# On the Power of Computationally Sound Interactive Proofs of Proximity

Thesis for the degree
Master of Science

By: Hadar Strauss

Advisor: Prof. Oded Goldreich

Submitted to the Scientific Council of the Weizmann Institute of Science
November 29, 2025

**Abstract**

Interactive proofs of proximity (IPPs) are a relaxation of interactive proofs, analogous to property testing, in which soundness is required to hold only for inputs that are $\epsilon$-*far* from the property being verified, where $\epsilon > 0$ is a proximity parameter. In such proof systems, the verifier has oracle access to the input, and it engages in two types of activities before making its decision: querying the input oracle and communicating with the prover. The main objective is to achieve protocols where both the query and communication complexities are extremely low.

In this work, we focus on *computationally sound* IPPs (cs-IPPs). We study their power in two aspects:

- *Query complexity:* We show that, assuming the existence of collision-resistant hashing functions (CRHFs), any public-coin cs-IPP that has query complexity $q$ can be transformed into a cs-IPP that makes only $O(1/\epsilon)$ queries, while increasing the communication complexity by roughly $q$. If we further assume the existence of a good computational PIR (private information retrieval) scheme, then a similar transformation holds for *general* (i.e., possibly private-coin) cs-IPPs.
- *Coordination:* Aside from the low query complexity, the resulting cs-IPP has only minimal coordination between the verifier's two activities. The general definition of IPPs allows the verifier to fully coordinate its interaction with the prover and its queries to the input oracle. Goldreich, Rothblum, and Skverer (ITCS 2023) introduced two restricted models of IPPs that are *minimally coordinated*: The pre-coordinated model, where no information flows between the querying and interacting activities, but they may use a common source of randomness, and the isolated model, where the two activities are fully independent, each operating with a separate source of randomness.

  Our transformation shows that (under the aforementioned computational assumptions) any cs-IPP can be made to be in the pre-coordinated model, while preserving its efficiency. Hence, pre-coordinated cs-IPPs are essentially as powerful as general cs-IPPs.

  In contrast, we show that cs-IPPs in the *isolated* model are extremely limited, offering almost no advantage over property testers. Specifically, extending on a result shown by Goldreich et al. for *unconditionally sound* IPPs in the isolated model, we show that if a property has a cs-IPP in the isolated model that makes $q$ queries and uses $c > 0$ bits of communication, then it has a tester with query complexity $O(c \cdot q)$.

# Contents

# 1 Introduction

Interactive proofs of proximity (IPPs) are the property testing analog of interactive proofs. The focus of this work is on studying the power of *computationally sound* IPPs. We begin with some background.

## 1.1 General background

**Property testing.** The field of property testing [10, 28] studies a relaxed notion of decision problems. Rather than deciding exact membership, a property tester is only required to distinguish (w.h.p.) between objects in the property and objects that are $\epsilon$-*far* from the property, where $\epsilon > 0$ is a proximity parameter. An object $x \in \{0,1\}^n$ is considered $\epsilon$-far from a property $\Pi_n \subseteq \{0,1\}^n$ if it differs from any object in the property on more than $\epsilon \cdot n$ locations.

Standard decision problems generally require reading the entire input, since flipping a single bit can change the decision. In contrast, relaxed decision opens the possibility of algorithms that (probabilistically) read only a sub-linear portion of the input. Thus, in property testing, the input is viewed as a huge object to which the tester gets only oracle access, and the goal is to obtain testers with extremely low query complexity (e.g., query complexity that is poly-logarithmic in the input size).

**Interactive proofs of proximity.** Interactive proofs of proximity (IPPs) [8, 27] extend the relaxation considered in property testing to the realm of proof systems, analogously to the extension of standard decision algorithms to standard interactive proofs (IPs). Specifically, an interactive proof of proximity for a property is a protocol between two parties, called a verifier and a prover. The verifier has oracle access to the input, and it interacts with an (untrusted) prover that tries to convince it to accept the input. The goal is for the verifier to be convinced to accept inputs that satisfy the property when interacting with an honest prover ("completeness"), and to not be fooled into accepting inputs that are far from the property, no matter what strategy is employed by the prover ("soundness"). The prover is assumed to have explicit access to the input and is computationally unbounded.

The main complexity measures considered in IPPs are the verifier's query complexity and the communication complexity (i.e., the total number of bits exchanged during the interaction with the prover). Like the query complexity, the communication complexity should be sublinear in the input length. With linear communication complexity, the prover could simply send the entire input, and the verifier could verify that (a) the alleged input is indeed in the property (which requires no queries to the oracle); and (b) the alleged input is $\epsilon$-close to the actual input, by checking for consistency with $O(1/\epsilon)$ random locations in the actual input. Another complexity measure of interest is the round complexity (the number of back-and-forth communication rounds). The goal is to obtain proof systems with significantly lower query complexity than a tester for the property can achieve, while also minimizing the communication and round complexities.

The *computational* complexity of the verifier is also an important complexity measure, although it is often a secondary consideration in the property testing literature, where the focus is on query complexity. In this work, we require that the verifier is implementable in probabilistic polynomial time. IPPs where also the honest prover is implementable in probabilistic polynomial-time are called doubly-efficient (see, e.g., [26]).[1]

[1] Doubly-efficient IPPs should not be confused with doubly-sublinear IPPs [1], which refer to the *query* complexity of the honest prover.

**The isolated and pre-coordinated models.** The verifier in an IPP performs two distinct activities: querying the input oracle and interacting with the prover. The general definition of IPPs allows the verifier to fully coordinate these two activities; that is, it can choose where to query the input based on its communication with the prover, and likewise, it can send challenges to the prover based on values seen in the input.

Goldreich, Rothblum, and Skverer [13] considered highly restricted models of IPPs where the querying and interacting activities are assigned to separate modules such that no information can flow between them. The two modules feed their final views to a separate deciding module that decides whether to accept or reject based on the combined views.

They introduced two versions of this model. In the first model, called the isolated model, the querying and the interacting modules each get a separate and independent source of randomness, making them completely independent. The second model, called the pre-coordinated model, provides both modules with a shared source of randomness, which allows for some amount of coordination.[2]

Goldreich et al. showed that the isolated model is extremely weak; that is, it can only offer a very limited advantage over property testers. Specifically, they showed that IPPs in the isolated model that use $q$ queries and $c > 0$ bits of communication can be emulated by property testers with query complexity $O(c \cdot q)$.

In contrast, they showed that the pre-coordinated model is much more powerful. In particular, they demonstrated that there are pre-coordinated IPPs of extremely low complexity for properties that are extremely hard to test. Still, they also showed that the pre-coordinated model is considerably limited compared to general IPPs. They showed that public-coin $O(1)$-round IPPs in the pre-coordinated model can be efficiently emulated by standard property testers.[3]

## 1.2 This work – computationally sound IPPs

In this work, we focus on *computationally sound* interactive proofs of proximity (cs-IPPs). That is, we consider IPPs in which the soundness condition is relaxed to hold only against computationally bounded provers.[4] Note that for the definition to be meaningful, cs-IPPs require that also the *honest* prover is computationally bounded; otherwise, any property may have a trivial cs-IPP in which the verifier simply checks whether the prover succeeds in solving a computationally hard task.[5] Hence, we actually consider computationally sound *doubly-efficient* IPPs.

### 1.2.1 Positive result in the pre-coordinated model

We exhibit the power of cs-IPPs, both for achieving low query complexity and for achieving pre-coordination. We show that, assuming the existence of collision-resistant hashing functions (CRHFs), any public-coin cs-IPP (and hence any public-coin *unconditionally* sound IPP with an efficient honest prover) can be efficiently emulated by a cs-IPP that makes only $O(1/\epsilon)$ queries, and is in the *pre-coordinated* model. The communication complexity of the resulting system is roughly $c + q$, where $c$ and $q$ are the communication and query complexity of the original system, respectively. Furthermore, the emulation adds only 2 rounds of interaction.

[2] The isolated and pre-coordinated models were originally studied in [13] as restricted versions of IPPs with proof-oblivious queries. Proof-oblivious queries restrict only the information flow from the interacting module to the querying module.

[3] We usually consider an emulation of one model in a second model to be efficient if the communication and query complexity in the second model are polynomial in the complexities in the first model.

[4] This notion was first discussed in passing in [27], and served as the focus of [21].

[5] Recall that we already assume that the *verifier* is computationally efficient.

We consider two types of collision-resistance: weak collision-resistance, which requires that finding collisions is hard for polynomial-size circuits, and strong collision-resistance, which requires that finding collisions is hard for *sub-exponential* size circuits (see Definition 2.3).

**Theorem 1.1** (public-coin cs-IPPs can be efficiently emulated in the pre-coordinated model)**.** *Let $\mathcal{H}$ be a family of* CRHFs. *Suppose that a property $\Pi$ has a* public-coin *cs*-IPP *with query complexity $q$, communication complexity $c$, and round complexity $r$. Then $\Pi$ has a cs*-IPP in the pre-coordinated model *with the following complexities:*

- Query complexity: $O(1/\epsilon)$. *Furthermore, the queries are uniformly and independently distributed.*[6]
- Communication complexity: $O\big(c(n) + \big(q(n) + \epsilon^{-1}\big) \cdot \tau(n) + \rho(n)\big)$, *where:*
  - $\tau(n) = \text{polylog}(n)$ *if $\mathcal{H}$ is strong collision-resistant, and $\tau(n) = O(n^\gamma)$ for any constant $\gamma > 0$ if $\mathcal{H}$ is weak collision-resistant.*
  - $\rho$ *is the randomness complexity of the original verifier.*
- Round complexity: $r(n) + 2$.

*Furthermore, perfect completeness is preserved, and if $\mathcal{H}$ is a* public-coin CRHF *family,*[7] *then the resulting cs*-IPP *is public-coin.*

The emulation also roughly preserves the *running time* of the verifier: If the running time of the original verifier is $t_{\mathsf{V}}$, then the resulting verifier runs in time $O\big(t_{\mathsf{V}}(n) + \big(q(n) + \epsilon^{-1}\big) \cdot \tau(n)\big)$ (where $\tau(n)$ is as in the statement of Theorem 1.1). In addition, the running time of the resulting honest prover is $O(t_{\mathsf{P}}(n) + t_{\mathsf{V}}(n) + n \cdot \tau(n))$, where $t_{\mathsf{P}}$ is the running time of the original honest prover.

The term $\rho(n)$ appearing in Theorem 1.1 actually represents only the additional coins that the verifier uses to generate its queries, beyond the coins sent during the interaction (which contribute at most $c$ coins). By standard techniques, this $\rho$ term can always be reduced to $O(\log n)$, at the cost of introducing non-uniformity; see Remark 3.9. Alternatively, we show that the $\rho$ term can be avoided altogether, at the cost of $q(n)$ additional rounds of communication in the general case, and at no extra cost when the original verifier uses non-adaptive queries. However, the resulting system will not be public-coin (see Theorem 3.5).

While Theorem 1.1 is stated for *public-coin* cs-IPPs, we show a similar transformation for *general* cs-IPPs when using a computational PIR (private information retrieval) scheme (see Theorem 3.10).

**Application to construction of zero-knowledge IPPs.** IPPs in the pre-coordinated model are a special type of IPPs with *proof oblivious queries* (cf. [13]). As noted by [13], one of the motivations for obtaining IPPs with proof oblivious queries is their use in facilitating the construction of *zero-knowledge* IPPs [4]. Loosely speaking, in the context of IPPs, zero-knowledge means that anything that can be obtained from the prover can also be obtained without access to the prover in roughly the same time and with roughly the same number of queries.

In [4, Thm. 6.3] it was shown that assuming the existence of one-way functions, any public-coin cs-IPP with proof-oblivious queries can be efficiently transformed to a zero-knowledge cs-IPP. Hence, combining Theorem 1.1 with [4, Thm. 6.3], it follows that assuming the existence of a public-coin CRHF family, *any public-coin cs*-IPP *can be efficiently transformed to a zero-knowledge cs*-IPP. In addition, as stated in [13], it seems that also general (i.e., private-coin) cs-IPPs with proof

[6] I.e., the resulting system is actually sample-based (cf. [12, 9]).

[7] $\mathcal{H}$ is a *public-coin* CRHF family if the collision resistance property still holds when the collision finder is given access to the random coins used by the sampler of $\mathcal{H}$.

oblivious queries can be transformed to zero-knowledge ones by using secure two-party computation (or, alternatively, fully-homomorphic encryption). Combined with our general emulation in the pre-coordinated model (i.e., Theorem 3.10), it follows that (under the corresponding assumptions) *general* cs-IPPs can be converted into zero-knowledge cs-IPPs.

### 1.2.2 Negative result for the isolated model

Having shown the power of computational soundness for the pre-coordinated model, we turn to consider the *isolated* model. In contrast to the pre-coordinated model, we show that cs-IPPs in the isolated model can be efficiently emulated by testers, as in the unconditional soundness case. In particular, this demonstrates the necessity of pre-coordination for the result of Theorem 1.1.

**Theorem 1.2** (cs-IPPs in the isolated model can be efficiently emulated by testers)**.** *If a property* $\Pi$ *has a cs-*IPP *in the isolated model with query complexity* $q$ *and communication complexity* $c > 0$*, then* $\Pi$ *has a tester with query complexity* $O(c \cdot q)$*.*

While Theorem 1.2 rules out an emulation in the isolated model with the efficiency of Theorem 1.1, in Section 3.3 we show that our pre-coordinated model emulation can be transformed to the isolated model at the cost of increasing the *product* of the query and communication complexities by roughly $n/\epsilon$. More precisely, we can get a general tradeoff between the resulting query and communication complexities, such that for any $q' > O(1/\epsilon)$, we can obtain query complexity $q'$ while increasing the communication complexity (over that obtained in Theorem 1.1) by an additive $O\big(\frac{n}{\epsilon \cdot q'}\big) \cdot \tau(n)$ term, where $\tau(n)$ is as in the statement of Theorem 1.1. Note that this tradeoff is nearly optimal given the negative result of Theorem 1.2, since there exist properties that have very efficient cs-IPPs but require nearly linear query complexity for testing (see details in Section 3.3).

**Our prior work.** Recall that Theorem 1.2 demonstrates that pre-coordination (rather than isolation) is *necessary* for the result of Theorem 1.1. While this is shown through a general emulation of the isolated model by testers, in our prior work [30] we show this necessity by a direct lower bound on the isolated model for a specific property.

The property that we consider is `PERM`, the set of all permutations over $[n]$. Applying Theorem 1.1 to the IPP for `PERM` shown in [15, Sec. 4.1], we get that assuming the existence of strong CRHFs, `PERM` has a cs-IPP in the *pre-coordinated* model with poly-logarithmic query and communication complexities. In contrast, in [30] we show that any cs-IPP for `PERM` in the *isolated* model must have either query complexity or communication complexity greater than $n^{\Omega(1)}$. The exact lower bound we show is $q^5 \cdot c = \Omega(n/\log(n))$ where $q > 0$ is a query complexity and $c > 0$ is the communication complexity in any isolated cs-IPP for `PERM`.

The emulation of isolated cs-IPPs by testers (established by Theorem 1.2) implies a similar lower bound for `PERM` that follows from a simpler argument. Specifically, since testing `PERM` requires $\Omega(\sqrt{n})$ queries (see [29, Apdx. A] following [15, Lem. 4.3]), it follows that any isolated cs-IPP for `PERM` with query complexity $q > 0$ and communication complexity $c > 0$ must satisfy $q \cdot c = \Omega(\sqrt{n})$.

