Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 131 (2024)

Emulating Computationally Sound Public-Coin IPPs in the Pre-Coordinated Model

Hadar Strauss
Department of Computer Science
Weizmann Institute of Science, Rehovot, ISRAEL.

September 20, 2024

Abstract

Interactive proofs of proximity (IPPs) for a property are relaxed proof systems analogous to property testers in which the goal is for the verifier to be convinced to accept inputs that are in the property, and to not be fooled into accepting inputs that are far from the property.

The general definition of IPPs allows the verifier to fully coordinate its interaction with the prover and its queries to the input oracle. In contrast, in IPPs with proof-oblivious queries in the pre-coordinated model, the verifier is decomposed into separate modules, one interacting with the prover and another querying the input, such that no information flow between the modules is allowed. The only coordination that is allowed between the modules is through shared randomness.

Goldreich, Rothblum, and Skverer (ECCC, TR22-124) showed that while the pre-coordinated model is significantly more powerful than standard property testers, it is also considerably limited compared to general IPPs. In this work we show that if we relax the soundness condition to computational soundness, then the pre-coordinated model becomes significantly stronger. Specifically, assuming the existence of collision-resistant hashing functions, we obtain a computationally sound (public-coin) IPP in the pre-coordinated model for any property that has a general computationally sound public-coin IPP, such that the complexities of the resulting IPP are related to the complexities of the original IPP.

Contents

ISSN 1433-8092

1 Introduction

1.1 Defining Property Testers, IPPs, and IPPs with PO-queries in the pre-coordinated model

A property is a collection of sets $\Pi = \bigcup_{n\in\mathbb{N}} \Pi_n$ such that Π_n is a set of functions from $[n]$ to Σ. A property tester for property Π is a probabilistic algorithm that has oracle access to a function $f : [n] \to \Sigma$, and also gets explicit inputs n and $\epsilon > 0$, where ϵ is called the proximity parameter. The aim of the tester is to determine whether f is in Π_n or is ϵ-far from Π_n, where f is ϵ-far from Π_n if for every $g \in \Pi_n$ it holds that $|\{i \in [n] : f(i) \neq g(i)\}| > \epsilon \cdot n$ (and otherwise it is ϵ-close to Π_n). Specifically, if f is in Π_n, the tester will output 1 with probability at least $2/3$, and if f is ϵ-far from Π_n, the tester will output 0 with probability at least $2/3$. The query complexity of the tester is $q : \mathbb{N} \times [0, 1] \to \mathbb{N}$ if, on input n, ϵ and oracle access to any $f : [n] \to \Sigma$, the tester makes at most $q(n, \epsilon)$ queries to f. We usually seek testers of extremely low query complexity (e.g., query complexity that is poly-logarithmic in n).

Interactive proofs of proximity are an extension of property testers, analogous to how IP is an extension of BPP. Specifically, an interactive proof of proximity for a property Π is a two-party protocol for parties called verifier and prover. The verifier has oracle access to a function $f : [n] \to \Sigma$, and also gets explicit inputs n and $\epsilon > 0$. The prover gets f as explicit input, and its aim is to convince the verifier that f is in Π. We require that the prover can convince the verifier to accept any f in Π_n (w.h.p.), but cannot fool the verifier into accepting f that is ϵ-far from Π_n (except for with low probability). The prover is defied by its strategy, where a party's strategy is a (computationally unbounded) function that maps the party's input and all messages it has received so far, to the next message it will send.

Definition 1.1 (interactive proofs of proximity systems (IPPs))**.** *A randomized, polynomial-time interactive oracle machine, denoted V, constitutes a verifier for an interactive proof of proximity for a property Π 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, ϵ and oracle access to any $f \in \Pi_n$, after interacting with P that gets f as explicit input, the verifier 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, ϵ and oracle access to any $f : [n] \to \Sigma$ that is ϵ-far from Π_n, after interacting with P, the verifier accepts with probability at most $1/3$.*[1]

The system has completeness error (resp., soundness error) $e : \mathbb{N} \to [0, 1]$ if it satisfies the completeness (resp., soundness) condition when the term $1/3$ is replaced with $e(n)$. The system has perfect completeness if the verifier accepts each $f \in \Pi$ 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, ϵ, on oracle access to any $f : [n] \to \Sigma$ and when interacting with any prover, the verifier makes at most $q(n, \epsilon)$ queries to f, 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 is a predetermined sub-sequence of the randomness of the verifier. The system uses non-adaptive queries if the verifier's queries to the input function 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.

