Probabilistically Checkable Arguments

Yael Tauman Kalai[1] and Ran Raz[2]

[1] Microsoft Research
[2] Weizmann Institute

Abstract. We give a general reduction that converts any public-coin interactive proof into a one-round (two-message) argument. The reduction relies on a method proposed by Aiello *et al.* [1], of using a Private-Information-Retrieval (PIR) scheme to collapse rounds in interactive protocols. For example, the reduction implies that for any security parameter t, the membership in any language in PSPACE can be proved by a one-round (two-message) argument of size $\text{poly}(n,t)$, which is sound for malicious provers of size 2^t. (Note that the honest prover in this construction runs in exponential time, since she has to prove membership in PSPACE, but we can choose t such that 2^t is significantly larger than the running time of the honest prover).
A *probabilistically checkable argument* (PCA) is a relaxation of the notion of probabilistically checkable proof (PCP). It is defined analogously to PCP, except that the soundness property is required to hold only *computationally*. We consider the model where the argument is of one round (two-message), where the verifier's message depends only on his (private) randomness. We show that for membership in many NP languages, there are PCAs (with efficient honest provers) that are of size polynomial in the size of the *witness*. This compares to the best PCPs that are of size polynomial in the size of the *instance* (that may be significantly larger). The number of queries to these PCAs is poly-logarithmic.
The soundness property, in all our results, relies on exponential hardness assumptions for PIR schemes.

1 Introduction

In this paper, we define and study the new notion of *probabilistically checkable argument* (PCA), (see Subsection 1.1). We give a general reduction that uses a *poly-logarithmic* PIR scheme to convert any public-coin interactive proof into a one-round (two-message) argument, (see Subsection 1.2). For example, the reduction shows that membership in any language in PSPACE can be proved by a one-round (two-message) argument of polynomial size, (see Subsection 1.3). Similarly, we give a general reduction that converts any efficient *interactive*-PCP, with certain properties, into a short PCA, (see Subsections 1.4, 1.5). Combined with a recent efficient construction of interactive-PCPs [14], the reduction gives, for membership in many NP languages, PCAs that are significantly shorter than the known PCPs for these languages, (see Subsection 1.6).

1.1 Probabilistically Checkable Arguments

The PCP theorem states that the satisfiability of a formula $\Phi(z_1, \ldots, z_k)$ of size n can be proved by a proof of size poly(n) that can be verified by reading only a constant number of its bits [5, 10, 3, 2]. Note, however, that in many cases the size of the witness, k, is significantly smaller than the size of the instance, n. A central line of research in the area of PCPs is devoted to constructing short PCPs. In particular, one could hope that the satisfiability of a formula $\Phi(z_1, \ldots, z_k)$ of size n could be proved by PCPs of size poly(k), rather than poly(n), (see for example [15]). However, a very interesting recent result of Fortnow and Santhanam shows that this is very unlikely, as it implies that $NP \subseteq coNP/poly$ [12].

In this paper, we consider the relaxed setting of *probabilistically checkable argument* (PCA). We show that for many NP languages, there are PCAs that are of size polynomial in the size of the witness, rather than polynomial in the size of the instance. The number of queries to these PCAs is poly-logarithmic in n.

Roughly speaking, a PCA is a relaxation of the notion of PCP, where the soundness property is required to hold only *computationally*. We consider the model where the argument is of one round (two-message), where the verifier's message depends only on his (private) randomness[3]. Before the protocol starts, the verifier generates two strings that we refer to below as a "secret key" and a "public key", and sends the public key in the first message. The prover's message depends on the verifier's public key. The verifier in turn will need to use his secret key for verification.[4]

More precisely, a PCA system is associated with three algorithms: a *key generation algorithm* $\mathcal{G}$, a *proof generation algorithm* $\mathcal{P}$, and a *verification algorithm* $\mathcal{V}$. It is also associated with five parameters t, p, q, c, s, where t, p, q are integers and c, s are reals, s.t. $0 \leq s < c \leq 1$. (Informally, t is the *security parameter*, p is the *size* of the PCA, q is the *number of queries* allowed to the PCA, c is the *completeness* parameter and s is the *soundness* parameter). We think of the parameters t, p, q, c, s as functions of the instance size n.

Let L be an NP language, defined by $L = \{x : \exists w \quad s.t. \quad (x, w) \in R_L\}$. Suppose that Alice wishes to prove to Bob that $x \in L$. Assume that Bob applied in the past the key generation algorithm $\mathcal{G}$, and thus is associated with a pair of secret and public keys $(SK, PK) \leftarrow \mathcal{G}(1^t)$. Assume that Bob sent to Alice the public key PK. We assume that both Alice and Bob know L and that they both get as input an instance x of size n. Alice gets an additional input w (supposedly a witness for the membership of $x \in L$). A PCA system allows

[3] We note that one could consider several other models for PCA, such as, PCA in the common random string model, where there is a public random string that both the prover and the verifier can see, and the argument is composed of only one message sent by the prover to the verifier.

[4] We note that each pair of secret and public keys can only be used once. Our soundness' proof are only valid in this case. Moreover, it was noted to us by Rafael Pass that by observing whether a verifier accepted or rejected a contrived message (sent by the prover), the prover can gain information about the secret key of the verifier.

Alice to generate a string $\pi \leftarrow \mathcal{P}(x, w, PK)$ of p bits. Bob is allowed to access at most q bits of the string π, and based on these bits he decides whether to accept or reject the statement $x \in L$. We require the following completeness and soundness properties:

1. **Completeness:** For any $x \in L$ and any witness w (given to the prover as input) such that $(x, w) \in R_L$, the verifier, associated with a pair of secret and public keys $(SK, PK) \leftarrow \mathcal{G}(1^t)$, accepts $\pi \leftarrow \mathcal{P}(x, w, PK)$ with probability at least c. Namely,
$$\Pr[\mathcal{V}^{\pi}(x, SK, PK) = 1] \geq c$$
where the probability is over $(SK, PK) \leftarrow \mathcal{G}(1^t)$, over $\pi \leftarrow \mathcal{P}(x, w, PK)$, and over the randomness of $\mathcal{V}$.
2. **Soundness:** For any $x \notin L$ and any cheating prover $\tilde{\mathcal{P}}$ of size $\leq 2^t$,
$$\Pr[\mathcal{V}^{\tilde{\pi}}(x, SK, PK) = 1] \leq s$$
where $\tilde{\pi} = \tilde{\mathcal{P}}(PK)$, and the probability is over $(SK, PK) \leftarrow \mathcal{G}(1^t)$ and over the randomness of $\mathcal{V}$.

For the formal definition of PCA, see Section 2.