## 1.3 Technical Overview

### 1.3.1 Emulation of cs-IPPs by cs-IPPs in the pre-coordinated model

Our emulation in the pre-coordinated model is based on the well-known technique of Kilian [22]. Recall that Kilian showed how to transform PCPs into computationally sound interactive proofs that have very low communication complexity, assuming the existence of collision-resistant hashing functions. Kilian's protocol uses a special commitment scheme, called a tree commitment scheme

(also known as Merkle-hashing), that allows one to commit to an $n$-bit string such that individual bits in the string can be revealed (and verified as correct) with very low communication cost (e.g., poly-logarithmic in $n$). In Kilian's protocol, the prover commits to the PCP-proof, and the verifier asks the prover to reveal the locations that the original PCP-verifier would query in the proof oracle. The commitment scheme ensures that a computationally bounded prover must respond consistently with a fixed proof string (i.e., it cannot cheat by answering differently based on the set of queries it has received).

Our emulation of public-coin cs-IPPs uses the tree commitment scheme to have the prover commit to the *input*. Recall that we are given a public-coin cs-IPP for some property $\Pi$, and aim to efficiently emulate it in the pre-coordinated model while making only $O(1/\epsilon)$ queries. The emulation will begin by executing the commitment phase of the tree commitment scheme, where the prover commits to a string that allegedly equals to the input. Next, we emulate the original interaction with the prover. Note that this can be done without access to the input oracle since the original interaction is public-coin. The emulation is done with respect to proximity parameter $\epsilon/2$ (where $\epsilon$ is our target proximity parameter). At the end of the emulated interaction, instead of making the queries that the original verifier would have made to the input oracle, we ask that the prover provide the answer to these queries by revealing the corresponding locations in the alleged input via the tree-commitment scheme. We check that the original verifier would have accepted given the answers the prover provided, along with the interaction transcript generated in the emulated interaction. In addition, we query the actual input oracle on $O(1/\epsilon)$ randomly chosen locations, ask the prover to reveal the same locations in the alleged (committed) input, and check for consistency. Note that the interacting and querying modules of the resulting system need only share the random consistency-check locations, and otherwise operate independently from one another.

To see why soundness holds, consider the foregoing emulation given an arbitrary input $x \in \{0,1\}^n$ that is $\epsilon$-far from the property $\Pi$, when interacting with an arbitrary computationally bounded prover. At a high level, once the tree commitment is established, the prover's answers to the queries must be consistent with some fixed string $x'$. If $x'$ is $(\epsilon/2)$-close to $x$, then $x'$ is $(\epsilon/2)$-far from $\Pi$, and so the prover will fail to fool the original verifier with high probability. Otherwise, $x'$ is $(\epsilon/2)$-far from $x$, and then by checking for consistency between $x$ and $x'$ on $O(1/\epsilon)$ random locations, we will detect a mismatch with high probability.

Note that it is important to first emulate the original interaction and only after ask the prover to provide the answer to the queries (which is possible due to the public-coin hypothesis). This is because, by asking the prover to provide the answers to the queries, we reveal to the prover their locations, whereas the soundness of the original verifier may rely on keeping these locations secret.

To extend the emulation to general cs-IPPs that are not necessarily public-coin, we interleave the emulation of the original interaction with the query requests, since the verifier's messages may depend on responses to its prior queries. To withhold the actual queries from the prover, we use a computational PIR scheme for providing answers to these queries.

While the technique of using the tree commitment scheme to force a computationally bounded prover to respond consistently with a fixed string is well-known, to the best of our knowledge, there is a lack of a source that presents this technique formally and in a general manner (with the exception of the concurrent [18]; see Section 1.4).[8] In this work, we provide a general lemma (see Lemma 3.3) that captures this property of the tree commitment scheme. The lemma asserts that in any interaction that uses the tree commitment scheme, with overwhelmingly high probability, at the

[8] One notable source that provides a formal treatment of this technique is [2]. The setting treated in [2] is a more complex one, but it is less generic.

end of the commitment phase there exists a fixed string, such that any location that is subsequently opened will either be opened consistently with this string or will be detected as faulty.

### 1.3.2 Emulation of cs-IPPs in the isolated model by testers

Recall that in the isolated model, the verifier's querying and interacting activities are totally independent from one another, each operating with a separate source of randomness. Specifically, the verifier is decomposed into a querying, interacting, and deciding modules $Q$, $I$, and $D$, respectively, such that its decision when interacting with a prover strategy $P$ on input $x$ is expressed as: $D(Q^x(R_Q), \langle P, I(R_I)\rangle)$, where $R_Q$ and $R_I$ are independent random variables representing the randomness of the querying and the interacting modules, respectively.

In [13] it was shown that *unconditionally sound* IPPs in the isolated model can be efficiently emulated by property testers: If a property $\Pi$ has an IPP in the isolated model that makes $q$ queries and uses $c > 0$ bits of communication, then $\Pi$ has a tester with query complexity $O(c \cdot q)$. We show the same is true for *computationally sound* IPPs.

Roughly speaking, the reason that the emulation of [13] does not carry over to the context of cs-IPPs, is that it emulates the verifier with the *optimal* prover strategy. In the context of cs-IPPs, the performance of the optimal strategy (on NO-instances) is not relevant, since the optimal strategy may not be efficiently implementable.

Instead of the optimal prover, we will consider the *honest prover*, which is guaranteed to be efficient. Note that we cannot directly emulate the interaction with the honest prover on the actual input, because doing so requires full access to the input. Thus, instead, we emulate $2^n$ interactions, each with a different possible $n$-bit input (actually, it will suffice to use only the YES-instances). Observe that if the actual input is a YES-instance, then there exists an emulation that will lead to accepting with probability at least 2/3. On the other hand, if the actual input is a NO-instance, then each of these emulations leads to rejection with probability at least 2/3.

However, we do not want to take a union bound over $2^n$ events. Instead, we observe that each of these emulations can be expressed as a (different) convex combination of the probabilities that the verifier accepts given each fixed interaction transcript $\tau \in \{0,1\}^c$. For simplicity, assume that the interaction transcript fully determines the randomness of the interacting module. Then, for any prover strategy $P$, we have:

$$\Pr\left[D\big(Q^x(R_Q), \langle P, I(R_I)\rangle\big) = 1\right] = \sum_{\tau\in\{0,1\}^c} \Pr\left[\langle P, I(R_I)\rangle = \tau\right] \cdot \Pr\left[D\big(Q^x(R_Q), \tau\big) = 1\right] \quad (1)$$

Thus, we can obtain an (additive constant) approximation of the acceptance probability of all $2^n$ emulations by obtaining an approximation of the $2^c$ fixed-transcripts probabilities:

$$p^x_\tau \stackrel{\text{def}}{=} \Pr\left[D\big(Q^x(R_Q), \tau\big) = 1\right].$$

As observed in [13],[9] an approximation of $p^x_\tau$ can be obtained by repeated invocations of the querying module $Q^x$, where the same invocations can be used towards approximating all $2^c$ values $p^x_\tau$ since the invocations are oblivious of the value of $\tau$. Hence, by invoking the querying module $O(c)$ times, we can obtain for each $p^x_\tau$ a constant additive approximation with error probability $O(2^{-c})$, and by a union bound we get an approximation of all $2^c$ values, with high constant probability. Using these values, we can obtain approximations of the success probability of the honest

[9] The fixed-transcript probabilities $p^x_\tau$ are used in [13] to compute the acceptance probability of the optimal prover via the max-average game tree of interactive proofs (where the probabilities $p^x_\tau$ are viewed as the leaves of the game tree).

prover strategy on all $2^n$ inputs, by computing, for each such strategy, the corresponding coefficients (i.e., $\Pr[\langle P, I(R_I)\rangle = \tau]$) in Eq. (1). Indeed, this will be computationally expensive, but the computational complexity is irrelevant to us.

### 1.4 Further related works

As mentioned in Section 1.2, computationally sound IPPs were briefly discussed in the work of Rothblum, Vadhan, and Wigderson [27], and then became the focus of the work of Kalai and Rothblum [21]. In the former work [27], the authors noted that in a similar manner to Kilian's technique, PCPs of *Proximity* (PCPPs) [3, 7] can be transformed to computationally sound IPPs. Note that this refers to committing to the *proof* oracle of the PCPP, as in Kilian's original protocol, whereas we use the commitment scheme to commit to the *input* oracle (and use it for a distinct setting and purpose). In addition, the work of [27] establishes a separation between what can be achieved with unconditionally sound versus computationally sound IPPs. The subsequent work of [21] strengthened this separation, under a stronger computational assumption. In addition, [21] showed that assuming the existence of a sub-exponentially secure FHE scheme, any language in P has a *one-round* cs-IPP with sublinear complexities (more specifically, the query complexity is $n^{1-\gamma}$ for some $\gamma > 0$).

More recently, independently and in parallel to us,[10] Herman and Rothblum [18] used an approach similar to ours in the context of *distribution testing*. Specifically, they use a tree commitment scheme to have the prover commit to an approximation of the entire input distribution. They use this to obtain a cs-IPP in which the verifier's complexities are all roughly $\sqrt{n}/\epsilon^2$ for any property of distributions decidable in polynomial time.

### 1.5 Organization

Section 2 contains preliminaries and definitions, which include definitions of the computational models discussed in the introduction as well as the cryptographic assumptions that we use. In Section 3 we present the emulation of cs-IPPs by cs-IPPs in the pre-coordinated model. In addition, in Subsection 3.3 we show how to transform the emulation to the isolated model (at a significant cost). In Section 4 we prove that cs-IPPs in the isolated model can be efficiently emulated by testers.

## 2 Preliminaries and definitions

### 2.1 Computational models

**Property testing.** A property is a collection of sets $\Pi = \bigcup_{n\in\mathbb{N}} \Pi_n$ such that $\Pi_n$ is a set of strings over $\{0,1\}^n$. The relative hamming distance between two strings $x, x' \in \{0,1\}^n$ is the fraction of bits on which they differ. We say $x$ is $\epsilon$-far from $x'$ if the relative hamming distance between $x$ and $x'$ is greater than $\epsilon$, and otherwise we say they are $\epsilon$-close. A string $x \in \{0,1\}^n$ is $\epsilon$-far from a property $\Pi = \bigcup_{n\in\mathbb{N}} \Pi_n$ if it is $\epsilon$-far from any $x' \in \Pi_n$; otherwise, it is $\epsilon$-close to $\Pi$. A tester for a property $\Pi$ is a probabilistic algorithm that, on input parameters $n \in \mathbb{N}$, $\epsilon > 0$ and oracle access to $x \in \{0,1\}^n$, outputs 1 with probability at least 2/3 if $x$ is in $\Pi$, and outputs 0 with probability at least 2/3 if $x$ is $\epsilon$-far from $\Pi$. The query complexity of the tester is $q : \mathbb{N} \times [0,1] \to \mathbb{N}$ if, on input $n$, $\epsilon$ and oracle access to any $x \in \{0,1\}^n$, the tester makes at most $q(n,\epsilon)$ queries to $x$.

[10] See preliminary version of Section 3 in this paper [29].

**Interactive proofs of proximity (IPPs).** An interactive proof of proximity for a property $\Pi$ is a two-party protocol for parties called verifier and prover. The verifier has oracle access to a string $x \in \{0,1\}^n$, and also gets explicit inputs $n$ and $\epsilon > 0$. The prover gets $x$ as explicit input, and its aim is to convince the verifier that $x$ is in $\Pi$. We require that the prover can convince the verifier to accept any $x$ in $\Pi$ (w.h.p.), but cannot fool the verifier into accepting $x$ that is $\epsilon$-far from $\Pi$ (except for with low probability). The prover is defined by its strategy, which is a (computationally unbounded) function that maps a party's input and all messages it has received so far, to the next message it will send.

**Definition 2.1** (interactive proofs of proximity (IPPs))**.** *Let $V$ be a randomized interactive oracle machine that gets explicit inputs $n \in \mathbb{N}$ and $\epsilon > 0$, as well as oracle access to a string $x \in \{0,1\}^n$, and runs in time polynomial in $n$. The machine $V$ constitutes a verifier for an interactive proof of proximity for a property $\Pi = \bigcup_{n\in\mathbb{N}} \Pi_n$, if for every $\epsilon > 0$ the following two conditions hold.*

- **(*completeness*):** *There exists a prover $P$, called the honest prover, such that for any $n \in \mathbb{N}$, on input $n$, $\epsilon$ and oracle access to any $x \in \{0,1\}^n$, after interacting with $P$ that gets $x$ as explicit input, $V$ rejects with probability at most $1/3$.*
- **(*soundness*):** *For any prover $P$ (referred to as a cheating prover), for any $n \in \mathbb{N}$, on input $n$, $\epsilon$ and oracle access to any $x \in \{0,1\}^n$ that is $\epsilon$-far from $\Pi_n$, after interacting with $P$, the verifier $V$ accepts with probability at most $1/3$.*

*The system has perfect completeness if the verifier accepts each $x \in \Pi$ with probability 1. The functions $q, c, r : \mathbb{N} \times [0,1] \to \mathbb{N}$ are the system's query complexity, communication complexity, and round complexity, respectively, if on input $n$, $\epsilon$, on oracle access to any $x \in \{0,1\}^n$ and when interacting with any prover, the verifier makes at most $q(n,\epsilon)$ queries to $x$, the parties exchange at most $c(n,\epsilon)$ bits, and the interaction consists of at most $r(n,\epsilon)$ communication rounds (with two messages per round). The system is public-coin if each message sent by the verifier consists only of the outcomes of its coin tosses. The system uses non-adaptive queries if the verifier's queries to the input are determined based on the verifier's randomness and the message it has received from the prover, but do not depend (directly) on the answers to prior queries. If the honest prover is implementable in probabilistic polynomial time, then the system is doubly-efficient.*

More generally, IPPs with general soundness (resp., completeness) error $e : \mathbb{N} \to [0,1]$ are defined by replacing the term $1/3$ in the soundness (resp., completeness) condition with $e(n)$.

**Computationally sound IPPs.** The soundness condition in Definition 2.1 (which holds against computationally unbounded provers) is sometimes referred to as statistical soundness. In computationally sound IPPs (cs-IPPs), the soundness condition is relaxed to hold only against computationally efficient provers. This leads to restricting also the computational power of the prover in the completeness condition. Actually, we even require the honest prover's strategy to be implementable in probabilistic polynomial-time, although all of our results hold also if it is allowed to be implemented by a non-uniform polynomial-size family of circuits.

**Definition 2.2** (computationally sound interactive proofs of proximity (cs-IPPs))**.** *Let $V$ be a randomized interactive oracle machine that gets explicit inputs $n \in \mathbb{N}$ and $\epsilon > 0$, as well as oracle access to a string $x \in \{0,1\}^n$, and runs in time polynomial in $n$. The machine $V$ constitutes a verifier for an interactive proof of proximity for a property $\Pi = \bigcup_{n\in\mathbb{N}} \Pi_n$, if for every $\epsilon > 0$ the following two conditions hold.*

(***completeness***)**:** *There exists a prover $P$, called the* honest prover*, that can be implemented by a randomized, polynomial-time machine, such that for any $n \in \mathbb{N}$, on input $n$, $\epsilon$ and oracle access to any $x \in \Pi_n$, after interacting with $P$ that gets $x$ as explicit input, $V$ rejects with probability at most $1/3$.*

(***computational soundness***)**:** *For any prover $P$ (referred to as a* cheating prover*) that can be implemented by a (non-uniform) polynomial-size family of circuits, for all sufficiently large $n$'s, on input $n$, $\epsilon$ and oracle access to any $x \in \{0,1\}^n$ that is $\epsilon$-far from $\Pi_n$, after interacting with $P$, the verifier $V$ accepts with probability at most $1/3$.*

In the above definitions of IPPs and cs-IPPs, the verifier is required to run in time polynomial in $n$. We note that our positive results (i.e., Theorem 3.4, 3.5 and 3.10) hold also if we had defined the verifier such that it is implementable by a probabilistic (*non-uniform*) polynomial-size family of circuits,[11] but not if we put no computational bounds on it. On the other hand, our negative result (i.e., Theorem 1.2) holds also if no computational bounds are put on the verifier.

