Succinct NP Proofs from an Extractability Assumption

Giovanni Di Crescenzo[1] and Helger Lipmaa[2]

[1] Telcordia Technologies, Piscataway, NJ, USA, giovanni@research.telcordia.com
[2] University College London, London, UK

Abstract We prove, *using a non-standard complexity assumption*, that any language in $\mathcal{NP}$ has a *1-round* (that is, the verifier sends a message to the prover, and the prover sends a message to the verifier) argument system (that is, a proof system where soundness holds against polynomial-time provers) with communication complexity *only polylogarithmic in the size of the $\mathcal{NP}$ instance*. We also show formal evidence that the nature of the non-standard complexity assumption we use is analogous to previous assumptions proposed in the cryptographic literature. The question of whether complexity assumptions of this nature can be considered acceptable or not remains of independent interest in complexity-theoretic cryptography as well as complexity theory.

1 Introduction

A conventional $\mathcal{NP}$ proof system requires a single message from prover to verifier and communication at most polynomial in the length of the instance to the $\mathcal{NP}$ language. Several variants of this proof system have been proposed in the literature, motivated by various theoretical and practical applications. In particular, interactive proof systems [12,4] added two major ingredients: interaction between prover and verifier, and randomization in messages exchanged, so that the traditional completeness and soundness were allowed not to hold with some very small probability. In this paper we focus on simultaneously minimizing the round complexity and the communication complexity of the messages exchanged between prover and verifier.

History and previous work. In [15], the author proposed a 2-round (or, 4-message) argument system for $\mathcal{NP}$ with polylogarithmic communication complexity under standard intractability assumptions; here, an argument system [7] is a proof system where soundness holds against all polynomial-time provers. (For related constructions see also universal arguments of [5] and CS proofs of [20].) It has long been an open question whether one can build a 1-round argument system for $\mathcal{NP}$ with even sublinear communication complexity. The impressive communication-efficiency property of both PCP schemes and PIR protocols motivated the authors of [6,1] to propose some combination of both tools to obtain such an argument system. As an example of how to combine these two tools, define a database equal to the PCP-transformed witness for the statement $x \in L$; then, the verifier's message contains multiple PIR queries for random and independent positions in this database, and the prover's message contains answers to these queries according to the PIR protocol; finally, the verifier can compute the database content with respect to the queried positions, and can apply the PCP verifier to accept or not the common input. (This is the protocol proposed in [6]; the protocol proposed in [1] is a variation of it, in that it adds some randomization steps to the verifier's message). As described, this protocol satisfies the completeness property, and would seem to only require a polylogarithmic (in n) number of queries to satisfy soundness, under an appropriate assumption about the privacy property of the PIR protocol used. (The intuition here being that the PIR privacy property implies that the prover cannot guess the random indices queried by the verifier, and therefore can only meet the verifier's checks with probability only slightly larger than the PCP error probability for inputs not in the language.) Unfortunately, this protocol was proved to be *not* sound in [11], one main technical reason being that a prover can use different databases to answer to each query by the verifier. Furthermore, the authors in [11] suggest the intuitive reasoning that the PIR privacy alone might be too weak to imply the soundness for a resulting argument system along these lines.

Our result and assumption. In this paper we break through the linear-size barrier for 1-round argument systems for NP languages, and in fact, achieve polylogarithmic communication complexity. Our departure point is again the 1-round protocol from [6] based on PIR schemes and PCP systems, but we add to it some crucial modifications based on hash function families and Merkle trees, bearing some similarities to the 2-round (4-message) protocol from [15] and to another 2-round protocol from [17] proposed for 2-party private computation. As we cannot use similar proofs as for 2-round protocols, and prove the soundness of our 1-round argument system using a non-black-box but *non-standard* assumption. However, we give formal evidence that our assumption is of similar nature to certain assumptions that were recently introduced in the cryptographic literature. Very roughly speaking, our assumption can be described as follows: the only way for a prover to send a 'correct' PIR answer to a PIR query from the verifier, along with a compression of the database and a proof that the answer belongs to the compressed database, is to actually know a "correct" database. Assumptions of this nature have been used to solve various problems, including long-standing ones, in cryptography (starting with [9]) and zero-knowledge (starting with [14]). While using these assumptions, researchers warned that they are very strong and suggested that their validity is further studied. Recently, in [10], these assumptions were abstracted into a class of "extractability-assumptions", and several variants of this class were defined, the strongest variants being proved to be false (assuming intractability assumptions often used in the cryptography literature and believed to be true), and the weakest variants (including the one we use here) remaining unsettled. All proofs are omitted due to space restriction.