1.2 From Interactive Proofs to One-Round Arguments

We propose a general method for reducing the number of rounds in any public-coin interactive proof (that is, an interactive proof where all the bits sent by the verifier are random, and consist of the verifier's random coin tosses). More specifically, our method uses a PIR scheme to convert any public-coin interactive proof into a one-round (two-message) argument. The idea of using a PIR scheme to reduce the round complexity in interactive protocols was proposed by Aiello *et al.* [1], who used a PIR scheme to convert the (short) 4-message argument for NP proposed by [19, 21], into a (short) 2-message protocol. Although their 2-message protocol is very natural, attempts to prove its soundness have failed. Moreover, Dwork *et al.* [9] exhibit inherent difficulties in such attempts (for the protocol of [1] and for extensions of this protocol). Dwork *et al.* note that the essence of the problem is that seemingly independant executions of PIR schemes may have, so called, *spooky interactions*. We prove that our method, which is based on the ideas of Aiello *et al.* [1], is sound, if the initial interactive protocol involves only one prover, and is a *proof*.

A *Private Information Retrieval* (PIR) scheme, a concept introduced by Chor, Goldreich, Kushilevitz, and Sudan [7] and by Kushilevitz and Ostrovsky [16][5], allows a user to retrieve information from a database in a private manner. More formally, the database is modeled as an N bit string $x = (x_1, \ldots, x_N)$,

[5] The original PIR scheme of [7] had information theoretic privacy but required several copies of the database that cannot interact with each other. The first PIR scheme with a single database (with privacy under computational assumptions) was obtained in [16].

out of which the user retrieves the i'th bit x_i, without revealing any information about the index i. A trivial PIR scheme consists of sending the entire database to the user, thus satisfying the PIR privacy requirement in the information-theoretic sense. A PIR scheme with communication complexity smaller than N is said to be *non-trivial*. In this paper, we are interested in *poly-logarithmic* PIR schemes, formally defined by Cachin *et al.* [6]. Roughly speaking, a poly-logarithmic PIR scheme is a PIR scheme with poly-logarithmic communication complexity. For the formal definition of poly-logarithmic PIR scheme, see Subsection 3.2.

Roughly speaking, we are able to prove the following result. Assume the existence of a poly-logarithmic PIR scheme (as defined in [6]). Assume that there exists a public-coin interactive proof system $(\mathcal{P}, \mathcal{V})$ for proving membership in some language L; with communication complexity ℓ, completeness c, and soundness s. Then for any security parameter $t \geq \max\{\ell, \log n\}$, there exists a one-round (two-message) argument system $(\mathcal{P}', \mathcal{V}')$ for L, with communication complexity $\text{poly}(t)$, completeness $c - 2^{-t^2}$, and soundness $s + 2^{-t^2}$ against malicious provers of size $\leq 2^t$. The verifier $\mathcal{V}'$ runs in time $\text{poly}(t, n)$ (assuming that $\mathcal{V}$ runs in time $\text{poly}(n)$). The prover $\mathcal{P}'$ runs in time $\text{poly}(T, t, 2^\lambda)$, where T is the running time of $\mathcal{P}$, and λ is the total number of bits sent from $\mathcal{V}$ to $\mathcal{P}$ in the interactive proof system $(\mathcal{P}, \mathcal{V})$.

Moreover, the resulting one-round argument system $(\mathcal{P}', \mathcal{V}')$ has the property that the first message, sent by $\mathcal{V}'$, depends only on the random coin tosses of $\mathcal{V}'$ (and is independent of the instance x), and can be computed in time $\text{poly}(t)$.

The main idea of the proof is as follows. For every round i of the protocol $(\mathcal{P}, \mathcal{V})$, the prover $\mathcal{P}'$ prepares a database DB_i (of size at most 2^λ) of the response of $\mathcal{P}$ (in round i) on all the possibilities of bits sent by $\mathcal{V}$ (in all rounds). The verifier $\mathcal{V}'$ retrieves the response of $\mathcal{P}$ from this database, using a PIR scheme. This is done simoulataneously for all rounds of the protocol. Intuitively, the use of a PIR scheme ensures that when the prover $\mathcal{P}'$ prepares the database DB_i, she cannot use information about the bits sent by the verifier in later rounds. As mentioned above, Dwork *et al.* show that this intuition is misleading in many cases [9]. Nevertheless, we are able to prove that our protocol is sound. Thus, in this particular case, spooky interactions are not a problem.

Note that the running time of the honest prover $\mathcal{P}'$ is exponential in λ, where λ is the total number of bits sent from $\mathcal{V}$ to $\mathcal{P}$ in the interactive proof system $(\mathcal{P}, \mathcal{V})$. This is the case because $\mathcal{P}'$ has to handle the databases DB_i of size 2^λ. We are able to improve a little bit over that and to prove a similar result, where the prover's running time is $\text{poly}(T, t, 2^\lambda)$, where this time λ is the maximum number of bits of the verifier's messages that the prover "needs" to read in order to compute one bit to be sent to the verifier.

For more details, see Section 4.

The problem of reducing the number of rounds in interactive protocols was previously studied in a large number of works. The most famous heuristic for reducing the number of rounds in general protocols is the Fiat-Shamir method [11]. It was shown in [4, 13] that in certain cases where the original protocol is an *ar-*

gument the obtained protocol is not sound. It is still not known whether or not the obtained protocol is sound if the original protocol is a *proof*.

A different approach, more related to ours, to reduce the number of rounds in interactive proofs was previously done by Damgard, Fazio and Nicolosi [8]. Damgard, Fazio and Nicolosi used a homomorphic encryption, rather than PIR, and their results only apply for 3 messages protocols that have the additional property that the third message is linear in the second message. The proof of soundness of their protocols is obtained by a similar approach to the one taken here.

1.3 One-Round Arguments for PSPACE

The reduction from Subsection 1.2 can be used to convert any public-coin interactive proof into a one-round argument. In particular, it can be used for giving one-round arguments for membership in PSPACE languages.

For any language L in PSPACE, there exists a public-coin interactive proof system $(\mathcal{P}, \mathcal{V})$ for proving membership in L; with communication complexity $\ell = \text{poly}(n)$, completeness 1, and exponentially small soundness [20, 22]. Using our general reduction, we can translate this interactive proof into a one-round argument as follows: For any security parameter $t \geq \max\{\ell, n\}$, there exists a one-round (two-message) argument system $(\mathcal{P}', \mathcal{V}')$ for L, with communication complexity $\text{poly}(t)$, completeness $1 - 2^{-t}$, and soundness 2^{-t} against malicious provers of size $\leq 2^t$. The verifier $\mathcal{V}'$ runs in time $\text{poly}(t)$. The prover $\mathcal{P}'$ runs in time $\text{poly}(t, 2^\ell)$.

Note that the running time of the honest prover $\mathcal{P}'$ is exponential. This seems necessary because the prover has to prove membership in PSPACE languages. Note, however, that we can choose t to be significantly larger than ℓ, say, $t = \ell^3$. In this case, the honest prover runs in time $\text{poly}(2^\ell)$, while the proof is sound for malicious provers of size $\leq 2^{\ell^3}$.