**The isolated and pre-coordinated models.** To define the isolated and pre-coordinated models, we follow the framework of [13], where the verifier is decomposed into three modules: the querying module $Q$, the interacting module $I$, and the deciding module $D$. The querying module is the only part that queries the input, and the interacting module is the only part that interacts with the prover. The final decision is made by the deciding module, which is fed with the outputs of the two other modules.

Both the isolated and pre-coordinated models are restricted forms of IPPs in which there is no information flow between the querying and the interacting modules. In the isolated model, the querying and the interacting modules each get a separate and independent source of randomness, whereas in the pre-coordinated model, the modules get a shared source of randomness.

Recall that in standard interactive proofs, one denotes the output of the verifier $V$ when interacting with a prover $P$ by $\langle P, V\rangle(x)$, where $x$ is the common input. In extensions that allow private inputs, one uses the notation $\langle P(y), V(z)\rangle(x)$, where $z$ and $y$ are private inputs given to $V$ and $P$, respectively. In the isolated model, we write the random variable representing the decision of the verifier as $D(Q^x(R_Q), \langle P(x), I(R_I)\rangle)$, where $R_Q$ and $R_I$ are independent random variables representing the randomness of each module. In the pre-coordinated model, we write the random variable representing the decision as $D(Q^x(R), \langle P(x), I(R)\rangle)$, where $R$ is a random variable representing the shared randomness of both modules.

## 2.2 Cryptographic assumptions

A function $\mu : \mathbb{N} \to \mathbb{R}$ is called negligible if for every positive polynomial $p$, for all sufficiently large $n$'s, it holds that $\mu(n) < \frac{1}{p(n)}$.

**Definition 2.3** (strong and weak collision-resistant hashing functions (CRHFs))**.** *Let $\mathcal{H} = \bigcup_{k\in\mathbb{N}} \mathcal{H}_k$ such that $\mathcal{H}_k \subseteq \{h : \{0,1\}^{2k} \to \{0,1\}^k\}$. $\mathcal{H}$ is said to be a family of* collision-resistant hashing functions (CRHFs) *if there exists a mapping from strings $s \in \{0,1\}^*$ to functions $h_s \in \mathcal{H}$, such that:*

1. (Efficient indexing): *There exists a probabilistic polynomial-time algorithm $I$, called an* indexing algorithm*, that for every $k \in \mathbb{N}$, given $1^k$, samples a string $s$ such that $h_s \in \mathcal{H}_k$.*

[11] In Theorem 3.4, if the verifier is allowed to be non-uniform, then also the resulting honest prover would be non-uniform.

2. (Efficient evaluation): *There exists a polynomial-time algorithm, called an* evaluation algorithm*, that, given $s$ and $x$, returns $h_s(x)$.*

3. (Hard-to-form collisions): *One of the following types of collision-resistance holds:*

   - *There exists a constant $\delta > 0$ such that for every (non-uniform) family of circuits $\{C_k\}_{k\in\mathbb{N}}$ of size at most $2^{O(k^\delta)}$, it holds that:*

   $$\Pr_{\substack{s\leftarrow I(1^k)\\(x_1,x_2)\leftarrow C_k(s)}}[h_s(x_1) = h_s(x_2) \wedge x_1 \neq x_2] \leq 2^{-\omega(k^\delta)}$$

   *In this case, $\mathcal{H}$ is said to be* strong collision-resistant *with* parameter *$\delta$.*

   - *For every (non-uniform) polynomial-size family of circuits $\{C_k\}_{k\in\mathbb{N}}$, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that for every $k$:*

   $$\Pr_{\substack{s\leftarrow I(1^k)\\(x_1,x_2)\leftarrow C_k(s)}}[h_s(x_1) = h_s(x_2) \wedge x_1 \neq x_2] \leq \mu(k)$$

   *In this case, $\mathcal{H}$ is said to be* weak collision-resistant*.*

*$\mathcal{H}$ is said to be* public-coin *[19] if the indexing algorithm $I$ outputs the outcome of its coin tosses.*

For simplicity, we will avoid referring to the indexing algorithm $I$ explicitly, and write $h \leftarrow \mathcal{H}_k$ as a shorthand for $h \leftarrow I(1^k)$. Abusing notation, we will sometimes treat the string $h$ (generated by $I$) as the actual hash function it corresponds to (e.g., writing $h(x)$ to denote evaluation of the hash function).

**Computational private information retrieval (PIR).** A computational private information retrieval (PIR) scheme is a protocol between two parties, called a sender and a receiver. The sender holds a string $x \in \{0,1\}^n$ and the receiver holds an index $i \in [n]$. Loosely speaking, the protocol enables the receiver to retrieve $x_i$ without the sender learning anything about $i$. The main complexity measure of the PIR scheme is its communication complexity, which should be sublinear in $n$.

**Definition 2.4** (computational private information retrieval (PIR))**.** *A* computational private information retrieval (PIR) scheme *is a two-party 1-round protocol between parties called a sender and receiver. The sender gets as input a string $x \in \{0,1\}^n$ and the receiver gets as input an index $i \in [n]$ as well as $n$, and both run in time at most polynomial in $n$. The scheme is defined by three algorithms $(Q, A, R)$ and proceeds as follows: The receiver sends to the sender a query $q = Q(n, i, r)$, where $r$ is the receiver's randomness, and it gets in return an answer $a = A(x, q)$. The receiver then runs $R(n, i, r, a)$ to reconstruct the $i^{th}$ bit of $x$. The scheme should satisfy:*

- Correctness*: For all $n$, for every $x \in \{0,1\}^n$ and every $i \in [n]$ it holds that:*

$$\Pr_r[q \leftarrow Q(n,i,r),\ a \leftarrow A(x,q),\ R(n,i,r,a) = x_i] = 1$$

- Security*: For any (non-uniform) polynomial-size family of circuits $\{D_n\}_{n\in\mathbb{N}}$, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that for all $n \in \mathbb{N}$ and for every $i, j \in [n]$ it holds that:*

$$\left|\Pr_r[D_n(Q(n,i,r)) = 1] - \Pr_r[D_n(Q(n,j,r)) = 1]\right| \leq \mu(n)$$

*The communication complexity of the scheme is $c : \mathbb{N} \to \mathbb{N}$ if the parties exchange at most $c(n)$ bits given any inputs $x \in \{0,1\}^n$ and $i \in [n]$.*

Constructions of computational PIR schemes that have very low communication complexity are known based on various hardness assumptions (e.g., quadratic residuosity). Examples include [23], which achieves communication complexity $O(n^\alpha)$ for any constant $\alpha > 0$, and [24], which achieves *poly-logarithmic* communication complexity.

**Computational indistinguishability.** Two probability ensembles, $X \stackrel{\text{def}}{=} \{X_n\}_{n\in\mathbb{N}}$ and $Y \stackrel{\text{def}}{=} \{Y_n\}_{n\in\mathbb{N}}$ are said to be computationally indistinguishable if for every (non-uniform) polynomial size family of circuits $\{D_n\}_{n\in\mathbb{N}}$, there exists a negligible functions $\mu : \mathbb{N} \to \mathbb{R}$ such that $|\Pr[D_n(X_n) = 1] - \Pr[D_n(Y_n) = 1]| \leq \mu(n)$.

# 3 The power of computationally sound IPPs

In this section we show that under standard computational assumptions, any cs-IPP can be efficiently emulated by a cs-IPP that makes only $O(1/\epsilon)$ queries and is in the pre-coordinated model. We begin, in Section 3.1, by presenting the tree commitment scheme which underlies our emulation. Then, in Section 3.2.1, we present the emulation of *public-coin* cs-IPPs based on CRHFs, and in Section 3.2.2 we extend this emulation to general cs-IPPs using a computational PIR scheme.

## 3.1 The tree commitment scheme

In this section, we present the tree commitment scheme and provide formal proofs of its properties. The key technical contribution of this section is Lemma 3.3, which formalizes the use of the tree commitment scheme in a general interactive setting for forcing the committing party to reveal all bits consistently with a fixed string. As discussed in Section 1.3.1, this technique is well known, but to the best of our knowledge, lacked a source providing a general and formal analysis.

The tree commitment scheme provides a mechanism for committing to an $n$-bit string such that individual bits in the string can be revealed (and verified as correct) with very low communication cost. The scheme relies on collision-resistant hashing functions (see Definition 2.3), where the type of the collision-resistance affects the communication cost.

**Definition 3.1** (tree commitment scheme (also known as Merkle tree [25]))**.** *Let $\mathcal{H} = \bigcup_{k\in\mathbb{N}} \mathcal{H}_k$ be a family of collision-resistant hashing functions. The tree commitment scheme is a protocol between two parties, called a sender and a receiver, that is composed of the following phases:*

- *Committing to a string $s \in \{0,1\}^n$:*

  *Let $k \in \mathbb{N}$ be a parameter defined according to the type of $\mathcal{H}$: If $\mathcal{H}$ is strong collision-resistant with parameter $\delta$, then $k = (\log n)^{1/\delta}$. If $\mathcal{H}$ is weak collision-resistant, then $k = n^\alpha$ for an arbitrary $\alpha > 0$.*

  - *The receiver selects a hash function $h \leftarrow \mathcal{H}_k$ and sends it to the sender.*
  - *The sender partitions $s$ into $m = \frac{n}{k}$ consecutive blocks, each of length $k$. It constructs a binary tree of depth $\log_2 m$, placing the $m$ blocks of $s$ in the corresponding leaves of the tree. In each internal node, the sender places the value obtained by applying $h$ on the values of the node's children. The sender sends the value of the root of the resulting tree, which is called the tree commitment.*

- *Opening block $i \in [m]$:*

  *The sender reveals the value of block $i$, and sends the values of all siblings along the path from the root to the $i^{th}$ leaf. This sequence of values is called the* opening *of block $i$.*

- *Validating an opening of block $i \in [m]$:*

  *Given the revealed block value $v \in \{0,1\}^k$ and the opening $\pi$, the receiver computes the values of all nodes along the path from the root to the $i^{th}$ leaf in the following manner. The value of the $i^{th}$ leaf is taken to be $v$. The value of each consecutive node is computed by applying $h$ on the value computed for the previous node concatenated with the value of its sibling (obtained from $\pi$), ordered according to their position in the tree. The receiver verifies that the value it computed for the root matches the one received during the commit phase.*

*An opening $\pi$ is a* valid opening *for $(h, tc, i, v)$ if the validation procedure accepts $\pi$ as an opening for block $i$, where $v$ is the revealed value of the block, and $h$ and $tc$ are the hash function and tree commitment sent in the commit phase, respectively.*

Note that the parameters $k$ and $m$ defined in the commitment scheme are functions of $n$, the length of the committed string. When $n$ is clear from the context, we omit the explicit dependence of $k$ and $m$ on $n$. Additionally, we will say that an opening is *valid*, without specifying with respect to which parameters $(h, tc, i, v)$, when those parameters are implied from the context.

**Lemma 3.2** (tree commitments are computationally binding)**.** *Let $\mathcal{H}$ and $k$ be as in Definition 3.1. For any (non-uniform) polynomial-size family of circuits $\{C_n\}_{n\in\mathbb{N}}$, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that for every $n$:*

$$\Pr_{\substack{h\leftarrow\mathcal{H}_{k(n)} \\ (tc,i,v_1,v_2,\pi_1,\pi_2)\leftarrow C_n(h)}} [v_1 \neq v_2 \text{ and } \forall j \in \{1,2\}\ \pi_j \text{ is a valid opening for } (h, tc, i, v_j)] \leq \mu(n)$$

We note that a similar claim is implicit in the proof of [2, Claim 3.5.2].

**Proof:** Assume towards contradiction that there exists a polynomial-size family of circuits $\{C_n\}_{n\in\mathbb{N}}$ and a polynomial $p$, such that for infinitely many $n$'s the probability in the statement of the lemma is greater than $\frac{1}{p(n)}$. We derive a contradiction by constructing a family of circuits $\{C'_k\}_{k\in\mathbb{N}}$ that breaks the collision resistance condition of $\mathcal{H}$. The circuit $C'_{k(n)}$ first emulates $C_n$ to obtain its output $(tc, i, v_1, v_2, \pi_1, \pi_2)$. Next, for both $j \in \{1,2\}$, using $v_j$ and $\pi_j$ it computes the values of all the nodes along the path from the root to the $i^{\text{th}}$ leaf in a similar manner to the validation procedure from Definition 3.1. Let $(u_1^1, ..., u_1^t)$ and $(u_2^1, ..., u_2^t)$ denote the computed values, where $t = \log_2 m$ denotes the height of the tree, and for each level $l \in [t]$ (starting from the leaves), $u_j^l$ is the value computed for the node at level $l$ when using $(v_j, \pi_j)$. The circuit then searches for the first level $l$ for which $u_1^l \neq u_2^l$ and $u_1^{l+1} = u_2^{l+1}$. If it finds such an $l$, it outputs $(x_1, x_2)$ such that each $x_i$ is the concatenation of $u_i^l$ with its sibling according to their order in the tree. Otherwise, it halts with an arbitrary output.

Suppose that $v_1 \neq v_2$ yet for both $j \in \{1,2\}$ it holds that $\pi_j$ is a valid openings for $(h, tc, i, v_j)$. Then, $u_1^1 = v_1 \neq v_2 = u_2^1$ and $u_1^t = tc = u_2^t$, so there must be some level $l$ for which $u_1^l \neq u_2^l$ and $u_1^{l+1} = u_2^{l+1}$. In this case, the circuit will output $(x_1, x_2)$ such that $x_j$ is the concatenation of $u_j^l$ with its sibling. Notice that $(x_1, x_2)$ form a collision under $\mathcal{H}$; that is, $x_1 \neq x_2$ (because $u_1^l \neq u_2^l$) and $h(x_1) = h(x_2)$ (because $h(x_1) = u_1^{l+1} = u_2^{l+1} = h(x_2)$).

Since $\{C_n\}_{n\in\mathbb{N}}$ is of polynomial size, also the size of $\{C'_{k(n)}\}_{n\in\mathbb{N}}$ is polynomial in $n$. Let $q$ denote this polynomial, i.e., $|C'_{k(n)}| \leq q(n)$ for all $n$. If $\mathcal{H}$ is strong collision-resistant with parameter $\delta > 0$

and $k = (\log n)^{1/\delta}$, we have that $q(n) = 2^{O(k^\delta)}$ and similarly $\frac{1}{p(n)} = 2^{-O(k^\delta)}$. So we obtain a family of circuits $\{C'_k\}_{k\in\mathbb{N}}$ of size smaller than $2^{O(k^\delta)}$ that for some constant $c > 0$ for infinitely many $k$'s succeeds in finding collisions under $\mathcal{H}$ with probability greater than $2^{-c\cdot k^\delta}$, reaching a contradiction. Similarly, if $\mathcal{H}$ is weak collision-resistant and $k = n^\alpha$ for $\alpha > 0$, we get a contradiction to the weak collision-resistant condition, since any polynomial in $n$ is upper bounded by a polynomial in $k$. ■

Next, we show that the tree commitment scheme forces a (computationally bounded) cheating sender to respond non-adaptively. We consider general interactions composed of two stages, where in the first stage the sender commits to a string, and in the second stage there is a randomized interactive process in which the receiver chooses blocks for the sender to open. We show that the sender is forced to respond consistently with a fixed string. More precisely, we show that with overwhelming probability over the hash function $h$ sampled in the commitment stage, at the end of the commitment stage there exists a string $s_h \in \{0,1\}^n$ such that with overwhelming probability over the choice of blocks to be opened, each block that is opened validly must be opened to a value that matches $s_h$.