[1] Having the prover get the input function is not necessary in the soundness condition. If there is a function f such that there is a prover with access to the input that can fool the verifier on f, then a prover without such access can achieve the same by hard-coding f into its strategy.

Note that we can equivalently formulate properties (and accordingly property testers and IPPs) to be sets of strings over alphabet Σ, rather than sets of functions from n to Σ. Furthermore, we can focus on strings over $\{0,1\}$, by encoding elements of Σ by codewords of a code with constant relative distance. This will increase the query complexity by at most a factor of $O(\log|\Sigma|)$. We will switch between these formulations according to what is more convenient in the given context.

IPPs with proof-oblivious queries (PO-queries) are a special class of IPPs where the verifier's queries to the input function are oblivious of the messages received from the prover. To formally define IPPs with PO-queries, we follow the framework introduced by Goldreich, Rothblum, and Skverer [3], 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 function, 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. 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 IPPs with PO-queries we can write the random variable representing the decision of the verifier as $D(\langle P(f), I(Q^f(R))\rangle)$ where R is a random variable representing the randomness of the verifier.

Goldreich et al. [3] introduced a restricted model of IPPs with PO-queries, called the pre-coordinated model. In this model, the querying and interacting modules share randomness but there is no information flow between them. In this case, we write the random variable representing the decision as $D(Q^f(R), \langle P(f), I(R)\rangle)$, where R is a random variable representing the shared randomness of both modules. In their work, it was shown that while IPPs in the pre-coordinated model are significantly more powerful than standard property testers, they are also considerably limited compared to general IPPs. Specifically, they showed that public-coin $O(1)$-round IPPs in the pre-coordinated model are as limited as standard property testers. Furthermore, they conjecture that $O(1)$-round IPPs in the pre-coordinated model are weaker than 1-round IPPs in the general PO-queries model, and that IPPs in the pre-coordinated model are weaker than 1/2-round general IPPs (MAPs).

In this work, we extend the study of IPPs with PO-queries in the pre-coordinated model to the context of *computationally sound* interactive proofs of proximity. That is, we relax the soundness condition of Definition 1.1 to hold only against efficient provers. This leads us to also restrict the computational power of the prover in the completeness condition.[2] Actually, we even require the honest prover's strategy to be implementable in probabilistic polynomial-time, although our result holds also if it is allowed to be implemented by a non-uniform polynomial-size family of circuits.

Definition 1.2 (computationally sound interactive proofs of proximity systems)**.** *A randomized, polynomial-time interactive oracle machine, denoted V, constitutes a verifier for a computationally sound interactive proof of proximity for a property Π 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, ϵ and oracle*

[2] It is unnatural to consider, in the completeness condition, honest provers that are more powerful than the potentially cheating provers considered in the (computational) soundness condition. In fact, the common goal is to have proof systems with an honest prover that is as efficient as possible, while ensuring soundness against the most powerful possible cheating provers. We note that avoiding this natural convention leads to weird proof systems in which the verifier ignores the input and accepts the prover's claim if the latter succeeds in solving a computationally hard problem (in NP or just in IP).

access to any $f \in \Pi_n$, after interacting with P that gets f as explicit input, the verifier 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, ϵ and oracle access to any $f : [n] \to \Sigma$ that is ϵ-far from Π_n, after interacting with P, the verifier accepts with probability at most* $1/3$.

In order to obtain computationally sound proof systems that do not satisfy standard soundness, we introduce a computational hardness assumption.[3] Specifically, we will assume the existence of collision-resistant hash functions.

Definition 1.3 (strong and weak collision-resistant hashing functions (CRHF))**.** *Let $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$ be an efficiently computable hash function ensemble such that $\mathcal{H}_k$ assumes values in the set $\{h : \{0,1\}^{2k} \to \{0,1\}^k\}$.*

- *$\mathcal{H}$ is said to be* strong collision-resistant (CRHF)*, if 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^{k^δ}, for all sufficiently large k's, it holds that:*

$$\Pr_{\substack{h\leftarrow\mathcal{H}_k\\(x_1,x_2)\leftarrow C_k(h)}}[h(x_1) = h(x_2) \wedge x_1 \neq x_2] \leq 2^{-k^\delta}$$