1.4 Interactive-PCP

An interactive-PCP (say, for the membership $x \in L$) is a combination of a PCP and a short interactive proof. Roughly speaking, an interactive-PCP is a proof that can be verified by reading only a small number of its bits, with the help of a short interactive proof.

More precisely, let L be an NP language, defined by $L = \{x : \exists w \ \ s.t. \ \ (x, w) \in R_L\}$. Let p, q, l, c, s be parameters as follows: p, q, l are integers and c, s are reals, s.t. $0 \leq s < c \leq 1$. (Informally, p is the *size of the PCP*, q is the *number of queries* allowed to the PCP, l is the *communication complexity* of the interactive proof, c is the *completeness* parameter and s is the *soundness* parameter). We think of the parameters p, q, l, c, s as functions of the instance size n. An interactive-PCP with parameters (p, q, l, c, s) for membership in L is an interactive protocol between an (efficient) prover P and an (efficient) verifier V, as follows:

We assume that both the prover and the verifier know L and get as input an instance x of size n, and the prover gets an additional input w (supposed to be a

witness for the membership $x \in L$). In the first round of the protocol, the prover generates a string π of p bits. (We think of π as an encoding of the witness w). The verifier is still not allowed to access π. The prover and the verifier then apply an interactive protocol, where the total number of bits communicated is l. During the protocol, the verifier is allowed to access at most q bits of the string π. After the interaction, the verifier decides whether to accept or reject the statement $x \in L$. We require the following completeness and soundness properties:

There exists an (efficient) verifier V such that:

1. Completeness: There exists an (efficient) prover P, such that: for every $x \in L$ and any witness w (given to the prover P as an input), if $(x, w) \in R_L$ then the verifier accepts with probability at least c.
2. Soundness: For any $x \notin L$ and any (not necessarily efficient) prover $\tilde{P}$, and any w (given to the prover $\tilde{P}$ as an input), the verifier accepts with probability at most s.

For the formal definition of interactive-PCP, see Subsection 3.1.

1.5 From Interactive-PCP to PCA

We give a general reduction that converts any efficient interactive-PCP, with certain properties, into a short PCA. The main idea is to use the reduction from Subsection 1.2 to convert the interactive phase of the interactive-PCP into a one-round argument.

Roughly speaking, we are able to prove the following result. Assume the existence of a poly-logarithmic PIR scheme (as defined in [6]). Assume that there exists an interactive-PCP system $(\mathcal{P}, \mathcal{V})$ with parameters p, q, ℓ, c, s for some NP language L, such that the interactive phase of $(\mathcal{P}, \mathcal{V})$ is public-coin, and each bit sent by the prover in the interactive phase depends on at most λ bits sent by the verifier. Then, for any security parameter $t \geq \max\{\ell, \log n\}$, there exists a PCA system $(\mathcal{G}', \mathcal{P}', \mathcal{V}')$ with parameters t, p', q', c', s' for the language L, where $p' = \text{poly}(p, t)$, $q' = \text{poly}(q, t)$, $c' \geq c - 2^{-t^2}$, and $s' \leq s + 2^{-t^2}$. The prover $\mathcal{P}'$ runs in time $\text{poly}(t, n, 2^{\lambda})$.

For more details, see Section 5.

1.6 Short PCAs for Satisfiability

Efficient constructions of interactive-PCPs were given in [18, 14]. In particular, in [14], the following theorem was proven.

Let $\Phi(z_1, \ldots, z_k)$ be a Boolean circuit of size n and depth d, and assume without loss of generality that $k \geq \log n$ (otherwise, it is easy to check the satisfiability of Φ in time $\text{poly}(n)$). Let s be such that, $\log n \leq s \leq \text{poly}(n)$. Then, the satisfiability of Φ can be proved by an interactive-PCP with the following parameters. Size of the PCP: $p = \text{poly}(k, d)$. Number of queries: $q = \text{poly}(s)$. Communication complexity of the interactive phase: $\ell = \text{poly}(d, s)$. Completeness: 1. Soundness: 2^{-s}. Moreover, the interactive phase is public-coin, and each

message sent by the prover depends only on the preceding $\lambda = O(\log n)$ bits sent by the verifier.

Using our reduction from interactive-PCP to PCA, together with the theorem of [14], one obtains the following result. (Our reduction and the theorem of [14] were obtained roughly at the same time. The result below follows from their combination).

Let $\Phi(z_1, \ldots, z_k)$ be a Boolean circuit of size n and depth d. Let t be a security parameter, such that, $\log n \leq t \leq \text{poly}(n)$. Then, the satisfiability of Φ can be proved by an efficient PCA system (i.e., PCA with an efficient prover), with the following parameters. Size of the PCA: $p = \text{poly}(k, d, t)$. Number of queries: $q = \text{poly}(d, t)$. Completeness: $1 - 2^{-t}$. Soundness: 2^{-t}.

In particular, if $\Phi(z_1, \ldots, z_k)$ is a Boolean formula of size n, and $\log n \leq t \leq \text{poly}(n)$, the satisfiability of Φ can be proved by an efficient PCA system, with the following parameters. Size of the PCA: $p = \text{poly}(k, t)$. Number of queries: $q = \text{poly}(t)$. Completeness: $1 - 2^{-t}$. Soundness: 2^{-t}.

2 Definition of PCA

Let L be any NP language defined by $L = \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}_L\}$. Let t, p, q, c, s be parameters that satisfy the following: The parameters $t, p, q : \mathbb{N} \to \mathbb{N}$ are integers, and the parameters $c, s : \mathbb{N} \to [0, 1]$ are reals, such that for every $n \in \mathbb{N}$, $0 \leq s(n) < c(n) \leq 1$.

Definition 1. *A triplet $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ of probabilistic Turing machines is a* **PCA system** *for L with parameters (t, p, q, c, s), if the following holds:*

- *$\mathcal{G}$ is a probabilistic Turing machine that runs in time $\text{poly}(t)$, and $\mathcal{V}$ is a probabilistic oracle machine that runs in time $\text{poly}(t, n)$.*
- *For every $(x, w) \in \mathcal{R}_L$ (where $|x| = n$) and every $(SK, PK) \leftarrow \mathcal{G}(1^{t(n)})$, the algorithm $\mathcal{P}(x, w, PK)$ generates a bit string π of size at most $p(n)$, and the oracle machine $\mathcal{V}^{\pi}(x, SK, PK)$ reads at most $q(n)$ bits of π.*
- **Completeness:** *For every $(x, w) \in \mathcal{R}_L$ (where $|x| = n$),*

$$\Pr[\mathcal{V}^{\pi}(x, SK, PK) = 1] \geq c(n)$$

  (where the probability is over $(SK, PK) \leftarrow \mathcal{G}(1^{t(n)})$, over $\pi \leftarrow \mathcal{P}(x, w, PK)$, and over the randomness of $\mathcal{V}$).
- **Soundness:** *For every $x \notin L$ (where $|x| = n$), and every cheating prover $\tilde{\mathcal{P}}$ of size $\leq 2^{t(n)}$,*

$$\Pr[\mathcal{V}^{\tilde{\pi}}(x, SK, PK) = 1] \leq s(n)$$

  (where $\tilde{\pi} = \tilde{\mathcal{P}}(PK)$, and the probability is over $(SK, PK) \leftarrow \mathcal{G}(1^{t(n)})$ and over the randomness of $\mathcal{V}$).