**Lemma 3.3** (interactive openings of tree commitments)**.** *Let $\mathcal{H} = \bigcup_{k\in\mathbb{N}} \mathcal{H}_k$ be a family of* CRHF*s, and let $k$ and $m$ be the corresponding parameters used in the tree commitment scheme as specified in Definition 3.1. Consider an interaction between a deterministic ("cheating") sender and a randomized receiver, where both parties' strategies are implementable by a (non-uniform) polynomial-size family of circuits, such that for each $n \in \mathbb{N}$, on common input $1^n$, the interaction proceeds in two stages:*

- *In the first stage, the parties execute the commitment phase of the tree commitment scheme for an $n$-bit long string, where only the receiver is assumed to execute its part honestly.*
- *In the second stage, there are iterative rounds in which the receiver requests to open blocks of its choice, based on all messages it has received so far and possibly also on fresh randomness, independent of the hash function sampled in the first stage.*

*For a fixed input $1^n$, let $h$ be the hash function sampled in the first stage, and let $r$ be the receiver's additional fresh randomness. Let $B^{\text{val}}_{h,r}$ denote the set of blocks $i \in [m]$ that the sender opens with a valid opening, and let $V_{h,r}(i)$ denote the value revealed by the sender for block $i \in B^{\text{val}}_{h,r}$.*[12] *Then, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that for all $n$, there exists a string $s_h \in \{0,1\}^n$ satisfying:*

$$\Pr_{h,r}\left[\exists i \in B^{\text{val}}_{h,r} \text{ s.t. } V_{h,r}(i) \neq s_h(i)\right] \leq \mu(n)$$

*where $s_h(i)$ denote the $i^{th}$ block in the partition of $s_h$ into $m$ consecutive blocks of length $k$.*[13]

**Proof:** We will show that the following string $s_h \in \{0,1\}^n$ satisfies the claim. For each $i \in [m]$, we define $s_h(i)$ as follows:

$$s_h(i) \stackrel{\text{def}}{=} \arg\max_{v\in\{0,1\}^k} \left(\Pr_r\left[V_{h,r}(i) = v \,\middle|\, B^{\text{val}}_{h,r} \ni i\right]\right).$$[14]

[12] Note that $h, r, B^{\text{val}}_{h,r}$, and $V_{h,r}$ all depend on $n$, although this dependency is left implicit in the notation.

[13] More intuitively, Lemma 3.3 implies that with overwhelmingly high probability over $h$, the probability over $r$ that $\exists i \in B^{\text{val}}_{h,r}$ *s.t.* $V_{h,r}(i) \neq s_h(i)$ is negligible.

[14] If the value $v$ maximizing the expression $\Pr_r[V_{h,r}(i) \neq v \mid B^{\text{val}}_{h,r} \ni i]$ is not unique, we arbitrarily take $s_h(i)$ to be the lexicographic first value that maximizes the expression.

Notice that we can express $s_h(i)$ as $\arg\max_{v\in\{0,1\}^k}\left(\Pr_r\left[B^{\text{val}}_{h,r}\ni i,\, V_{h,r}(i)=v\right]\right)$, since $\Pr_r\left[B^{\text{val}}_{h,r}\ni i,\, V_{h,r}(i)=v\right]=\Pr_r\left[V_{h,r}(i)=v\,|\,B^{\text{val}}_{h,r}\ni i\right]\cdot\Pr_r\left[B^{\text{val}}_{h,r}\ni i\right]$. Furthermore, we can write:

$$s_h(i)=\arg\min_{v\in\{0,1\}^k}\left(\Pr_r\left[B^{\text{val}}_{h,r}\ni i,\, V_{h,r}(i)\neq v\right]\right) \tag{2}$$

since $\Pr_r[B^{\text{val}}_{h,r}\ni i,\, V_{h,r}(i)=v]=\Pr_r[B^{\text{val}}_{h,r}\ni i]-\Pr_r[B^{\text{val}}_{h,r}\ni i,\, V_{h,r}(i)\neq v]$.

Assume, towards contradiction, that there exists a polynomial $p$ such that for infinitely many $n$'s it holds that

$$\Pr_{h,r}\left[\exists i\in B^{\text{val}}_{h,r}\ s.t.\ V_{h,r}(i)\neq s_h(i)\right]\geq\frac{1}{p(n)} \tag{3}$$

We will show a family of circuits $\{C_n\}_{n\in\mathbb{N}}$ that contradicts the binding property of the tree commitment scheme. Given input $h\leftarrow\mathcal{H}_{k(n)}$, the circuit $C_n$ simulates the interaction between the sender and receiver on input $1^n$ in the following manner. First, $C_n$ simulates the first stage of the interaction using $h$ as the sampled hash function. Then, $C_n$ independently samples $r_1$ and $r_2$ from the distribution of $r$, and simulates the rest of the interaction twice, once with randomness $r_1$ and once with $r_2$. It then searches for a block that was opened validly in both simulations, such that the revealed values for this block differ between the two simulations. Note that this family of circuits $\{C_n\}_{n\in\mathbb{N}}$ is of polynomial size, due to the assumption that the strategies of the sender and receiver can be implemented by a polynomial-size family of circuits.

Consider any $n$ satisfying Eq. (3). Using a reverse of the union bound, there exists a block $i\in[m]$ such that

$$\Pr_{h,r}\left[B^{\text{val}}_{h,r}\ni i,\, V_{h,r}(i)\neq s_h(i)\right]\geq\frac{1}{m\cdot p(n)}>\frac{1}{n\cdot p(n)} \tag{4}$$

We will show that with non-negligible probability, this block $i$ is opened validly in both simulations, such that the values revealed for block $i$ are different between the simulations. Thus, by checking all blocks, the circuit will succeed in finding a block with valid openings to two different values with non-negligible probability, reaching a contradiction.

$$\begin{aligned}
&\Pr_{h,r_1,r_2}\left[B^{\text{val}}_{h,r_1}\ni i,\, B^{\text{val}}_{h,r_2}\ni i,\, V_{h,r_1}(i)\neq V_{h,r_2}(i)\right]\\
&=\sum_{h'}\left(\Pr_h\left[h=h'\right]\cdot\sum_{r'_2:B^{\text{val}}_{h',r'_2}\ni i}\left(\Pr_{r_2}\left[r_2=r'_2\right]\cdot\Pr_{r_1}\left[B^{\text{val}}_{h',r_1}\ni i,\, V_{h',r_1}(i)\neq V_{h',r'_2}(i)\right]\right)\right)\\
&\geq\sum_{h'}\left(\Pr_h\left[h=h'\right]\cdot\sum_{r'_2:B^{\text{val}}_{h',r'_2}\ni i}\left(\Pr_{r_2}\left[r_2=r'_2\right]\cdot\Pr_{r_1}\left[B^{\text{val}}_{h',r_1}\ni i,\, V_{h',r_1}(i)\neq s_{h'}(i)\right]\right)\right)\\
&=\sum_{h'}\left(\Pr_h\left[h=h'\right]\cdot\Pr_{r_2}\left[B^{\text{val}}_{h',r_2}\ni i\right]\cdot\Pr_{r_1}\left[B^{\text{val}}_{h',r_1}\ni i,\, V_{h',r_1}(i)\neq s_{h'}(i)\right]\right)
\end{aligned} \tag{5}$$

where the first equality follows from the mutual independence of $h$, $r_1$, and $r_2$, and the inequality is due to Eq. (2). Let $\gamma=\frac{1}{2}\cdot\frac{1}{n\cdot p(n)}$, and define the set of 'Good' hash functions:

$$G=\left\{h':\Pr_r\left[B^{\text{val}}_{h',r}\ni i,\, V_{h',r}(i)\neq s_{h'}(i)\right]\geq\gamma\right\}$$

Then for every $h' \in G$, both the terms $\Pr_{r_2}[B^{\text{val}}_{h',r_2} \ni i]$ and $\Pr_{r_1}[B^{\text{val}}_{h',r_1} \ni i,\, V_{h',r_1}(i) \neq s_{h'}(i)]$ in Eq. (5) are at least $\gamma$. From Eq. (4) and Markov's inequality, it holds that $\Pr_h[h \in G] \geq \gamma$. Thus, Eq. (5) is lower bound by

$$\sum_{h' \in G} \left( \Pr_h \left[ h = h' \right] \cdot \gamma \cdot \gamma \right) = \Pr_h \left[ h \in G \right] \cdot \gamma^2 \geq \gamma^3$$

This non-negligible probability leads to the desired contradiction. ■

## 3.2 Emulation in the pre-coordinated model

In this section we present an emulation of cs-IPPs by cs-IPPs in the pre-coordinated model. We begin, in Section 3.2.1, by presenting an emulation of *public-coin* cs-IPPs, and in Section 3.2.2 we extend the emulation to general cs-IPPs.

### 3.2.1 Emulation of public-coin cs-IPPs

In this section we show that, assuming the existence of CRHFs, any public-coin cs-IPP can be efficiently emulated by a cs-IPP in the pre-coordinated model that makes only $O(1/\epsilon)$ queries.

**Theorem 3.4** (restatement of Theorem 1.1)**.** *Let $\mathcal{H} = \bigcup_{k \in \mathbb{N}} \mathcal{H}_k$ be a family of* CRHFs*. Suppose that a property* $\Pi$ *has a* public-coin *cs-*IPP *with query complexity $q$, communication complexity $c$, and round complexity $r$. Then* $\Pi$ *has a cs-*IPP in the pre-coordinated model *with the following complexities:*

- Query complexity: $O(1/\epsilon)$*. Furthermore, the queries are uniformly and independently distributed.*
- Communication complexity: $O\big(c\big(n, \frac{\epsilon}{2}\big) + \big(q\big(n, \frac{\epsilon}{2}\big) + \epsilon^{-1}\big) \cdot \tau(n) + \rho\big(n, \frac{\epsilon}{2}\big)\big)$*, where:*
  - $\tau(n) = \text{polylog}(n)$ *if $\mathcal{H}$ is strong collision-resistant, and $\tau(n) = O(n^\gamma)$ for any constant $\gamma > 0$ if $\mathcal{H}$ is weak collision-resistant.*
  - *$\rho$ is the randomness complexity of the original verifier.*
- Round complexity: $r\big(n, \frac{\epsilon}{2}\big) + 2$*.*

*Furthermore, perfect completeness is preserved, and if $\mathcal{H}$ is a* public-coin CRHF *family, then the resulting cs-*IPP *is public-coin.*

The resulting verifier runs in time $O\big(t_{\mathsf{V}}(n, \frac{\epsilon}{2}) + \big(q(n, \frac{\epsilon}{2}) + \epsilon^{-1}\big) \cdot \tau(n)\big)$ and the resulting honest prover runs in time $O\big(t_{\mathsf{P}}(n, \frac{\epsilon}{2}) + t_{\mathsf{V}}(n, \frac{\epsilon}{2}) + n \cdot \tau(n)\big)$, where $t_{\mathsf{V}}$ and $t_{\mathsf{P}}$ denote the running times of the original verifier and honest prover, respectively.

Theorem 3.4 includes the randomness complexity $\rho$ of the original verifier in the communication complexity. As noted in the introduction, this term actually represents only the additional coins that the verifier uses to generate its queries, beyond the coins sent during the interaction (which contribute at most $c$ coins). By standard techniques, this $\rho$ term can always be reduced to $O(\log n)$ at the cost of making both the resulting verifier and the resulting honest prover *non-uniform*; see Remark 3.9. Alternatively, the $\rho$ term can be avoided at the cost of additional rounds, as shown in the following theorem, which is identical to Theorem 3.4 except that: (i) the communication complexity does not include the randomness $\rho$, (ii) the round complexity increase by $q$ when the original protocol uses adaptive queries, and (iii) the public-coin property is not preserved.

**Theorem 3.5.** *Let $\mathcal{H}$ and $\tau(n)$ be as in Theorem 3.4. Suppose that a property $\Pi$ has a* public-coin *cs*-IPP *with query complexity $q$, communication complexity $c$, and round complexity $r$. Then $\Pi$ has a cs*-IPP *in the pre-coordinated model with the following complexities:*

- Query complexity: *As in Theorem 3.4.*
- Communication complexity: $O\big(c\big(n, \frac{\epsilon}{2}\big) + \big(q\big(n, \frac{\epsilon}{2}\big) + \epsilon^{-1}\big) \cdot \tau(n)\big)$.
- Round complexity: $O\big(r\big(n, \frac{\epsilon}{2}\big) + q\big(n, \frac{\epsilon}{2}\big)\big)$ *in general, and* $r\big(n, \frac{\epsilon}{2}\big) + 2$ *if the original cs*-IPP *uses non-adaptive queries.*

*Furthermore, perfect completeness is preserved.*

The running time of the resulting verifier and honest prover is as in Theorem 3.4, except that the term $t_{\mathsf{V}}\big(n, \frac{\epsilon}{2}\big)$ can be removed from the honest prover's running time.

The emulations in Theorem 3.4 and 3.5 preserve the completeness error. Furthermore, they can be modified to preserve the soundness error $s$ up to a negligible term, at the cost of increasing the query complexity, as well as the term $\epsilon^{-1}$ in the communication complexity, by a factor of $\log(1/s)$ (see Remark 3.8).

We begin by proving the latter Theorem 3.5, and then show how to modify it to get Theorem 3.4.

**Proof of Theorem 3.5:** Fix a tree commitment scheme using $\mathcal{H}$ as the CRHF. Let $k$ be the corresponding block size according to Definition 3.1, and let $m = n/k$ be the corresponding number of blocks. Our pre-coordinated cs-IPP proceeds as follows.

**Construction 3.6.** *On input $n$, $\epsilon$ and oracle access to $x \in \{0,1\}^n$, the emulation proceeds as follows:*

- *The shared randomness: A sequence of $O(1/\epsilon)$ uniformly and independently distributed locations in $[n]$.*
- *The querying module obtains the value of $x$ at the random locations specified in the shared randomness.*
- *The interacting module proceeds in 4 stages.*

  1. *In the first round of the interaction, the interacting module and the prover execute the commitment phase of the tree commitment scheme for an $n$-bit long string, where the honest prover commits to the input string $x$. Specifically, the interacting module sends a random hash function $h \leftarrow \mathcal{H}_{k(n)}$ to the prover, who then sends the corresponding tree commitment.*
  2. *The interacting module emulates a random execution of the original verifier's interaction with the prover, using proximity parameter $\epsilon/2$. (Since the original interaction is public-coin, this can be done with no access to the input oracle.)*

  Terminology: *Throughout the rest of the proof,* querying the prover with location $j \in [n]$ *means sending $j$ to the prover and expecting it to open the block containing location $j$ (i.e., to send the revealed value for the block together with its corresponding opening). The* prover's answer to the query *is the value within the revealed block corresponding to $j$.*

  3. *For every query $j$ made by the original verifier, the interacting module queries the prover with location $j$. In the general case this is done sequentially in $q$ rounds. However, if the original* IPP *uses non-adaptive queries, then this is done in parallel in a single round.*

4. *In addition, the interacting module queries the prover with all locations specified by the shared randomness.*

- *The deciding module verifies that the following checks pass:*
  - Original Check: *The original verifier would have accepted given the interaction transcript generated in Stage 2 and the prover's answers to the queries made in Stage 3, as well as the random string of the emulated execution.*
  - Consistency Check: *The prover's answers to the queries made in Stage 4 match the values of $x$ obtained by the querying module.*
  - Openings Validity Check: *All the openings sent in Stages 3 and 4 are valid.*

Having described the emulation, we now turn to its analysis. First, note that since we emulate the original verifier with proximity parameter $\epsilon/2$, all of its complexities are taken with respect to this parameter. Let $\tilde{q} = q(n, \frac{\epsilon}{2})$, $\tilde{c} = c(n, \frac{\epsilon}{2})$, and $\tilde{r} = r(n, \frac{\epsilon}{2})$.