  In this case we will refer to δ as the parameter *of $\mathcal{H}$.*

- *$\mathcal{H}$ is said to be* weak collision-resistant (CRHF)*, if for every (non-uniform) polynomial-size family of circuits $\{C_k\}_{k\in\mathbb{N}}$, there exists a negligible[4] function $\mu : \mathbb{N} \to \mathbb{R}$ such that for every k:*

$$\Pr_{\substack{h\leftarrow\mathcal{H}_k\\(x_1,x_2)\leftarrow C_k(h)}}[h(x_1) = h(x_2) \wedge x_1 \neq x_2] \leq \mu(k)$$

1.2 Our main result

Our main result is showing that any computationally sound public-coin IPP can be emulated by a computationally sound IPP in the pre-coordinated model, with related complexities. The emulation introduces an overhead to the resulting complexities which depends on the type of CRHF we assume to have, where the stronger the assumption the less overhead we incur.

Theorem 1.4 (computationally sound public-coin IPPs can be efficiently emulated by computationally sound IPPs in the pre-coordinated model)**.** *Let Π be a property that can be verified by a computationally sound public-coin IPP. Let q, c and r denote the query, communication and round complexities of this IPP, respectively.*

Let n denote the input size, and ϵ denote the proximity parameter. Let $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$ be a CRHF, and let $k(n) = (\log n)^{1+\frac{1}{\delta}}$ if $\mathcal{H}$ is a strong CRHF with parameter δ, and $k(n) = n^\alpha$ for any $\alpha > 0$ if $\mathcal{H}$ is a weak CRHF.

Then, Π has a computationally sound public-coin IPP in the pre-coordinated model with query complexity $O(\epsilon^{-1})$, communication complexity $O\big(c + \log|\mathcal{H}_{k(n)}| + (q + \epsilon^{-1}) \cdot k(n) \cdot \log n\big)$, and $O(r+q)$ rounds. If the original IPP uses non-adaptive queries, then the emulation can be done in $r + 2$ rounds. Furthermore, the resulting IPP has the same completeness error as the original IPP.

[3] A computational hardness assumption is needed, since computationally sound proof systems that do not satisfy standard soundness imply that the prover's optimal strategy is not in $P/poly$, which would prove $PSPACE \not\subseteq P/poly$.

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

Note that in particular the emulation preserves perfect completeness. We note that the result stated in Theorem 1.4 will hold also if we define the verifier such that its strategy is implementable by a (non-uniform) polynomial-size family of circuits, but not if we put no computational bounds on it. Additionally, the result holds also if we define the honest prover to be implementable by a (non-uniform) polynomial-size family of circuits.

Remark 1.5 (the effect of the emulation in Theorem 1.4 on the soundness error)**.** *The emulation in Theorem 1.4 can be modified to preserve the computational-soundness error up to a negligible term. If the original IPP has computational-soundness error s, then the modified resulting IPP will have computational-soundness error $s + \mu$, where μ is a negligible function that depends on the cheating prover. This modification affects only the resulting query and communication complexities: The resulting query complexity becomes $O(\epsilon^{-1} \cdot \log(s^{-1}))$, and the resulting communication complexity becomes $O\big(c + \log|\mathcal{H}_{k(n)}| + (q + \epsilon^{-1} \cdot \log(s^{-1})) \cdot k(n) \cdot \log n\big)$.*

We note that it is possible to do error reduction by parallel repetitions in public-coin computationally sound IPPs (see [1], which extends to the IPP setting). Remark 1.5 shows a way to reduce the computational-soundness error of the resulting system by first reducing the computational-soundness error of the original system, which can be done by parallel repetitions of the original system. Alternatively, a similar effect can be achieved by parallel repeating the resulting system itself, with only a small extra cost to the communication complexity relative to the first method.[5]

Theorem 1.4 implies a conditional separation between computationally sound public-coin $O(1)$-round IPPs in the pre-coordinated model and public-coin $O(1)$-round IPPs in the pre-coordinated model (see Corollary 2.5).

1.3 Our techniques and related works

Our construction of computationally sound IPPs in the pre-coordinated model is heavily inspired by Kilian's protocol [5]. Recall that Kilian showed how to transform PCPs into computationally sound (zero-knowledge) interactive proofs that have very low communication complexity, assuming the existence of collision-resistant hashing functions. Specifically, he used a commitment scheme (called a tree commitment scheme) that allows to commit to a string of length n such that individual bits of this string can be revealed (and verified as correct) with very small communication cost. Kilian used this scheme to have the prover commit to a PCP proof, forcing the prover to respond consistently with a fixed proof.