Remark. Note that in Definition 1 we did not specify the complexity of $\mathcal{P}$. We say that a PCA system $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is *efficient* if $\mathcal{P}$ runs in time $\text{poly}(t, n)$.

3 Preliminaries

3.1 Interactive-PCP (IPCP)

Let L be any NP language defined by $L = \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}_L\}$. Let p, q, ℓ, c, s be parameters that satisfy the following: The parameters $p, q, \ell : \mathbb{N} \to \mathbb{N}$ are integers, and the parameters $c, s : \mathbb{N} \to [0, 1]$ are reals, such that for every $n \in \mathbb{N}$, $0 \leq s(n) < c(n) \leq 1$.

Definition 2. *A pair $(\mathcal{P}, \mathcal{V})$ of probabilistic polynomial time interactive Turing machines is an* interactive-PCP *for L with parameters (p, q, ℓ, c, s), if for every $(x, w) \in \mathcal{R}_L$ the prover $\mathcal{P}(x, w)$ generates a bit string π of size at most $p(n)$ (where $n = |x|$), such that the following properties are satisfied.*

- **Completeness:** *For every $(x, w) \in \mathcal{R}_L$,*

$$\Pr[(\mathcal{P}(x, w), \mathcal{V}^{\pi}(x)) = 1] \geq c(n)$$

  (where $n = |x|$, and the probability is over the random coin tosses of $\mathcal{P}$ and $\mathcal{V}$).
- **Soundness:** *For every $x \notin L$, every (unbounded) interactive Turing machine $\tilde{\mathcal{P}}$, and every string $\tilde{\pi} \in \{0, 1\}^*$,*

$$\Pr[(\tilde{\mathcal{P}}(x), \mathcal{V}^{\tilde{\pi}}(x)) = 1] \leq s(n)$$

  (where $n = |x|$, and the probability is over the random coin tosses of $\mathcal{V}$).
- **Complexity:** *The communication complexity of the protocol $(\mathcal{P}(x, w), \mathcal{V}^{\pi}(x))$ is at most $\ell(n)$, and $\mathcal{V}$ reads at most $q(n)$ bits of π.*

3.2 Private Information Retrieval (PIR)

A PIR scheme consists of three algorithms: Q^{PIR}, D^{PIR} and R^{PIR}. The query algorithm Q^{PIR} takes as input a security parameter t, the database size N, and an index $i \in [N]$ (that the user wishes to retrieve from the database). It outputs a query q, which should reveal no information about the index i, together with an additional output s, which is kept secret by the user and will later assist the user in retrieving the desired element from the database. The database algorithm D^{PIR} takes as input a security parameter t, the database $(x_1, \ldots, x_N)$ and a query q, and outputs an answer a. This answer enables the user to retrieve x_i, by applying the retrieval algorithm R^{PIR}, which takes as input a security parameter t, the database size N, an index $i \in [N]$, a corresponding pair (q, s) obtained from the query algorithm, and the database answer a corresponding to the query q. It outputs a value which is supposed to be the i'th value of the database.

In this paper we are interested in *poly-logarithmic* PIR schemes, formally defined by Cachin *et al.* [6], as follows.[6]

[6] Definition 3 is not worded exactly as the one in [6], but was shown to be equivalent to it in [17].

Definition 3. *Let t be the security parameter and N be the database size. Let Q^{PIR} and D^{PIR} be probabilistic circuits, and let R^{PIR} be a deterministic circuit. We say that $(Q^{PIR}, D^{PIR}, R^{PIR})$ is a* poly-logarithmic private information retrieval scheme *if the following conditions are satisfied:*

1. (Size Restriction:) *Q^{PIR} and R^{PIR} are of size $\leq \text{poly}(t, \log N)$, and D^{PIR} is of size $\leq \text{poly}(t, N)$. The output of Q^{PIR} and D^{PIR} is of size $\leq \text{poly}(t, \log N)$.*
2. (Correctness:) *$\forall N$, $\forall t$, $\forall$database $x = (x_1, \ldots, x_N) \in \{0,1\}^N$, and $\forall i \in [N]$,*

$$\Pr[R^{PIR}(t, N, i, (q, s), a) = x_i \mid (q, s) \leftarrow Q^{PIR}(t, N, i), a \leftarrow D^{PIR}(t, x, q)]$$

$$\geq 1 - 2^{-t^3}.$$

3. (User Privacy:) *$\forall N$, $\forall t$, $\forall i, j \in [N]$, and $\forall$adversary $\mathcal{A}$ of size at most 2^{t^3},*

$$\big|\Pr[\mathcal{A}(t, N, q) = 1 \mid (q, s) \leftarrow Q^{PIR}(t, N, i)] -$$
$$\Pr[\mathcal{A}(t, N, q) = 1 \mid (q, s) \leftarrow Q^{PIR}(t, N, j)]\big| \leq 2^{-t^3}.$$

4 From Interactive Proofs to One-Round Arguments

In this section, we propose a general method for reducing the number of rounds in any public-coin interactive proof (that is, an interactive proof where all the bits sent by the verifier are random, and consist of the verifier's random coin tosses). More specifically, our method uses a PIR scheme to convert any public-coin interactive proof into a one-round (two-message) argument.

Lemma 1. *Assume the existence of a (uniform) poly-logarithmic PIR scheme (as defined in Definition 3).*[7] *Assume that there exists a public-coin interactive proof system $(\mathcal{P}, \mathcal{V})$ for proving membership in some language L, with communication complexity ℓ, completeness c, and soundness s. Then for any security parameter $t \geq \max\{\ell, \log n\}$, there exists a one-round (two-message) argument system $(\mathcal{P}', \mathcal{V}')$ for L, with communication complexity $\ell' = \text{poly}(t, \ell) = \text{poly}(t)$, completeness $c' \geq c - 2^{-t^2}$, and soundness $s' \leq s + 2^{-t^2}$ against provers of size $\leq 2^t$. The verifier $\mathcal{V}'$ runs in time $\leq \text{poly}(t, n)$ (assuming that $\mathcal{V}$ runs in time $\text{poly}(n)$). The prover $\mathcal{P}'$ runs in time $\leq \text{poly}(T, t, 2^\lambda)$, where T is the running time of $\mathcal{P}$, and λ is the total number of bits sent from $\mathcal{V}$ to $\mathcal{P}$ in the interactive proof system $(\mathcal{P}, \mathcal{V})$.*

Moreover, the resulting one-round argument system $(\mathcal{P}', \mathcal{V}')$ has the property that the first message, sent by $\mathcal{V}'$, depends only on the random coin tosses of $\mathcal{V}'$ (and is independent of the instance x),[8] *and can be computed in time $\leq \text{poly}(t)$.*

[7] We assume the existence of such a PIR scheme for any parameters N and t.

