexity $r = \lambda$. For every instance x, given a PCP proof $\Pi \in \Sigma$, the PCP verifier V works as follows:

 $\mathbf{V}^{\Pi}(\mathbb{x})$:

- 1. Sample randomness $\rho \leftarrow \{0, 1\}^{\lambda}$.
- 2. Check that $\Pi = \rho$.

The soundness error of PCP is $\epsilon_{\text{PCP}} = 2^{-\lambda}$.

Let ARG := Kilian[PCP, VC]. Consider the optimal adversary \tilde{P} for the Schnorr identification scheme. We construct an argument adversary $\tilde{\mathcal{P}}$ against the argument verifier for Kilian's protocol. Note that the argument adversary $\mathcal P$ has access to the public parameter for the VC scheme in Section [2.4.2,](#page-16-0) which consists of $((\mathbb{G}, p, q), h)$ where (\mathbb{G}, p, q) is sampled by GroupGen and $h \in \mathbb{G}$ is a random group element.

 $\widetilde{\mathcal{D}}$.

- 1. $\tilde{\mathcal{P}}$'s commitment:
	- (a) Set the instance $x_{Schnor} := ((\mathbb{G}, p, q), h)$ (using the public parameter of VC).
	- (b) Run $(\alpha, \text{aux}) \leftarrow P(\mathbf{x}_{\text{Schnor}})$.
	- (c) Output (cm, aux) := (α, aux) .
- 2. P's opening given verifier challenge ρ :
	- (a) Run $\gamma \leftarrow P(\mathbf{aux}, \rho)$.
	- (b) Output $(Q := \{1\}, \text{ans} := \rho, \text{pf} = \gamma)$.

The running time of \tilde{P} is $O(t_{1D})$, where t_{1D} is the running time of \tilde{P} . Moreover, $\langle \tilde{P}, V(\mathbf{x}) \rangle = 1$ if and only if $\langle P(x_{Schnor}), V(x_{Schnor}) \rangle = 1$. Hence, we conclude that for every instance $x \notin L(R)$ the following holds:

$$
\epsilon_{\text{Schnor}}(\lambda, t_{\text{ID}}) \leq \epsilon_{\text{ARG}}(\lambda, \mathbf{x}, O(t_{\text{ID}})) \enspace .
$$

Similarly, in the expected-time setting, we can show that

$$
\epsilon_\text{Schnor}^\star(\lambda,t_\text{ID}^\star) \leq \epsilon_{\text{ARG}}^\star(\lambda,\mathbf{x},O(t_\text{ID}^\star))\enspace.
$$

2.5 Knowledge soundness analysis of Kilian's protocol

We wish to upper bound the knowledge soundness error of Kilian [PCP, VC]. As claimed in Section [1.1,](#page-2-1) we argue that, for every $\epsilon > 0$, there exists a probabilistic extractor $\mathcal E$ that runs in time $O\left(\frac{\ell}{\epsilon}\right)$ $\frac{\ell}{\epsilon} \cdot t_{\text{ARG}}$ such that, for every instance x, time bound $t_{\text{ARG}} \in \mathbb{N}$, and t_{ARG} -size adversary $\widetilde{\mathcal{P}}$,

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

By construction of the argument verifier V , the above probability is equivalent to the following:

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

We construct $\mathcal E$ using the PCP prover $\widetilde{\mathbf{P}}$ described in Section [2.2.2](#page-9-2) and the PCP extractor $\mathbf E$ (which is given by the underlying PCP system):

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

Using the law of total probability,

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

where the last two probabilities are with respect to the experiment

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

.

The term on the right is bounded by $\epsilon_{\text{VC}}(\lambda, \ell, \mathsf{q}, t_{\text{VC}}) + \epsilon$ due to Lemma [1.](#page-9-1)

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

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

2.6 Succinct interactive arguments with adaptive security

For simplicity, we described our security analyses in the *plain model*, where there are no public parameters available to all parties; in particular, the argument verifier is responsible for sampling and sending VC's public parameters to the argument prover. However, in the technical sections (Sections [5](#page-25-0) to [7\)](#page-39-0) we prove stronger versions of Theorems [1](#page-3-0) and [2](#page-3-1) 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\)](#page-20-1), which require error bounds to hold for any instance \bf{x} chosen by the malicious argument prover up to an instance size bound $n⁶$ $n⁶$ $n⁶$ In particular, the (soundness and knowledge soundness) error bounds depend on n rather than x .

We achieve adaptive security in the CRS model by following the structure sketched in the sections above, with only syntactic modifications due to the different target definitions. (E.g., modifying experiments to replace a fixed instance 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 [5](#page-25-0) to [7\)](#page-39-0) are stronger than the (informal) statements in Theorems [1](#page-3-0) and [2](#page-3-1) because we achieve adaptive security in the CRS model.[7](#page-0-0)

For consistency, the formal statement of Theorem [3](#page-3-2) (our lower bounds for Kilian's protocol) in the technical section (Section [8\)](#page-42-0) is also proved in the setting of adaptive security in the CRS model. Nevertheless, as noted in Remark [8.12,](#page-45-0) the results hold even for non-adaptive security, which is an even stronger statement.

⁶For convenience, we use soundness and knowledge notions for the PCPs in which the malicious prover chooses the instance (see Section [3.3\)](#page-22-0). In to the information-theoretic setting, these definitions are equivalent to the standard ones with fixed instances.

⁷Adaptive 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 [3.7.](#page-21-1)

3 Preliminaries

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

3.1 Interactive arguments

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

Definition 3.2 (Perfect completeness). ARG = $(G, \mathcal{P}, \mathcal{V})$ *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 $pp \in \mathcal{G}(1^{\lambda}, n)$, and instance-witness *pair* $(x, w) \in R$ *with* $|x| \leq n$,

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

Definition 3.3 (Adaptive soundness). ARG = $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *for a relation R has* (adaptive strict-time) 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} *, adversary time bound* $t_{\text{ARG}} \in \mathbb{N}$ *, and* t_{ARG} -time algorithm P *,*

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

Definition 3.4 (Adaptive expected-time soundness). ARG = $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *for a relation R has* (adaptive) expected**time soundness error** ϵ_{ARG} *if for every security parameter* $\lambda \in \mathbb{N}$ *, instance size bound* $n \in \mathbb{N}$ *, auxiliary input* distribution D, adversary time bound $t_{\text{ARG}} \in \mathbb{N}$, and algorithm \widetilde{P} with expected running time t_{ARG}^{\star} ,

$$
\Pr\left[\begin{array}{c}|\mathbf{x}|\leq n\\\wedge\mathbf{x}\notin L(R)\\\wedge b=1\end{array}\middle|\begin{array}{c} \mathsf{pp}\leftarrow\mathcal{G}(1^\lambda,n)\\\eta\leftarrow\mathcal{D}\\\ (\mathbf{x},\mathsf{aux})\leftarrow\widetilde{\mathcal{P}}(\mathsf{pp},\eta)\\\ b\leftarrow\langle\widetilde{\mathcal{P}}(\mathsf{aux}),\mathcal{V}(\mathsf{pp},\mathbf{x})\rangle\end{array}\right]\right.\leq\epsilon^\star_{\text{ARG}}(\lambda,n,t^\star_{\text{ARG}})\enspace.
$$

Definition 3.5 (Adaptive knowledge soundness). ARG = (G, P, V) *for a relation R has* (adaptive) knowledge soundness error κ_{ARG} with extraction time t_{ε} *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} *, adversary time bound* $t_{\text{ARG}} \in \mathbb{N}$ *, and* t_{ARG} -time algorithm \mathcal{P} *,*

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

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

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

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

Remark 3.6. The argument generator G receives two inputs: the security parameter λ and an instance size bound n. This means that the public parameter sampled by 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 G receives only λ as input. Our analysis works for both cases; see Remark [4.2.](#page-24-1)

Remark 3.7 (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 $(G, \mathcal{P}, \mathcal{V})$ 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 x and witness w, and the argument verifier V' receives as input the instance x ; both also receive as input the security parameter λ (in unary). They interact as follows:

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

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.^{[8](#page-0-0)} 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* [\[CF13\]](#page-48-14) over alphabet Σ is a tuple of algorithms

 $VC = (Gen, Committee, Open, Check)$

with the following syntax.

- VC.Gen $(1^{\lambda}, \ell) \to \text{pp}$: On input a security parameter $\lambda \in \mathbb{N}$ and message size bound $\ell \in \mathbb{N}$, VC.Gen samples public parameter pp.
- VC.Commit(pp, $m) \to$ (cm, aux): On input a public parameter pp and a message $m \in \Sigma^{\ell}$, VC.Commit produces a commitment cm and the corresponding auxiliary state aux.
- VC.Open(pp, aux, Q) \rightarrow pf: On input a public parameter pp, an auxiliary state aux, and a query set $\mathcal{Q} \subset [\ell],$ VC. Open outputs an opening proof string pf attesting that $m[Q]$ is a restriction of m to Q.
- VC.Check(pp, cm, Q, ans, pf) \rightarrow {0, 1}: On input a public parameter pp, a commitment cm, a query set $Q \subseteq [\ell]$, an answer string ans $\in \Sigma^Q$, and an opening proof string pf, VC. Check determines if pf is a valid proof for 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 3.8 (Completeness). VC = (Gen, Commit, Open, 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 $\mathcal{Q} \subseteq [\ell],$

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

Definition 3.9 (Position binding). VC = (Gen, Commit, Open, Check) *has* (strict-time) 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* D *, adversary time bound* $t_{\text{VC}} \in \mathbb{N}$ *, and* t_{VC} *-time algorithm* A_{VC} *,*

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

 8 These standard definitions can be derived from Definitions [3.2,](#page-20-2) [3.3](#page-20-3) and [3.5](#page-20-4) by setting pp to be empty.

Definition 3.10 (Expected-time position binding). VC = (Gen, Commit, Open, Check) *has* expected-time **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 D , adversary time bound $t_{\text{VC}}^* \in \mathbb{N}$, and an algorithm A_{VC} with expected running time t^{\star}_{VC} ,

$$
\Pr\left[\begin{array}{c} |{\mathcal Q}|=|{\mathcal Q}'|=s\\ \wedge\ \exists\ i\in {\mathcal Q}\cap {\mathcal Q}' : \mathsf{ans}[i]\neq \mathsf{ans}'[i] \\ \wedge\ \mathsf{VC}.\mathsf{Check}(pp,cm,{\mathcal Q},\mathsf{ans},\mathsf{pf})=1\\ \wedge\ \mathsf{VC}.\mathsf{Check}(pp,cm,{\mathcal Q}',\mathsf{ans}',\mathsf{pf}')=1\end{array}\right|\; \left|\begin{array}{c} \mathsf{pp}\leftarrow \mathsf{VC}.\mathsf{Gen}(1^\lambda,\ell)\\ \eta\leftarrow {\mathcal D}\\ \left(\mathsf{cm},\mathsf{ans},\mathsf{ans}',\\ {\mathcal Q},{\mathcal Q}',\mathsf{pf},\mathsf{pf}'\right)\leftarrow A_\mathsf{VC}(\mathsf{pp},\eta)\end{array}\right.\right|\leq \epsilon_\mathsf{VC}^\star(\lambda,\ell,s,t^\star_\mathsf{VC})\enspace.
$$

Remark 3.11 (Monotonicity of ϵ_{VC}). We assume hereafter that the position binding error ϵ_{VC} is monotone in each coordinate in the natural direction:

- $\epsilon_{\text{VC}}(\cdot, \ell, s, t_{\text{VC}})$ is non-increasing (larger security parameters decrease an adversary's success);
- $\epsilon_{\text{VC}}(\lambda, \cdot, s, t_{\text{VC}})$ is non-decreasing (opening some set in a string is easier than opening in a substring);
- $\epsilon_{\text{VC}}(\lambda, \ell, \cdot, t_{\text{VC}})$ is non-decreasing (finding a collision in a set is easier than finding one in a subset); and
- $\epsilon_{\text{VC}}(\lambda, \ell, s, \cdot)$ 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 commitment schemes); in any case, otherwise one may replace, in our computations, expressions of the type $\epsilon_{\rm VC}(\lambda, \ell_{\rm max}, s_{\rm max}, t_{\rm VC})$, when $\ell_{\rm max} = \max_i {\{\ell_i\}}$ and $s_{\text{max}} = \max_j \{s_j\}$, with

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

Analogously, we assume the expected-time position binding error ϵ_{VC}^* has monotonicity as well.