We first consider the communication complexity. From the efficient-indexing property of $\mathcal{H}$, sending the hash function at Stage 1 requires $\text{poly}(k)$ bits. Emulating the original interaction requires $\tilde{c}$ bits. The length of each opening is less than $k \cdot \log(n)$ (since an opening consists of a $k$-bit value for each level of the tree, and the tree is of height $\log(m) < \log(n)$). Therefore, opening the $\tilde{q}$ original queries and the $O(\epsilon^{-1})$ shared random locations requires $(\tilde{q} + O(\epsilon^{-1})) \cdot O(k \cdot \log(n))$ bits. The total communication complexity is thus

$$\text{poly}(k) + \tilde{c} + \left(\tilde{q} + O(\epsilon^{-1})\right) \cdot O(k \cdot \log n) \tag{6}$$

Recall that if $\mathcal{H}$ is strong collision-resistant then $k = \text{polylog}(n)$, and if $\mathcal{H}$ is weak collision-resistant then $k = n^{\alpha}$ for arbitrary $\alpha > 0$. Hence, we can express the total communication complexity as $\tilde{c} + \left(\tilde{q} + \epsilon^{-1}\right) \cdot \tau(n)$, where $\tau(n) = \text{polylog}(n)$ if $\mathcal{H}$ is strong collision-resistant, and $\tau(n) = O(n^{\gamma})$ for any constant $\gamma > 0$ if $\mathcal{H}$ is weak collision-resistant.

A similar analysis yields that the time complexity of the resulting verifier is $\tilde{t}_{\mathsf{V}} + \left(\tilde{q} + \epsilon^{-1}\right) \cdot \tau(n)$, where $\tilde{t}_{\mathsf{V}} \stackrel{\text{def}}{=} t_{\mathsf{V}}(n, \frac{\epsilon}{2})$ is the running time of the original verifier. This stems from the fact that in order to check the validity of each tree-commitment opening, the verifier needs to evaluate the hash function for each level of the tree, which takes $\text{poly}(k)$ time. The running time of the resulting honest prover is $\tilde{t}_{\mathsf{P}} + n \cdot \tau(n)$, where $\tilde{t}_{\mathsf{P}} \stackrel{\text{def}}{=} t_{\mathsf{P}}(n, \frac{\epsilon}{2})$ is the running time of the original honest prover, because the resulting honest prover needs to compute the entire Merkle tree.

As for the round complexity, in the general case we have 1 round in Stage 1, $\tilde{r}$ rounds in Stage 2, and $\tilde{q}$ rounds in Stages 3 and 4 combined. Therefore, the total number of rounds is $\tilde{r} + \tilde{q} + 1$. If the original IPP uses non-adaptive queries, then Stages 3 and 4 combined are executed in a single round, and thus the total number of rounds is $\tilde{r} + 2$.

The completeness of the system follows from the completeness of the original verifier. We thus turn to the computational soundness condition.

Claim 3.7 (computational soundness claim). *Let $P$ be a cheating prover that can be implemented by a polynomial-size family of circuits, and let $\{x_n\}_n$ be an infinite sequence of strings such that $x_n$ is $\epsilon$-far from $\Pi_n$. Then, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that for all $n$, on oracle access to $x_n$ and after interacting with $P$, the verifier in Construction 3.6 accepts with probability at most $1/3 + \mu(n)$.*

Proof: Starting with the high-level idea of the proof, consider an input $x$ that is $\epsilon$-far from $\Pi$. The idea is that once the tree commitment is established, the prover's answers to the queries must be

consistent with some fixed string $x'$. If $x'$ is $(\epsilon/2)$-close to $x$, then $x'$ is $(\epsilon/2)$-far from $\Pi$, and so the prover will fail to fool the original verifier with high probability. Otherwise, $x'$ is $(\epsilon/2)$-far from $x$, and then by randomly sampling $O(\epsilon^{-1})$ matching locations in $x'$ and $x$, we will detect a mismatch with high probability.

Fixing $n$, consider the interaction between $P$ and the verifier of Construction 3.6 on input $x_n$. Let $h$ be the hash function sampled in the first round, and let $r$ denote the randomness of the interacting module, excluding $h$ (i.e., $r$ consists of the randomness of the original verifier and the shared randomness). Let $B^{\text{val}}_{h,r}$ denote the set of blocks $i \in [m]$ that $P$ opens in Stages 3 and 4, for which it provides a valid opening. For $i \in B^{\text{val}}_{h,r}$, let $V_{h,r}(i)$ denote the value that $P$ reveals for block $i$. Note that $h, r, B^{\text{val}}_{h,r}$, and $V_{h,r}$ all depend on $n$, although this dependency is left implicit in the notation.

For each $n$, let $x'_h \in \{0,1\}^n$ be the string guaranteed by Lemma 3.3,[15] such that there exist a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ satisfying, for all $n$:

$$\Pr_{h,r}\left[\exists i \in B^{\text{val}}_{h,r} \; s.t. \; V_{h,r}(i) \neq x'_h(i)\right] \leq \mu(n)$$

We next analyze the probability of passing the Original Check and the Consistency Check of the deciding module. We will first consider an idealized prover $P'$ that is always consistent with $x'_h$. That is, in the first two stages $P'$ sends the same messages as $P$, and in Stages 3 and 4, for any block $i \in [m]$ that $P'$ opens, it reveals the value $x'_h(i)$, where $h$ is the hash function sent in the first round. Note that we do not require $P'$ to provide valid openings for the revealed values.

For each of the checks of the deciding module, we say that a prover passes the check on input $x$ if, on oracle access to $x$ and when interacting with this prover, the check passes. We next show that for all sufficiently large $n$'s, the probability that $P'$ passes both the Original Check and the Consistency Check on input $x_n$ is less than $1/3$. For each $n$, fix $h$ to an arbitrary hash function $h'$. Now, consider any $n$ for which $x'_{h'}$ is $(\epsilon/2)$-far from $x_n$. The prover $P'$ will pass the Consistency Check on input $x_n$ only if all the random locations in the shared randomness are locations where $x'_{h'}$ and $x_n$ agree, therefore

$$\Pr\left[P' \text{ passes the Consistency Check on input } x_n \,|\, h = h'\right] \leq \left(1 - \frac{\epsilon}{2}\right)^{O(1/\epsilon)} \leq \frac{1}{3} \tag{7}$$

It is left to deal with $n$'s for which $x'_{h'}$ is $(\epsilon/2)$-close to $x_n$. Since $x_n$ is $\epsilon$-far from $\Pi_n$, if $x'_{h'}$ is $(\epsilon/2)$-close to $x_n$, then $x'_{h'}$ is $(\epsilon/2)$-far from $\Pi_n$. Therefore, from the computational soundness of the original verifier, for all sufficiently large $n$'s such that $x'_{h'}$ is $(\epsilon/2)$-close to $x_n$, it holds that

$$\Pr\left[P' \text{ passes the Original Check on input } x_n \,|\, h = h'\right] \leq \frac{1}{3} \tag{8}$$

Having analyzed the idealized prover $P'$, we return to the actual prover $P$. Consider the event that the prover $P$ passes all checks of the deciding module on input $x_n$. There are two cases:

- The prover $P$ is inconsistent with $x'_h$; that is, there exists $i \in B^{\text{val}}_{h,r}$ such that $V_{h,r}(i) \neq x'_h(i)$. By the definition of $x'_h$, the probability of this event is negligible.
- The prover $P$ is consistent with $x'_h$; that is, for every block $i \in [m]$ that $P$ opens, either the opening it provides is invalid, or $V_{h,r}(i) = x'_h(i)$. Since we are considering the event where $P$ passes all checks, including the Openings Validity Check, it must be the case that for every

[15] By considering the parties that given common input $1^n$, simulate the current interaction on input $x_n$.

block $i \in [m]$ that $P$ opens, $V_{h,r}(i) = x'_h(i)$. Consequently, in this case, $P$ and $P'$ provide identical answers to all queries in addition to providing the same messages in Stages 1 and 2. Therefore, since $P$ passes the Original Check and the Consistency Check, so does $P'$. However, we have shown that for all sufficiently large $n$'s the probability of $P'$ passing both these checks is at most 1/3.

Hence, the probability that $P$ passes all checks on input $x_n$ is at most negligibly higher than 1/3. □

**Completing the proof of Theorem 3.5.** Claim 3.7 shows that the verifier in Construction 3.6 has computational-soundness error $1/3 + \mu$, where $\mu$ is a negligible function that depends on the cheating prover. Note that the constant 1/3 is arbitrary, and if the original soundness error was any other constant $e > 0$, then we would get soundness error $e + \mu$. Hence, to obtain soundness error at most 1/3, we can first perform $O(1)$ parallel repetitions of the original system. Note that error reduction by parallel repetitions is possible since the *original* system is public-coin (see [6, 5, 17]). The repetitions will increase the query and communication complexities (but not the round complexity) by a constant factor. ■

**Remark 3.8** (the effect of the emulation on the soundness error)**.** *We can modify Construction 3.6 so as to preserve the soundness error of the original verifier up to a negligible term: Suppose that the original verifier has computational-soundness error $s$, and we increase the number of shared random locations to $O(\epsilon^{-1} \cdot \log(s^{-1}))$. Then in Eq.* (7) *and* (8)*, the term $\frac{1}{3}$ will be replaced by $s$, and the resulting computational-soundness error will reduce to $s+\mu$, where $\mu$ is a negligible function that depends on the cheating prover. The resulting query complexity, as well as the term $\epsilon^{-1}$ in the communication complexity, will increase to $O(\epsilon^{-1} \cdot \log(s^{-1}))$, to account for the increase in the number of locations that we query the input oracle and the prover. Note that the round complexity does not increase.*

Remark 3.8 is especially important due to the fact that there is no parallel repetition theorem for general computationally sound systems.

We next show how to modify Construction 3.6 to reduce the number of rounds as per Theorem 3.4.

**Proof of Theorem 3.4:** The proof is identical to that of Theorem 3.5, where the only difference is that we modify Stage 3 of the interacting module as follows: Instead of requesting the original verifier's queries explicitly, the interacting module sends to the prover the entire random string of the original verifier (which is being used in the current emulated execution), and expects it to open all blocks corresponding to the locations that the original verifier would have queried in this execution. The rest of the construction and analysis remain exactly the same, except that the locations that are being queried (as well as the corresponding blocks being opened) are now *implicit* in the interaction transcript generated in Stage 2 and the random string sent in Stage 3, as well as the answers to prior queries according to the sequence of values that the prover provides. Note that (the new) Stage 3 and Stage 4 can be executed in a single round. ■

**Remark 3.9** (reducing the randomness complexity)**.** *In the above emulation (i.e., in the proof of Theorem 3.4), the new verifier sends to the prover the entire random string of the original verifier. First, note that the coins the original verifier sends during the interaction with the prover are already sent during Stage 2 of the emulation. Thus, in Stage 3 it is left only to send the outcome of additional coin tosses that the original verifier makes in order to generate its queries. By standard*

*techniques (see Appendix A), the number of these additional coin tosses can always be reduced to $O(\log n)$, at the cost of making the original verifier non-uniform (that is, implementable by a* non-uniform *polynomial-size family of circuits). When the original verifier is non-uniform, both the resulting verifier and the resulting honest prover (which emulates the queries of the original verifier) will be non-uniform. Hence, the randomness term $\rho$ in the statement of Theorem 3.4 can be reduced to $O(\log n)$, at the cost of making both the resulting verifier and the resulting honest prover non-uniform.*

### 3.2.2 Emulation of general cs-IPPs

We extend the pre-coordinated model emulation from the previous section, which handled *public-coin* cs-IPPs, to general (i.e., possibly *private*-coin) cs-IPPs. This extension relies on a computational PIR scheme (see Definition 2.4), where the communication complexity of the resulting cs-IPP depends on the communication complexity of the underlying PIR construction. This is established by the following theorem, which is identical to Theorem 3.5 except that: (1) the term $\tau(n)$ is replaced by $\tau(n) + c_{\mathcal{R}}(n)$ where $c_{\mathcal{R}}$ denote the communication complexity of the PIR scheme. (2) in the case of non-adaptive queries, the resulting round complexity is $O(r)$ rather than $r + 2$.[16]

**Theorem 3.10** (general cs-IPPs can be efficiently emulated in the pre-coordinated model)**.** *Let $\mathcal{H}$ be a family of* CRHFs*, and let $\tau(n) = \text{polylog}(n)$ if $\mathcal{H}$ is strong collision-resistant, and $\tau(n) = O(n^{\gamma})$ for any constant $\gamma > 0$ if $\mathcal{H}$ is weak collision-resistant. Let $\mathcal{R}$ be a computational* PIR *scheme with communication complexity $c_{\mathcal{R}}$. Suppose that a property $\Pi$ has a (possibly* private-coin*) cs-*IPP *with query complexity $q$, communication complexity $c$, and round complexity $r$. Then $\Pi$ has a cs-*IPP *in the pre-coordinated model with the following complexities:*

- *Query complexity: $O(1/\epsilon)$. Furthermore, the queries are uniformly and independently distributed.*
- *Communication complexity: $O\big(c(n, \frac{\epsilon}{2}) + \big(q(n, \frac{\epsilon}{2}) + \epsilon^{-1}\big) \cdot (\tau(n) + c_{\mathcal{R}}(n))\big)$.*
- *Round complexity: $O\big(r(n, \frac{\epsilon}{2}) + q(n, \frac{\epsilon}{2})\big)$ in general, and $O\big(r(n, \frac{\epsilon}{2})\big)$ if the original cs-*IPP *uses non-adaptive queries.*

*Furthermore, perfect completeness is preserved.*

The *time complexity* of the resulting verifier is $O\big(t_{\mathsf{V}}(n, \frac{\epsilon}{2}) + \big(q(n, \frac{\epsilon}{2}) + \epsilon^{-1}\big) \cdot (\tau(n) + t_{\mathsf{R}}(n))\big)$, where $t_{\mathsf{V}}$ is the running time of the original verifier, and $t_{\mathsf{R}}$ is the running time of the receiver in the PIR scheme. The time complexity of the resulting honest prover is $O\big(t_{\mathsf{P}}(n, \frac{\epsilon}{2}) + n \cdot \tau(n) + q(n, \frac{\epsilon}{2}) \cdot t_{\mathsf{S}}(n)\big)$, where $t_{\mathsf{P}}$ is the running time of the original honest prover, and $t_{\mathsf{S}}$ is the running time of the sender in the PIR scheme.

We remark that in [20] it was shown that the existence of a computational PIR scheme with communication complexity $o(n/\log n)$ implies the existence of *weak* CRHFs.[17] In Appendix B we show that the existence of a computational PIR scheme with *poly-logarithmic* communication complexity implies the existence of *a relaxed version* of *strong* CRHFs that suffices for Theorem 3.10 to hold. However, when using this relaxed form of strong CRHFs, the *time complexity* of the resulting verifier and resulting honest prover will increase, such that the term $\tau(n) = \text{polylog}(n)$

[16] Let us explain what goes behind the $O(r)$ expression. First, the queries made after each interaction round are replaced by parallel executions of the PIR scheme, and therefore $r$ rounds are replaced by $2r$. A minor technicality is that the soundness error may grow by an additive negligible term, and so we perform error reduction via *sequential* repetitions (where 3 repetitions suffice).

[17] The statement in [20] only claims this implication when the communication complexity of the PIR scheme is $O(n^c)$ for some constant $c < 1$, but their proof supports the foregoing stronger claim (see Appendix B).

appearing in the respective time complexities will be replaced by $O(n^\gamma)$ for an arbitrarily small constant $\gamma > 0$ (see details in Appendix B). In summery, putting aside the time complexity, we have that *assuming only the existence of a computational PIR scheme* with communication complexity polylog($n$) (resp., $O(n^\gamma)$ for an arbitrarily small constant $\gamma > 0$), we can get the guarantees of Theorem 3.10 where the factor $(\tau(n) + c_{\mathcal{R}}(n))$ equals polylog($n$) (resp., $O(n^\gamma)$ for an arbitrarily small constant $\gamma > 0$). That is, there is no need to assume the existence of CRHFs.

**Proof:** We modify Construction 3.6 to handle the case where the original verifier is not necessarily public-coin. Recall that in the public-coin case, we could first emulate the original interaction and only then ask the prover to provide the answers to the queries, since the messages of the original verifier were independent of the answers to those queries. To extend Construction 3.6 to handle general cs-IPPs, where the original verifier's messages may depend on the answers to its prior queries, we will ask the prover to provide the answers to those queries *during* the emulation of the original interaction. Note, however, that revealing to the prover the query locations may compromise the soundness of the original verifier. Thus, we use the PIR scheme to request the answers to the queries without revealing their locations.