2 Definitions and Tools

We recall formal definitions for the main tools and protocols used in our main construction: 1-round argument systems, PCP proofs, private information retrieval schemes, collision-resistant function families, and Merkle trees. We assume some familiarity with properties of these tools and with various other notions of proof systems for $\mathcal{NP}$ languages. We use the notation $y \leftarrow A(x)$ to denote the probabilistic experiment of running (the possibly probabilistic) algorithm A on input x, and denoting its output as y.

Definition 1 (One-round argument systems). Let L be a language in $\mathcal{NP}$. Let (P, V) be a pair where $V = (V_1, V_2)$, and P, V_1, V_2 are probabilistic algorithms running in time polynomial in their first input. We say that (P, V) is a *1-round argument system with parameters* (δ_c, δ_s) for L if the following properties hold:
Completeness. $\forall x \in L$, and all witnesses w for x, it holds that $V_2(x, s, vmes, pmes) =$ accept with probability $\geq 1 - \delta_c(n)$, where $(vmes, s) \leftarrow V_1(x)$, $pmes \leftarrow P(x, w, vmes)$.
Soundness. $\forall x \notin L$, for all probabilistic polynomial time algorithms P', it holds that $V_2(x, s, vmes, pmes) =$ accept with probability $\leq \delta_s(n)$, where $(vmes, s) \leftarrow V_1(x)$ and $pmes \leftarrow P'(x, vmes)$.

(P, V) is a *1-round argument system* for L if there exists negligible functions δ_c, δ_s such that (P, V) is a 1-round argument system with parameters (δ_c, δ_s) for L.

Probabilistically Checkable Proof (PCP) Systems. In this paper we consider PCP systems [2,3] that are non-adaptive and have efficient generation (as essentially all such systems in the literature). In the formal definition of PCP systems, by π_i we denote the bit at location i of m-bit string π.

Definition 2. Let L be a language in $\mathcal{NP}$. Let (P, V) be a pair where $V = (V_1, V_2)$, and P, V_1, V_2 are probabilistic algorithms running in time polynomial in their first input. We say that (P, V) is a *(non-adaptive) probabilistically checkable proof system with parameters* (q, δ_c, δ_s) for L if for some polynomial p the next properties hold:
Completeness. $\forall x \in L$ such that $|x| = n$, for all witnesses w certifying that $x \in L$, $P(x, w)$ returns with probability $\geq 1 - \delta_c(n)$ a proof tape $\pi = \pi_1 | \cdots | \pi_{p(n)}$ such that $V_2(x, (i_1, \pi_{i_1}), \ldots, (i_q, \pi_{i_q})) =$ accept, where $(i_1, \ldots, i_q) \leftarrow V_1(x)$.
Soundness. $\forall x \notin L$, for all proof tapes π', $V_2(x, (i_1, \pi'_{i_1}), \ldots, (i_q, \pi'_{i_q})) =$ accept holds with probability $\leq \delta_s(n)$, where $(i_1, \ldots, i_q) \leftarrow V_1(x)$.

We say that (P, V) is a *(non-adaptive) probabilistically checkable proof system* for L if there exists a polynomial q and negligible functions δ_c, δ_s such that (P, V) is a (non-adaptive) probabilistically checkable proof system with parameters (q, δ_c, δ_s) for L.
In this paper we use PCP systems with slightly superlogarithmic query complexity and negligible soundess error; that is, having parameters $(q, 1 - o(1/\text{poly}(n)), o(1/\text{poly}(n)))$, for $q = (\log n)^{1+\epsilon}$, for some constant $\epsilon > 0$. PCP systems in the literature having such parameters include [22].

Private Information Retrieval (PIR) Schemes. We review the definition of PIR schemes in the single-database model [16].