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 3.12 (Completeness). PCP = (P, V) *for a relation* R *has* perfect completeness *if, for every instance-witness pair* $(x, w) \in R$,

$$
\Pr\left[\mathbf{V}^{\Pi}(\mathbf{x};\rho) = 1 \left| \begin{array}{c} \Pi \leftarrow \mathbf{P}(\mathbf{x},\mathbf{w}) \\ \rho \leftarrow \{0,1\}^r \end{array} \right. \right] = 1 \enspace .
$$

Definition 3.13 (Soundness). PCP = (P, V) *for a relation R has* **soundness error** ϵ_{PCP} *if, for every (unbounded) circuit* ^P^e *and auxiliary input distribution* ^D*,*

$$
\Pr\left[\begin{array}{c}|\mathbf{x}| \leq n \\ \wedge \mathbf{x} \notin L(R) \\ \wedge \mathbf{V}^{\widetilde{\Pi}}(\mathbf{x};\rho) = 1\end{array} \middle| \begin{array}{c} \mathbf{ai} \leftarrow \mathbf{D} \\ (\mathbf{x},\widetilde{\Pi}) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \\ \rho \leftarrow \{0,1\}^r \end{array} \right] \leq \epsilon_{\text{PCP}}(n) .
$$

Definition 3.14 (Knowledge soundness). PCP = (P, V) *for a relation* R *has* knowledge soundness error κ_{PCP} with extraction time t_{E} *if there exists a probabilistic algorithm* **E** *such that, for every adversary* **P** *and auxiliary input distribution* D*,*

$$
\Pr\left[\begin{array}{c}|\mathbf{x}| \leq n \\ \wedge(\mathbf{x}, \mathbf{w}) \notin R \\ \wedge \mathbf{V}^{\widetilde{\Pi}}(\mathbf{x}; \rho) = 1\end{array}\middle|\begin{array}{c} \mathbf{ai} \leftarrow \mathbf{D} \\ (\mathbf{x}, \widetilde{\Pi}) \leftarrow \widetilde{\mathbf{P}}(\mathbf{ai}) \\ \rho \leftarrow \{0, 1\}^r \\ \mathbf{w} \leftarrow \mathbf{E}(\mathbf{x}, \widetilde{\Pi})\end{array}\right]\right] \leq \kappa_{\text{PCP}}(n) ;
$$

moreover, **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* $q \in [\ell]$ is the number of queries that the PCP verifier makes to the PCP string (each query is an index in $[\ell]$ 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 x (e.g., of its size $|x|$).

4 Kilian's protocol

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

Construction 4.1. The argument generator 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: $pp_{\nu C} \leftarrow \text{VC.Gen}(1^{\lambda}, \ell(n)).$
- 2. Set public parameter for the interactive argument: $pp := pp_{\vee C}$.
- 3. Output pp.

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

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

4. V's decision: check that $\mathbf{V}^{[\mathcal{Q},\text{ans}]}(\mathbf{x};\rho) = 1$ and VC. Check(pp, cm, \mathcal{Q} , ans, 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 $|pp_{\text{VC}}|$ bits;
- the prover-to-verifier communication consists of $|cm| + q \cdot (\log \ell + \log |\Sigma|) + |pf|$ bits;
- the verifier-to-prover communication consists of r bits;
- the time complexity of the argument generator is $t_{\text{VC.Gen}}$.
- the time complexity of the argument prover is $t_P + t_{\text{VC.Commit}}} + t_V + t_{\text{VC.Open}};$
- the time complexity of the argument verifier is $t_V + t_{\text{VC-Check}}$.

Remark 4.2. There are vector commitments for which VC.Gen needs only the security parameter λ as input (i.e., VC.Gen works for every message size); for example, Merkle commitment schemes are vector commitment schemes with this property, because the public parameter consists of (the description of) a hash function, which suffices for every message size. In this case, the argument generator G in Construction [4.1](#page-24-2) requires only λ as input and works for every instance size. This leads the notion of an interactive argument discussed in Remark [3.6.](#page-20-5)

Remark 4.3. In the plain-model variant of Construction [4.1](#page-24-2) (see Remark [3.7\)](#page-21-1), the public parameters $pp := pp_{VC}$ 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) VC.Gen is a public-coin algorithm (its output includes all of its randomness).

5 Strict-time security analysis

Theorem 5.1. *Consider these two ingredients:*

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

Then $ARG = (G, P, V) :=$ Kilian $[PCP, VC]$ *(Construction [4.1\)](#page-24-2) is a three-message public-coin interactive argument system for* R*, whose soundness error* ϵARG *and knowledge soundness error* κARG *satisfy the following for every* $\epsilon > 0$ *and* $t_{\text{ARG}} \ge t_{\text{V}} + t_{\text{VC.Check}} + \log|\Sigma| + \log \ell$:

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

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

Corollary 5.2. Let ARG be as in Theorem [5.1.](#page-25-2) Assume that for any $n \in \mathbb{N}$, $\epsilon_{\text{VC}}(\cdot, \cdot, \cdot, t_{\text{VC}}) = \text{negl}(n)$ if $t_{\text{VC}} = \text{poly}(n)$ *. Then, given that* $t_{\text{ARG}} = \text{poly}(n)$ *, we have*

$$
\epsilon_{\text{ARG}}(\lambda, n, t_{\text{ARG}}) \le \epsilon_{\text{PCP}}(n) + \text{negl}(n) \text{ and}
$$

$$
\kappa_{\text{ARG}}(\lambda, n, t_{\text{ARG}}) \le \kappa_{\text{PCP}}(n) + \text{negl}(n) .
$$

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

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

Since p is an arbitrary polynomial, we conclude that

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

An analogous argument holds for κ_{ARG} .

5.1 Security reduction

To analyze the soundness and knowledge soundness for the argument system of Construction [4.1,](#page-24-2) 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 \tilde{P} from an argument prover \tilde{P} (which may or may not be malicious). More precisely, given a convincing argument prover \tilde{P} , we want to obtain a convincing PCP prover \tilde{P} , which we achieve via the *reductor* algorithm R in Construction [5.5.](#page-26-0)

Recall that if V accepts if and only if both V and VC. Check accept. Hence, Lemma [5.3](#page-25-3) shows that PCP strings generated by the reductor R are, up to small errors, as convincing to the PCP verifier V as the argument prover P is to the argument verifier V; in other words, R transforms an argument prover \tilde{P} into a PCP prover \overline{P} .^{[9](#page-0-0)} We later use this lemma to prove (adaptive) soundness and knowledge soundness of ARG in Sections [5.2](#page-28-0) and [5.3,](#page-30-0) respectively.

 \Box

⁹Moreover, $\mathcal R$ preserves uniformity: if $\widetilde{\mathcal P}$ is a uniform algorithm, then so is \widetilde{P} .

Lemma 5.3. *There exists a probabilistic algorithm* $\mathcal R$ *which, for every* $\epsilon > 0$ *, auxiliary input distribution* $\mathcal D$ *, time bound* $t_{\text{ARG}} \geq t_{\text{VC.Check}} + 2q \cdot (\log|\Sigma| + \log \ell)$ *, and* t_{ARG} -size circuit P, satisfies

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

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

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

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

Construction 5.4. We construct the *sampler* S as follows.

 $\mathcal{S}^{\mathcal{P}(\mathsf{aux},\cdot)}(\mathsf{pp},\mathsf{cm},\mathsf{N})$:

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

The algorithm S makes N queries to \tilde{P} , and runs in time^{[10](#page-0-0)}

$$
t_{\mathcal{S}} \leq \mathsf{N} \cdot (t_{\text{ARG}} + t_{\text{VC.Check}}) \leq 2\mathsf{N} \cdot t_{\text{ARG}}.
$$

Construction 5.5. The *reductor* \mathcal{R} is defined as follows. (Below, σ is an arbitrary symbol in the alphabet Σ .)

 $\mathcal{R}^{\mathcal{P}(\mathsf{aux},\cdot)}(\mathsf{pp},\mathsf{cm},\epsilon)$: 11 11 11 1. Set $N := \frac{\ell}{\epsilon}$ $\frac{\ell}{\epsilon}$ and run $\mathcal{K} \leftarrow \mathcal{S}^{\mathcal{P}(\text{aux},\cdot)}(\text{pp},\text{cm},\text{N}).$ 2. Run $(\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}_{\text{post}}(\mathcal{K}, \ell)$. 3. Output (Q, Π) .

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

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

¹⁰Note that the 2q($log|\Sigma| + log \ell$) overhead incurred by copying (Q' , ans', pf') into K is accounted for in the difference between t_{ARG} and the other terms.

¹¹We also denote by $\mathcal{R}^{\mathcal{P}(\text{aux},\cdot)}(\text{pp},\text{cm},\epsilon;\boldsymbol{\rho})$, where $\boldsymbol{\rho}=(\rho^{(\ell)})_{\ell\in\mathbb{N}}$ (and similarly for S) the deterministic algorithm that uses $\rho^{(\ell)}$ as the randomness for S 's ℓ -th sample. This allows the PCP prover of Construction [6.6](#page-36-0) to be deterministic.

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

Note that R makes $N = \ell/\epsilon$ queries to \tilde{P} by construction, whose total $N \cdot t_{\text{ABC}}$ time dominates that of R.

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

$$
\begin{bmatrix}\n\mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}, n) \\
\eta \leftarrow \mathcal{D} \\
(\mathbf{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta) \\
(\mathsf{cm}, \mathsf{aux}_1) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_0) \\
\rho \leftarrow \{0, 1\}^r \\
(\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_1, \rho)\n\end{bmatrix}\n=\n\begin{bmatrix}\n\mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}, n) \\
\eta \leftarrow \mathcal{D} \\
(\mathbf{x}, (\mathsf{cm}, \mathsf{aux})) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta) \\
(\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}, \cdot)}(\mathsf{pp}, \mathsf{cm}, \epsilon)\n\end{bmatrix}.\n\tag{1}
$$

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

$$
\begin{bmatrix}\n\mathbf{V}^{[Q,\Pi]}(\mathbf{x};\rho) = 0 \\
\wedge \mathbf{V}^{[Q,\text{ans}]}(\mathbf{x};\rho) = 1 \\
\wedge \mathbf{V} \mathbf{C}.\text{Check}(\mathbf{pp}, \mathbf{cm}, \mathcal{Q}, \mathbf{ans}, \mathbf{pf}) = 1\n\end{bmatrix}.
$$
\n(2)

Observe that Eq. [2](#page-27-0) implies that either (i) $\tilde{\Pi}$ and ans disagree at a position $q \in \mathcal{Q} \cap \tilde{\mathcal{Q}}$; or (ii) there is a query q in $Q \setminus Q$. We analyze the two cases separately.

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

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

where $t_{\text{VC}} \leq 3\text{N} \cdot t_{\text{ARG}}$.

Consider the following adversary $A_{\nu c}$ against the vector commitment scheme, which follows Experiment [1](#page-27-1) (without executing $\mathcal{R}_{\text{post}}$) and attempts to find a collision using the output K of the sampler S.

 $A_{\text{VC}}(\text{pp}, \eta)$:

1. Run $(\mathbf{x},(\mathsf{cm},\mathsf{aux})) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp},\eta)$.

2. Sample $\rho \leftarrow \{0, 1\}^r$.

3. Run $(Q, \text{ans}, \text{pf}) \leftarrow \widetilde{\mathcal{P}}(\text{aux}, \rho).$

4. Run $K \leftarrow \mathcal{S}^{\mathcal{P}(\text{aux},\cdot)}(\text{pp},\text{cm},\text{N}), \text{with } \text{N} = \frac{\ell}{\epsilon}$ $\frac{\ell}{\epsilon}$ (as in Construction [5.5\)](#page-26-0).

- 5. If there are $(Q', \textsf{ans}', \textsf{pf}') \in \mathcal{K}$ and $q \in \mathcal{Q}' \cap \mathcal{Q}$ with ans' $[q] \neq \textsf{ans}[q]$, output (cm, ans, ans', $\mathcal{Q}, \mathcal{Q}', \textsf{pf}', \textsf{pf}'$).
- 6. Otherwise, output (the "dummy" tuple) (cm, ans, ans, Q, Q, pf, pf).

The time complexity of the sampler S is at most $2N \cdot t_{ABC}$ and the collision-finding check (Step [5\)](#page-27-2) runs in time $N \cdot 2q(\log|\Sigma| + \log \ell) \le N \cdot t_{\text{ARG}},$ ^{[12](#page-0-0)} so the time complexity of A_{VC} is $t_{\text{VC}} \le 3N \cdot t_{\text{ARG}}$.

Therefore, according to Definition [3.9,](#page-21-2)

$$
\Pr\left[\begin{array}{c} \exists\,q\in\mathcal{Q}\cap\widetilde{\mathcal{Q}}:\,\mathsf{ans}[q]\neq\widetilde{\Pi}[q] \\ \wedge\mathsf{VC}\mathsf{Check}(\mathsf{pp},\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1 \end{array}\right] \\ = \Pr\left[\begin{array}{c} |\mathcal{Q}| = |\mathcal{Q}'| = \mathsf{q} \\ \wedge\exists\,q\in\mathcal{Q}\cap\mathcal{Q}':\,\mathsf{ans}[q]\neq\mathsf{ans}'[q] \\ \wedge\mathsf{VC}\mathsf{Check}(\mathsf{pp},\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1 \\ \wedge\mathsf{VC}\mathsf{Check}(\mathsf{pp},\mathsf{cm},\mathcal{Q}',\mathsf{ans}',\mathsf{pf}')=1 \end{array}\right]\left[\begin{array}{c} \mathsf{pp}\leftarrow\mathsf{VC}\mathsf{Gen}(1^\lambda,\ell) \\ \eta\leftarrow\mathcal{D} \\ \left(\mathsf{cm},\mathsf{ans},\mathsf{ans}',\mathsf{ns}'\right) \right\} \leftarrow A_\mathsf{VC}(\mathsf{pp},\eta) \end{array}\right]
$$

¹²Searching for an intersection between Q' and Q takes 2q log 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 running time of S).

$$
\leq \epsilon_{\text{VC}}(\lambda,\ell,\mathsf{q},t_{\text{VC}}) \ .
$$

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

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

Fix a public parameter-auxiliary input pair (p, η) (recall that these are obtained in the first steps of Experiment [1\)](#page-27-1), and define the *weight* of a coordinate $q \in [\ell]$ with respect to (pp, η) as