We use this commitment scheme to have the prover commit to the input. Then, instead of making the possibly proof-dependent queries to the input oracle, we request that the prover reveal the committed values at the locations the verifier would have queried. To ensure that the prover has committed to the actual input, we use the shared randomness between the interacting module and the querying module (postulated in the pre-coordinated model) to query both the actual input (provided by the oracle) and the alleged input (provided by the prover) at randomly chosen matching locations, verifying their consistency.

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 no source that presents this technique with a formal proof. In this work, we formalize the notion that the tree commitment scheme forces non-adaptive responses. This is captured by Lemma 2.3, which asserts that in any interaction that uses the tree commitment scheme, with overwhelmingly high probability over the randomness of the commitment phase, at the end of the commitment

[5] Performing parallel repetitions of the resulting system unnecessarily repeats the first round, in which the verifier sends a random hash function to the prover.

phase there is a fixed string such that, with overwhelmingly high probability over the locations chosen to be opened, every location that is opened will either be opened to a value consistent with that string or be detected as faulty. This lemma may be used whenever applying the aforementioned technique. Additionally, we provide a proof that the tree commitment scheme is computationally binding; that is, efficient devices cannot successfully produce a commitment together with valid openings to two different values for the same location, with more than negligible probability (see Lemma 2.2). To the best of our knowledge, also this property does not have a proof in the existing literature.

2 Proof of Theorem 1.4

We begin by presenting the tree commitment scheme that we will use in the emulation, then proceed to the emulation itself.

2.1 The tree commitment scheme

Definition 2.1 (tree commitment scheme (also known as Merkle tree [6]))**.** *Let $\{\mathcal{H}_k\}_{k\in\mathbb{N}}$ be a collision-resistant hash function ensemble. Let $n \in \mathbb{N}$. The tree commitment scheme is a protocol between two parties, called a sender and a receiver, which is composed of the following phases:*

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

  Let $k \in \mathbb{N}$ defined according to the type of $\mathcal{H}$: If $\mathcal{H}$ is a strong CRHF with parameter δ, then $k = (\log n)^{1+\frac{1}{\delta}}$. If $\mathcal{H}$ is a weak CRHF, then $k = n^{\alpha}$ for some $\alpha > 0$.

  - *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. Next, the sender 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 the block and sends the values of all siblings along the path from the root to the i^{th} leaf. This message 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 π, 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 π), ordered according to their position in the tree. The receiver verifies that the value he computed for the root matches the one received during the commit phase.

An opening π is a valid opening *for (h, tc, i, v) if the validation procedure accepts π 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: in the following, we will refer to the parameters k and m defined in the commitment scheme. These parameters are functions of n, the length of the committed string. When n is clear from the context, we will omit the explicit dependency 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 2.2 (tree commitments are computationally binding)**.** *Let $\mathcal{H}$ and k be as in Definition 2.1. Then, 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:*

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

Proof. Assume towards contradiction that there is 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 claim is greater than $\frac{1}{p(n)}$. We derive a contradiction by constructing a family of circuits $\{C'_k\}_{k\in\mathbb{N}}$ that break 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 $j \in \{1, 2\}$, it computes the values of all the nodes along the path from the root to the i^{th} leaf, using (v_j, π_j), in a similar manner to the validation procedure from Definition 2.1. Denote the computed values as $(u_1^1, ..., u_1^t)$ and $(u_2^1, ..., u_2^t)$ such that u_j^l is the value computed for the node in level l (starting from the leaves) when using (v_j, π_j), and $t = \log_2 m$ is the height of the tree. 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 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 $v_1 \neq v_2$ and $\forall j \in \{1, 2\}$ π_j is a valid opening 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 $\{C'_{k(n)}\}_{n\in\mathbb{N}}$ has size polynomial in n. Let q denote this polynomial, i.e., $|C'_{k(n)}| \leq q(n)$ for all n.