[8] The fact that this message depends *only* on the random coin tosses of $\mathcal{V}'$ (and is independent of x) will be crucial when converting an interactive-PCP system into a PCA system in Section 5.

Rather than proving Lemma 1 directly, we prove a stronger and more general lemma. The more general lemma converts any interactive proof $(\mathcal{P}, \mathcal{V})$, in which the verifier's messages depend only on the verifier's random coin tosses (and in particular, any public-coin interactive proof), into a one-round argument $(\mathcal{P}', \mathcal{V}')$. Moreover, the resulting prover $\mathcal{P}'$ is more efficient than the one in the statement of Lemma 1. More specifically, the running time of $\mathcal{P}'$ is $\leq \text{poly}(T, t, 2^{\lambda})$, where T, t are as in the statement of Lemma 1 (i.e., T is the running time of $\mathcal{P}$, and t is the security parameter of the underlying PIR scheme), but λ here is defined differently than in Lemma 1. Recall that in Lemma 1, λ was the *total* number of bits sent by the verifier. Intuitively, here λ is the maximum number of bits of the verifier's messages, that the prover "needs" to read in order to compute one bit to be sent to the verifier. Namely, if λ_i denotes the number of bits of the verifier's messages, that the prover "needs" to read in order to compute its i'th bit (to be sent to the verifier), then $\lambda = \max\{\lambda_1, \ldots, \lambda_\ell\}$. We formalize this via the following definition of a *history-ignorant* interactive proof.

Definition 4. *An ℓ-round interactive proof $(\mathcal{P}, \mathcal{V})$ for proving membership in a language L is said to be* history-ignorant*, if for every input $x \in L$, every auxiliary input $w \in \{0,1\}^*$,[9] and for every $i \in [\ell]$, the message sent by the (honest) prover $\mathcal{P}(x, w)$ in the i'th round of the protocol $(\mathcal{P}(x, w), \mathcal{V}(x))$ depends only on the message sent by $\mathcal{V}$ in the i'th round of the protocol (and on x, w and the random coin tosses of $\mathcal{P}$),[10] and does not depend on the messages sent by $\mathcal{V}$ before the i'th round.*

In the generalized lemma we propose a method for converting any ℓ-round *history-ignorant* interactive proof $(\mathcal{P}, \mathcal{V})$ (where the verifier's messages depend only on the verifier's random coin tosses) into a one-round argument $(\mathcal{P}', \mathcal{V}')$. As in Lemma 1, we show that the completeness and soundness parameters remain almost the same, and the communication complexity increases by at most a polynomial factor in the security parameter t. However, here the running time of the resulting prover $\mathcal{P}'$ is $\leq \text{poly}(T, t, 2^{\lambda})$, where now λ is the length of the longest message sent by $\mathcal{V}$ (rather than the total number of bits sent by $\mathcal{V}$). Namely, if $m_1, \ldots, m_\ell$ are the ℓ messages sent by $\mathcal{V}$ throughout the protocol $(\mathcal{P}, \mathcal{V})$ (where the message m_i is sent by $\mathcal{V}$ in round i), then

$$\lambda \stackrel{\text{def}}{=} \max_{i \in [\ell]} \{|m_i|\}.$$

Lemma 2. *Assume the existence of a (uniform) poly-logarithmic PIR scheme (as defined in Definition 3). Assume that there exists a history-ignorant interactive proof system $(\mathcal{P}, \mathcal{V})$ for proving membership in some language L, where the verifier's messages depend only on the verifier's random coin tosses (and are independent of the interaction and the input). Let ℓ be the communication*

[9] As is common, we allow the prover in the interactive proof system to use an auxiliary input, supposedly a witness for $x \in L$.

[10] We think of each round as consisting of a message sent by the verifier $\mathcal{V}$ followed by a message sent by the prover $\mathcal{P}$.

complexity, c be the completeness parameter, and s be the soundness parameter of the proof system $(\mathcal{P}, \mathcal{V})$. Denote by λ the length of the longest message sent by $\mathcal{V}$, and assume that $\mathcal{V}$ uses at most $O(\ell \cdot \lambda)$ random bits. Then for any security parameter $t \geq \max\{\ell, \log n\}$, there exists a one-round (two-message) argument system $(\mathcal{P}', \mathcal{V}')$ for L, with communication complexity $\ell' = \text{poly}(t, \ell) = \text{poly}(t)$, completeness $c' \geq c - 2^{-t^2}$, and soundness $s' \leq s + 2^{-t^2}$ against provers of size $\leq 2^t$. The verifier $\mathcal{V}'$ runs in time $\leq \text{poly}(t, n)$ (assuming that $\mathcal{V}$ runs in time $\text{poly}(n)$). The prover $\mathcal{P}'$ runs in time $\leq \text{poly}(T, t, 2^\lambda)$, where T is the running time of $\mathcal{P}$.

Moreover, the resulting one-round argument system $(\mathcal{P}', \mathcal{V}')$ has the property that the first message, sent by $\mathcal{V}'$, depends only on the random coin tosses of $\mathcal{V}'$ (and is independent of the instance x). If each message sent by $\mathcal{V}$ can be computed in time $\leq \text{poly}(t)$ then the first message sent by $\mathcal{V}'$ can also be computed in time $\leq \text{poly}(t)$.

Before proving Lemma 2, we show that this lemma can be used to convert any (not necessarily history-ignorant) interactive proof $(\mathcal{P}, \mathcal{V})$, where the verifier's messages depend only on the verifier's random coin tosses, into a one-round argument $(\mathcal{P}', \mathcal{V}')$, where the running time of the resulting prover $\mathcal{P}'$ is $\leq \text{poly}(T, t, 2^\lambda)$, and where λ is the maximum number of bits of the verifier's messages that the underlying prover $\mathcal{P}$ "needs" to read in order to compute one bit to be sent to the verifier. We assume for simplicity that in the protocol $(\mathcal{P}, \mathcal{V})$, the prover $\mathcal{P}$ sends in each round a *single* bit. This is without loss of generality since we can always increase the number of rounds artificially.