Definition 3. Let D, Q, R be algorithms running in polynomial time in the length of their first input (Q, R may be probabilistic). Let n be a security parameter, m be the length of the database and ℓ the bit length of the database elements. We say that (D, Q, R) is a *(single-database) PIR scheme with parameters* $(n, m, \ell, \delta_c, \delta_p)$ if:
Correctness: For any m-element database db of ℓ-bit strings and any location $i \in [m]$, with probability $\geq 1 - \delta_c(n)$ it holds that $R(1^n, 1^m, 1^\ell, i, (q, s), a) = db[i]$, where $(q, s) \leftarrow Q(1^n, 1^m, 1^\ell, i)$, and $a \leftarrow D(1^n, 1^\ell, db, q)$.
Privacy: For any family of probabilistic circuits $\{A_n\}_{n \in \mathbb{N}}$ running in time $t(n)$ and any $i, j \in [m]$, it holds that $|p_i - p_j| \leq \delta_p(n)$, where for $h = i, j$, it holds that $p_h = \text{Prob}\left[\,(q, s) \leftarrow Q(1^n, 1^m, 1^\ell, h)\ :\ A_n(1^n, q) = 1\,\right]$,

PIR schemes with communication complexity only polylogarithmic in the size m of the database, have been proposed, under hardness assumptions about number-theoretic problems, in [8,18,13]. We can use any of these schemes in our main result.

Collision-resistant (CR) hash function families. This popular cryptographic primitive can be informally described as a family of compression functions for which it is hard to compute two preimages of the same output. We recall the formal definition of a CR hash function secure against adversaries that run in time superpolynomial in the hash function's security parameter. This will allow us to use CR hash functions with a security parameter polylogarithmic in another, global, security parameter.

Definition 4. Let $\mathcal{H} = \{H_u\}$ be a family of functions $H_u : \{0,1\}^{2v} \rightarrow \{0,1\}^v$, where u is a function index satisfying $|u| = n$. We say that $\mathcal{H}$ is a *collision-resistant function family with parameters* $(n, 2v, v, t, \epsilon)$ if for any algorithm A running in time $t(n)$, it holds that Prob $[\, u \leftarrow \{0,1\}^n; (x_1, x_2) \leftarrow A(u) \,:\, H_u(x_1) = H_u(x_2) \,] \,\leq\, \epsilon(n)$. We say that $\mathcal{H}$ is a *superpolynomial-time collision-resistant function family* if it is a collision-resistant function family with parameters $(n, 2v, v, t, \epsilon)$, for some v polylogarithmic in n, some ϵ negligible in n and some t superpolynomial in n.

Merkle trees. Starting from any collision-resistant hash function family, with hash functions H_u mapping 2ℓ-bit inputs to ℓ-bit outputs, Merkle defined in [19] the following tree-like construction to compress a polynomial number $m = 2^t$ of ℓ-bit strings $x_0, \ldots, x_{m-1}$ into a single ℓ-bit string y. The output y =MTree$(H_u; x_0, \ldots, x_{m-1})$ is recursively defined as $H_u($MTree$(H_u; x_0, \ldots, x_{m/2-1}),$ MTree$(H_u; x_{m/2}, \ldots, x_{m-1}))$, where MTree$(H_u; x) = x$, for any ℓ-bit string x. In the rest of the paper, we will use the following notation: the output computed as y =MTree$(H_u; x_0, \ldots, x_{m-1})$ is also denoted as *root*; for any ℓ-bit string x_i associated to the i-th leaf of the tree, we define the *i-th certification path* as the sequence of values that are necessary to certify that x_i is a leaf of the Merkle tree with root y. This construction has been often used in several results in complexity theory and interactive proofs, including [15,20,5].

3 An Extractability Assumption

We now present the assumption that will be used in the rest of the paper, and prove that it is an extractability assumption, a notion recently introduced in [10] which generalizes assumptions studied in various papers (starting with [9]). We start by recalling definitions from [10].