In case $\mathcal{H}$ is a strong CRHF with parameter $\delta > 0$, and $k(n) = (\log n)^{1+\frac{1}{\delta}}$, for all sufficiently large n's it holds that

$$q(n) < 2^{(\log n)^{1+\delta}} = 2^{k(n)^\delta}, \quad \frac{1}{p(n)} > 2^{-(\log n)^{1+\delta}} = 2^{-k(n)^\delta}$$

So we obtain a family of circuits $\{C'_k\}_{k\in\mathbb{N}}$ of size smaller than 2^{k^δ} that for infinitely many k's succeeds in finding collisions under $\mathcal{H}$ with probability greater than 2^{-k^δ}, reaching a contradiction.

Similarly, in case $\mathcal{H}$ is a weak CRHF and $k(n) = n^\alpha$ for some $\alpha > 0$, we get a contradiction to the weak collision resistant condition, as any polynomial in n is upper bounded by a polynomial in $k(n)$. ■

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. 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 will show that the sender is forced to respond consistently with a fixed string, regardless of which blocks are chosen. More precisely, we will show that there is a string which is determined at the end of the commitment stage, such that with overwhelmingly high probability over the hash function sampled in the commitment stage, with overwhelmingly high probability over the choice of blocks to be opened, for each of the opened blocks, if the sender provides a valid opening for this block, then the value it reveals for the block matches the corresponding block of this string.

Lemma 2.3. *Fix a tree commitment scheme. Let k and m be the parameters used in tree commitment scheme as specified in Definition 2.1. 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 his 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 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 $i \in B^{\text{val}}_{h,r}$.[6]

Then, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that for all n, and for every h, there exists a string $s_h \in \{0,1\}^n$ satisfying:

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

where $s_h(i)$ denote the i^{th} block in the partition of s_h into m consecutive blocks of length k.[7]

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:

$$s_h(i) = \arg\max_{v \in \{0,1\}^k}\left(\Pr_r[V_{h,r}(i) = v \mid B^{\text{val}}_{h,r} \ni i]\right).^8$$

Notice that $s_h(i) = \arg\max_{v\in\{0,1\}^k}\left(\Pr_r[B^{\text{val}}_{h,r} \ni i, V_{h,r}(i) = v]\right)$, because $\Pr_r[B^{\text{val}}_{h,r} \ni i, V_{h,r}(i) = v] = \Pr_r[V_{h,r}(i) = v \mid B^{\text{val}}_{h,r} \ni i] \cdot \Pr_r[B^{\text{val}}_{h,r} \ni i]$, and the second factor is positive and independent of v. Additionally, it holds that

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

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

$$\Pr_{h,r}\left[\exists i \in B^{\text{val}}_{h,r} \ s.t.\ V_{h,r}(i) \neq s_h(i)\right] \geq \frac{1}{p(n)} \tag{2}$$

[6] 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.

[7] More intuitively, Lemma 2.3 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.

[8] In case there is no unique value v that maximizes the expression $\Pr_r[V_{h,r}(i) \neq v \mid B^{\text{val}}_{h,r} \ni i]$, we arbitrarily take $s_h(i)$ to be the lexicographic first value that maximises the expression.

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. Next, C_n searches for a block that was opened validly in both simulations, such that the revealed values for this block differ between the two simulations. The 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 Equation 2. Using a reverse of the union bound, we get that there exists $i \in [m]$ such that

$$\Pr_{h,r}[B^{\text{val}}_{h,r} \ni i, V_{h,r}(i) \neq s_h(i)] \geq \frac{1}{m \cdot p(n)} > \frac{1}{n \cdot p(n)} \tag{3}$$

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.

$$\begin{aligned}
&\Pr_{h,r_1,r_2}[B^{\text{val}}_{h,r_1} \ni i, B^{\text{val}}_{h,r_2} \ni i, V_{h,r_1}(i) \neq V_{h,r_2}(i)] \\
&= \sum_{h'} \left(\Pr_h[h = h'] \cdot \sum_{r_2' : B^{\text{val}}_{h',r_2'} \ni i} \left(\Pr[r_2 = r_2'] \cdot \Pr_{r_1}[B^{\text{val}}_{h',r_1} \ni i, V_{h',r_1}(i) \neq V_{h',r_2'}(i)] \right) \right) \\
&\geq \sum_{h'} \left(\Pr_h[h = h'] \cdot \sum_{r_2' : B^{\text{val}}_{h',r_2'} \ni i} \left(\Pr[r_2 = r_2'] \cdot \Pr_{r_1}[B^{\text{val}}_{h',r_1} \ni i, V_{h',r_1}(i) \neq s_{h'}(i)] \right) \right) \\
&= \sum_{h'} \left(\Pr_h[h = h'] \cdot \Pr_{r_2}[B^{\text{val}}_{h',r_2} \ni i] \cdot \Pr_{r_1}[B^{\text{val}}_{h',r_1} \ni i, V_{h',r_1}(i) \neq s_{h'}(i)] \right)
\end{aligned} \tag{4}$$