The idea is the following:[11] First convert the interactive proof $(\mathcal{P}, \mathcal{V})$ into the following *history-ignorant* interactive proof $(\mathcal{P}'', \mathcal{V}'')$: The verifier $\mathcal{V}''$ will first prepare all the messages to be sent by $\mathcal{V}$ in the protocol $(\mathcal{P}, \mathcal{V})$. Note that this can be done in advance since according to our assumption, the verifier's messages depend only on the verifier's random coin tosses (and are independent of the interaction and the input). Then, in each round i, the verifier $\mathcal{V}''$ will send $\mathcal{P}''$ all the bits that $\mathcal{P}$ "needs" in order to compute its i'th bit (i.e., its i'th round message) in the protocol $(\mathcal{P}, \mathcal{V})$. The prover $\mathcal{P}''$ will emulate the prover $\mathcal{P}$, while reading only the message sent by $\mathcal{V}''$ in the i'th round. This results with a history-ignorant protocol, with the same completeness and soundness parameters, and where the communication complexity increases by at most a polynomial factor. The thing to notice is that in the i'th round the verifier $\mathcal{V}''$ sends a message of size λ_i (where λ_i is the number of bits of the verifier's messages, that the prover $\mathcal{P}$ "needs" to read in order to compute its i'th bit). What remains is to apply Lemma 2 to the history-ignorant protocol $(\mathcal{P}'', \mathcal{V}'')$ in order to obtain the desired one-round argument $(\mathcal{P}', \mathcal{V}')$.

Proof of Lemma 2. (In this version, due to space limitation, we only describe the protocol and do not give the full proof). Fix any history-ignorant interactive

[11] The discussion in this paragraph is only an intuition. The formal result is given by Definition 4 and Lemma 2.

proof system $(\mathcal{P}, \mathcal{V})$ as in the statement of the lemma, for proving membership in some language L. Denote by λ the length of the longest message sent by $\mathcal{V}$ in the interactive proof $(\mathcal{P}, \mathcal{V})$ ($\lambda \leq \ell$). We assume for simplicity (and without loss of generality) that this protocol consists of exactly ℓ rounds, where in each round $\mathcal{V}$ sends a message of size exactly λ, and $\mathcal{P}$ sends a single bit.

Fix any security parameter $t \geq \max\{\ell, \log n\}$. Let

$$(Q^{PIR}, D^{PIR}, R^{PIR})$$

be a poly-logarithmic PIR scheme, with respect to security parameter t and database size $N \stackrel{\text{def}}{=} 2^{\lambda}$. (We refer the reader to Subsection 3.2 for the definition of a poly-logarithmic PIR scheme.) We next describe how to convert $(\mathcal{P}, \mathcal{V})$ into a one-round argument system $(\mathcal{P}', \mathcal{V}')$ as in the statement of the lemma.

Fix any $x \in \{0,1\}^*$ (supposedly $x \in L$) and any string $w \in \{0,1\}^*$ given to the prover as auxiliary input. The one-round argument $(\mathcal{P}'(x,w), \mathcal{V}'(x))$ proceeds as follows:

- The verifier $\mathcal{V}'(x)$ sends the first message, computed as follows:
  1. Choose a random string $r_v \in_R \{0,1\}^{O(\ell \cdot \lambda)}$, to be used when emulating the underlying verifier $\mathcal{V}(x)$.
  2. Compute the ℓ messages $m_1, \ldots, m_\ell \in \{0,1\}^{\lambda}$ sent by $\mathcal{V}(x)$ (with randomness r_v). Note that these messages can be computed in advance since in the protocol $(\mathcal{P}, \mathcal{V})$ all the messages sent by $\mathcal{V}$ depend only on $\mathcal{V}$'s random coin tosses.
  3. For each $i \in [\ell]$, let $(q_i, s_i) \leftarrow Q^{PIR}(t, N, m_i)$.
  4. Save the values $(r_v, m_1, \ldots, m_\ell, s_1, \ldots, s_\ell, q_1, \ldots, q_\ell)$.[12]
  5. Send $(q_1, \ldots, q_\ell)$ to $\mathcal{P}'$.

  Note that if each message of $\mathcal{V}$ can be computed in time $\leq \text{poly}(t)$ then the message sent by $\mathcal{V}'$ can be computed in time $\leq \text{poly}(t, \ell) = \text{poly}(t)$.
- Upon receiving a message $(q_1, \ldots, q_\ell)$ from $\mathcal{V}'$, the algorithm $\mathcal{P}'(x,w)$ operates as follows:
  1. Choose a random string $r_p \in_R \{0,1\}^T$, to be used when emulating the underlying prover $\mathcal{P}(x,w)$.
  2. For each $i \in [\ell]$, compute an N-size database DB_i as follows: The $m \in \{0,1\}^{\lambda}$ entry of DB_i contains the i'th bit that the prover $\mathcal{P}(x,w)$ (with randomness r_p) would have sent to $\mathcal{V}(x)$, if the i'th-round message sent by $\mathcal{V}(x)$ was m.[13]
  3. For each $i \in [\ell]$, compute $a_i \leftarrow D^{PIR}(t, DB_i, q_i)$.
  4. Send the message $(a_1, \ldots, a_\ell)$ to $\mathcal{V}'$.
- Upon receiving the message $(a_1, \ldots, a_\ell)$ from $\mathcal{P}'$, the algorithm $\mathcal{V}'(x)$ operates as follows:

[12] We note that the messages $m_1, \ldots, m_\ell$ do not need to be saved since they can be recomputed from r_v. We save them for simplicity.

[13] We are using here the fact that the interactive proof system $(\mathcal{P}, \mathcal{V})$ is history-ignorant.

1. Restore the saved values $(r_v, m_1, \ldots, m_\ell, s_1, \ldots, s_\ell, q_1, \ldots, q_\ell)$.
2. For every $i \in [\ell]$, compute $b'_i \stackrel{\text{def}}{=} R^{PIR}(t, N, m_i, (q_i, s_i), a_i)$.
3. Accept if and only if $\mathcal{V}(x)$, with (initial) randomness r_v, would have accepted the messages $(b'_1, \ldots, b'_\ell)$.[14]

■

5 From Interactive-PCPs to PCAs

In this section, we propose a general method for converting an interactive-PCP system (with certain properties) into a PCA system. This method is very similar to the method of converting an interactive proof into a one-round argument (presented in Section 4). Namely, it uses a PIR scheme to reduce the round complexity of the interactive phase of the interactive-PCP system into one round (two-messages). Then, the first message (sent by the verifier) in this one-round protocol, is interpreted as the verifier's public-key in the PCA system; and the second message (sent by the prover) in the one-round protocol, together with the interactive-PCP oracle, are interpreted as the PCA string.