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

Then, by a union bound over $q \in [\ell]$,

$$
\Pr\left[\begin{array}{l}\mathcal{Q}\setminus\widetilde{\mathcal{Q}}\neq\emptyset\\\wedge\mathsf{VC}.\mathsf{Check}(pp,cm,\mathcal{Q},\mathsf{ans},\mathsf{p}\mathsf{f})=1\end{array}\right] \n= \Pr\left[\begin{array}{l}\exists q\in[\ell]:q\in\mathcal{Q}\wedge q\notin\widetilde{\mathcal{Q}}\\\wedge\mathsf{VC}.\mathsf{Check}(pp,cm,\mathcal{Q},\mathsf{ans},\mathsf{p}\mathsf{f})=1\end{array}\right] \n\leq \max_{(pp,\eta)}\left\{\Pr\left[\begin{array}{l}\exists q\in[\ell]:q\in\mathcal{Q}\wedge q\notin\widetilde{\mathcal{Q}}\\\wedge\mathsf{VC}.\mathsf{Check}(pp,cm,\mathcal{Q},\mathsf{ans},\mathsf{p}\mathsf{f})=1\end{array}\middle|\begin{array}{l}\n(\mathbf{x},\mathsf{cm},\mathsf{aux})\leftarrow\widetilde{\mathcal{P}}(\mathsf{pp},\eta)\n\end{array}\right]\right\} \n= \max_{(pp,\eta)}\left\{\sum_{q\in[\ell]}\delta_{\mathsf{pp},\eta}(q)\cdot\left(1-\delta_{\mathsf{pp},\eta}(q)\right)^{\mathsf{N}}\right\} \n\leq \frac{\ell}{\mathsf{N}},
$$

where the last inequality follows from $\delta \cdot (1-\delta)^N \leq 1/N$ for any $\delta \in [0,1]$.^{[13](#page-0-0)} Finally, plugging in $N = \frac{\ell}{\epsilon}$ $\overline{\epsilon}$ bounds Eq. [2](#page-27-0) as desired, concluding the proof:

$$
\Pr\left[\begin{array}{l}\n\mathbf{V}^{[\tilde{\mathcal{Q}},\tilde{\Pi}]}(\mathbf{x};\rho) = 0 \\
\wedge \mathbf{V}^{[\mathcal{Q},\text{ans}]}(\mathbf{x};\rho) = 1 \\
\wedge \mathbf{V} \mathbf{C}.\text{Check}(pp, cm, \mathcal{Q}, \text{ans}, pf) = 1\n\end{array}\right] \\
\leq \Pr\left[\begin{array}{l}\n\exists q \in \mathcal{Q} \cap \tilde{\mathcal{Q}} : \text{ans}[q] \neq \tilde{\Pi}[q] \\
\wedge \mathbf{V}\mathbf{C}.\text{Check}(pp, cm, \mathcal{Q}, \text{ans}, pf) = 1\n\end{array}\right] + \Pr\left[\begin{array}{l}\n\mathcal{Q} \setminus \tilde{\mathcal{Q}} \neq \emptyset \\
\wedge \mathbf{V}\mathbf{C}.\text{Check}(pp, cm, \mathcal{Q}, \text{ans}, pf) = 1\n\end{array}\right] \\
\leq \epsilon_{\text{VC}}(\lambda, \ell, \mathsf{q}, t_{\text{VC}}) + \epsilon
$$

5.2 Adaptive soundness

 \sim $^{-1}$

Lemma 5.6. *For every* $\epsilon > 0$, security parameter $\lambda \in \mathbb{N}$, instance size bound $n \in \mathbb{N}$, auxiliary input distribution \mathcal{D} , adversary time bound $t_{\text{ARG}} \ge t_{\text{VC-Check}} + 2q \cdot (\log|\Sigma| + \log \ell)$ and t_{ARG} -size circuit $\widetilde{\mathcal{P}}$, the soundness error of *the argument system in Construction [4.1](#page-24-2) satisfies*

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

where $t_{\text{VC}} = O\left(\frac{\ell}{\epsilon}\right)$ $\frac{\ell}{\epsilon} \cdot t_{\sf ARG}$).

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

Proof. Recall, from Definition [3.3](#page-20-3) and Construction [4.1,](#page-24-2) that our goal is to upper bound

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

(Since \tilde{P} sends cm as the first message in Construction [4.1,](#page-24-2) 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 [5.3,](#page-25-3) we consider the following experiment (a restatement of Experiment [1\)](#page-27-1), which augments the above by executing R , and thus leaves the probability unchanged.

$$
\begin{bmatrix}\n\mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}, n) \\
\eta \leftarrow \mathcal{D} \\
(\mathbf{x}, (\mathsf{cm}, \mathsf{aux})) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta) \\
(\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}^{\mathcal{P}(\mathsf{aux}, \cdot)}(\mathsf{pp}, \mathsf{cm}, \epsilon) \\
\rho \leftarrow \{0, 1\}^r \\
(\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho)\n\end{bmatrix}
$$

.

.

By total probability,

$$
\Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\ \wedge \mathbf{x} \notin L(R)\\ \wedge \mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho) = 1\\ \wedge \mathbf{V} \subset \mathrm{Check}(\mathrm{pp},\mathrm{cm},\mathcal{Q},\mathrm{ans},\mathrm{pf}) = 1 \end{array}\right]
$$
\n
$$
= \Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\ \wedge \mathbf{x} \notin L(R)\\ \wedge \mathbf{V}^{[\mathcal{Q},\bar{\mathrm{fl}}]}(\mathbf{x};\rho) = 1\\ \wedge \mathbf{V}^{[\mathcal{Q},\bar{\mathrm{ans}}]}(\mathbf{x};\rho) = 1\\ \wedge \mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho) = 1\\ \wedge \mathbf{V} \subset \mathrm{Check}(\mathrm{pp},\mathrm{cm},\mathcal{Q},\mathrm{ans},\mathrm{pf}) = 1 \end{array}\right] + \Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\ \wedge \mathbf{x} \notin L(R)\\ \wedge \mathbf{V}^{[\mathcal{Q},\bar{\mathrm{ans}}]}(\mathbf{x};\rho) = 0\\ \wedge \mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho) = 1\\ \wedge \mathbf{V} \subset \mathrm{Check}(\mathrm{pp},\mathrm{cm},\mathcal{Q},\mathrm{ans},\mathrm{pf}) = 1 \end{array}\right]
$$

We first bound the probability of the leftmost term by the PCP system's soundness error (i.e., Definition [3.13](#page-22-1) with respect to PCP).

Construction 5.7. We define the auxiliary input distribution **D** of the PCP prover \widetilde{P} as follows:

D:

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

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

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

- 1. Parse ai as (pp, η, ρ) .
- 2. Run $(\mathbf{x}, \mathsf{aux}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta)$.
- 3. Set $\textsf{aux} \coloneqq (\textsf{pp}, \textsf{aux}_0, \boldsymbol{\rho}).$ 4. Output (x, aux) . $\widetilde{\mathbf{P}}(\mathbf{aux})$: 1. Parse aux as (pp, aux_0, ρ). 2. Run (cm, aux_1) $\leftarrow \widetilde{\mathcal{P}}(aux_0)$.
- 3. Run $(\widetilde{\mathcal{Q}}, \widetilde{\Pi}) \leftarrow \mathcal{R}^{\mathcal{P}(\textsf{aux}_1, \cdot)}(\textsf{pp}, \textsf{cm}, \epsilon; \boldsymbol{\rho}).$ 4. Output Π .

Using Definition 3.13 , 14

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

$$
\leq \Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\ \wedge \mathbf{V}^{[\tilde{\mathcal{Q}},\tilde{\Pi}]}(\mathbf{x};\rho) = 1 \end{array}\right] \quad \begin{array}{l} \text{ai} \leftarrow \mathbf{D} \\ \wedge \mathbf{x} \notin L(R)\\ \wedge \mathbf{V}^{\tilde{\Pi}}(\mathbf{x}) = 1 \end{array}\right] \quad \begin{array}{l} \text{ai} \leftarrow \mathbf{D} \\ (\mathbf{x},\text{aux}) \leftarrow \tilde{\mathbf{P}}(\text{ai})\\ \tilde{\Pi} \leftarrow \tilde{\mathbf{P}}(\text{aux}) \end{array}\right] \leq \epsilon_{\text{PCP}}(n) \quad .
$$

Lastly, an application of Lemma [5.3](#page-25-3) yields

$$
\Pr\left[\begin{array}{l}|\mathbf{x}|\leq n\\\wedge\mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbf{x};\rho)=0\\\wedge\mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho)=1\\\wedge\mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho)=1\\\wedge\mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho)=1\end{array}\right]\leq \Pr\left[\begin{array}{l}\mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbf{x};\rho)=0\\\wedge\mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho)=1\\\wedge\mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho)=1\\\leq \epsilon_{\text{VC}}(\lambda,\ell,\mathsf{q},t_{\text{VC}})+\epsilon\end{array}\right]
$$

where $t_{\text{VC}} \leq \frac{3\ell}{\epsilon}$ $\frac{3\ell}{\epsilon} \cdot t_{\text{ARG}}$, which concludes the proof.

5.3 Adaptive knowledge soundness

Lemma 5.8. *For every* $\epsilon > 0$, security parameter $\lambda \in \mathbb{N}$, instance size bound $n \in \mathbb{N}$, auxiliary input distribution \mathcal{D} , adversary time bound $t_{\text{ARG}} \ge t_{\text{VC-Check}} + 2q \cdot (\log|\Sigma| + \log \ell)$ and t_{ARG} -size circuit $\widetilde{\mathcal{P}}$, the knowledge soundness *error of the argument system obtained by Construction [4.1](#page-24-2) satisfies*

 \Box

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

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

Construction 5.9. Let E be the extractor for PCP. We use \tilde{P} (Construction [6.6\)](#page-36-0) and E to construct the knowledge extractor E for ARG as follows.

 $\mathcal{E}^{\mathcal{P}(\mathsf{aux})}(\mathsf{pp}, \mathbb{x}, \mathsf{tr})$: 1. Sample $\rho \leftarrow (\{0, 1\}^r)^N$.

¹⁴Note that the prover in Definition [3.13](#page-22-1) corresponds to the sequential execution of both steps in Construction [6.6.](#page-36-0)

- 2. Set $\text{aux} := (pp, \text{aux}, \rho)$ and run $\tilde{\Pi} \leftarrow \tilde{\mathbf{P}}(\text{aux})$.^{[15](#page-0-0)}
- 3. Run w \leftarrow $\mathbf{E}(\mathbf{x}, \tilde{\Pi})$.
- 4. Output w.

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

Proof. From Definition [3.5](#page-20-4) and Construction [4.1,](#page-24-2) our goal is to upper bound

$$
\Pr\left[\begin{array}{l}|\mathbf{x}| \leq n & \mathbf{p} \neq \mathcal{G}(1^{\lambda}, n) \\ \wedge(\mathbf{x}, \mathbf{w}) \notin R & (\mathbf{x}, \mathbf{aux}) \leftarrow \tilde{\mathcal{P}}(\mathbf{p} \mathbf{p}, \eta) \\ \wedge b = 1 & b \stackrel{\text{tr}}{\leftarrow} \langle \tilde{\mathcal{P}}(\mathbf{aux}), \mathcal{V}(\mathbf{p} \mathbf{p}, \mathbf{x}) \rangle \\ \mathbf{w} \leftarrow \mathcal{E}^{\tilde{\mathcal{P}}(\mathbf{aux})}(\mathbf{p} \mathbf{p}, \mathbf{x}, \mathbf{tr})\end{array}\right]\right] \\
= \Pr\left[\begin{array}{l}|\mathbf{x}| \leq n & \mathbf{x} \\ \wedge(\mathbf{x}, \mathbf{w}) \notin R & (\mathbf{x}, \mathbf{aux}) \in \tilde{\mathcal{P}}(\mathbf{p} \mathbf{p}, \mathbf{r}) \\ \wedge \mathbf{V}^{[\mathcal{Q},\mathsf{ans}]}(\mathbf{x}; \rho) = 1 & (\mathbf{cm}, \mathbf{aux}_1) \leftarrow \tilde{\mathcal{P}}(\mathbf{p} \mathbf{p}, \eta) \\ \wedge \mathbf{V} \mathbf{C}.\mathbf{Check}(\mathbf{p} \mathbf{p}, \mathbf{cm}, \mathcal{Q}, \mathbf{ans}, \mathbf{p} \mathbf{f}) = 1 & \left(\begin{array}{l} \mathcal{O}, 1 \end{array}\right)^r \\ (\mathcal{Q}, \mathsf{ans}, \mathbf{p} \mathbf{f}) \leftarrow \tilde{\mathcal{P}}(\mathsf{aux}_1, \rho) \\ \mathbf{w} \leftarrow \mathcal{E}^{\tilde{\mathcal{P}}(\mathsf{aux}_1)}(\mathbf{p} \mathbf{p}, \mathbf{x}, \mathbf{tr})\end{array}\right].
$$

Now, note that by Constructions [5.9](#page-30-1) and [6.6,](#page-36-0) the experiment above is equivalent to the following.

$$
\begin{bmatrix}\n\mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}, n) \\
\eta \leftarrow \mathcal{D} \\
(\mathbf{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta) \\
(\mathsf{cm}, \mathsf{aux}_1) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_0) \\
\rho \leftarrow \{0, 1\}^r \\
(\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}_1, \rho) \\
\mathsf{aux} := (\mathsf{pp}, \mathsf{aux}_1, \rho) \\
\widetilde{\Pi} \leftarrow \widetilde{\mathsf{P}}(\mathsf{aux}) \\
\mathsf{w} \leftarrow \mathsf{E}(\mathbf{x}, \widetilde{\Pi})\n\end{bmatrix}\n\begin{bmatrix}\n\mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}, n) \\
\eta \leftarrow \mathcal{D} \\
(\mathbf{x}, (\mathsf{cm}, \mathsf{aux})) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta) \\
(\mathbf{x}, (\mathsf{cm}, \mathsf{aux})) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta) \\
(\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho) \\
(\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho) \\
(\mathcal{Q}, \widetilde{\Pi}) \leftarrow \mathcal{R}^{\widetilde{\mathcal{P}}(\mathsf{aux}, \cdot)}(\mathsf{pp}, \mathsf{cm}, \epsilon)\n\end{bmatrix}
$$

.