We modify Construction 3.6 by replacing Stages 2 and 3 of the interacting module with the following:

**Construction 3.11** (Modified Steps 2&3 of the interacting module in Construction 3.6)**.**

> *The interacting module emulates a random execution of the original verifier with proximity parameter $\epsilon/2$, such that whenever the original verifier makes a query to the input-oracle, the interacting module instead requests the answer from the prover in the form of a* PIR *query. Specifically, if the original verifier queries position $i$, the interacting module sends a* PIR *query for position $i$, receives the prover's response, and uses it to reconstruct the bit value according to the* PIR *scheme. This reconstructed value is set as the (alleged) answer to the query, and the interacting module continues the execution of the original verifier using this answer.*
>
> *In the general case, each* PIR *execution is performed in a separate round. However, if the original verifier makes non-adaptive queries, then the* PIR *executions corresponding to the queries made between each interaction round can be carried out in parallel.*[18] *Each of the* PIR *executions is made with fresh randomness.*
>
> *At the last round of the foregoing emulation, the interacting module sends to the prover the locations of all the queries that the original verifier made during the emulated execution, and the prover is meant to respond with the corresponding tree-commitment openings for these locations.*[19]
>
> *The resulting interaction transcript, alleged answers, and alleged openings are used by the deciding module in the same manner as in Construction 3.6.*

All other components (the shared randomness, querying module, deciding module, and Stages 1 and 4 of the interacting module) remain identical to Construction 3.6.

We turn to the analysis of the resulting system. As before, we use the notation $\tilde{q} = q(n, \frac{\epsilon}{2})$ and $\tilde{r} = r(n, \frac{\epsilon}{2})$. Starting with the communication complexity, we only need to increase the communication bound from Eq. (6) by the communication cost of executing the PIR scheme for each of the

[18] Since the PIR scheme consists of only 1 round, it is straightforward that its security extends to parallel executions. That is, we have that any two *sequences* of (at most polynomially many) PIR queries are indistinguishable.

[19] I.e., for each location, the prover should provide the value of the entire corresponding tree-commitment block (which should be consistent with the query answer that was retrieved for the same location), along with the opening of that block.

$\tilde{q}$ original queries. As for the round complexity, recall that Stages 1 and 4 (in Construction 3.6) remain unchanged, and each requires 1 interaction round. In the general case, the modified Stages 2 and 3 require $\tilde{r} + \tilde{q}$ rounds since each query may require a separate round. If the original verifier uses non-adaptive queries, then the queries that the verifier makes between each interaction round can be emulated in parallel. Hence, in this case the modified Stages 2 and 3 can be executed in $2 \cdot \tilde{r}$ rounds,[20] and the total round complexity is thus $O(\tilde{r})$.

Completeness follows from the completeness of the original verifier and the correctness of the PIR scheme. We turn to the soundness condition.

Claim 3.12 (computational soundness claim). *Let $V'$ denote the resulting verifier in Construction 3.6 as modified by Construction 3.11. Let $P$ be a cheating prover that can be implemented by a polynomial-size family of circuits, and let $\{x_n\}_n$ be an infinite sequence of strings such that $x_n$ is $\epsilon$-far from $\Pi_n$. Then, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that for all $n$, on oracle access to $x_n$ and after interacting with $P$, the verifier $V'$ accepts with probability at most $1/3+\mu(n)$.*

Proof: We define $h$, $r$, $B^{\text{val}}_{h,r}$, $V_{h,r}(i)$, and $x'_h$ in the same way as in the proof of Theorem 3.5. Similarly to the proof of Theorem 3.5, we consider an idealized prover $P'$ that acts identically to $P$, except that it answers all queries consistently with $x'_h$. That is, for each PIR-query that $P'$ receives, $P'$ follows the honest sender strategy in the PIR scheme with respect to input $x'_h$. Additionally, when receiving the random consistency-check locations in Stage 4, $P'$ answers with the corresponding values in $x'_h$. In the rest of the interaction, $P'$ follows the strategy of $P$.

Similarly to the proof of Theorem 3.5, it suffices to show that $P'$ cannot pass both the Original and Consistency Checks with high probability. For each $n$, fix $h$ to an arbitrary hash function $h'$. By the same argument as Eq. (7), for each $n$ such that $x'_{h'}$ is $(\epsilon/2)$-far from $x_n$ it holds that $P'$ passes the Consistency Check with probability at most $1/3$. It is left to deal with $n$'s for which $x'_{h'}$ is $(\epsilon/2)$-close to $x_n$. We show that there exists a negligible function $\mu'$ such that for each such $n$ it holds that:

$$\Pr\left[P' \text{ passes the Original Check on input } x_n \,|\, h = h'\right] \leq \frac{1}{3} + \mu'(n) \tag{9}$$

Let $P'_{h'}$ and $V'_{h'}$ denote the strategies of $P'$ and $V'$ from the second round onward, after fixing the hash function in the first round to $h'$. Consider a prover $\widetilde{P}$ that interacts with the *original verifier* by emulating the messages of $P'_{h'}$, such that whenever $P'_{h'}$ expects to receive a PIR-query, $\widetilde{P}$ feeds it with a "dummy" PIR query that it obtains by running the query algorithm of the PIR scheme on index 1, using fresh randomness. Note that $\widetilde{P}$ can be implemented by a polynomial-size family of circuits.

Let $r_V$ denote the randomness of the original verifier and $r_R$ denote the randomness used to generate a single PIR query. Let $\bar{r}_R = (r_1, \ldots, r_{\tilde{q}})$ be a sequence of $\tilde{q} = q(n, \frac{\epsilon}{2})$ independent random variables each distributed according to $r_R$ (representing the randomness for generating $\tilde{q}$ PIR queries). Let $U = U(r_V, \bar{r}_R)$ denote the emulated view of the original verifier created in the interaction between $P'_{h'}$ and $V'_{h'}$ on input $x_n$ (in the modified Steps 2&3), where $V'_{h'}$ uses randomness $r_V$ for the original verifier emulation and randomness $\bar{r}_R$ for the PIR queries. Let $\widetilde{U} = \widetilde{U}(r_V, \bar{r}_R)$ denote the view of the original verifier when interacting with $\widetilde{P}$ on input $x'_{h'}$, where the original verifier uses randomness $r_V$, and $\widetilde{P}$ uses randomness $\bar{r}_R$ to generate the dummy queries. Note that $r_V$, $r_R$, $U$ and $\widetilde{U}$ implicitly depend on $n$.

[20] Note that, as will be discussed later, to obtain soundness error $1/3$ we will perform $O(1)$ *sequential* repetitions. Thus, the round complexity will increase by an additional constant factor.

As we will prove below, we claim that $U$ and $\widetilde{U}$ are computationally indistinguishable. This implies that the probability that $P'_{h'}$ passes the Original Check on input $x_n$ is at most negligibly higher than the probability that the original verifier accepts when interacting with $\widetilde{P}$ on input $x'_{h'}$ (as otherwise the efficient decision process of the original verifier distinguishes the views $U$ and $\widetilde{U}$). From the computational soundness of the original verifier, for all sufficiently large $n$'s such that $x'_{h'}$ is $(\epsilon/2)$-close to $x_n$ (and hence $(\epsilon/2)$-far from $\Pi$), the probability that the original verifier accepts $x'_{h'}$ when interacting with $\widetilde{P}$ is at most 1/3. Hence, we get Eq. (9), as desired.

*Proof that $U$ and $\widetilde{U}$ are computationally indistinguishable.* We essentially follow a standard argument for showing preservation of security under sequential composition.[21] In the following, we will refer to emulation rounds as the rounds in the interaction with $V'_{h'}$ that correspond to an emulation of the original interaction, whereas we will refer to PIR rounds as the rounds in which the PIR scheme is executed. Note that both $U$ and $\widetilde{U}$ describe the view of the original verifier on input $x'_{h'}$ when interacting with the strategy of $P'_{h'}$ in the emulation rounds, where the difference is what is being fed to $P'_{h'}$ in the PIR rounds. In the case of $\widetilde{U}$, we have that $P'_{h'}$ is fed with dummy PIR queries, whereas in the case of $U$, we have that $P'_{h'}$ is fed with PIR queries corresponding to the actual locations that the original verifier queries in the respective part of the interaction.

Assume towards contradiction that there exists a polynomial-size family of circuits $\{D_n\}_n$ that distinguishes between $U$ and $\widetilde{U}$.[22] For starters, suppose that the original verifier makes only one query (i.e., $\tilde{q} = 1$). Since $\{D_n\}_n$ distinguishes $U$ and $\widetilde{U}$, there exists (for each $n$) a fixed randomness string $r'_V$ such that $\{D_n\}_n$ distinguishes $U(r'_V, \bar{r}_R)$ and $\widetilde{U}(r'_V, \bar{r}_R)$. We construct a polynomial-size family of circuits $\{D'_n\}_n$ that distinguishes between a dummy PIR query and a PIR query corresponding to the actual location that the original verifier queries when using randomness $r'_V$ (and receiving the messages that $P'_{h'}$ sends prior to the query). Given a query $Q$, the circuit $D'_n$ emulates the interaction between the original verifier and the strategy of $P'_{h'}$ in the emulation rounds, where the verifier gets oracle access to $x'_{h'}$ and uses randomness $r'_V$, and $P'_{h'}$ is fed with $Q$ in the PIR round. It then runs the distinguisher $D_n$ on the resulting view of the original verifier. Note that if $Q$ is the actual PIR query, then the view is distributed according to $U(r'_V, \bar{r}_R)$, and if $Q$ is the dummy query, then the view is distributed according to $\widetilde{U}(r'_V, \bar{r}_R)$. Since $\{D_n\}_n$ distinguishes $U(r'_V, \bar{r}_R)$ and $\widetilde{U}(r'_V, \bar{r}_R)$, it follows that $\{D'_n\}_n$ distinguishes between the aforementioned queries, contradicting the security of the PIR scheme.

Turning to the actual case, where there can be multiple queries, we use a hybrid argument. For each $i \in [\tilde{q}] \cup \{0\}$, we define a hybrid view $U_i = U_i(r_V, \bar{r}_R)$. The hybrid $U_i$ describes the view of the original verifier when given randomness $r_V$, oracle access to $x'_{h'}$, and interacting with the strategy of $P'_{h'}$ in the emulation rounds, where in the PIR rounds $P'_{h'}$ is fed with PIR queries generated using $\bar{r}_R$ as follows: In the first $i$ PIR executions, $P'_{h'}$ is fed with PIR queries corresponding to the actual locations that the original verifier queries in the respective part of the interaction, and in the remaining $\tilde{q} - i$ PIR executions, $P'_{h'}$ is fed with dummy PIR queries.[23] Note that $U_0 = \widetilde{U}$ and $U_{\tilde{q}} = U$.

Since $\{D_n\}_n$ distinguishes the views $U_0$ and $U_{\tilde{q}}$, it holds that (for each $n$) there exists $i \in [\tilde{q}]$ such that $\{D_n\}_n$ distinguishes $U_{i-1}$ and $U_i$. Thus, there exist fixed randomness strings $r'_V, r'_1, \ldots, r'_{i-1}$

[21] As mentioned before, security under *parallel* composition (which we use in the non-adaptive case) is straightforward here, because indistinguishability of the queries easily extends to indistinguishability of any two *sequences* of (at most polynomially many) queries.

[22] That is, there exists a positive polynomial $p$ such that for infinitely many $n$'s it holds that $|\Pr[D_n(U(r_V, \bar{r}_R)) = 1] - \Pr[D_n(\widetilde{U}(r_V, \bar{r}_R)) = 1]| \geq 1/p(n)$.

[23] Note that if PIR executions are performed in parallel, then we may feed $P'_{h'}$ in the same PIR round a mix of actual PIR queries and dummy PIR queries.

such that $\{D_n\}_n$ distinguishes $U'_{i-1} = U_{i-1}(r'_V, r'_1, \ldots, r'_{i-1}, r_i, \ldots, r_{\tilde{q}})$ and $U'_i = U_i(r'_V, r'_1, \ldots, r'_{i-1}, r_i, \ldots, r_{\tilde{q}})$. We construct a polynomial-size family of circuits $\{D'_n\}_n$ that distinguishes between a dummy PIR query and the PIR query corresponding to the actual location that the original verifier queries in its $i^{\text{th}}$ query in the execution described by $U'_{i-1}$, contradicting the security of the PIR scheme. The circuit $D'_n$ has $i$ and $r'_V, r'_1, \ldots, r'_{i-1}$ hard-wired. Given a query $Q$, it emulates the interaction described by $U'_{i-1}$, except that in the $i^{\text{th}}$ PIR execution it feeds $P'_{h'}$ with $Q$. It then runs the distinguisher $D_n$ on the resulting view of the verifier. Note that if $Q$ is the actual $i^{\text{th}}$ PIR query, then the resulting view is distributed according to $U'_i$, and if $Q$ is the dummy query, then the resulting view is distributed according to $U'_{i-1}$. Since $\{D_n\}_n$ distinguishes $U'_i$ and $U'_{i-1}$, it follows that $\{D'_n\}_n$ distinguishes between the aforementioned queries, contradicting the security of the PIR scheme. This concludes the proof that $U$ and $\widetilde{U}$ are computationally indistinguishable.

*Completing the proof of Claim 3.12.* As described above, from the indistinguishability of $U$ and $\widetilde{U}$ we derive Eq. (9), and therefore we conclude that the probability that $P'$ passes both the Original and Consistency Checks is at most negligibly higher than 1/3. By the same argument used in the proof of Theorem 3.5, this implies that the probability that the actual prover $P$ passes all checks is at most negligibly higher than 1/3, completing the proof. □

Claim 3.12 shows that the resulting verifier $V'$ has computational-soundness error $1/3 + \mu$, where $\mu$ is a negligible function that depends on the cheating prover. To reduce this error to 1/3, we use $O(1)$ *sequential* repetitions. ■

### 3.3 Transforming the emulation to the isolated model (at a significant cost)

In the previous section, we showed that (under standard computational assumptions) cs-IPPs can be efficiently emulated in the pre-coordinated model. We turn to consider what can be achieved in the *isolation* model.

We observe that all of our emulations in the pre-coordinated model (i.e., Theorem 3.4, 3.5 and 3.10) can be transformed to the isolated model at the cost of increasing the *product* of the query and communication complexities by roughly $n/\epsilon$. More precisely, we can get the following tradeoff between the resulting query and communication complexities: For any $q' > O(1/\epsilon)$, we can obtain query complexity $q'$ while increasing the communication complexity (relative to that obtained in the respective theorems) by an additive $O\left(\frac{n}{\epsilon \cdot q'}\right) \cdot \tau(n)$ term, where $\tau(n)$ is as in the statement of Theorem 3.4. This tradeoff is nearly optimal, since we show in Section 4 that cs-IPPs in the isolated model that use $q$ queries and $c > 0$ bits of communication can be emulated by testers with query complexity $O(c \cdot q)$, whereas there exist properties that have very efficient cs-IPPs that require nearly-linear queries to test. Concretely, there exists a property $\Pi$ that requires $\Omega(n^{0.999})$ queries to test, but can be verified by a doubly-efficient MAP with proof length $O(\log n)$ and query complexity $\text{poly}(1/\epsilon)$ (see [16] followed by [11, Thm. 1.3]). By the aforementioned emulation of isolated cs-IPPs by testers, the product of the query and communication complexities in any *isolated* cs-IPP for $\Pi$ must be greater than $\Omega(n^{0.999})$.