Theorem 1. *Assume the existence of a (uniform) poly-logarithmic PIR scheme (as defined in Definition 3). Assume that there exists an interactive-*PCP *system $(\mathcal{P}, \mathcal{V})$ with parameters (p, q, ℓ, c, s) for some NP language L, such that the interactive phase is history-ignorant,*[15] *and each message sent by the verifier in this phase depends only on the verifier's random coin tosses (and is independent of the interaction, the* PCP *string π, and the input x), and can be computed in time $\leq \text{poly}(\ell)$. Denote by λ the length of the longest message sent from $\mathcal{V}$ to $\mathcal{P}$ in the interactive phase of the interactive-*PCP *system $(\mathcal{P}, \mathcal{V})$. Assume that $\mathcal{V}$ uses at most $O(\ell \cdot \lambda)$ random bits. Then, for any security parameter $t \geq \max\{\ell, \log n\}$ there exists a* PCA *system $(\mathcal{G}', \mathcal{P}', \mathcal{V}')$ with parameters (t, p', q', c', s') for the language L, where $p' = \text{poly}(p, t)$, $q' = \text{poly}(q, t)$, $c' \geq c - 2^{-t^2}$, and $s' \leq s + 2^{-t^2}$. The prover $\mathcal{P}'$ runs in time $\leq \text{poly}(t, n, 2^\lambda)$.*

The proof of this theorem is very similar to the proof of Lemma 2.

Proof of Theorem 1. (In this version, due to space limitation, we only describe the protocol and do not give the full proof). Fix an NP language $L = \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}_L\}$. Let $(\mathcal{P}, \mathcal{V})$ be an interactive-PCP system for L with parameters (p, q, ℓ, c, s), as in the statement of the theorem. Denote by λ the length of the longest message sent by $\mathcal{V}$ in the interactive phase of $(\mathcal{P}, \mathcal{V})$ ($\lambda \leq \ell$). We assume for simplicity (and without loss of generality) that the interactive phase consists of exactly ℓ rounds. In each round $i \in [\ell]$, $\mathcal{V}$ sends a message m_i of size exactly λ (which depends only on the verifier's random coin tosses), and $\mathcal{P}$ sends

[14] This verdict test may require $\mathcal{V}'(x)$ to use additional (fresh) randomness.

[15] A history-ignorant interactive phase is defined in the same way as a history-ignorant interactive proof. See Definition 4.

a single bit b_i. For every $(x, w) \in \mathcal{R}_L$, the prover $\mathcal{P}(x, w)$ generates a bit string π of size at most $p(n)$ (where $n = |x|$). We assume for simplicity (and without loss of generality) that π is of size exactly $p(n)$.

Fix any security parameter $t \geq \max\{\ell, \log n\}$. Let $(Q^{PIR}, D^{PIR}, R^{PIR})$ be a poly-logarithmic PIR scheme, with respect to security parameter t and database size $N \stackrel{\text{def}}{=} 2^{\lambda}$. (We refer the reader to Subsection 3.2 for the definition of a poly-logarithmic PIR scheme.) We next describe how to convert the interactive-PCP system $(\mathcal{P}, \mathcal{V})$ into a PCA system $(\mathcal{G}', \mathcal{P}', \mathcal{V}')$ as in the statement of the theorem.

- $\mathcal{G}'(1^{t(n)})$ operates as follows:
  1. Choose a random string $r_v \in_R \{0,1\}^{O(\ell \cdot \lambda)}$ (to be used when emulating the messages sent by the underlying PCP verifier $\mathcal{V}$).
  2. Compute the ℓ messages $m_1, \ldots, m_\ell \in \{0,1\}^{\lambda}$ sent by $\mathcal{V}$ (with randomness r_v) in the interactive phase of $(\mathcal{P}, \mathcal{V})$. Note that these messages can be computed in advance (since all the messages sent by $\mathcal{V}$ depend only on $\mathcal{V}$'s randomness).
  3. For each $i \in [\ell]$, let $(q_i, s_i) \leftarrow Q^{PIR}(t, N, m_i)$.
  4. Let $PK = (q_1, \ldots, q_\ell)$ and let $SK = (r_v, m_1, \ldots, m_\ell, s_1, \ldots, s_\ell, q_1, \ldots, q_\ell)$.[16]
  5. Output the pair (SK, PK).

  Note that $\mathcal{G}'$ runs in time $\leq \text{poly}(t)$, since each message sent by $\mathcal{V}$ can be computed in time $\leq \text{poly}(\ell) \leq \text{poly}(t)$.
- For every $(x, w) \in \mathcal{R}_L$ (where $|x| = n$) and every $(SK, PK) \leftarrow \mathcal{G}'(1^{t(n)})$, the algorithm $\mathcal{P}'(x, w, PK)$ operates as follows.
  1. Parse $PK = (q_1, \ldots, q_\ell)$.
  2. Choose a random string $r_p \in_R \{0,1\}^{\text{poly}(n)}$ (to be used when emulating the underlying PCP prover $\mathcal{P}(x, w)$).
  3. Compute the oracle π, as computed by $\mathcal{P}(x, w)$ (with randomness r_p).
  4. For each $i \in [\ell]$, compute an N-size database DB_i as follows: The $m \in \{0,1\}^{\lambda}$ entry of DB_i contains the bit b_i that the PCP prover $\mathcal{P}(x, w)$ (with randomness r_p) would have sent to $\mathcal{V}(x)$ in the i'th round of the interactive phase of $(\mathcal{P}, \mathcal{V})$, if the i'th-round message sent by $\mathcal{V}(x)$ was m.[17]
  5. For each $i \in [\ell]$, compute $a_i \leftarrow D^{PIR}(t, DB_i, q_i)$.
  6. Output $\pi' \stackrel{\text{def}}{=} (\pi, a_1, \ldots, a_\ell)$
- For every $(x, w) \in \mathcal{R}_L$ (where $|x| = n$), every $(SK, PK) \leftarrow \mathcal{G}'(1^{t(n)})$, and every string π' (supposedly of the form $\pi' = (\pi, a_1, \ldots, a_\ell)$), the oracle machine $(\mathcal{V}')^{\pi'}(x, SK, PK)$ operates as follows:
  1. Query the oracle π' at all the coordinates $i > p(n)$. Denote the values obtained by the oracle by $(a_1, \ldots, a_\ell)$.
  2. Parse $PK = (q_1, \ldots, q_\ell)$ and $SK = (r_v, m_1, \ldots, m_\ell, s_1, \ldots, s_\ell, q_1, \ldots, q_\ell)$.
  3. For every $i \in [\ell]$, compute $b'_i \stackrel{\text{def}}{=} R^{PIR}(t, N, m_i, (q_i, s_i), a_i)$.

[16] We note that the messages $m_1, \ldots, m_\ell$ do not need to be part of the secret key since they can be recomputed from r_v. We save them for simplicity.

[17] We are using here the fact that the interactive phase of $(\mathcal{P}, \mathcal{V})$ is history-ignorant.

4. Compute the q oracle queries made by $\mathcal{V}(x)$ (with randomness r_v), assuming the ℓ messages that the prover $\mathcal{P}$ sent $\mathcal{V}$ during the interactive phase were $b'_1, \ldots, b'_\ell$. Denote these q queries by $i_1, \ldots, i_q \in [p(n)]$.
5. Query the oracle at the coordinates $i_1, \ldots, i_q$, and obtain q answers, denoted by $\pi_{i_1}, \ldots, \pi_{i_q}$.
6. Output 1 if and only if the verifier $\mathcal{V}^\pi(x)$ (with randomness r_v) outputs 1 after receiving the messages $b'_1, \ldots, b'_\ell$ from the prover $\mathcal{P}$ and receiving the bits $\pi_{i_1}, \ldots, \pi_{i_q}$ from the oracle.