We thus consider the above experiment, which augments that of Lemma [5.3](#page-25-3) by appending an execution of E, for the rest of the proof. Note that, as in Lemma [5.6,](#page-28-1) the experiment consisting of $(x, aux_0) \leftarrow \mathcal{P}(pp, \eta)$ followed by $(cm, aux_1) \leftarrow \widetilde{P}(aux_0)$ can be replaced by $(x, (cm, aux)) \leftarrow \widetilde{P}(pp, \eta)$: 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 \overline{P} on the left-hand side is replaced with sampling by R on the right-hand side.

By total probability,

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

¹⁵Note that, by Construction [6.6,](#page-36-0) \widetilde{P} only executes $\widetilde{P}(aux)$ (calls to $\widetilde{P}(aux', \rho)$ are continuations of executions of $\widetilde{P}(aux)$). Therefore, an equivalent construction gives \widetilde{P} oracle access to \widetilde{P} (aux) (rather than to \widetilde{P}), excluding aux from aux.

$$
= \Pr\left[\begin{array}{c} |\mathbf{x}| \leq n \\ \wedge (\mathbf{x}, \mathbf{w}) \notin R \\ \wedge \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbf{x};\rho) = 1 \\ \wedge \mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho) = 1 \\ \wedge \mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho) = 0 \\ \wedge \mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho) = 0 \end{array}\right] + \Pr\left[\begin{array}{c} |\mathbf{x}| \leq n \\ \wedge (\mathbf{x}, \mathbf{w}) \notin R \\ \wedge \mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbf{x};\rho) = 0 \\ \wedge \mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho) = 1 \\ \wedge \mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho) = 1 \end{array}\right].
$$

With the PCP prover \widetilde{P} in Construction [6.6](#page-36-0) and using Definition [3.14,](#page-22-2) we have

$$
\Pr\left[\begin{array}{c} \left|\mathbf{x}\right| \leq n\\ \wedge\left(\mathbf{x},\mathbf{w}\right) \notin R\\ \wedge\mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbf{x};\rho)=1\\ \wedge\mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho)=1\\ \wedge\mathbf{V} \text{C.Check}(pp,cm,\mathcal{Q},\mathrm{ans},\mathrm{pf})=1 \end{array}\right]\leq \Pr\left[\begin{array}{c} \left|\mathbf{x}\right| \leq n\\ \wedge\left(\mathbf{x},\mathbf{w}\right) \notin R\\ \wedge\mathbf{V}^{\widetilde{\Pi}}(\mathbf{x};\rho)=1\\ \wedge\mathbf{V}^{\widetilde{\Pi}}(\mathbf{x};\rho)=1 \end{array}\right]\begin{array}{c} \text{ai} \leftarrow \mathbf{D}\\ (\mathbf{x},\widetilde{\Pi}) \leftarrow \widetilde{\mathbf{P}}\\ \rho \leftarrow \{0,1\}^r\\ \mathbf{w} \leftarrow \mathbf{E}(\mathbf{x},\widetilde{\Pi}) \end{array}\right] \leq \kappa_{\mathsf{PCP}}(n) \ .
$$

Finally, Lemma [5.3](#page-25-3) implies

$$
\Pr\left[\begin{array}{l}|\mathbbm{x}|\leq n\\\wedge\mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbbm{x};\rho)=0\\\wedge\mathbf{V}^{[\mathcal{Q},\text{ans}]}(\mathbbm{x};\rho)=1\\\wedge\mathbf{V} \text{C.Check}(pp,cm,\mathcal{Q},\text{ans},\text{pf})=1\end{array}\right]\leq \Pr\left[\begin{array}{l}\mathbf{V}^{[\widetilde{\mathcal{Q}},\widetilde{\Pi}]}(\mathbbm{x};\rho)=0\\\wedge\mathbf{V}^{[\mathcal{Q},\text{ans}]}(\mathbbm{x};\rho)=1\\\wedge\mathbf{V}\text{C.Check}(pp,cm,\mathcal{Q},\text{ans},\text{pf})=1\end{array}\right]\leq \epsilon_{\text{VC}}(\lambda,\ell,q,t_{\text{VC}})+\epsilon\enspace,
$$

where $t_{\text{VC}} \leq \frac{3\ell}{\epsilon}$ $\frac{3\ell}{\epsilon} \cdot t_{\text{ARG}}$, which concludes the proof. \Box

6 Expected-time soundness analysis

Theorem 6.1. *Consider these two ingredients:*

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

Then $\text{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V}) := \text{Kilian}[\text{PCP}, \text{VC}]$ *is a three-message public-coin interactive argument system for* R, whose soundness error $\epsilon_{\rm ARG}$ and expected-time soundness error $\epsilon^{\star}_{\rm ARG}$ satisfies the following for every $\epsilon > 0$ and $t_{\text{ARG}} \geq t_{\text{VC.Check}} + \log|\Sigma| + \log \ell$

$$
\epsilon^\star_{\text{ARG}}(\lambda,n,t^\star_{\text{ARG}}) \leq \epsilon_{\text{PCP}}(n) + \text{q} \cdot \epsilon^\star_{\text{VC}}(\lambda,\ell,\text{q},t^\star_{\text{VC}}) + \epsilon \enspace,
$$

where ϵ_{PCP} is the soundness error of PCP and $t_{\text{VC}}^* = O\left(\ell \cdot \log \frac{q}{\epsilon} \cdot t_{\text{ARG}}^*\right)$.

It is clear that $\epsilon_{ARG}(\lambda, n, t_{ARG}) \leq \epsilon_{ARG}^{\star}(\lambda, n, t_{ARG})$, so Theorem [6.1](#page-33-2) gives an alternative bound of soundness error of ARG as well.

6.1 Security reduction

Similar to Section [5,](#page-25-0) we prove a slightly different security reduction lemma. We first introduce a few new definitions.

Definition 6.2. *For every argument prover* $\widetilde{\mathcal{P}}$ *, public parameter* pp*, auxiliary input* $\eta \in \mathcal{D}$ *, coordinate* $q \in [\ell]$ *and* $\sigma \in \Sigma$ *, we define* $p_{\text{pp},\eta}(q,\sigma)$ *as follows:*

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

We also define the optimal PCP string $\widetilde{\Pi}^{\star}_{\text{pp},\eta}$ (with respect to pp, $\eta \in \mathcal{D}$ *and* $\widetilde{\mathcal{P}}$) by

$$
\widetilde{\Pi}_{\mathsf{pp},\eta}^{\star}[q] \coloneqq \underset{\sigma \in \Sigma}{\arg \max} \{ p_{\mathsf{pp},\eta}(q,\sigma) \},
$$

with ties broken lexicographically.

Lemma 6.3 (Alternative security reduction lemma). *There exist probabilistic algorithms* $A_{\text{VC}}^{(i)}$ *for each* $i \in [q]$ $\textit{such that, for every } C \in \mathbb{N}$, adversary time bound $t_{\text{ARG}}^* \geq t_{\text{VC-Check}} + \log|\Sigma| + \log \ell$ and expected t_{ARG}^* -time *adversary* ^Pe*, satisfies*

$$
\Pr\left[\begin{array}{c} \mathbf{V}^{\widetilde{\Pi}_{\text{pp},\eta}^{*}(\mathbf{x};\rho)=0} \\ \wedge \mathbf{V}^{[\mathcal{Q},\text{ans}]}(\mathbf{x};\rho)=1 \\ \wedge \mathbf{V}c.\text{Check}(pp,cm,\mathcal{Q},\text{ans},\text{pf})=1 \\ \wedge \forall i \in [q], \text{ans}[\mathcal{Q}[i]]=\text{ans}^{(i)}[\mathcal{Q}[i]] \end{array}\right] \left[\begin{array}{c} \mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda},n) \\ \eta \leftarrow \mathcal{D} \\ \left(\mathbf{x},(\text{cm},\text{aux})\right) \leftarrow \widetilde{\mathcal{P}}(\text{pp},\eta) \\ \rho \leftarrow \{0,1\}^{r} \\ \left(\mathcal{Q},\text{ans},\text{pf}\right) \leftarrow \widetilde{\mathcal{P}}(\text{aux},\rho) \\ \text{For } i \in [q]: \\ \left(\begin{array}{c} \text{cm},\mathcal{Q},\text{ans},\text{pf}, \\ \mathcal{Q}^{(i)},\text{ans}^{(i)},\text{pf}^{(i)} \end{array}\right) \leftarrow A_{\text{VC}}^{(i)}(\text{pp},\eta,\rho) \end{array}\right] \leq \epsilon \enspace .
$$

Moreover, $A_{\text{VC}}^{(i)}$ runs in expected time $t_{\text{VC}}^{\star} = O\left(\ell \cdot \log \frac{\mathsf{q}}{\epsilon} \cdot t_{\text{ARG}}^{\star}\right)$ for all i. We construct $A_{\text{VC}}^{(i)}$ below.

Construction 6.4. Given an argument prover \tilde{P} , we construct each adversary $A_{VC}^{(i)}$ as follows.

 $A_{\mathsf{VC}}^{(i)}(\mathsf{pp},\eta,\rho)$:

- 1. Run (x, cm, aux) $\leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta)$ and (Q, ans, pf) $\leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho)$.
- 2. Check that VC.Check(pp, cm, Q , ans, pf) = 1. If not, output (cm, ans, ans, Q , Q , pf, pf).
- 3. Define $q := \mathcal{Q}(i)$ and set $j := 0$.
- 4. Repeat the following:
	- (a) Sample $\rho' \leftarrow \{0, 1\}^r$ and run $(Q', \text{ans}', \text{pf}') \leftarrow \widetilde{\mathcal{P}}(\text{aux}, \rho').$
	- (b) If $q \in \mathcal{Q}'$ and VC. Check(pp, cm, \mathcal{Q}' , ans', pf') = 1:
		- i. If ans $[q]\neq$ ans $'[q],$ output $(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}, \mathcal{Q}', \mathsf{ans}', \mathsf{pf}').$ ii. If ans $[q] = \text{ans}'[q]$, set $j := j + 1$. Further, if $j = \log \frac{q}{\epsilon}$, output (cm, ans, ans, Q, Q, pf, pf).

We first analyze the expected running time of $A_{\text{VC}}^{(i)}$. For every (pp, η) and $q \in [\ell]$, we define $\xi_{\text{pp},\eta}^{(i)}(q)$ as follows:

$$
\xi_{\mathsf{pp},\eta}^{(i)}(q) \coloneqq \Pr\left[\begin{array}{c} \mathcal{Q}(i) = q \\ \wedge \mathsf{VC}.\mathsf{Check}(\mathsf{pp},\mathcal{Q},\mathsf{ans},\mathsf{pf}) = 1 \\ \end{array} \middle| \begin{array}{c} (\mathbf{x},(\mathsf{cm},\mathsf{aux})) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp},\eta) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ (\mathcal{Q},\mathsf{ans},\mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux},\rho) \end{array} \right] \right].
$$

Define, also, $\xi_{\mathsf{pp},\eta}(q) \coloneqq \sum_{i \in [\mathsf{q}]} \xi_{\mathsf{pp},\eta}^{(i)}(q)$; note that this is the probability of the event

 $[q \in \mathcal{Q} \land \mathsf{VC}.\mathsf{Check}(\mathsf{pp}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1]$

(under the same experiment). Finally, denote by $\mathcal T$ and $\mathcal M_{\mathsf{pp},\eta}$ the distributions of running time of $\widetilde{\mathcal P}$ and the number of iterations of Step [4](#page-34-0) in an execution of $A_{\rm VC}(pp, \eta, \rho)$, respectively. Note that the conditional expectation of $M \leftarrow \mathcal{M}_{\mathsf{pp},\eta}$ on the above event is $O\left(\log \frac{q}{\epsilon}/\xi_{\mathsf{pp},\eta}(q)\right)$.