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

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

Then for every $h' \in G$, both $\Pr_{r_2}[B^{\text{val}}_{h',r_2} \ni i]$ and $\Pr_{r_1}[B^{\text{val}}_{h',r_1} \ni i, V_{h',r_1}(i) \neq s_{h'}(i)]$ are greater than or equal to γ. From Equation 3 and Markov's inequality, it holds that $\Pr_h[h \in G] \geq \gamma$. Thus, Equation 4 is lower bound by

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

This non-negligible probability leads to the desired contradiction.

■

2.2 The emulation

The emulation will utilize the tree commitment scheme, using $\mathcal{H}$ as the CRHF. Let k and m be as in Definition 2.1.

Construction 2.4. *On input n, ϵ and oracle access to $x \in \{0,1\}^n$, the emulation proceeds as follows:*

- *The shared randomness R: A sequence of $O(\epsilon^{-1})$ uniformly and independently distributed elements in $[n]$.*
- *The querying module obtains the value of x at the indices in R.*
- *The interacting module proceeds in 4 stages.*

  1. *The interaction starts by establishing a tree commitment to an n-bit long string (which, for the honest prover, would be the input 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, with proximity parameter $\epsilon/2$. This is done with no access to the input oracle per the public-coin assumption.*

  Terminology: *Throughout the rest of the proof,* querying the prover with index *$j \in [n]$ means that the prover opens the block containing index j (i.e., it reveals the value of the block and sends the corresponding opening). The* prover's answer to the query *is the value within the revealed block corresponding to index j.*

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

- *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.*
  - 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, we consider the communication complexity. In the first round, the verifier sends the hash function which is of length $\log|\mathcal{H}_k|$. In Stage 2, the parties send the messages of the original IPP, which have c bits in total. In Stage 3 and 4, the prover sends openings for each of the q queries made by the original verifier, and each of the $O(\epsilon^{-1})$ random locations in R. Recall that the height of the tree is $\log(m)$, where $m = \frac{n}{k} < n$, and that an opening consists of a k-bit value for each level of the tree. Thus, the length of each opening is less than $k \cdot \log(n)$. Hence, the total communication is $O\big(c + \log|\mathcal{H}_{k(n)}| + (q + \epsilon^{-1}) \cdot k(n) \cdot \log n\big)$. As for the round complexity, in the general case we have 1 round in Stage 1, r rounds in Stage 2, and q rounds in Stages 3 and 4 combined. Therefore, the total number of rounds is $r+q+1$. In case the original IPP uses non-adaptive queries, then Stages 3 and 4 combined can be done in a single round, and therefore the total number of rounds in this case is $r+2$. The completeness of the system follows from the completeness of the original verifier. We thus turn to the computational soundness condition.

Claim 2.4.1 (computational soundness claim)**.** *Let P be a cheating prover that can be implemented by a polynomial-size family of circuits. Let $\{x_n\}_n$ be an infinite sequence of strings such that x_n is ϵ-far from Π_n. Then, there exist 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 2.4 accepts with probability at most $1/3 + \mu(n)$.*

Proof. We start with a high-level idea of the proof. Consider the interaction on an input x that is ϵ-far from Π. 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 Π, 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 2.4 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 R). Let $B^{\text{val}}_{h,r}$ denote the set of blocks $i \in [m]$ that the prover 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 prover's revealed value 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 2.3,[9] 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 will next analyze the probability of passing the Original Check and the Consistency Check of the deciding module. We'll first consider an idealized prover that is always consistent with x'_h.

Consider a prover P' that behaves as follows: 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' is asked to open, P' 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, nor do we impose any efficiency requirements on it. For any 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, this check passes. We will 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 $1/3$.