Extractability Hardness Assumptions. Informally speaking, an extractability assumption considers any probabilistic polynomial time algorithm A that, on input a security parameter in unary and an index, returns a secret output and a public output. Then, the assumption states that if A satisfies certain efficiency or hardness properties (to be defined later), then for any adversary algorithm Adv trying to simulate A, there exists an efficient algorithm Ext that, given the security parameter, the index, Adv's public output and random bits, can compute a matching secret output. Actually, it is more appropriate to talk about a class of extractability assumptions, varying over the specific algorithms A, and the algorithms that generate the index taken as input by A. Towards formal definitions, we first note that the problem of generating an extractability assumption may not be well-defined for all probabilistic polynomial-time algorithms (for instance, some algorithms may not have a secret output at all), but, instead, has to be defined for algorithms with very specific properties. This motivates the following definition of an extractable-algorithm candidate.

Definition 5. Let Ind be a set samplable in polynomial time whose elements we also call *indices*. Let A be a probabilistic polynomial time algorithm (or, alternatively, A is a deterministic algorithm that takes as input a sufficiently long and uniformly distributed string R). On input a security parameter 1^n, random string R and an index $ind \in Ind$, A returns a triple (s, m, h) in time polynomial in n. Let Setup $= (\text{Sample}, \text{Verify})$ be a pair of probabilistic polynomial time algorithms such that Sample, on input 1^n, generates pairs $(ind, sind)$, where $ind \in Ind$, and Verify, on input $(1^n, ind, sind, m, h)$, returns $accept$ with probability 1 if $\exists\, R, s$ such that $A(1^n, R, ind) = (s, m, h)$ or with probability negligible in n otherwise. We say that (A, Ind, Setup) is an *extractable-algorithm candidate* if there exists a polynomial r such that:
Efficient public output computation. There exists an efficient algorithm Eval that, on input $ind \in Ind$ and s, returns values (R, m, h) such that $(s, m, h) = A(1^n, R, ind)$
Secret output hardness. For any efficient algorithm Adv the probability p_s that $\exists R'$ such that $A(1^n, R', ind) = (s', m, h)$ is negligible in n. Here, $R \leftarrow \{0,1\}^{r(n)}$, $(ind, sind) \leftarrow \text{Sample}(1^n)$, $(s, m, h) \leftarrow A(1^n, R, ind)$, and $s' \leftarrow Adv(ind, m, h)$.
Hard-core output hardness. For any efficient algorithm Adv, the probability p_h that $\exists R'$ such that $A(1^n, R', ind) = (s, m, h')$ is negligible in n. Here, $R \leftarrow \{0,1\}^{r(n)}$, $(ind, sind) \leftarrow \text{Sample}(1^n)$, $(s, m, h) \leftarrow A(1^n, R, ind)$, and $h' \leftarrow Adv(ind, m)$.

Towards recalling the formal definition of the class of extractability assumptions, we note that even if an adversary A succeeds in generating m without any knowledge of s, then the hard-core output hardness requirement would make it hard for this specific adversary to generate h. The latter fact of course does not imply a proof that any A returning m, h actually knows s, but this class of assumptions postulates that this is indeed the case, by allowing s to be efficiently extracted from A, given the randomness used in computing 'valid' outputs m, h (for any algorithm A that satisfies the above three properties). In the formalization, we also need a pair of algorithms Setup $= (\text{Sample}, \text{Verify})$, where Sample generates the index ind taken as input by A, and Verify checks whether the output (m, h) returned by A has the correct form, by returning $accept$ with probability 1 if $\exists\, R, s$ such that $A(1^n, R, ind) = (s, m, h)$ or with probability negligible in n otherwise.

Assumption 1 [EA assumption]. Let (A, Ind, Setup), be an extractability assumption candidate, where Setup $= (\text{Sample}, \text{Verify})$ and Ind is a set that is samplable in polynomial time. For any polynomial-time algorithm Adv, there exists a polynomial time algorithm Ext, called the *A-extractor*, such that, denoting by aux a polynomial-length auxiliary-input (modeling A's history), the probability that $\text{Verify}(1^n, ind, sind, m, h) = accept$ and $\nexists R' \in \{0,1\}^{r(n)}$ such that $A(1^n, R', ind) = (s', m, h)$, is negligible in n, where $R \leftarrow \{0,1\}^{r(n)}$, $(ind, sind) \leftarrow \text{Sample}(1^n)$, $(m, h) \leftarrow Adv(1^n, R, ind, aux)$ and $s' \leftarrow Ext(1^n, R, ind, m, h, aux)$.