Then the running time of $A_{\text{VC}}^{(i)}$ is $O(N \cdot t_{\text{VC.Check}} + \sum_{k \in [M]} T_k)$, where $T_k \leftarrow \mathcal{T}$ for all $k \in [M]$. Therefore,

$$
\mathbb{E}\left[\mathbb{E}\left[M \cdot t_{\text{VC,Check}} + \sum_{k \in [M]} T_k \middle| \begin{array}{c} M \leftarrow \mathcal{M}_{\text{pp},\eta} \\ \text{For } k \in [M]: \\ T_k \leftarrow \mathcal{T} \end{array} \middle| \begin{array}{c} \text{pp} \leftarrow \mathcal{G}(1^{\lambda},n) \\ (\text{x, (cm, aux))} \leftarrow \tilde{\mathcal{P}}(\text{pp},\eta) \\ (\mathcal{Q}, \text{ans}, \text{pf}) \leftarrow \tilde{\mathcal{P}}(\text{up},\eta) \right. \\ (\mathcal{Q}, \text{ans}, \text{pf}) \leftarrow \tilde{\mathcal{P}}(\text{aux},\rho) \right. \right] \right] \\ \left. \left(\mathcal{Q}, \text{ans}, \text{pf} \right) \leftarrow \tilde{\mathcal{P}}(\text{aux},\rho) \right] \\ \mathbb{E}\left[\mathbb{E}\left[M \cdot t_{\text{VC,Check}} + \sum_{k \in [M]} \mathbb{E}\left[T_k \mid T_k \leftarrow \mathcal{T}\right] \middle| M \leftarrow \mathcal{M}_{\text{pp},\eta} \right] \middle| \begin{array}{c} \text{pp} \leftarrow \mathcal{G}(1^{\lambda},n) \\ \eta \leftarrow \mathcal{D} \\ (\text{x, (cm, aux))} \leftarrow \tilde{\mathcal{P}}(\text{pp},\eta) \\ (\text{x, (cm, aux))} \leftarrow \tilde{\mathcal{P}}(\text{pp},\eta) \end{array} \right] \right] \\ \left. \mathbb{E}\left[M \cdot (t_{\text{VC,Check}} + t_{\text{ARG}}^{\star}) \mid M \leftarrow \mathcal{M}_{\text{pp},\eta} \right] \middle| \begin{array}{c} \text{pp} \leftarrow \mathcal{G}(1^{\lambda},n) \\ \eta \leftarrow \mathcal{D} \\ (\text{x, (cm, aux))} \leftarrow \tilde{\mathcal{P}}(\text{pp},\eta) \\ (\text{x, (cm, aux))} \leftarrow \tilde{\mathcal{P}}(\text{pp},\eta) \end{array} \right] \\ \left. \mathbb{E}\left[M \cdot (t_{\text{VC,Check}} + t_{\text{ARG}}^{\star}) \cdot \mathbb{E}\left[\xi_{\text{pp},\eta}(q) \cdot \frac{\log \frac{q}{\epsilon}}{\xi_{\text{pp},\eta}(q)} \middle| \begin{array}{c} \text
$$

Proof of Lemma [6.3.](#page-33-3) We consider the following experiment throughout the proof unless otherwise specified:

$$
\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda,n) \\ \eta \leftarrow \mathcal{D} \\ (\mathbf{x}, (\mathsf{cm}, \mathsf{aux})) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho) \\ \text{For } i \in [\mathsf{q}] : \\ (\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}, \mathcal{Q}^{(i)}, \mathsf{ans}^{(i)}, \mathsf{pf}^{(i)}) \leftarrow A_\mathsf{vc}^{(i)}(\mathsf{pp}, \eta, \rho) \end{array}\right]\right]
$$

.

.

Then, we can deduce that

$$
\Pr\left[\begin{array}{l} \mathbf{V}^{\widetilde{\Pi}^*_{\text{PP}},\eta}(\mathbf{x};\rho)=0 \\ \wedge\mathbf{V}^{[\mathcal{Q},\text{ans}]}(\mathbf{x};\rho)=1 \\ \wedge\mathsf{V}\mathsf{C}.\text{Check}(\text{pp},\text{cm},\mathcal{Q},\text{ans},\text{pf})=1 \\ \wedge\forall i\in[q],\text{ans}[\mathcal{Q}(i)]=\text{ans}^{(i)}[\mathcal{Q}(i)] \end{array}\right] \\ \leq \Pr\left[\begin{array}{l} \exists i\in[q],\text{ans}[\mathcal{Q}(i)]\neq \widetilde{\Pi}^*_{\text{PP},\eta}[\mathcal{Q}(i)] \\ \wedge\mathsf{V}\mathsf{C}.\text{Check}(\text{pp},\text{cm},\mathcal{Q},\text{ans},\text{pf})=1 \\ \wedge\forall i\in[q],\text{ans}[\mathcal{Q}(i)]=\text{ans}^{(i)}[\mathcal{Q}(i)] \end{array}\right] \\ \leq \Pr\left[\begin{array}{l} \exists i\in[q],\text{ans}[\mathcal{Q}(i)]=\text{ans}^{(i)}[\mathcal{Q}(i)]\neq \widetilde{\Pi}^*_{\text{PP},\eta}[\mathcal{Q}(i)] \\ \wedge\mathsf{VC}.\text{Check}(\text{pp},\text{cm},\mathcal{Q},\text{ans},\text{pf})=1 \\ \wedge\mathsf{VC}.\text{Check}(\text{pp},\text{cm},\mathcal{Q},\text{ans},\text{pf})=1 \end{array}\right] \\ \leq q\cdot 2^{-\log\frac{q}{\epsilon}}=\epsilon \enspace,
$$

where the last inequality follows because $A_{\text{VC}}^{(i)}$ outputs ans⁽ⁱ⁾ = ans and ans $[Q[i]] \neq \widetilde{\Pi}_{\text{pp},\eta}^{\star}[Q(i)]$, so that $\log \frac{q}{\epsilon}$ uniformly random valid openings matched ans in $\mathcal{Q}[i]$; since $\widetilde{\Pi}^{\star}_{\text{pp},\eta}[\mathcal{Q}(i)]$ is the most frequent answer, ans $[\mathcal{Q}(i)]$ appears with probability at most $1/2$ (otherwise the sum of both probabilities would exceed 1). \Box

6.2 Adaptive soundness

Lemma 6.5. *For every* $\epsilon > 0$, *security parameter* $\lambda \in \mathbb{N}$, *instance size bound* $n \in \mathbb{N}$, *auxiliary input distribution* \mathcal{D} , adversary time bound $t_{\text{ARG}} \ge t_{\text{VC-Check}} + 2q \cdot (\log|\Sigma| + \log \ell)$ and t_{ARG} -size circuit $\widetilde{\mathcal{P}}$, the soundness error of *the argument system in Construction [4.1](#page-24-2) satisfies*

$$
\epsilon_{\text{ARG}}(\lambda,n,t_{\text{ARG}}) \leq \epsilon_{\text{PCP}}(n) + \mathsf{q} \cdot \epsilon^{\star}_{\text{VC}}(\lambda,\ell,\mathsf{q},t^{\star}_{\text{VC}}) + \epsilon \enspace,
$$

where $t_{\text{VC}}^* = O\left(\ell \log \frac{q}{\epsilon} \cdot t_{\text{ARG}}\right)$.

Proof. Recall, from Definition [3.3](#page-20-3) and Construction [4.1,](#page-24-2) that our goal is to upper bound

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

(Since \tilde{P} sends cm as the first message in Construction [4.1,](#page-24-2) 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.)

We consider the following experiment, which augments the above by executing R and $A_{\nu c}^{(i)}$, and thus leaves the probability unchanged.

$$
\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda,n) \\ \eta \leftarrow \mathcal{D} \\ (\mathbf{x}, (\mathsf{cm}, \mathsf{aux})) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho) \\ \text{For } i \in [\mathsf{q}] : \\ (\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}, \mathcal{Q}^{(i)}, \mathsf{ans}^{(i)}, \mathsf{pf}^{(i)}) \leftarrow A_\mathsf{VC}^{(i)}(\mathsf{pp}, \eta, \rho) \end{array}\right]\right]
$$

.

.

By total probability,

$$
\Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\ \wedge \mathbf{x} \notin L(R)\\ \wedge \mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho) = 1\\ \wedge \mathbf{V} \mathbf{C}.\mathrm{Check}(\mathrm{pp},\mathrm{cm},\mathcal{Q},\mathrm{ans},\mathrm{pf}) = 1 \end{array}\right]
$$
\n
$$
= \Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\ \wedge \mathbf{x} \notin L(R)\\ \wedge \mathbf{V}^{\widetilde{\Pi}_{\mathrm{pp},\eta}}(\mathbf{x};\rho) = 1\\ \wedge \mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho) = 1\\ \wedge \mathbf{V} \mathbf{C}.\mathrm{Check}(\mathrm{pp},\mathrm{cm},\mathcal{Q},\mathrm{ans},\mathrm{pf}) = 1 \end{array}\right] + \Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\ \wedge \mathbf{x} \notin L(R)\\ \wedge \mathbf{V}^{\widetilde{\Pi}_{\mathrm{pp},\eta}}(\mathbf{x};\rho) = 0\\ \wedge \mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho) = 1\\ \wedge \mathbf{V} \mathbf{C}.\mathrm{Check}(\mathrm{pp},\mathrm{cm},\mathcal{Q},\mathrm{ans},\mathrm{pf}) = 1 \end{array}\right]
$$

We first bound the probability of the leftmost term by the PCP system's soundness error (i.e., Definition [3.13](#page-22-1) with respect to PCP).

Construction 6.6. We define the auxiliary input distribution **D** of the PCP prover \tilde{P} as follows:

D:

- 1. Sample $pp \leftarrow \mathcal{G}(1^{\lambda}, n)$ and $\eta \leftarrow \mathcal{D}$.
- 2. Output $\mathbf{ai} := (pp, \eta)$.

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

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

- 1. Parse ai as (pp, η) .
- 2. Run $(\mathbf{x}, \mathsf{aux}_0) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta)$.
- 3. Set $aux := (pp, \eta, aux_0)$.
- 4. Output (x, aux) .

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

- 1. Parse aux as (pp, η, aux_0) .
- 2. Run (cm, aux_1) $\leftarrow \widetilde{\mathcal{P}}(aux_0)$.
- 3. Run $(Q_\rho, \text{ans}_\rho, \text{pf}_\rho) \leftarrow \widetilde{\mathcal{P}}(\text{aux}_1, \rho)$ for all $\rho \leftarrow \{0, 1\}^r$.
- 4. Construct and output $\widetilde{\Pi}_{\mathsf{pp},\eta}^{\star}$.^{[16](#page-0-0)}

¹⁶Recall that, by Definition [6.2,](#page-33-4) $\tilde{\Pi}^*_{\text{pp},\eta}[q] = \arg \max_{\sigma \in \Sigma} \{p_{\text{pp},\eta}(\text{p},\eta)(q,\sigma)\}\text{, where}$

 $p_{\mathsf{pp},\eta}(q,\sigma) = 2^{-\mathsf{r}} \cdot |\big\{ \rho \in \{0,1\}^{\mathsf{r}} : q \in \mathcal{Q}_\rho \wedge \mathsf{ans}_\rho[q] = \sigma \wedge \mathsf{VC}.\mathsf{Check}(\mathsf{pp},\mathsf{cm},\mathcal{Q}_\rho,\mathsf{ans}_\rho,\mathsf{pf}_\rho) = 1 \big\}|.$

Using Definition [3.13,](#page-22-1)^{[17](#page-0-0)}

$$
\Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\ \wedge \mathbf{x} \notin L(R)\\ \wedge \mathbf{V}^{\tilde{\Pi}_{\mathsf{pp},\eta}}(\mathbf{x};\rho) = 1\\ \wedge \mathbf{V}^{[\mathcal{Q},\mathrm{ans}]}(\mathbf{x};\rho) = 1\\ \wedge \mathbf{V} \mathsf{C}.\mathrm{Check}(\mathsf{pp},\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf}) = 1 \end{array}\right] \leq \Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\ \wedge \mathbf{x} \notin L(R)\\ \wedge \mathbf{V}^{\tilde{\Pi}_{\mathsf{pp},\eta}}(\mathbf{x};\rho) = 1 \end{array}\right]
$$
\n
$$
= \Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\ \wedge \mathbf{V}^{\tilde{\Pi}_{\mathsf{pp},\eta}}(\mathbf{x};\rho) = 1 \end{array}\right] \quad \begin{array}{l}\text{ai} \leftarrow \mathbf{D} \\ \wedge \mathbf{x} \notin L(R)\\ \wedge \mathbf{V}^{\tilde{\Pi}_{\mathsf{pp},\eta}}(\mathbf{x}) = 1 \end{array}\right] \quad \begin{array}{l}\text{ai} \leftarrow \mathbf{D} \\ (\mathbf{x},\mathbf{aux}) \leftarrow \tilde{\mathbf{P}}(\mathbf{ai})\\ \wedge \mathbf{V}^{\tilde{\Pi}_{\mathsf{pp},\eta}}(\mathbf{x}) = 1 \end{array}\right] \quad \begin{array}{l}\n\text{ai} \leftarrow \mathbf{D} \\ (\mathbf{x},\mathbf{aux}) \leftarrow \tilde{\mathbf{P}}(\mathbf{ai})\\ \tilde{\Pi} \leftarrow \tilde{\mathbf{P}}(\mathbf{aux})\n\end{array}\right]\n\end{array}
$$

Then we bound the remaining term. By total probability,

$$
\Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\[3mm]\wedge\mathbf{x} \notin L(R)\\[3mm]\wedge\mathbf{V}^{[\Omega,\mathrm{ans}]}(\mathbf{x};\rho) = 1\\[3mm]\wedge\mathbf{V} \subset \mathrm{Check}(pp,\mathrm{cm},\mathcal{Q},\mathrm{ans},\mathrm{pf}) = 1 \end{array}\right]
$$
\n
$$
\leq \Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\[3mm]\wedge\mathbf{x} \notin L(R)\\[3mm]\wedge\mathbf{V}^{[\Omega,\mathrm{ans}]}(\mathbf{x};\rho) = 0\\[3mm]\wedge\mathbf{V}^{[\Omega,\mathrm{ans}]}(\mathbf{x};\rho) = 0\\[3mm]\wedge\mathbf{V}^{[\Omega,\mathrm{ans}]}(\mathbf{x};\rho) = 1\\[3mm]\wedge\mathbf{V} \subset \mathrm{Check}(pp,\mathrm{cm},\mathcal{Q},\mathrm{ans},\mathrm{pf}) = 1\\[3mm]\wedge\mathbf{V} \in [q],\mathrm{ans}[\mathcal{Q}[i]] = \mathrm{ans}^{(i)}[\mathcal{Q}[i]] \end{array}\right]+\Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\[3mm]\wedge\mathbf{x} \notin L(R)\\[3mm]\wedge\mathbf{V}^{[\Omega,\mathrm{ans}]}(\mathbf{x};\rho) = 0\\[3mm]\wedge\mathbf{V}^{[\Omega,\mathrm{ans}]}(\mathbf{x};\rho) = 1\\[3mm]\wedge\mathbf{V} \subset \mathrm{Check}(pp,\mathrm{cm},\mathcal{Q},\mathrm{ans},\mathrm{pf}) = 1\\[3mm]\wedge\mathbf{V} \subset \mathrm{Check}(pp,\mathrm{cm},\mathcal{Q},\mathrm{ans},\mathrm{pf}) = 1\\[3mm]\wedge\exists i \in [q],\mathrm{ans}[\mathcal{Q}[i]] \neq \mathrm{ans}^{(i)}[\mathcal{Q}[i]] \end{array}\right],
$$

where the last inequality follows from Lemma [6.3.](#page-33-3) The last remaining term can be upper bounded by q · $\epsilon_{\text{VC}}^{\star}(\lambda, \ell, \mathsf{q}, t_{\text{VC}}^{\star})$, as we explain below.