First, consider any n for which x'_h is $(\epsilon/2)$-far from x_n. The prover P' passes the Consistency Check on input x_n only if all the elements in R are locations where x'_h and x_n agree, therefore

$$\Pr\left[P' \text{ passes the Consistency Check on input } x_n\right] \leq (1-\frac{\epsilon}{2})^{O(\epsilon^{-1})} \leq \frac{1}{3} \tag{5}$$

It is left to deal with n's for which x'_h is $(\epsilon/2)$-close to x_n. Since x_n is ϵ-far from Π_n, if x'_h is $(\epsilon/2)$-close to x_n, then x'_h is $(\epsilon/2)$-far from Π_n. Therefore, from the computational soundness of

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

the original verifier, for all sufficiently large n's such that x'_h is $(\epsilon/2)$-close to x_n it holds that

$$\Pr[P' \text{ passes the Original Check on input } x_n] \leq \frac{1}{3} \tag{6}$$

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 how we defined 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 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 1.4. Claim 2.4.1 shows that the verifier in Construction 2.4 has computational-soundness error $1/3 + \mu$, where μ is a negligible function that depends on the cheating prover. To reduce this error to 1/3, we can parallel repeat the system $O(1)$ times. Error reduction by parallel repetitions is possible since the system is public-coin [1].

Proof of Remark 1.5. Suppose that the original verifier has computational-soundness error s, and we increase the number of random locations in R to $O(\epsilon^{-1} \cdot \log(s^{-1}))$. Then in Equations 6 and 5, the term $\frac{1}{3}$ will be replaced by $s(n)$, and the resulting computational-soundness error will reduce to $s + \mu$, where μ is a negligible function that depends on the cheating prover. The resulting query and communication complexities will increase accordingly, to account for the increase in the number of random locations for which we query the input oracle and the prover.

Corollary 2.5 (separation between computationally sound public-coin $O(1)$-round IPPs in the pre-coordinated model and public-coin $O(1)$-round IPPs in the pre-coordinated model)**.** *Assuming the existence of strong CRHF, there exists a property that has an efficient computationally sound public-coin $O(1)$-round IPP in the pre-coordinated model but has no efficient public-coin $O(1)$-round IPP in the pre-coordinated model, where efficient means poly-logarithmic query and communication complexity.*

Proof. Consider the property PERM, which consists of all permutations over $[n]$. An efficient 1-round public-coin IPP for PERM that uses non-adaptive queries and has a polynomial-time honest prover, was presented in [4, Sec. 4.1]. By applying Theorem 1.4, we conclude that PERM also has an efficient computationally sound public-coin $O(1)$-round IPP in the pre-coordinated model. On the other hand, [3, Sec. 4.4] showed that any $O(1)$-round public-coin IPP in the pre-coordinated model can be efficiently emulated by a standard tester. Since the query complexity of testing PERM is $\Omega(n^{1/2})$,[10] it follows that PERM does not have an efficient $O(1)$-round public-coin IPP in the pre-coordinated model. ■

[10] The lower bound follows as a special case of [4, Lem. 4.3], which should be restated as saying that any MAP for PERM with proof complexity p and query complexity q must satisfy $(p+1) \cdot q = \Omega(n^{1/2})$, where a tester may be viewed

Acknowledgments

I am grateful to Oded Goldreich for encouraging the initial idea that led to this work and expanding on it, as well as for his ongoing guidance throughout the research process and detailed feedback during the writing process.

References

[1] 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 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II.* 2015, pp. 229–246.

[2] Oded Goldreich. "Introduction to Property Testing". In: *Cambridge University Press.* 2017.

[3] Oded Goldreich, Guy N. Rothblum, and Tal Skverer. "On Interactive Proofs of Proximity with Proof-Oblivious Queries". In: *Electronic Colloquium on Computational Complexity (ECCC)* TR22-124 (2022).

[4] 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.

[5] 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.

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

as a MAP with $p = 0$. The original text, which implicitly assumes $p \geq 1$, states that $p \cdot q = \Omega(n^{1/2})$; however, the existing proof supports the foregoing claim. (The issue is that the proof of [4, Thm. 3] starts by reducing the error to $2^{-p}/c$, for some constant $c > 10$, by performing $O(p)$ repetitions, which should actually be $O(1) + O(p) = O(p + 1)$ repetitions.) We note that an alternative, simpler proof of the lower bound for the case of testers only is presented in Appendix A.

Appendices

A Lower bound on the complexity of testing PERM

The following Theorem A.1 follows as a special case from a stronger result that refers to MAPs [4, Lem. 4.3] (see footnote 10). The purpose of this appendix is to provide a direct proof for the special case of testers.