3.1 Our EA assumption

We now formally describe the EA assumption that we will use in this paper, which is based on PIR schemes and Merkle trees based on CR hash functions. Informally speaking, we consider an extractable-algorithm candidate that randomly chooses a large string r and compresses into a much shorter string $root$ using Merkle trees, and defines a database db as follows: the i-th record db_i is set equal to the i-th bit r_i of string r, concatenated with the logarithmic number of strings that are used to compute $root$ from r_i within the Merkle tree computation. The index ind is generated as a PIR query to a random index $j \in \{1, \ldots, |r|\}$, and the algorithm A returns, on input the security parameter 1^n and ind, a main output, computed as the string $root$ associated with the root of the Merkle tree, a hard-core output, computed as the PIR answer using ind as a query and db as a database, and a secret output, computed as the string r.

Before proceeding more formally, we sketch why this construction of an extractable-algorithm candidate satisfies Definition 5. First, it satisfies the efficient output computation requirement as the efficient computability of the main output follows from the analoguous property of the Merkle tree, and the efficient computability of the secret and hard-core outputs follow from the analoguous property of the answers in a PIR scheme. Second, it satisfies the secret output hardness as an adversary able to compute the secret output from the main output and the index can be used to contradict the collision-resistance property of the hash function family. Third, it satisfies the hard-core output unpredictability requirement as it is hard to compute a valid PIR answer only from the PIR query and the root of the Merkle tree, as for database db, this would imply a way to break the collision-resistance property of the hash function family used.

Formal description. We formally define set Ind, algorithm A and the pair of algorithms Setup $=$ (Sample, Verify), and then prove that (A, Ind, Setup) is an extractable-algorithm candidate under appropriate assumptions. We will consider databases with $m = \text{poly}(n)$ records (the actual polynomial not being important for our result to hold).

Set Ind: This is the set of CR hash functions indices and PIR queries on databases with m records; formally: $Ind = \{(u, query) \mid u \leftarrow \{0,1\}^k; (query, s_q) \leftarrow Q(1^n, 1^m, 1^\ell, i)$ for some $i \in \{1, \ldots, m\}$ and some random string used by $Q\}$.

Algorithm Sample*:* This is the querying algorithm in the PIR scheme; formally, Sample(1^n) randomly chooses $i \in \{1, \ldots, m\}, u \in \{0,1\}^k$, computes $(query, s_q) \leftarrow Q(1^n, 1^m, 1^\ell, i)$ and returns: $ind = (1^m, u, query)$ and $sind = (i, s_q)$.

Algorithm A: On input $1^n, ind$, algorithm A first randomly chooses an m-bit string r and computes a Merkle tree compression of r, thus obtaining $root$ and the i-th certification path $path_i$ from the i-th bit r_i in r to $root$, for $i = 1, \ldots, m$. Then A defines an m-record database db as follows: for $i = 1, \ldots, m$, the i-th record of db contains a unique v-bit representation of bit r_i, concatenated with the i-th certification path $path_i$. Then, A computes

1. the main output as $main = root$;
2. the hard-core output h equal to the PIR answer to the query from ind using db as a database; that is, $h = D(1^n, 1^m, 1^\ell, db, query)$, where $ind = (1^m, u, query)$;
3. the secret output s equal to the m-bit string r.

Algorithm Verify*:* This is the retrieving algorithm in the PIR scheme; formally, algorithm Verify$(1^n, ind, sind, main, h)$ is defined as follows:

1. rewrite ind as $ind = (1^m, query)$, $sind$ as $sind = (i, s_q)$, $main$ as $main = root$;
2. compute $db[i] = R(1^n, 1^m, 1^\ell, i, (query, s_q), h)$ and rewrite $db[i]$ as $r_i|path_i$;
3. check that $path_i$ is a valid i-th certification path from r_i to $root$ using hash function H_u; if yes, then return: $accept$ otherwise return: $reject$.

We obtain the following theorem.