■

5.1 Corollaries

We next show how Theorem 1, together with the interactive-PCP system constructed in [14], yields an *efficient* PCA system. We use the following theorem from [14].

Theorem 2. *[14] Let $\Phi(z_1, \ldots, z_k)$ be a (fanin 2) Boolean circuit of size n and depth d, and assume without loss of generality that $k \geq \log n$.*[18] *Let s be such that, $\log n \leq s \leq \text{poly}(n)$. Then, the satisfiability of Φ can be proved by an interactive-PCP with the following parameters. Size of the PCP: $p = \text{poly}(k, d)$. Number of queries: $q = \text{poly}(s)$. Communication complexity of the interactive phase: $\ell = \text{poly}(d, s)$. Completeness: 1. Soundness: 2^{-s}.*

Moreover, the interactive phase is public-coin, the verifier uses at most $O(\ell)$ random bits, and each message sent by the prover depends only on the preceding $\lambda = O(\log n)$ bits sent by the verifier.

The following is an immediate corollary of Theorem 1 and Theorem 2.

Corollary 1. *Assume the existence of a (uniform) poly-logarithmic PIR scheme (as defined in Definition 3). Let $L = \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}_L\}$ be any NP language, and assume that $\mathcal{R}_L$ is given by a Boolean circuit of size $\text{poly}(n)$ and depth d, where $n = |x|$ denotes the instance size. Let $k = |w|$ denote the witness size. Let t be a security parameter, such that, $\log n \leq t \leq \text{poly}(n)$. Then, the satisfiability of Φ can be proved by an efficient* PCA *system (i.e., PCA with a (honest) prover that runs in time $\text{poly}(n)$), with the following parameters. Size of the PCA: $p = \text{poly}(k, d, t)$. Number of queries: $q = \text{poly}(d, t)$. Completeness: $1 - 2^{-t}$. Soundness: 2^{-t}.*

Proof. Fix any NP language $L = \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}_L\}$, as above. Fix $s = t+1$. Theorem 2 implies that L has an interactive-PCP system with the following parameters: $p = \text{poly}(k, d, \log n)$, $q = \text{poly}(s)$, $\ell = \text{poly}(d, s)$, completeness 1, and soundness 2^{-s}.

The fact that each message sent by the (interactive-PCP) prover depends only on the preceding $O(\log n)$ bits sent by the verifier, implies that the interactive-PCP system can be converted into a *history-ignorant* one, where the length of the

[18] Otherwise, it is easy to check the satisfiability of Φ in time $\text{poly}(n)$.

longest message sent by the verifier in the interactive phase is $\lambda = O(\log n)$. This increases the communication complexity ℓ by at most a quadratic factor, and does not change the other parameters. The fact that the interactive phase of the (original) interactive-PCP is public-coin, implies that in the resulting (history-ignorant) interactive-PCP, each message sent by the verifier depends only on the verifier's random coin tosses, and can be computed in time $\leq \text{poly}(\ell)$.

Applying Theorem 1 (with security parameter $t' = \text{poly}(\ell, t)$) to this interactive-PCP system, results with a PCA system for L, with the desired parameters, where the prover runs in time $\text{poly}(n)$.

References

1. W. Aiello, S. N. Bhatt, R. Ostrovsky, and S. Rajagopalan. Fast Verification of Any Remote Procedure Call: Short Witness-Indistinguishable One-Round Proofs for NP. In Proc. 27th Interantional Colloquium on Automata, Language and Programming, (ICALP 2000). (manuscript withdrown by the authors prior to ICALP).
2. S. Arora, C. Lund, R. Motwani, M, Sudan, and M. Szegedy. Proof Verification and Hardness of Approximation Problems. In *FOCS 1992: 14-23.* Also in *J. ACM 45(3): 501-555 (1998).*
3. S. Arora, and S. Safra: Probabilistic Checking of Proofs: A New Characterization of NP. In *FOCS 1992: 2-13.* Also in *J. ACM 45(1): 70-122 (1998).*
4. B. Barak. How to Go Beyond the Black-Box Simulation Barrier. *FOCS 2001: 106-115*
5. L. Babai, L. Fortnow, and C. Lund. Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. In *FOCS 1990: 16-25.* Also In *Computational Complexity 1: 3-40 (1991).*
6. C. Cachin, S. Micali, and M. Stadler. Computationally Private Information Retrieval with Polylogarithmic Communication. In *EUROCRYPT 1999*, pages 402-414.
7. B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private Information Retrieval. In *J. ACM 45(6), 1998*, pages 965-981.
8. I. Damgard, N. Fazio, A. Nicolosi. Non-interactive Zero-Knowledge from Homomorphic Encryption. In *TCC 2006: 41-59.*
9. C. Dwork, M. Langberg, M. Naor, K. Nissim, and O. Reingold. Succinct Proofs for NP and Spooky Interactions. Unpublished Manuscript.
10. U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Interactive Proofs and the Hardness of Approximating Cliques. In *FOCS 1991: 2-12.* Also in *J. ACM 43(2): 268-292 (1996).*
11. A. Fiat, A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO 1986:186-194*
12. L. Fortnow, and R. Santhanam. Infeasibility of Instance Compression and Succinct PCPs for NP. In *STOC 2008: 133-142.*
13. S. Goldwasser, Y. T. Kalai. On the (In)security of the Fiat-Shamir Paradigm. In *FOCS 2003*
14. S. Goldwasser, Y. T. Kalai, and G. Rothblum. Delegating Computation: Interactive Proofs for Muggles. In *FOCS 2007: 113-122.*
15. H. Harnik, and M. Naor. On the Compressibility of NP instances and Cryptographic Applications. In *FOCS 2006: 719-728.*
16. E. Kushilevitz, R. Ostrovsky. Replication is NOT Needed: SINGLE Database, Computationally-Private Information Retrieval. In *FOCS 1997: 364-373.*

17. Y. T. Kalai, and R. Raz. Succinct Non-Interactive Zero-Knowledge Proofs with Preprocessing for LOGSNP. In *FOCS 2006: 355-366.*
18. Y. T. Kalai, and R. Raz. Interactive PCP. In *ICALP 2008: 536-547.*
19. J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *STOC 1992: 723-732.*
20. C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems. In *FOCS 1990: 2-10.* Also in *J. ACM 39(4): 859-868 (1992).*
21. S. Micali. CS Proofs (Extended Abstracts). In *FOCS 1994: 436-453.*
22. A. Shamir. IP=PSPACE. In *J. ACM 39(4): 869-877 (1992).*