According to Definition [3.10,](#page-21-3) and by the fact that VC. Check(cm, $\mathcal{Q}^{(i)}$, ans $^{(i)}$, pf $^{(i)}$) = 1 for all i (by construction of $A_{\text{VC}}^{(i)}$),

$$
\Pr\left[\begin{array}{l} \mathsf{VC}.Check(\mathsf{pp},\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1\\[.2em] \begin{aligned} &\wedge \exists i \in [\mathsf{q}],\mathsf{ans}[\mathcal{Q}[i]] \neq \mathsf{ans}^{(i)}[\mathcal{Q}[i]] \end{aligned}\right] \\ &=\Pr\left[\begin{array}{l} \mathsf{VC}.Check(\mathsf{pp},\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1\\[.2em] &\wedge \mathsf{VC}.Check(\mathsf{pp},\mathsf{cm},\mathcal{Q}^{(i)},\mathsf{ans}^{(i)},\mathsf{pf}^{(i)})=1\\[.2em] &\wedge \exists i \in [\mathsf{q}],\mathsf{ans}[\mathcal{Q}[i]] \neq \mathsf{ans}^{(i)}[\mathcal{Q}[i]] \end{array}\right] \\ &\leq \sum_{i \in [\mathsf{q}]} \Pr\left[\begin{array}{l} \mathsf{VC}.Check(\mathsf{pp},\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1\\[.2em] &\wedge \exists i \in [\mathsf{q}],\mathsf{ans}[\mathcal{Q}[i]] \neq \mathsf{ans}^{(i)}[\mathcal{Q}[i]] \end{array}\right] \\ &\leq \sum_{i \in [\mathsf{q}]} \Pr\left[\begin{array}{l} |\mathcal{Q}|=|\mathcal{Q}^{(i)}|= \mathsf{q}\\[.2em] &\wedge \exists q \in \mathcal{Q} \cap \mathcal{Q}^{(i)}: \mathsf{ans}[q] \neq \mathsf{ans}^{(i)}[q] \\[.2em] &\wedge \mathsf{VC}.Check(\mathsf{pp},\mathsf{cm},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1\\[.2em] &\wedge \mathsf{VC}.Check(\mathsf{cp},\mathsf{cm},\mathcal{Q}^{(i)},\mathsf{ans}^{(i)},\mathsf{pf}^{(i)})=1 \end{array}\right]\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{VC}.Gen(1^{\lambda},\ell) \\[.2em] &\rho \leftarrow \{0,1\}^r \\[.2em] &\mathsf{cm},\mathsf{ans},\mathsf{ans}^{(i)}, \end{array}\right) \leftarrow A_{\mathsf{VC}}^{(i)}(\mathsf
$$

 17 Note that the prover in Definition [3.13](#page-22-1) corresponds to the sequential execution of both steps in Construction [6.6.](#page-36-0)

$$
\leq \sum_{i \in [q]} \epsilon^{\star}_{\text{VC}}(\lambda, \ell, q, t^{ \star}_{\text{VC}}) \\ = q \cdot \epsilon^{\star}_{\text{VC}}(\lambda, \ell, q, t^{ \star}_{\text{VC}}) \enspace,
$$

where $t_{\text{VC}}^* = O\left(\ell t_{\text{ARG}} \log \frac{q}{\epsilon}\right)$ is the expected runtime of the adversary that samples ρ then runs $A_{\text{VC}}^{(i)}(\text{pp}, \eta, \rho)$.

7 Expected-time soundness analysis for PCPs with non-adaptive verifiers

Theorem 7.1. *Consider these two ingredients:*

- PCP = (P, V) *, a PCP system with non-adaptive verifier for a relation R with alphabet* Σ *, proof length* ℓ *, and query complexity* q*; and*
- $VC = (Gen, Committee, Open, Check), a vector commitment scheme over alphabet Σ .$

Then $ARG = (G, P, V) :=$ Kilian $[PCP, VC]$ *is a three-message public-coin interactive argument system for* R, ϵ xpected-time soundness error $\epsilon_{\text{ARG}}^{\star}$ satisfies the following for every $\epsilon > 0$ and $t_{\text{ARG}}^{\star} \ge t_{\text{VC,Check}} + \log|\Sigma| + \log \ell$.

 $\epsilon^{\star}_{\texttt{ARG}}(\lambda, n, t^{\star}_{\texttt{ARG}}) \leq \epsilon_{\texttt{PCP}}(n) + \mathsf{q} \cdot \epsilon^{\star}_{\texttt{VC}}(\lambda, \ell, \mathsf{q}, t^{\star}_{\texttt{VC}}) + \epsilon \enspace ,$

where ϵ_{PCP} is the soundness error of PCP and $t_{\text{VC}}^* = O\left(\log \frac{q}{\epsilon} \cdot (t_{\text{ARG}}^* + \ell \cdot t_{\text{V}})\right)$.

7.1 Security reduction

We prove the same security reduction lemma as in Section [6,](#page-33-0) but an improved bound on the expected running time of the VC adversaries. The corresponding improvement on ϵ_{ARG} then follows by the same argument as in Section [6.](#page-33-0)

Lemma 7.2 (Alternative security reduction lemma). *There exist probabilistic algorithms* $A_{\text{VC}}^{(i)}$ *for each* $i \in [q]$ $\textit{such that, for every } C \in \mathbb{N}, \textit{adversary time bound } t^{\star}_{\text{ARG}} \geq t_{\text{VC.Check}} + \log|\Sigma| + \log \ell \textit{ and expected } t^{\star}_{\text{ARG}}\textit{-time}$ *adversary* ^Pe*, satisfies*

$$
\Pr\left[\begin{array}{c} \mathbf{V}^{\widetilde{\Pi}_{\text{pp},\eta}^{\star}(\mathbf{x};\rho)=0 \\ \wedge\mathbf{V}^{[\mathcal{Q},\text{ans}]}(\mathbf{x};\rho)=1 \\ \wedge\mathbf{V}c\text{.Check}(pp,cm,\mathcal{Q},\text{ans},\text{pf})=1 \\ \wedge\forall i\in[q],\text{ans}[\mathcal{Q}[i]]=\text{ans}^{(i)}[\mathcal{Q}[i]] \end{array}\right]\left[\begin{array}{c} \mathsf{pp}\leftarrow\mathcal{G}(1^{\lambda},n) \\ \left(\mathbf{x},(\mathsf{cm},\mathsf{aux})\right)\leftarrow\widetilde{\mathcal{P}}(\mathsf{pp},\eta) \\ \rho\leftarrow\{0,1\}^r \\ (\mathcal{Q},\text{ans},\text{pf})\leftarrow\widetilde{\mathcal{P}}(\text{aux},\rho) \\ \text{For }i\in[q]: \\ \left(\begin{array}{c} \mathsf{cm},\mathcal{Q},\text{ans},\text{pf}, \\ \mathcal{Q}^{(i)},\text{ans}^{(i)},\text{pf}^{(i)} \end{array}\right)\leftarrow A_{\text{VC}}^{(i)}(\mathsf{pp},\eta,\rho) \end{array}\right]\leq \epsilon \right].
$$

Moreover, $A_{\text{VC}}^{(i)}$ runs in expected time $t_{\text{VC}}^{\star} = O\left(\log \frac{\mathsf{q}}{\epsilon} \cdot (t_{\text{ARG}}^{\star} + \ell \cdot t_{\text{V}})\right)$ for all i.

We construct $A_{\text{VC}}^{(i)}$ below.

Construction 7.3. Given an argument prover \tilde{P} , we construct each adversary $A_{VC}^{(i)}$ as follows.

 $A_{\mathsf{VC}}^{(i)}(\mathsf{pp},\eta,\rho)$:

1. Run $(\mathbf{x}, \text{cm}, \text{aux}) \leftarrow \tilde{\mathcal{P}}(\text{pp}, \eta)$ and $(\mathcal{Q}, \text{ans}, \text{pf}) \leftarrow \tilde{\mathcal{P}}(\text{aux}, \rho)$.

- 2. Check that VC.Check(pp, cm, Q, ans, pf) = 1. If not, output (cm, ans, ans, Q, Q, pf, pf).
- 3. Define $q := \mathcal{Q}(i)$ and set $j := 0$.
- 4. Repeat the following:
	- (a) Repeat the following:
		- i. Sample $\rho' \leftarrow \{0, 1\}^r$.
		- ii. Run the PCP verifier V to obtain the query set $\mathcal{Q}'(\rho')$ corresponding to ρ' .
		- iii. If $Q'(i) = q$, exit this loop.
	- (b) Run $(Q', \text{ans}', \text{pf}') \leftarrow \widetilde{P}(\text{aux}, \rho').$
	- (c) If VC. Check(pp, cm, Q' , ans', pf') = 1:
		- i. If ans $[q]\neq$ ans $'[q],$ output $(\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}, \mathcal{Q}', \mathsf{ans}', \mathsf{pf}').$

ii. If ans $[q] = \text{ans}'[q]$, set $j \coloneqq j + 1$. Further, if $j = \log \frac{q}{\epsilon}$, output (cm, ans, ans, $\mathcal{Q}, \mathcal{Q}, \mathsf{pf}, \mathsf{pf}$).

We first analyze the expected running time of $A_{\text{VC}}^{(i)}$. For every (pp, η) and $q \in [\ell]$, we define $p_{\text{pp},\eta}^{(i)}(q)$ be the probability that verifier's i -th query is q for a uniformly sampled randomness:

$$
p_{\mathsf{pp},\eta}^{(i)}(q) := \Pr\left[\mathcal{Q}(i) = q \mid \begin{array}{l} (\mathbf{x}, (\mathsf{cm}, \mathsf{aux})) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta) \\ \rho \leftarrow \{0,1\}^r \\ (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux}, \rho) \end{array}\right] .
$$

For every (pp, η) and $q \in [\ell]$, we define $\psi_{\mathsf{pp},\eta}^{(i)}(q)$ as follows:

$$
\psi_{\mathsf{pp},\eta}^{(i)}(q) \coloneqq \Pr\left[\begin{array}{l}\mathsf{VC}. \mathsf{Check}(\mathsf{pp},\mathcal{Q},\mathsf{ans},\mathsf{pf})=1\\ \text{conditioned on}\\ \mathcal{Q}(i)=q\end{array}\middle|\begin{array}{l}(\mathbb{x},(\mathsf{cm},\mathsf{aux}))\leftarrow\widetilde{\mathcal{P}}(\mathsf{pp},\eta)\\ \rho\leftarrow\{0,1\}^r\\ (\mathcal{Q},\mathsf{ans},\mathsf{pf})\leftarrow\widetilde{\mathcal{P}}(\mathsf{aux},\rho)\end{array}\right]\right].
$$

Finally, denote by X, \mathcal{T} and $\mathcal{M}_{\text{pp},\eta}$ the distribution of the running time of Item [4a,](#page-15-3) distributions of the running time of \tilde{P} and the number of iterations of Step [4](#page-34-0) in an execution of $A_{\rm VC}(pp, \eta, \rho)$, respectively. Then the running time of $A_{\text{VC}}^{(i)}$ is

$$
O\left(M \cdot t_{\text{VC.Check}} + \sum_{k \in [M]} X_k + \sum_{k \in [M]} T_k\right) ,
$$

where $X_k \leftarrow \mathcal{X}$ and $T_k \leftarrow \mathcal{T}$ for all $k \in [M]$. Therefore,

$$
\mathbb{E}\left[\mathbb{E}\left[M \cdot t_{\text{VC,Check}} + \sum_{k \in [M]} X_k + \sum_{k \in [M]} T_k \middle| \begin{array}{c} M \leftarrow M_{\text{pp},\eta} \\ \text{For } k \in [M]: \\ X_k \leftarrow \mathcal{X} \\ T_k \leftarrow \mathcal{T} \end{array}\right] \middle| \begin{array}{c} \text{pp} \leftarrow \mathcal{G}(1^{\lambda}, n) \\ (\text{x, (cm, aux)}) \leftarrow \tilde{\mathcal{P}}(\text{pp}, \eta) \\ (\text{x, (cm, aux)}) \leftarrow \tilde{\mathcal{P}}(\text{pp}, \eta) \right] \right] \\ (\text{x, (cm, aux)}) \leftarrow \tilde{\mathcal{P}}(\text{pp}, \eta) \right] \\ (\text{Q, ans, pf}) \leftarrow \tilde{\mathcal{P}}(\text{aux}, \rho) \right] \\ \mathbb{P} \left[\mathbb{E}\left[M \cdot (t_{\text{VC,Check}} + t_{\text{ARC}}^{\star}) + \sum_{k \in [M]} \mathbb{E}\left[X_k \mid X_k \leftarrow \mathcal{X}\right] \middle| M \leftarrow M_{\text{pp},\eta} \right] \middle| \begin{array}{c} \text{pp} \leftarrow \mathcal{G}(1^{\lambda}, n) \\ \text{p} \leftarrow \{0, 1\}^r \\ (\text{x, (cm, aux)}) \leftarrow \tilde{\mathcal{P}}(\text{pu}, \eta) \end{array}\right] \right] \\ = \mathbb{E}\left[\sum_{q \in [\ell]} p_{\text{pp},\eta}^{(i)}(q) \cdot \psi_{\text{pp},\eta}^{(i)}(q) \cdot \left(t_{\text{VC,Check}} + t_{\text{ARC}}^{\star} + \frac{t_{\text{V}}}{p_{\text{pp},\eta}^{(i)}(q)}\right) \cdot \mathbb{E}\left[M \mid M \leftarrow M_{\text{pp},\eta}\right] \middle| \begin{array}{c} \text{pp} \leftarrow \mathcal{G}(1^{\lambda}, n) \\ \text{p} \leftarrow \{0, 1\}^r \\ \text{p} \leftarrow \mathcal{P}(1^{\lambda}, n) \end{array}\right] \\ = \mathbb{E}\left[\sum_{q \in [\ell]} p_{\text{pp},\eta}^{(i)}(q) \cdot \psi_{\text{pp},\eta}^{(i)}(q) \cdot \left(t_{\text{VC,Check}} + t_{\text{ARC}}^
$$