Theorem 1. Let n be a security parameter. Assume the existence of a family of CR hash functions with parameters $(n', 2v, v, t, \epsilon)$, such that n', v are polylogarithmic in n and t is superpolynomial in n but subexponential in v. Also, assume there exists a (single-database) PIR scheme having parameters $(n, m, \ell, \delta_c, \delta_p)$, with communication complexity polylogarithmic in n, where m is polynomial in n, ℓ is polylogarithmic in n and δ_c is negligible in n. Then the above triple $(A, Ind,$ Setup$)$ is an extractable-algorithm candidate with parameters (n, p_s, p_h), where: (1) p_s is negligible in n; (2) if δ_p is negligible in n then so is p_h; (3) if (s, h, m) denotes A's output on input $(1^n, ind)$, then $|s| + |h| + |m|$ is at most polylogarithmic in n.

4 A Low-Communication 1-Round Argument for $\mathcal{NP}$

We are now ready to present the main result of the paper.

Theorem 2. Let L be a language in $\mathcal{NP}$ and let $(A, Ind,$ Setup$)$ be the triple proposed in Section 3, and proved to be an extractable-algorithm candidate assuming the existence of PIR schemes and CR hash functions. If $(A, Ind,$ Setup$)$ satisfies the EA assumption, then there exists a 1-round argument system (P, V) for L such that: if the assumed PIR scheme has communication complexity polylogarithmic in the database size then (P, V) has communication complexity polylogarithmic in n, the length of the common input to the argument system.

We once again caution the reader that this result is based on a quite non-standard hardness assumption. Preliminary studies on variants of this assumption [10] indicate that the strongest variants are actually false (under intractability assumptions that are often used in the cryptography literature and believed to be true) no matter what is the specific extractable-algorithm candidate. Luckily, the variant used here seems significantly different and it is still open whether it can be proved to be false for all extractable-algorithm candidates (under some conventional intractability assumption) or can be considered a reasonable assumption for at least one of them.

Informal Description of (P, V). Our argument system is obtained as an appropriate combination of the following tools: PIR schemes with efficient communication complexity, PCP systems, CR hash function families and Merkle trees. As in [1], the starting point is the protocol from [6]: the verifier asks to receive some random entries in the PCP-transformed witness through some random PIR queries; the prover computes the PCP-transformed witness and uses it as a database from which to compute and send the PIR answers to the verifier; finally, the latter can check that the indices retrieved from the database corresponding to entries that would be accepted by the PCP verifier.

As this protocol was shown to be not sound from [11], we attempt to modify it so that it achieves soundness under the EA assumption for the extractable-algorithm candidate presented in Section 3. Consequently, we modify the prover so that for every PIR query, it also computes a Merkle-tree compression of the PCP-transformed witness and defines each database record to contain not only a bit of the PCP-transformed witness but also the certification path to the Merkle-tree root. We then note that this modification, while enabling us to use the EA assumption, may still not be very helpful as a cheating prover might choose to apply the Merkle-tree compression algorithm to a new string π^j for every query $query^j$ made by the verifier. (In essence, this is a variant of the main objection raised by [11] about the protocol in [1].) In our protocol such attacks are avoided by modifying prover and verifier so that the prover only computes a single Merkle-tree root and the verifier can efficiently check that, for each of the verifier's PIR queries, the prover uses certification paths that refer to the same root (and thus to the same single database containing them).

Formal Description. By x we denote the n-bit common input to our argument system (P, V). Protocol (P, V), formally described in Figure 1, uses the following tools:

1. A collision-resistant hash function family $\mathcal{H} = \{H_u\}$, such that $H_u : \{0,1\}^{2v} \rightarrow \{0,1\}^v$, where $|u|, v$ are polylogarithmic in n.
2. The Merkle tree construction $Mtree$ defined in Section 2, based on the collision-resistant hash function family $\mathcal{H}$.
3. A (non-adaptive) PCP system (pcpP,pcpV), where pcpV=(pcpV$_1$,pcpV$_2$), with parameters (q, δ_c, δ_s), where q is polylogarithmic in n; δ_c, δ_s are negligible in n.
4. A (single-database) PIR scheme (D, Q, R) with parameters $(n, m, \ell, \delta_c, \delta_p)$ with communication complexity polylogarithmic in m, and where m is polynomial in n, ℓ is polylogarithmic in n and δ_c, δ_p are negligible in n.