Theorem A.1. *Testing* PERM *requires* $\Omega(n^{1/2})$ *queries.*

Proof. Let n denote the domain size of the input and let ϵ denote the proximity parameter. Let $\alpha > 0$ and let $q(n) = (\alpha \cdot n)^{1/2}$. We will show that for sufficiently small ϵ and sufficiently small α, testing PERM requires more than $q(n)$ queries. We will use the technique presented in [2, Sec. 7.3]. We define two distributions of functions. In one distribution all functions are in PERM, and in the other almost all functions are ϵ-far from PERM. We then show that no algorithm that makes at most $q(n)$ queries can distinguish (with sufficiently high probability) between these distributions.

Let F_1 denote the uniform distribution over all permutations over $[n]$, and let F_0 denote the uniform distribution over all function from $[n]$ to $[n]$. As shown in [4, Claim 4.5], for all sufficiently small ϵ, $\Pr[F_0 \text{ is } \epsilon\text{-close to } \texttt{PERM}] \leq e^{-\frac{n}{10}}$. We turn to upper bound the probability of distinguishing between F_0 and F_1. Let M be a deterministic oracle machine that makes at most $q(n)$ queries. We will upper bound $\left|\Pr[M^{F_1}(n,\epsilon) = 1] - \Pr[M^{F_0}(n,\epsilon) = 1]\right|$. Assume without loss of generality that M does not make the same query more than once. Let $A_M(f)$ be the sequence of answers to the queries made by M given oracle access to f. Observe that the output of M is determined by $A_M(f)$, that is, there is a function $g : \{0,1\}^* \to \{0,1\}$ such that for every $f : [n] \to [n]$, $M^f(n,\epsilon) = g(A_M(f))$. Therefore, it suffices to bound the statistical distance between $D_1 = A_M(F_1)$ and $D_0 = A_M(F_0)$.[11] Let C be the set of all sequences $(a_1, \ldots, a_q) \in [n]^q$ for which there exist $i, j \in [q]$ such that $a_i = a_j$. Observe that D_0 is a random sequence in $[n]^q$ and D_1 is a random sequence in $[n]^q \setminus C$. To prove this, consider a sequence $a = (a_1, \ldots, a_q) \in [n]^q$, and let i_k be the k^{th} location queried by M when the answer to the first $k-1$ queries are $a_1, \ldots, a_{k-1}$. Then $D_0 = a$ when $F_0(i_k) = a_k$ for each $k \in [q]$. The probability of this event is the same for all $a \in [n]^q$, as we only need to count the options for the remaining $n-q$ locations. Similarly, the probability that $D_1 = a$ is the same for all $a \notin C$ (and equals 0 for $a \in C$). Next, observe that D_1 is identically distributed to the distribution D_0 conditioned on $D_0 \notin C$ (since for $a \in C$, $\Pr[D_0 = a \mid D_0 \notin C] = 0$ and for $a \notin C$, $\Pr[D_0 = a \mid D_0 \notin C] = \frac{\Pr[D_0 = a]}{\Pr[D_0 \notin C]}$ which is identical for all a). For any distribution X and set S, the statistical distance between X and "X conditioned on $X \notin S$" equals $\Pr[X \in S]$.[12] Hence, we get that $\Delta(D_0, D_1) = \Pr[D_0 \in C]$. Lastly, we have that

$$\Pr[D_0 \notin C] = \frac{n!}{n^q \cdot (n-q)!} > \frac{(n-q)^q}{n^q} = \left(1 - \frac{q}{n}\right)^q > 1 - \frac{q^2}{n} = 1 - \alpha$$

Therefore $\Delta(D_0, D_1) = \Pr[D_0 \in C] < \alpha$. By choosing a sufficiently large n and a sufficiently small α such that $\alpha + e^{-\frac{n}{10}} < 1/3$, we conclude the proof. ■

[11] The statistical distance between two distributions D_0 and D_1 is defined as $\Delta(D_0, D_1) = \max_T |\Pr[D_0 \in T] - \Pr[D_1 \in T]|$, or equivalently $\Delta(D_0, D_1) = \max_{g:\{0,1\}^* \to \{0,1\}} |\Pr[g(D_0) = 1] - \Pr[g(D_1) = 1]|$.

[12] Because in this case $\max_T |\Pr[D_0 \in T] - \Pr[D_1 \in T]|$ is obtained at $T = \bar{S}$.