Proof of Lemma [6.3.](#page-33-3) We consider the following experiment throughout the proof unless otherwise specified:

$$
\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda,n) \\ \eta \leftarrow \mathcal{D} \\ (\mathbf{x}, (\mathsf{cm}, \mathsf{aux})) \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp},\eta) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ (\mathcal{Q}, \mathsf{ans}, \mathsf{pf}) \leftarrow \widetilde{\mathcal{P}}(\mathsf{aux},\rho) \\ \text{For } i \in [\mathsf{q}]: \\ (\mathsf{cm}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}, \mathcal{Q}^{(i)}, \mathsf{ans}^{(i)}, \mathsf{pf}^{(i)}) \leftarrow A_\text{VC}^{(i)}(\mathsf{pp},\eta,\rho) \end{array}\right]\right]
$$

.

Then, we can deduce that

$$
\Pr\left[\begin{array}{l} \mathbf{V}^{\widetilde{\Pi}^*_{\text{PP}},\eta}(\mathbf{x};\rho)=0 \\ \wedge\mathbf{V}^{[\mathcal{Q},\text{ans}]}(\mathbf{x};\rho)=1 \\ \wedge\mathsf{V}\mathsf{C}.\text{Check}(\text{pp},\text{cm},\mathcal{Q},\text{ans},\text{pf})=1 \\ \wedge\forall i\in[q],\text{ans}[\mathcal{Q}(i)]=\text{ans}^{(i)}[\mathcal{Q}(i)] \end{array}\right] \\ \leq \Pr\left[\begin{array}{l} \exists i\in[q],\text{ans}[\mathcal{Q}(i)]\neq \widetilde{\Pi}^*_{\text{PP},\eta}[\mathcal{Q}(i)] \\ \wedge\mathsf{V}\mathsf{C}.\text{Check}(\text{pp},\text{cm},\mathcal{Q},\text{ans},\text{pf})=1 \\ \wedge\forall i\in[q],\text{ans}[\mathcal{Q}(i)]=\text{ans}^{(i)}[\mathcal{Q}(i)] \end{array}\right] \\ \leq \Pr\left[\begin{array}{l} \exists i\in[q],\text{ans}[\mathcal{Q}(i)]=\text{ans}^{(i)}[\mathcal{Q}(i)]\neq \widetilde{\Pi}^*_{\text{PP},\eta}[\mathcal{Q}(i)] \\ \wedge\mathsf{VC}.\text{Check}(\text{pp},\text{cm},\mathcal{Q},\text{ans},\text{pf})=1 \\ \wedge\mathsf{VC}.\text{Check}(\text{pp},\text{cm},\mathcal{Q},\text{ans},\text{pf})=1 \end{array}\right] \\ \leq q\cdot 2^{-\log\frac{q}{\epsilon}}=\epsilon \enspace,
$$

where the last inequality follows because $A_{\text{VC}}^{(i)}$ outputs ans⁽ⁱ⁾ = ans and ans $[Q[i]] \neq \widetilde{\Pi}_{\text{pp},\eta}^{\star}[Q(i)]$, so that $\log \frac{q}{\epsilon}$ uniformly random valid openings matched ans in $\mathcal{Q}[i]$; since $\widetilde{\Pi}^{\star}_{\text{pp},\eta}[\mathcal{Q}(i)]$ is the most frequent answer, ans $[\mathcal{Q}(i)]$ appears with probability at most $1/2$ (otherwise the sum of both probabilities would exceed 1).

8 Lower bounds from Schnorr identification scheme

In this section, we show that a generic bound (i.e., for arbitrary PCP and VC) for the security of Kilian's argument system implies the same bound for the security of the Schnorr identification scheme [\[Sch89;](#page-50-0) [Sch91\]](#page-50-1).

8.1 Schnorr identification scheme

An identification scheme (ID scheme) for a relation R consists of a tuple $ID = (G, P, V, C)$ that works as follows:

 $G(1^{\lambda})$: Sample an instance-witness pair $(x, w) \leftarrow G(1^{\lambda})$ in the relation R.

The prover P and verifier V interact as follows:

- 1. P's first message: $(\alpha, \text{aux}) \leftarrow P(\mathbf{x}, \mathbf{w})$, then P sends α to V.
- 2. V's challenge: sample $\beta \leftarrow \mathbf{C}$ and send β to P.
- 3. P's second message: $\gamma \leftarrow P(\mathbf{aux}, \beta)$ and send γ to V.
- 4. V's decision: output $V(\mathbf{x}, \alpha, \beta, \gamma)$.

Definition 8.1 (Impersonation security). $ID = (G, P, V, C)$ *for a relation R has (strict-time) impersonation error* ϵ_{ID} *if for every security parameter* $\lambda \in \mathbb{N}$ *, adversary time bound* t_{ID} *, and* t_{ID} *-time adversary* P*,*

$$
\Pr\left[V(\textbf{x},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma})=1\left|\begin{array}{c}(\textbf{x},\textbf{w})\leftarrow G(1^{\lambda})\\\left(\boldsymbol{\alpha}, \textbf{aux}\right)\leftarrow \widetilde{P}(1^{\lambda},\textbf{x})\\\boldsymbol{\beta} \leftarrow \mathbf{C}\\\boldsymbol{\gamma}\leftarrow \widetilde{P}(\textbf{aux},\boldsymbol{\beta})\end{array}\right.\right\}\leq \epsilon_{\text{ID}}(\lambda,t_{\text{ID}})\enspace.
$$

In prior literature, security against passive impersonation attacks assume adversaries that have access to an oracle that produces honestly-generated transcripts for the instance x. However, in this work we only consider identification schemes that are *simulatable*: there exists an efficient algorithm that samples a transcript from the distribution of honest executions of the protocol on input (x, w) . Hence, it is equivalent to define the adversaries without the oracle access because they can simulate the oracle themselves.

Definition 8.2 (Expected-time impersonation security). $ID = (G, P, V, C)$ *for a relation* R *has (expected-time)* $impersonation error \epsilon_{ID}^{\star}$ *if for every security parameter* $\lambda \in \mathbb{N}$, *adversary time bound* $t_{ID}^{\star} \in \mathbb{N}$, *and adversary* \widetilde{P} with expected running time t_{ID}^{\star} ,

$$
\Pr\left[\mathsf{V}(\mathbf{x}, \alpha, \beta, \gamma) = 1 \left| \begin{array}{c} (\mathbf{x}, \mathbf{w}) \leftarrow \mathsf{G}(1^{\lambda}) \\ (\alpha, \mathbf{aux}) \leftarrow \widetilde{\mathsf{P}}(1^{\lambda}, \mathbf{x}) \\ \beta \leftarrow \mathbf{C} \\ \gamma \leftarrow \widetilde{\mathsf{P}}(\mathbf{aux}, \beta) \end{array} \right. \right] \leq \epsilon_{\mathsf{ID}}^{\star}(\lambda, t_{\mathsf{ID}}^{\star}) \enspace.
$$

Now we recall the construction of the Schnorr identification scheme. We rely on a group generation algorithm GroupGen that takes as input the security parameter 1^{λ} and outputs a description (\mathbb{G}, p, g) of a cyclic group \mathbb{G} of prime order p and the generator g of \mathbb{G} . In particular, the security parameter λ determines a lower bound on the group order: $p \ge 2^{\lambda}$.

Construction 8.3 (Schnorr identification scheme). $ID_{Schner} = (G, P, V, C)$ works as follows:

- $G(1^{\lambda})$:
	- 1. $(\mathbb{G}, p, g) \leftarrow$ GroupGen (1^{λ}) .
	- 2. Sample w $\leftarrow \mathbb{Z}_p$.
	- 3. Set $x := ((\mathbb{G}, p, g), g^w)$.

4. Output (x, w) .

- $P(x, w)$:
	- 1. Parse \mathbf{x} as $((\mathbb{G}, p, q), h)$.
	- 2. Sample $r \leftarrow \mathbb{Z}_p$.
	- 3. Compute the first message $\alpha := g^r$.
	- 4. Set $aux := (w, r)$.
	- 5. Output (α, \mathbf{aux}) .
- P(aux, β):
	- 1. Parse aux as (w, r) .
	- 2. Output $\gamma \coloneqq \mathbf{w} \cdot \beta + r \bmod p$.
- $V(\mathbf{x}, \alpha, \beta, \gamma)$:
	- 1. Parse x as $((\mathbb{G}, p, g), h)$.
	- 2. Check that $g^{\gamma} = \alpha \cdot h^{\beta}$.

The challenge space C is \mathbb{Z}_p .

The security of the Schnorr identification scheme comes from the hardness of discrete logarithm problem.

Definition 8.4 (Discrete logarithm assumption). *The discrete logarithm assumption holds with error* ϵ_{DLOG} *if for every security parameter* λ *, adversary time bound* $t_{DLOG} \in \mathbb{N}$ *and* t_{DLOG} *-size adversary* A_{DLOG} *,*

$$
\Pr\left[x = g^y \left| \begin{array}{c} (\mathbb{G}, p, g) \leftarrow \text{GroupGen}(1^{\lambda}) \\ x \leftarrow \mathbb{Z}_p^{\times} \\ y \leftarrow \mathbf{A}_{\text{DLOG}}((\mathbb{G}, p, g), x) \end{array} \right] \right| \leq \epsilon_{\text{DLOG}}(\lambda, t_{\text{DLOG}}) .
$$

Definition 8.5 (expected-time discrete logarithm assumption). *The expected-time discrete logarithm assumption holds with error* ϵ_{DLOG}^* *if for every security parameter* λ , adversary time bound $t_{\text{DLOG}}^* \in \mathbb{N}$ and adversary A_{DLOG} with expected running time t_{DLOG}^{\star} ,

$$
\Pr\left[x = g^y \middle| \begin{array}{c} (\mathbb{G}, p, g) \leftarrow \text{GroupGen}(1^{\lambda}) \\ x \leftarrow \mathbb{Z}_p^{\times} \\ y \leftarrow \mathbf{A}_{\text{DLOG}}((\mathbb{G}, p, g), x) \end{array} \right] \le \epsilon_{\text{DLOG}}^{\star}(\lambda, t_{\text{DLOG}}^{\star}) \end{array} \right.
$$

8.2 From Kilian to Schnorr

We show that the security of the Schnorr identification scheme is bounded by the soundness error of an argument system constructed by Kilian's construction (Construction [4.1\)](#page-24-2).

Theorem 8.6. *There exists* PCP *and* VC *such that for every* $n \in \mathbb{N}$ *,*

$$
\epsilon_{\text{Schnorr}}(\lambda,\mathbf{t}_{\text{ID}}) \leq \epsilon_{\text{ARG}}(\lambda,n,t_{\text{ARG}}) ,
$$

where ϵ_{Schnor} *is the impersonation security of the Schnorr identification scheme,* ϵ_{ARG} *is the soundness error of* ARG := Kilian [PCP, VC] *and* $t_{\text{ARG}} = O(t_{\text{ID}})$.

Theorem 8.7. *There exists* PCP *and* VC *such that for every* $n \in \mathbb{N}$ *,*

$$
\epsilon_\text{Schnor}^\star(\lambda,t_\text{ID}^\star) \leq \epsilon_{\text{ARG}}^\star(\lambda,n,t_{\text{ARG}}^\star) \enspace,
$$

where $\epsilon_{\text{Schnor}}^{\star}$ is the expected-time impersonation security of the Schnorr identification scheme, $\epsilon_{\text{ARG}}^{\star}$ is the *expected-time soundness error of* ARG $:=$ Kilian[PCP, VC] *and* $t_{\text{ARG}} = O(t_{\text{ID}})$ *.*

We construct the PCP and VC in Theorems [8.6](#page-43-1) and [8.7](#page-43-2) below.

Construction 8.8. Consider the empty relation $R = \emptyset$. Fix $r \in \mathbb{N}$. We construct PCP = (P, V) for R as follows:

- $\mathbf{P}(\mathbf{x}, \mathbf{w})$: Output 0^r.
- $\mathbf{V}^{\Pi}(\mathbf{x})$:
	- 1. Sample randomness $\rho \leftarrow \{0, 1\}^r$.
	- 2. Check that $\Pi[1] = \rho$.

Observe that PCP has alphabet $\Sigma = \{0, 1\}^r$, proof length $\ell = 1$ and query complexity $q = 1$. Moreover, it's easy to show that the soundness error of PCP is $\epsilon_{\text{PCP}}(n) = 2^{-r}$.