We now prove that (P, V) (formally described at the end of the section) satisfies Theorem 2. We start by noting that V runs in polynomial time. This follows since algorithms Q, R from the assumed PIR scheme and algorithm pcpV from the assumed PCP scheme run in polynomial time; and, furthermore, since checking whether a given string is an i-th certification path in a Merkle tree can be done in polynomial time.

Communication complexity: The communication complexity of (P, V) is polylog(n) as: both the value u and each PIR query sent by V have length polylogarithmic in n; the number q of PIR queries sent to P is also polylogarithmic in n; since the database record length ℓ is $O(v \log m)$, and v is chosen to be polylogarithmic in n, then so is ℓ and so is the length of each of the PIR answers sent by P.

Completeness. Assume $x \in L$. Then the completeness (with δ_c negligible in n) follows from the correctness property of the PIR scheme used and the completeness of the PCP proof system used.

Soundness (main ideas). Assume that $x \notin L$ and that there exists a cheating prover making V accept with non-negligible probability. Then with the same probability this prover produces a main output $main$ and q corresponding hard-core outputs $h_1, \ldots, h_q$ of algorithm A, in correspondence of the PIR queries from V, from which one can obtain indices $ind_1, \ldots, ind_q$ for A. Now, we distinguish two cases, according to whether, after applying the EA assumption to triple $(ind_i, main, h_i)$ and thus extracting string W_i, for $i = 1, \ldots, q$, the extracted strings $W_1, \ldots, W_q$ are all equal or not.

Case (a): there exists $a, b \in \{1, \ldots, q\}$ such that $W_a \neq W_b$. In this case we can derive an efficient algorithm that breaks the collision-resistance of the hash function family $\mathcal{H}$. Even in this case, as while proving the secret output hardness in the proof of Theorem 1, we need to use the extractable algorithm assumption to extract two different strings W_a, W_b such that $MTree(H_u; W_a) = MTree(H_u; W_b)$.
Case (b): $W_1 = \cdots = W_q$. In this case, we can derive an efficient algorithm that distinguishes which among two q-tuples of random values in $\{1, \ldots, m\}$ was used to compute V's PIR queries (by a simple hybrid argument, this is then used to efficiently break the privacy of the PIR scheme used). Very roughly speaking, this is done by observing the following. First, for the q-tuple actually used by V's PIR queries, the prover is able to provide entries from the PCP-transformed witness that would be accepted by the PCP verifier. Instead, the q-tuple not used by V's PIR queries has distribution uniform and independent from the exchanged communication. Then the probability that the string W_1 used by the prover contains entries from the PCP-transformed witness that would be accepted by the PCP verifier in correspondence with this q-tuple can be showed to be negligible using the soundness of the PCP proof system used.

```
Common input: $n$-bit instance $x$
$P$'s private input: a witness $w$ certifying that $x \in L$.

$V$(message 1):
  1. Randomly choose an index $u$ for a hash function $H_u$ from $\mathcal{H}$;
  2. for $j = 1, \ldots, q$,
       randomly and independently choose database index $i_j \in \{1, \ldots, m\}$;
       compute PIR query $(query_j, aux_j) = Q(1^n, 1^m, 1^\ell, i_j)$;
  3. send $u, query_1, \ldots, query_q$ to $P$.
$P$(message 2):
  1. Run the PCP prover on input instance $x$ and witness $w$ and let
     $\pi =$pcpP$(x, w)$;
  2. compute $root = Mtree(H_u; \pi)$ and send $root$ to $V$;
  3. for $i = 1, \ldots, m$;
       let $path_i$ be the $i$-th certification path for the $i$-th bit $\rho_i$ of $\pi$,
       define the content of the $i$-th record of database $db$ as $(\pi_i | path_i)$;
  4. for $j = 1, \ldots, q$,
       compute $ans_j = D(1^n, 1^m, 1^\ell, db, query_j)$ and send $ans_j$ to $V$.
$V$(decision):
  1. For $j = 1, \ldots, q$,
       compute $db_{i_j} = R(1^n, 1^m, 1^\ell, i_j, (query_j, aux_j), ans_j)$;
       rewrite $db_{i_j}$ as $db_{i_j} = (path_{i_j} | \pi_{i_j})$;
       check that $path_{i_j}$ is an $i_j$-th certification path for $\pi_{i_j}$ and $root$;
  2. check that pcpV$_2(x, (i_1, \pi_{i_1}), \ldots, (i_q, \pi_{i_q})) =$accept;
  3. if all verifications are satisfied then accept else reject.
```

Acknowledgements. The second author was partially supported by the the Estonian Science Foundation, grant 6848.