To transform our emulations to the isolated model, we only modify the way that we check for consistency between the committed string and the actual input. Recall that in the pre-coordinated emulations, consistency is verified by comparing the two strings at $O(1/\epsilon)$ random locations that are *shared* between the querying and interacting modules. To transform the emulation to the isolated model, where the two modules can no longer share randomness, we proceed as follows: Let $C > 0$ be a sufficiently large constant, and let $q' > C \cdot \frac{1}{\epsilon}$ be the desired query complexity. The querying module queries the input at $q'$ distinct random points in $[n]$. Independently, the interacting module asks the prover (in Stage 4) to reveal the committed string at $\ell \stackrel{\text{def}}{=} C \cdot \frac{n}{\epsilon \cdot q'}$ distinct random points

in $[n]$. We say that a collision occurs when a location is both queried by the querying module and revealed by the prover. The deciding module checks (in the Consistency Check) that at every collision, the value provided by the prover matches the value obtained by the querying module.

To establish the correctness of the emulations under this modification, we only need to show that if the prover provides values consistent with some string $x'$ that is $(\epsilon/2)$-far from the actual input $x$, then the Consistency Check fails with high probability. First, observe that the expected number of collisions is $\frac{q' \cdot \ell}{n} = C \cdot \frac{1}{\epsilon}$. Hence, with probability at least $1 - \exp(-\Omega(C))$, there are at least $\frac{C}{2} \cdot \frac{1}{\epsilon}$ collisions.[24] Second, consider the probability space under the condition of having at least $k \stackrel{\text{def}}{=} \frac{C}{2} \cdot \frac{1}{\epsilon}$ collisions, and consider the first $k$ queries made by the querying module that are a part of a collision. Observe that (conditioned on having at least $k$ collisions) these queries are distributed *uniformly* over all possible $k$ distinct points in $[n]$. Therefore, since $x$ and $x'$ disagree on at least an $\epsilon/2$ fraction of locations, the probability, conditioned on having at least $\frac{C}{2} \cdot \frac{1}{\epsilon}$ collisions, that $x$ and $x'$ agree on all collision locations, is less than $(1 - \epsilon/2)^{C/(2\epsilon)} \leq \exp(-\Omega(C))$. Overall, the deciding module detects a disagreement between $x$ and $x'$ with probability at least $(1 - \exp(-\Omega(C)))^2$, which can be made sufficiently high by setting $C$ to be sufficiently large.

# 4 The limits of computationally sound IPPs in the isolated model

In this section we prove Theorem 1.2 which asserts that any property that can be verified by a cs-IPP in the *isolated* model using $q$ queries and $c > 0$ bits of communication, can be tested using $O(c \cdot q)$ queries.

**Proof:** Let $(Q, I, D)$ be the querying, interacting, and deciding modules of a verifier in an isolated cs-IPP for a property $\Pi = \bigcup_n \Pi_n$, and let $R_Q$ and $R_I$ be the randomness of the querying and interacting modules, respectively. For each input $x \in \{0,1\}^n$ and each fixed interaction transcript $\tau \in \{0,1\}^c$, let $p^x_\tau$ denote the probability that the deciding module accepts $x$ given interaction transcript $\tau$:

$$p^x_\tau \stackrel{\text{def}}{=} \Pr\left[D(Q^x(R_Q), \tau, R^\tau_I) = 1\right]$$

where $R^\tau_I$ denotes a random variable that is uniform over all randomness strings of the interacting module that are consistent with the transcript $\tau$.[25]

Let $P$ denote the honest prover. For each $x' \in \{0,1\}^n$, let $v^x_{P(x')}$ denote the probability that the deciding module accepts $x$ when interacting with the strategy of the honest prover on input $x'$:

$$\begin{aligned} v^x_{P(x')} &\stackrel{\text{def}}{=} \Pr\left[D\big(Q^x(R_Q), \langle P(x'), I(R_I)\rangle\big) = 1\right] \\ &= \sum_{\tau \in \{0,1\}^c} \Pr\left[\mathcal{T}(P(x'), I(R_I)) = \tau\right] \cdot p^x_\tau \end{aligned} \tag{10}$$

where $\mathcal{T}(P(x'), I(R_I))$ denotes the transcript of the interaction between $P(x')$ and $I(R_I)$.

We construct a tester for $\Pi$. Given oracle access to $x \in \{0,1\}^n$, the tester works as follows. For each $\tau \in \{0,1\}^c$ the tester obtains an estimate of $p^x_\tau$, denoted $\tilde{p}^x_\tau$, by making $O(c)$ repeated invocations of $Q^x$. Specifically, it samples $r \sim R_Q$, invokes $Q^x$ on $r$, and uses the result to compute

[24] This follows from the extension of the Chernoff Bound for sampling *without replacement* (i.e., for the Hypergeometric distribution).

[25] That is, a randomness string $r$ is consistent with the transcript $\tau$ if for every prefix of $\tau$ of the form $(\tau', \alpha)$, where $\alpha$ represents a verifier message, it holds that the next message of the interacting module given randomness string $r$ and partial interaction transcript $\tau'$ is $\alpha$.

$\Pr_{R_I^\tau}[D(Q^x(r),\tau,R_I^\tau)=1]$ for each $\tau\in\{0,1\}^c$.[26] It repeats this process independently $O(c)$ times, and takes $\tilde{p}_\tau^x$ to be the average of the corresponding computed values. We stress that the same invocations of $Q^x$ are used towards all $2^c$ estimates. Hence, the tester makes a total of $O(c\cdot q)$ queries. Based on these estimates, for each $x'\in\Pi_n$ the tester obtains an estimate of $v^x_{P(x')}$, denoted $\tilde{v}^x_{P(x')}$, by substituting $\tilde{p}_\tau^x$ for $p_\tau^x$ in Eq. (10) (and computing all coefficients $\Pr[\mathcal{T}(P(x'),I(R_I))=\tau]$ by brute-force). It then accepts if and only if there exists an $x'\in\Pi_n$ such that $\tilde{v}^x_{P(x')}>1/2$.

*Correctness.* By Chernoff's Bound, using $O(c)$ independent samples for the estimates $\tilde{p}_\tau^x$ we can ensure that each estimate $\tilde{p}_\tau^x$ falls within additive deviation less than $1/6$ from $p_\tau^x$ with probability at least $1-\frac{1}{3}\cdot 2^{-c}$. By the union bound, with probability at least $2/3$ all $2^c$ estimates $\tilde{p}_\tau^x$ are successful, which in turn implies that all approximations $\tilde{v}^x_{P(x')}$ (for all $x'\in\Pi_n$) are successful; that is, fall within additive deviation less than $1/6$ from $v^x_{P(x')}$ (because the values $v^x_{P(x')}$ are convex combinations of the $p_\tau^x$'s).

We next show that if all estimates $\tilde{v}^x_{P(x')}$ are successful, then if $x$ is in $\Pi_n$ the tester accepts, and, for all sufficiently large $n$'s, if $x$ is $\epsilon$-far from $\Pi_n$ the tester rejects.[27] First, if $x$ is in $\Pi_n$, then (by the completeness condition) it holds that $v^x_{P(x)}\geq 2/3$, and hence, if all estimates are successful, we have that $\tilde{v}^x_{P(x)}>2/3-1/6=1/2$, and the tester will accept.

We turn to the case where $x$ is $\epsilon$-far from $\Pi_n$. Let $\Gamma_\epsilon(\Pi_n)\stackrel{\text{def}}{=}\{x\in\{0,1\}^n : x \text{ is } \epsilon\text{-far from } \Pi_n\}$. For each $n$, define a cheating prover strategy $\widetilde{P}_n$ that equals $P(x_n^*)$ such that $x_n^*$ is an input in $\Pi_n$ that maximizes the probability that the verifier falsely accepts some $x\in\{0,1\}^n$ which is $\epsilon$-far from $\Pi_n$ when interacting with $P(x_n^*)$ (that is, $x_n^*$ is the input that among all $x'\in\Pi_n$ maximizes the value $\max_{x\in\Gamma_\epsilon(\Pi_n)}(v^x_{P(x')})$). A polynomial-size family of circuits can implement $\{\widetilde{P}_n\}_n$, because it may have $x_n^*$ hard-wired and emulate the polynomial-time honest prover. Therefore, from the computational soundness of the verifier, for all sufficiently large $n$'s, for all $x\in\Gamma_\epsilon(\Pi_n)$, the verifier rejects $x$ when interacting with $\widetilde{P}_n$ with probability at least $2/3$. Thus, for all sufficiently large $n$'s, for all $x\in\Gamma_\epsilon(\Pi_n)$, by the maximality of $x_n^*$ we have that $v^x_{P(x')}\leq 1/3$ for all $x'\in\Pi_n$. Hence, if all estimates are successful, then $\tilde{v}^x_{P(x')}<1/3+1/6=1/2$ for all $x'\in\Pi_n$, and the tester will reject. ■

**Remark:** We can modify the foregoing emulation (of isolated cs-IPPs by testers) so as to get a tester with query complexity $O((\ell+\log n)\cdot q)$, where $\ell$ is the total length of the *prover's messages*. An analogous result was shown in [13] for their emulation of *statistically sound* isolated IPPs (see the remark after the proof of [13, Thm. 1.2]). There, they obtain a tester with query complexity $O((\ell+r\cdot\log\ell)\cdot q)$ where $r$ is the number of rounds of the isolated IPP.

To prove the foregoing claim, consider again the expression from Eq. (10):

$$v^x_{P(x')}=\sum_{\tau\in\{0,1\}^c}\Pr\left[\mathcal{T}(P(x'),I(R_I))=\tau\right]\cdot p_\tau^x$$

The idea is that we can approximate $v^x_{P(x')}$ by sampling a few $R_I$'s, thereby reducing the number of fixed-transcript probabilities $p_\tau^x$ we need to approximate. Similar to the argument we had when sampling from $R_Q$, the same samples of $R_I$ can be used towards approximating all values $v^x_{P(x')}$ (i.e., for all $x'$). Specifically, in order to get with high constant probability a constant additive approximation of all (possibly $2^n$) values $v^x_{P(x')}$, we need to take $t=O(n)$ samples of $R_I$. Once we

[26] The probabilities $\Pr_{R_I^\tau}[D(Q^x(r),\tau,R_I^\tau)=1]$ can be computed by scanning all possible values for $R_I^\tau$.

[27] This is sufficient because we only care about asymptotic complexities. For smaller $n$'s the tester can decide if the input is in the property by reading the entire input.

have sampled the $R_I$'s, it suffices to approximate only the $p^x_\tau$'s for transcripts $\tau$ that are consistent with the messages that the interacting module sends when using the random strings that were sampled. Note that once the random string of the interacting module is fixed, its messages are determined from the messages of the prover. Hence, for each sample $r \sim R_I$, there are $2^\ell$ possible transcripts that are consistent with the messages that $I$ sends when using randomness $r$, where $\ell$ is the total length of the prover's messages. Since we take $t$ samples of $R_I$, we need to approximate $t \cdot 2^\ell$ of the fixed-transcript probabilities $p^x_\tau$ in total. To achieve this, we need only $O(\log(t \cdot 2^\ell)) = O(\ell + \log n)$ samples of $R_Q$.

## Acknowledgments

I am deeply grateful to my advisor, Oded Goldreich, for his kind guidance and support, which went above and beyond and always centered on instilling the underlying principles and fostering learning. I also wish to thank Tom Gur, Ron Rothblum, and Guy Rothblum for helpful discussions, and anonymous reviewers for helpful comments.

## References

[1] Noga Amir, Oded Goldreich, and Guy N. Rothblum. "Doubly Sub-Linear Interactive Proofs of Proximity". In: *16th Innovations in Theoretical Computer Science Conference*. Vol. 325. LIPIcs. 2025, 6:1–6:25.

[2] Boaz Barak and Oded Goldreich. "Universal Arguments and their Applications". In: *Proceedings of the 17th IEEE Annual Conference on Computational Complexity*. CCC '02. IEEE Computer Society, 2002, p. 194.

[3] Eli Ben-Sasson et al. "Robust pcps of proximity, shorter pcps and applications to coding". In: *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*. STOC '04. Association for Computing Machinery, 2004, pp. 1–10.

[4] Itay Berman, Ron D. Rothblum, and Vinod Vaikuntanathan. "Zero-Knowledge Proofs of Proximity". In: *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Vol. 94. Leibniz International Proceedings in Informatics (LIPIcs). 2018, 19:1–19:20.

[5] Kai-Min Chung and Feng-Hao Liu. "Parallel Repetition Theorems for Interactive Arguments". In: *Theory of Cryptography - 7th Theory of Cryptography Conference, TCC, Proceedings*. Vol. 5978. Lecture Notes in Computer Science. Springer, 2010, pp. 19–36.

[6] Kai-Min Chung and Rafael Pass. "Tight Parallel Repetition Theorems for Public-Coin Arguments Using KL-Divergence". In: *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC, Proceedings, Part II*. 2015, pp. 229–246.

[7] Irit Dinur and Omer Reingold. "Assignment Testers: Towards a Combinatorial Proof of the PCP Theorem". In: *SIAM J. Comput.* 36.4 (2006), pp. 975–1024.

[8] Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. "Fast approximate probabilistically checkable proofs". In: *Information and Computation* 189.2 (2004), pp. 135–159.

[9] Guy Goldberg and Guy N. Rothblum. "Sample-Based Proofs of Proximity". In: *13th Innovations in Theoretical Computer Science Conference*. Vol. 215. Leibniz International Proceedings in Informatics (LIPIcs). 2022, 77:1–77:19.

[10] Oded Goldreich, Shafi Goldwasser, and Dana Ron. "Property testing and its connection to learning and approximation". In: *J. ACM* 45.4 (1998), pp. 653–750.

[11] Oded Goldreich, Tom Gur, and Ilan Komargodski. "Strong Locally Testable Codes with Relaxed Local Decoders". In: *30th Conference on Computational Complexity*. Vol. 33. Leibniz International Proceedings in Informatics (LIPIcs). 2015, pp. 1–41.

[12] Oded Goldreich and Dana Ron. "On Sample-Based Testers". In: *ACM Trans. Comput. Theory* 8.2 (2016).

[13] Oded Goldreich, Guy N. Rothblum, and Tal Skverer. "On Interactive Proofs of Proximity with Proof-Oblivious Queries". In: *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*. Vol. 251. 2023, 59:1–59:16.

[14] Oded Goldreich and Or Sheffet. "On the randomness complexity of property testing". In: *Computational Complexity* 19 (2010), pp. 99–133.

[15] Tom Gur, Yang P. Liu, and Ron D. Rothblum. "An Exponential Separation Between MA and AM Proofs of Proximity". In: *Comp. Complexity* 30.2 (2021), p. 12.

[16] Tom Gur and Ron Rothblum. "Non-interactive proofs of proximity". In: *Computational Complexity* 27.1 (2018), pp. 99–207. Preliminary version in ECCC, TR13-078, 2013.

[17] Johan Håstad et al. "An Efficient Parallel Repetition Theorem". In: *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC. Proceedings*. Vol. 5978. Lecture Notes in Computer Science. Springer, 2010, pp. 1–18.

[18] Tal Herman and Guy N. Rothblum. "How to Verify Any (Reasonable) Distribution Property: Computationally Sound Argument Systems for Distributions". In: *The Thirteenth International Conference on Learning Representations*. 2025.

[19] Chun-Yuan Hsiao and Leonid Reyzin. "Finding Collisions on a Public Road, or Do Secure Hash Functions Need Secret Coins?" In: *Advances in Cryptology - CRYPTO 2004*. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 92–105.

[20] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. "Sufficient Conditions for Collision-Resistant Hashing". In: *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, 2005, Proceedings*. Vol. 3378. Lecture Notes in Computer Science. Springer, 2005, pp. 445–456.

[21] Yael Tauman Kalai and Ron D. Rothblum. "Arguments of Proximity - [Extended Abstract]". In: *CRYPTO*. Springer, 2015, pp. 422–442.

[22] Joe Kilian. "A note on efficient zero-knowledge proofs and arguments (extended abstract)". In: *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*. STOC '92. 1992, pp. 723–732.