Construction 8.9 (VC scheme from Schnorr). We construct a VC scheme as follows:

- VC.Gen (1^{λ}) :
	- 1. Run $(\mathbb{G}, p, g) \leftarrow$ GroupGen (1^{λ}) .
	- 2. Sample w $\leftarrow \mathbb{Z}_p$.
	- 3. Output pp := $((\mathbb{G}, p, g), g^{\mathbb{W}})$.
- VC.Commit (pp, m) :
	- 1. Parse pp as $((\mathbb{G}, p, g), h)$.
	- 2. Sample $r \leftarrow \mathbb{Z}_p$.
	- 3. Set cm $:= g^r$.
	- 4. Set aux $:= (r, m)$.
	- 5. Output (cm, aux).
- VC.Open(pp, aux, $\{1\}$):
	- 1. Parse aux as (r, m) .
	- 2. Output pf $:= r + m$.
- VC. Check(pp, cm, Q , ans, pf):
	- 1. Parse pp as $((\mathbb{G}, p, q), h)$.
	- 2. Check that $g^{pf} = \text{cm} \cdot h^{\text{ans}}$.

Lemma 8.10 (Position binding of Construction [8.9\)](#page-44-0). *Assume that the discrete logarithm assumption (Definition* [8.4\)](#page-43-3) *holds with error* $\epsilon_{DLOG} = \epsilon_{DLOG}(\lambda, t_{DLOG})$ *. Let* VC *be constructed as in Construction* [8.9.](#page-44-0) *Then* VC *has binding error* ϵ_{VC} *such that*

$$
\epsilon_{\text{VC}}(\lambda, \ell = 1, s = 1, t_{\text{VC}}) \leq \epsilon_{\text{DLOG}}(\lambda, t_{\text{VC}}) .
$$

Proof. Let $A_{\nu c}$ be an adversary for VC. We construct an adversary $A_{\nu c}$ for discrete log.

 $\mathbf{A}_{\text{DLOG}}((\mathbb{G},p,q),h)$: 1. Set $pp \coloneqq ((\mathbb{G}, p, q), h)$. 2. Run $(cm, {1}, {1},$ ans, ans', pf, pf') $\leftarrow A_{\text{VC}}(pp)$. 3. Set $y := (pf' - pf) \cdot (ans' - ans)^{-1}$. 4. Output y.

If $A_{\nu c}$ succeeds, then ans \neq ans', $g^{\rho f} = \text{cm} \cdot h^{\text{ans}}$ and $g^{\rho f'} = \text{cm} \cdot h^{\text{ans'}}$. Therefore, y is well-defined and $g^y = h$, so \mathbf{A}_{DLOG} succeeds as $g^{\beta} = g^{y \cdot c} = h^c = \alpha \cdot h^c$. \Box Lemma 8.11 (expected-time position binding of Construction [8.9\)](#page-44-0). *Assume that the expected-time discrete logarithm assumption (Definition [8.5\)](#page-43-4) holds with error* $\epsilon_{DLOG}^* = \epsilon_{DLOG}^*(\lambda, t_{DLOG}^*)$. Let VC *be constructed as in Construction* [8.9.](#page-44-0) Then VC has expected-time position binding error $\epsilon_{\text{VC}}^{\star}$ such that

$$
\epsilon^{\star}_{\text{VC}}(\lambda,\ell=1,s=1,t^{\star}_{\text{VC}}) \leq \epsilon^{\star}_{\text{DLOG}}(\lambda,t^{\star}_{\text{VC}}) \ .
$$

The proof of Lemma [8.11](#page-44-1) is the same as the proof of Lemma [8.10.](#page-44-2)

Proof of Theorem [8.6.](#page-43-1) Let PCP be constructed as in Construction [8.8](#page-44-3) and VC be constructed as in Construc-tion [8.9.](#page-44-0) Let $ARG :=$ Kilian $[PCP, VC]$.

Let \tilde{P} be an adversary for the Schnorr identification scheme with running time t_{ID} and success probability $\epsilon_{Schnor}(\lambda, t_{ID})$. We construct an argument adversary $\widetilde{\mathcal{P}}$ as follows:

$$
\mathcal{P}(\mathsf{pp})
$$
:

- 1. \mathcal{P} 's commitment:
	- (a) Parse pp as $((\mathbb{G}, p, g), h)$.
	- (b) Run $(\alpha, \text{aux}) \leftarrow P((\mathbb{G}, p, q), h).$
	- (c) Output $(x, cm, aux) := (\perp, \alpha, aux).$
- 2. $\widetilde{\mathcal{P}}$'s response given verifier randomness ρ :
	- (a) Run $\gamma \leftarrow \widetilde{P}(\mathbf{aux}, \rho)$.
	- (b) Output ($Q := \{1\}$, ans $:= \rho$, pf $:= \beta$).

Note that the running time of $\widetilde{\mathcal{P}}$ is $t_{\text{ARG}} = O(t_{\text{ID}})$. It follows that

$$
\epsilon_{\text{ARC}}(\lambda, n, t_{\text{ARC}}) \geq \Pr\left[\begin{array}{l}|\mathbf{x}| \leq n\\ \wedge \mathbf{x} \notin L(R)\\ \wedge \mathcal{V}(\mathsf{pp}, \mathbf{x}, \mathcal{Q}, \mathsf{ans}, \mathsf{pf}) = 1\\ \end{array}\middle|\begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}, n)\\ \langle \mathbf{x}, \mathsf{cm}, \mathsf{aux} \rangle \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta)\\ \langle \mathcal{Q}, \mathsf{ans}, \mathsf{pf} \rangle \leftarrow \widetilde{\mathcal{P}}(\mathcal{P}, \rho)\end{array}\right]\right]
$$
\n
$$
= \Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}, n)\\ \mathsf{p} \leftarrow \{0, 1\}^{r}\\ \langle \mathcal{Q}, \mathsf{ans}, \mathsf{pf} \rangle \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta)\end{array}\right]\right] \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}, n)\\ \langle \mathcal{L}, \mathsf{cm}, \mathsf{aux} \rangle \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta)\end{array}\right]
$$
\n
$$
= \Pr\left[\begin{array}{c} \mathsf{aps} = \rho\\ \wedge g^{\mathsf{pf}} = \mathsf{cm} \cdot h^{\mathsf{ans}}\\ \end{array}\right|\left.\begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}, n)\\ \langle (\mathbb{G}, p, g), h \rangle := \mathsf{pp}\\ \langle \mathcal{L}, \mathsf{cm}, \mathsf{aux} \rangle \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta)\end{array}\right]
$$
\n
$$
= \Pr\left[\begin{array}{c} \mathsf{ans} = \rho\\ \wedge g^{\mathsf{pf}} = \mathsf{cm} \cdot h^{\mathsf{ans}}\\ \langle \mathcal{L}, \mathsf{cm}, \mathsf{aux} \rangle \leftarrow \widetilde{\mathcal{P}}(\mathsf{pp}, \eta)\end{array}\right]\right] \begin{array}{l} (\mathbb{X}, \mathsf{w}) \leftarrow \mathsf{G}(1^{\lambda})\\ \langle \mathcal{L}, \mathsf{ans},
$$

Theorem [8.7](#page-43-2) can be proved in the same way as Theorem [8.6.](#page-43-1)

Remark 8.12. Note that the choice of the instance x does not matter in the above proof. Hence, it is straightforward to extend the proofs in this section to hold in the setting of a non-adaptively chosen instance x.

 \Box

8.3 Lower bound from discrete logarithm assumption

We explain how to obtain barriers for security of Kilian's construction using known bounds for the Schnorr identification scheme.

Lemma 8.13 (Schnorr security from discrete log [\[PS00;](#page-49-1) [BN06;](#page-48-15) [KMP16\]](#page-49-18)). *Assume the discrete logarithm assumption holds with error* $\epsilon_{\text{DLOG}} = \epsilon_{\text{DLOG}}(\lambda, t_{\text{DLOG}})$ *. For every security parameter* $\lambda \in \mathbb{N}$ *, adversary time bound* t_{ID} ∈ N, and t_{ID}-time Schnorr impersonation adversary P, let ϵ_{Schner} be the impersonation error of the Schnorr *identification scheme, the following holds:*

$$
\epsilon_{\text{Schnorr}}(\lambda, t_{\text{ID}}) \leq \sqrt{\epsilon_{\text{DLOG}}(\lambda, O(t_{\text{ID}}))} \enspace .
$$

Theorem [8.6](#page-43-1) implies the following corollary.

Corollary 8.14 (Lower bound from position binding). *Consider these two ingredients:*

• PCP = (P, V) *, a PCP system for a relation* R with alphabet Σ *, proof length* ℓ *, and query complexity* q*; and* • $VC = (Gen, Committee, Open, Check), a vector commitment scheme over alphabet Σ .$

Let ARG := Kilian[PCP, VC] *and let* ϵ_{ARG} *be the soundness error of* ϵ_{ARG} *. If we have a security analysis such that for every* $t \in \mathbb{N}$,

$$
\epsilon_{\text{ARG}}(\lambda, n, t_{\text{ARG}}) < \epsilon_{\text{PCP}}(n) + o\left(\sqrt{\epsilon_{\text{VC}}(\lambda, \ell, \mathsf{q}, O(t_{\text{ARG}}))}\right)
$$

then

$$
\epsilon_{\text{Schnor}}(\lambda, t_{\text{ID}}) < o\left(\sqrt{\epsilon_{\text{DLOG}}(\lambda, O(t_{\text{ID}}))}\right) + 2^{-\lambda} \enspace,
$$

where $t_{ID} = \Omega(t_{ABC})$ *.*

For any cyclic group of order 2^{λ} , where the hardness of discrete logarithm is believed to hold [\[Sho97\]](#page-50-2),

$$
\epsilon_{\text{DLOG}}(\lambda, t_{\text{DLOG}}) \leq \frac{t_{\text{DLOG}}^2}{2^{\lambda}}
$$

.

Hence,

$$
\epsilon_{\text{VC}}(\lambda, 1, 1, t_{\text{VC}}) \leq \epsilon_{\text{DLOG}}(\lambda, O(t_{\text{VC}})) \leq O\left(\frac{t_{\text{VC}}^2}{2^{\lambda}}\right)
$$

According to Theorem [1,](#page-3-0) we have the following bound on the soundness error of ARG $:=$ Kilian [PCP, VC]:

$$
\epsilon_{\text{ARG}}(\lambda, \mathbf{x}, t_{\text{ARG}}) \leq \epsilon_{\text{PCP}}(n) + \epsilon_{\text{VC}}(\lambda, \ell, \mathbf{q}, t_{\text{VC}}) + \epsilon
$$

$$
\leq 2^{-\lambda} + \epsilon_{\text{VC}}(\lambda, 1, 1, t_{\text{VC}}) + \epsilon
$$

$$
\leq 2^{-\lambda} + O\left(\frac{t_{\text{VC}}^2}{2^{\lambda}}\right) + \epsilon.
$$

Setting $\epsilon := \Theta((\ell \cdot t_{\text{ARG}})^{2/3} \cdot 2^{-\lambda/3})$ minimizes the right-hand side at

$$
2^{-\lambda} + \ell^{2/3} \cdot \Theta\left(\sqrt[3]{\frac{t_{\text{ARG}}^2}{2^{\lambda}}}\right) \tag{3}
$$

,

,

.

Hence,

$$
\epsilon_{\text{Schnor}}(\lambda, t_{\text{ID}}) \le O\left(\sqrt{\frac{t_{\text{ID}}^2}{2^{\lambda}}}\right) , \text{ and}
$$

$$
\epsilon_{\text{ARG}}(\lambda, x, t_{\text{ARG}}) \le 2^{-\lambda} + \ell^{2/3} \cdot \Theta\left(\sqrt[3]{\frac{t_{\text{ARG}}^2}{2^{\lambda}}}\right)
$$

showing a polynomial gap between the best analysis of the Schnorr identification scheme and our analysis of Kilian's protocol.

8.4 Lower bound from expected-time discrete logarithm assumption

Lemma 8.15 (Expected-time security of the Schnorr identification scheme). *Assume the expected-time discrete* $logarithm$ assumption holds with error $\epsilon_{\text{DLOG}}^* = \epsilon_{\text{DLOG}}^*(\lambda, t_{\text{DLOG}}^*)$. For every security parameter $\lambda \in \mathbb{N}$, adversary time bound $t_{\text{DLOG}}^* \in \mathbb{N}$, and Schnorr adversary \widetilde{P} with expected running time t_{DLOG}^* , let $\epsilon_{\text{Schorr}}^*$ be the expected-time *impersonation error of the Schnorr identification scheme, the following holds:*

$$
\epsilon_\text{Schnor}^\star(\lambda,t_\text{ID}^\star) \leq \epsilon_\text{DLOG}^\star(\lambda,O(t_\text{ID}^\star))\enspace.
$$

From Theorem [6.1,](#page-33-2)

$$
\begin{aligned} \epsilon^{\star}_{\text{ARG}}(\lambda, \mathbf{x}, t^{\star}_{\text{ARG}}) & \leq \epsilon_{\text{PCP}}(\mathbf{x}) + \mathsf{q} \cdot \epsilon^{\star}_{\text{VC}}(\lambda, \ell, \mathsf{q}, t^{\star}_{\text{VC}}) + \epsilon \\ & \leq 2^{-\lambda} + \mathsf{q} \cdot \epsilon^{\star}_{\text{DLOG}}(\lambda, O(\log \frac{\mathsf{q}}{\epsilon} \cdot t^{\star}_{\text{ARG}})) + \epsilon \enspace. \end{aligned}
$$

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