References

1. William Aiello, Sandeep N. Bhatt, Rafail Ostrovsky and S. Raj Rajagopalan, *Fast Verification of Remote procedure Calls: Short Witness-Indistinguishable One-Round Proofs for $\mathcal{NP}$*, Proc. of ICALP 2000, LNCS, Springer-Verlag.
2. Sanjeev Arora and Shmuel Safra, *Probabilistic Checking of Proofs: A New Characterization of NP*, Journal of the ACM, 45(1):70–122, 1998.
3. Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy, *Proof verification and the hardness of approximation problems*, Journal of the ACM, 45(3):501-555, 1998.
4. Laszlo Babai and Shmolo Moran, *Arthur-Merlin Games: a Randomized Proof System, and a Hierarchy of Complexity Classes*, Journal of Computer and System Sciences, 36:254-276, 1988.
5. Boaz Barak and Oded Goldreich, *Universal Arguments and Their Applications*, Proc. of IEEE Conference on Computational Complexity 2002.
6. Ingrid Biehl, Bernd Meyer and Susanne Wetzel, *Ensuring the Integrity of Agent-Based Computation by Short Proofs*, Proc. of Mobile Agents 98, LNCS, Springer-Verlag.
7. Guilles Brassard, David Chaum, and Claude Crépeau, *Minimum Disclosure Proofs of Knowledge*, Journal of Computer and System Sciences, Academic Press, 37(2):156–189, 1988
8. Christian Cachin, Silvio Micali, and Markus Stadler, *Computationally Private Information Retrieval with Polylogarithmic Communication*, Proc. of EUROCRYPT 99, LNCS, Springer-Verlag.
9. Ivan Damgård, *Towards Practical Public-key Systems Secure against Chosen Ciphertext Attack*, Proc. of CRYPTO 91, LNCS, Springer-Verlag.
10. Giovanni Di Crescenzo, *Extractability Complexity Assumptions*, unpublished manuscript, Aug 06.
11. Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim and Omer Reingold, *Succinct NP Proofs and Spooky Interactions*, unpublished manuscript, Dec 04.
12. Shafi Goldwasser, Silvio Micali, and Charles Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, SIAM Journal on Computing, vol. 18, n. 1, 1989.
13. Craig Gentry and Zulfikar Ramzan, *Single-Database Private Information Retrieval with Constant Communication Rate*, Proc. of 32nd ICALP 2005, LNCS, Springer-Verlag.
14. Satoshi Hada and Toshiaki Tanaka, *On the existence of 3-round Zero-Knowledge Protocols*, Proc. of CRYPTO 98. See also IACR Eprint version.
15. Joe Kilian, *A Note on Efficient Zero-knowledge Proofs and Arguments*, Proc. of ACM STOC 1991.
16. Eyal Kushilevitz and Rafail Ostrovsky, *Replication is not needed: Single Database, computationally-private information retrieval*, Proc. of 38th IEEE FOCS 1997.
17. Sven Laur and Helger Lipmaa, *Consistent Adaptive Two-Party Computations*, Cryptology ePrint Archive, Report 2006/088, 2006.
18. Helger Lipmaa, *An Oblivious Transfer Protocol with Log-Squared Communication*, Proc. of 8th Information Security Conference (ISC'05), LNCS, Springer-Verlag.
19. Ralph Merkle, *A Certified Digital Signature*, Proc. of CRYPTO 1989, LNCS, Springer-Verlag.
20. Silvio Micali, *CS proofs*, Proc. of 35th IEEE FOCS 1994.
21. Alexander Russell, *Necessary and Sufficient Condtions for Collision-Free Hashing*, J. Cryptology 8(2): 87-100 (1995)
22. Alex Samorodnitsky and Luca Trevisan, *A PCP characterization of NP with Optimal Amortized Query Complexity*, Proc. of the 32nd ACM STOC 2000.