[23] E. Kushilevitz and R. Ostrovsky. "Replication is not needed: single database, computationally-private information retrieval". In: *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*. FOCS '97. IEEE Computer Society, 1997, p. 364.

[24] Helger Lipmaa. "An Oblivious Transfer Protocol with Log-Squared Communication". In: *ISC*. 2004, pp. 314–328.

[25] Ralph C. Merkle. "Protocols for Public Key Cryptosystems". In: *Proceedings of the 1980 Symposium on Security and Privacy*. 1980, pp. 122–134.

[26] Guy N. Rothblum and Ron D. Rothblum. "Batch Verification and Proofs of Proximity with Polylog Overhead". In: *Theory of Cryptography: 18th International Conference, TCC, Proceedings, Part II*. Springer-Verlag, 2020, pp. 108–138.

[27] Guy N. Rothblum, Salil Vadhan, and Avi Wigderson. "Interactive proofs of proximity: delegating computation in sublinear time". In: *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*. STOC '13. Association for Computing Machinery, 2013, pp. 793–802.

[28] Ronitt Rubinfeld and Madhu Sudan. "Robust Characterizations of Polynomials with Applications to Program Testing". In: *SIAM Journal on Computing* 25.2 (1996), pp. 252–271.

[29] Hadar Strauss. "Emulating Computationally Sound Public-Coin IPPs in the Pre-Coordinated Model". In: *Electronic Colloquium on Computational Complexity (ECCC)* TR24-131 (2024).

[30] Hadar Strauss. "On the Limits of Computationally Sound IPPs in the Isolated Model". In: *Electronic Colloquium on Computational Complexity (ECCC)* TR25-097 (2025).

# Appendices

## A Reducing the amount of randomness in public-coin IPPs

The following claim shows that for any public-coin cs-IPP, the number of coin-tosses made by the verifier, excluding those that it sends during the interaction with the prover, can be reduced to $O(\log n)$ at the cost of making the verifier non-uniform. We define cs-IPPs with a non-uniform verifier to be a relaxation of cs-IPPs that allows the verifier to be implementable by a (non-uniform) polynomial-size family of circuits.

**Claim A.1.** *Suppose that a property $\Pi$ has an $r$-round public-coin cs-*IPP *with query complexity $q$ and communication complexity $c$, and assume that $c \leq n$. Then, $\Pi$ has an $r$-round public-coin cs-*IPP *with a non-uniform verifier that has query complexity $O(q)$ and communication complexity $O(c)$, such that the number of coin-tosses made by the verifier, excluding those it sends during the interaction with the prover, is at most $O(\log n)$. Furthermore, perfect completeness is preserved, and if the original cs-*IPP *uses non-adaptive queries, then so does the resulting cs-*IPP*.*

We note that the claim holds also if the *original* cs-IPP has a non-uniform verifier. Note that the overall randomness complexity of the resulting verifier is at most $O(c + \log n)$.

**Proof:** Since the original verifier is public-coin, we can (w.l.o.g.) decompose it to a querying, interacting, and deciding modules $Q$, $I$, and $D$, respectively, such that its decision when interacting with a prover $P$ on input $x$ is expressed as:

$$D(Q^x(\langle P(x), I(R_I)\rangle, R_Q))$$

where $R_I$ represents the coins that the interacting module sends during the interaction with the prover, and $R_Q$ represents *independent* coins used by the querying module. Observe that the verifier's acceptance probability when interacting with prover $P$ on input $x$ can be decomposed as follows:

$$\sum_{\tau \in \{0,1\}^c} \Pr[\langle P(x), I(R_I)\rangle = \tau] \cdot \Pr[D(Q^x(\tau, R_Q)) = 1] \tag{11}$$

We construct a new querying module $Q'$ that, together with the original modules $I$ and $D$, will constitute the claimed verifier. More specifically, we will construct a querying module $Q'$ that has randomness $R'_Q$ of length at most $O(\log n)$, such that for all inputs $x \in \{0,1\}^n$ and for all possible interaction transcripts $\tau \in \{0,1\}^c$, it holds that

$$\Pr[D(Q'^x(\tau, R'_Q)) = 1] \in \Pr[D(Q^x(\tau, R_Q)) = 1] \pm 1/7$$

That is, $Q'$ preserves the second term in Eq. (11) up to an additive deviation of $1/7$. This implies that the acceptance probability of the new verifier on any input $f$ and when interacting with any prover $P$ is the same as that of the original verifier up to an additive deviation of $1/7$, which in turn implies that the completeness and (computational) soundness errors of the new system are at most $1/3 + 1/7$. We reduce the error via $O(1)$ parallel repetitions (error reduction by parallel repetitions is possible since the system is public-coin [6, 5, 17]). The repetitions will increase the communication and query complexities (but not the round complexity) by a constant factor.

To construct the described querying module $Q'$, we follow the standard argument for randomness reduction (see, e.g., [14, Thm. 3] and [15, Apdx. A]). Consider a matrix in which each column

corresponds to a possible pair $(\tau, x)$ consisting of an interaction transcript $\tau \in \{0,1\}^c$ and an input string $x \in \{0,1\}^n$, and each row corresponds to a possible randomness string $r_Q$ in the support of $R_Q$. Each entry in the matrix that corresponds to column $(\tau, x)$ and row $r_Q$, contains the corresponding decision of the original verifier: $D(Q^x(\tau, r_Q))$.

Note that the matrix has at most $2^c \cdot 2^n$ columns. If we proceed in the standard argument for randomness reduction, we get that there exists a multi-set of size $s = O(\log(2^c \cdot 2^n)) = O(c+n)$ of the matrix's rows that preserves the average of all columns up to an additive deviation of $1/7$. The new querying module $Q'$ will have this multi-set hard-wired, and will emulate the original querying module $Q$ with a randomness string selected uniformly at random from the multi-set. Due to the hypothesis that $c \leq n$, we have that $s = O(n)$. Hence, the randomness complexity of $Q'$ is $\log(s) = O(\log n)$, as desired. Furthermore, since the size of the multi-set (which will be hard-wired) is polynomial in $n$ (specifically, $O(n)$), we have that $Q'$ can be implemented by a (non-uniform) polynomial-size family of circuits. ■

# B Poly-logarithmic PIR implies a relaxed form of strong CRHFs

In [20], it was shown that if there exists a computational PIR scheme with communication complexity $o(n/\log n)$, then there exists a family of (private-coin) *weak* CRHFs.[28] In this section, we show that the construction of [20] can be modified to establish that if there exists a computational PIR scheme with *poly-logarithmic* communication complexity, then there exists *a relaxed version* of (private-coin) *strong* CRHFs, where, as will be explained below, this relaxed version suffices for most of the purposes of this work. In the relaxed version, rather than requiring the running time of the indexing and the evaluation algorithms to be polynomial, we require only that the *description length* of the hash function outputted by the indexing algorithm is polynomial, and allow the algorithms to run in time $2^{O(k^\delta)}$, where $\delta$ is the parameter of the strong collision-resistance. Specifically, we refer to the following definition:

**Definition B.1** (strong collision-resistant hashing functions – a relaxed definition)**.** *Let $\mathcal{H} = \bigcup_{k\in\mathbb{N}} \mathcal{H}_k$ such that $\mathcal{H}_k \subseteq \{h : \{0,1\}^{2k} \to \{0,1\}^k\}$. $\mathcal{H}$ is said to be a family of strong collision-resistant hashing functions (CRHFs) if there exists a mapping from strings $s \in \{0,1\}^*$ to functions $h_s \in \mathcal{H}$, and a parameter $\delta > 0$ such that:*

1. (Efficient indexing): *There exists a probabilistic algorithm $I$, called an indexing algorithm, that for every $k \in \mathbb{N}$, given $1^k$, samples a string $s$ such that $h_s \in \mathcal{H}_k$, and: (1) the length of $s$ is at most polynomial in $k$, and (2) the running time of $I$ is at most $2^{O(k^\delta)}$.*

2. (Efficient evaluation): *There exists an algorithm, called an evaluation algorithm, that, given $x \in \{0,1\}^{2k}$ and a string $s$ in the range of $I(1^k)$, returns $h_s(x)$, and runs in time at most $2^{O(k^\delta)}$.*

3. (Hard-to-form collisions): *For every (non-uniform) family of circuits $\{C_k\}_{k\in\mathbb{N}}$ of size at most $2^{O(k^\delta)}$, it holds that:*

$$\Pr_{\substack{s\leftarrow I(1^k)\\(x_1,x_2)\leftarrow C_k(s)}}[h_s(x_1) = h_s(x_2) \wedge x_1 \neq x_2] \leq 2^{-\omega(k^\delta)}$$

[28] The statement in [20] only claims this implication when the communication complexity of the PIR scheme is $O(n^c)$ for some constant $c < 1$; however, as shown later in this section, their proof supports the foregoing stronger claim.

Using the above relaxed form of strong CRHFs in the construction of the tree-commitment scheme will affect *only the time complexity* (but *not* the communication complexity) of executing the tree-commitment scheme. Recall that in the tree-commitment scheme, when committing to an $n$-bit string, the string is partitioned into blocks of size $k$, and the CRHF family is used with this parameter $k$. If we use the above relaxation of strong CRHFs, then we can set $k = (\alpha \cdot \log n)^{1/\delta}$ for an arbitrarily small constant $\alpha > 0$. This ensures that the running time of the indexing and evaluation algorithms is $2^{O(k^\delta)} = \text{poly}(n^\alpha)$, which can be bounded by $O(n^\gamma)$ for any desired small constant $\gamma > 0$ (by choosing $\alpha$ sufficiently small). That is, whereas standard strong CRHFs yield time complexity $\text{poly}(k) = \text{polylog}(n)$ for the indexing and evaluation algorithms, the relaxed variant increases this to $O(n^\gamma)$ for arbitrarily small $\gamma > 0$. We stress that the *communication complexity* of the scheme remains $\text{polylog}(n)$ due to the requirement that the description length of the hash functions remains polynomial in $k$.

We proceed to establish the following:

**Theorem B.2.** *If there exists a computational* PIR *scheme with poly-logarithmic communication complexity (according to Definition 2.4), then there exists a family of* strong CRHF*s according to Definition B.1.*

We start by describing the construction of [20] (which showed how to obtain *weak* CRHF from a PIR scheme with communication complexity $o(n/\log n)$), and then show how to modify it to get Theorem B.2.

**The construction of [20].** Consider a computational PIR scheme with communication complexity $c = o(n/\log n)$, defined by a Query, Answer, and Reconstruction algorithms $(Q, A, R)$, respectively. We shall construct a weak CRHF family. The construction uses an error correcting code $\text{ECC}(\cdot)$ that takes as input a string of size $2k$ and outputs a string of size $n(k) = O(k)$, such that for any two distinct strings $x \neq x' \in \{0,1\}^{2k}$, the relative Hamming distance between $y = \text{ECC}(x)$ and $y' = \text{ECC}(x')$ is at least some constant $\epsilon > 0$.

The *basic* construction proceeds as follows: The indexing algorithm, on input $1^k$, outputs $q = Q(n, i, r)$, where $n = n(k)$ (as in the error correcting code), $i$ is a uniformly random index in $[n]$, and $r$ is the randomness used by the query algorithm $Q$. The hash function $h_q$, on input $x \in \{0,1\}^{2k}$, is defined as $h_q(x) = A(y, q)$, where $y = \text{ECC}(x)$. The *actual* construction takes $t = \omega(\log n)$ independent copies of the basic construction. That is, the indexing algorithm outputs $\bar{q} = (q_1, \ldots, q_t) = (Q(n, i_1, r_1), \ldots, Q(n, i_t, r_t))$, and the hash function evaluates to $h_{\bar{q}}(x) = (A(y, q_1), \ldots, A(y, q_t))$, where $i_1, \ldots, i_t$ and $r_1, \ldots, r_t$ are independent random variable distributed identically to $i$ and $r$, respectively, and as before $y = \text{ECC}(x)$.

Note that the indexing and evaluation algorithms run in time $\text{poly}(k)$, because the query and answer algorithms run in time $\text{poly}(n)$. (In particular, this implies that the hash function description length $|\bar{q}|$ is polynomial in $k$.) We next show that $h_{\bar{q}}$ shrinks its input. Observe that $h_{\bar{q}}$ takes strings of length $2k$ to strings of length $t \cdot |A(y, q)| \leq t \cdot c$. Hence, if $c = o(n/\log n)$ then we can set $t = \omega(\log n)$ such that the output length is at most $k$.

We turn to analyze the collision resistance property. Suppose that a family of circuits of size $\text{poly}(k)$ (equiv., $\text{poly}(n)$) finds, given $q$, two distinct strings $x \neq x' \in \{0,1\}^{2k}$ that collide under $h_q$; that is, $h_q(x) = A(y, q) = A(y', q) = h_q(x')$, where $y = \text{ECC}(x)$ and $y' = \text{ECC}(x')$. By the correctness of the PIR scheme, $y$ and $y'$ must agree on $i$ (since $y_i = R(n, i, r, A(y, q)) = R(n, i, r, A(y', q)) = y'_i$). However, since $x \neq x'$, we have that $y$ and $y'$ differ on at least $\epsilon \cdot n$ locations. Thus, finding a collision implies finding $\epsilon \cdot n$ locations that $i$ is not equal to. Without access to the query $q$, the probability of guessing such locations is at most $1 - \epsilon$. Now, finding a

collision under $h_{\bar{q}}$ implies finding, *for each index* $i_j$ (where $j \in [t]$), a fraction of $\epsilon$ locations that $i_j$ is not equal to. Without access to $\bar{q}$, the probability of succeeding is at most $(1-\epsilon)^t$, which by the choice of $t$ is *negligible* in $n$ (equiv., in $k$). By the semantic security of the PIR query algorithm $Q$, we conclude that finding a collision under $h_{\bar{q}}$ (when given $\bar{q}$) is feasible only with negligible probability, establishing the (weak) collision-resistance property.

**Adjusting the construction to get *strong* collision resistance: Proof of Theorem B.2.** Let $(Q, A, R)$ be a computational PIR scheme with communication complexity $c = \log^{\alpha}(n)$, for some constant $\alpha > 0$. Let $\delta > 0$ be a parameter to be set later such that $\delta = \frac{1}{\alpha + O(1)}$. We construct a family of *strong* CRHFs with parameter $\delta$ (according to the relaxed Definition B.1). The construction is identical to the construction of [20] described above, where the only exception is that we use $n = 2^{k^\delta}$ (rather than $n = O(k)$).

We begin by showing that $h_{\bar{q}}$ shrinks its input. As before, $h_{\bar{q}}$ takes strings of length $2k$ to strings of length $t \cdot |A(y, q)| \leq t \cdot c$. We have $c = \log^{\alpha}(n)$, and recall that $t$ needs to satisfy $t = \omega(\log n)$; e.g., we can set $t = \log^2(n)$. Therefore, the output length is at most $\log^{\alpha+O(1)}(n) = k^{\delta \cdot (\alpha + O(1))}$. Hence, we can set $\delta = \frac{1}{\alpha + O(1)}$ such that the output length is at most $k$.

The description length of a hash function outputted by the indexing algorithm (given $1^k$) is $t \cdot |Q(n, i, r)| \leq t \cdot c = \log^{\alpha+O(1)}(n) = \mathrm{poly}(k)$. The running time of the indexing algorithm and the evaluation algorithm is at most $2^{O(k^\delta)}$ since the running time of both the PIR query algorithm $Q$ and the PIR answer algorithm $A$ is polynomial in $n$.

Finally, we consider the (strong) collision-resistance property. By the same analysis of the original construction above, given $\bar{q}$, any family of $\mathrm{poly}(n)$-size circuits can succeed in finding a collision under $h_{\bar{q}}$ with probability at most negligible in $n$. Since we have set $n = 2^{k^\delta}$, this implies that, given $\bar{q}$, any family of circuits of size at most $2^{O(k^\delta)}$, can succeed in finding a collision under $h_{\bar{q}}$ with probability at most $2^{-\omega(k^\delta)}$, as desired.