Universal Arguments and their Applications*

Boaz Barak
Department of Computer Science
Princeton University
Princeton, NJ 08544, USA.
boaz@cs.princeton.edu

Oded Goldreich
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel.
oded.goldreich@weizmann.ac.il

July 26, 2007

Abstract

We put forward a new type of computationally-sound proof systems, called universal-arguments. Universal-arguments are related but different from both CS-proofs (as defined by Micali [*SICOMP*, Vol. 37, pages 1253–1298, 2000]) and arguments (as defined by Brassard, Chaum, and Crépeau [*JCSS*, Vol. 37, pages 156–189, 1988]). In particular, we adopt the *instance-based* prover-efficiency paradigm of CS-proofs, but follow the computational-soundness condition of argument systems (i.e., we consider only cheating strategies that are implementable by *polynomial-size circuits*).

We show that universal-arguments can be constructed based on standard intractability assumptions that refer to polynomial-size circuits (rather than based on assumptions that refer to subexponential-size circuits as used in the construction of CS-proofs). Furthermore, these protocols have a constant number of rounds and are of the public-coin type. As an application of these universal-arguments, we weaken the intractability assumptions used in the non-black-box zero-knowledge arguments of Barak [*42nd FOCS*, 2001]. Specifically, we only utilize intractability assumptions that refer to polynomial-size circuits (rather than assumptions that refer to circuits of some "nice" super-polynomial size).

Keywords: Probabilistic proof systems, computationally-sound proof systems, zero-knowledge proof systems, proofs of knowledge, probabilistic checkable proofs (PCP), collision-resistant hashing, witness indistinguishable proof systems, error-correcting codes, tree hashing.

Comment: The current version differs from prior versions of this paper mainly in Section 4.1. The presentation in prior versions (see [8]) relied on the existence of constant-round, public-coin *strong*-WI proofs-of-knowledge for any NP set. Unfortunately, in contrary to prior misconceptions (cf. [16, Sec. 4.6]), such protocols are not known to exist [17, Apdx. C.3]. Indeed, this gap can be bypassed – as done in [7] and in the current version, by using a rather ugly patch – but our hope was to bridge the gap (i.e., prove the existence of the said strong-WI protocols) and maintain the original presentation. Having failed to do so for several years, we decided to archive the current version.

*An extended abstract has appeared in the proceedings of the *17th IEEE Conference on Computational Complexity*, pages 194–203, 2002.

Contents

1 Introduction

Various types of *probabilistic* proof systems have played a central role in the development of computer science in the last two decades. The best known ones are interactive proofs [20], zero-knowledge proofs [20], and probabilistic checkable proofs [15, 5, 14, 2], but other notions such as various types of computationally-sound proofs (e.g., arguments [12] and CS-proofs [23]) and multi-prover interactive proofs [11] have made a prominent appearance as well. *Do we really need yet another type of probabilistic proof systems?*

We believe that the answer is positive: The number of different related notions that "we need" is exactly the number of different notions that are natural, interesting and/or useful. Confining ourself to usefulness, we note that the new type of computationally-sound proof systems introduced in this paper has emerged in the context of trying to improve the constructions of non-black-box zero-knowledge arguments of Barak [6]. Furthermore, these proof system seem inherent to certain diagonalization techniques used in [6] and (in a different context) in [13].

1.1 Motivation: Applying diagonalization in cryptography

A naive idea, which was discarded for decades in Cryptography, is constructing a cryptographic scheme by "diagonalization"; for example, enumerating all probabilistic polynomial-time adversaries and making sure that each of them fails. The main reason that this idea was discarded is that the resulting scheme will (necessary) be more complex than the class of adversaries it defeats, while in cryptography a scheme should withstand adversaries that are (at least slightly) more complex than the scheme.

Still, as observed by Canetti, Goldreich and Halevi [13] and Barak [6], a small twist on diagonalization may be useful in Cryptography. The twist is using diagonalization in order to build a "trapdoor" so that the trapdoor can be used in some "imaginary setting" (e.g., by the simulator [6]), but not in the "real" setting (e.g., an actual execution of the proof system [6]).[1] Thus, the complexity of the simulator is effected by the diagonalization, whereas the complexity of the actual execution of the interactive proof is independent of the diagonalization. Specifically, the trapdoor constructed by Barak [6] is knowledge of the (adversarial) verifier's strategy, where this strategy (which is the locus of diagonalization) may be any polynomial-size circuit (where the polynomial is determined only after the proof system is specified). In the actual execution, the (zero-knowledge) prover does not use this trapdoor (but rather uses an NP-witness to the real input), and so its complexity is independent of the complexity of the trapdoor (i.e., the cheating verifier's strategy). However, the simulator uses the trapdoor, and so its complexity depends on the latter (and so every polynomial-size adversary yields a related polynomial-time simulation).

For the foregoing idea to make sense, the verifier should not be able to distinguish the case in which the (real) prover uses an NP-witness to the real input from the case in which the (simulated) prover uses the trapdoor (i.e., the cheating verifier's strategy). Indeed, Barak's protocol utilizes a witness indistinguishable (WI) proof for which both the NP-witness (to the real input) and the trapdoor (i.e., verifier's strategy) are valid witnesses. Thus, the honest verifier strategy in the WI proof must be independent of the length of the witness used by the prover. This is because in one of the cases the length of the witness is determined only after the proof system is specified (i.e.,

[1]In [13], the "imaginary setting" is an implementation of the random-oracle by a function ensemble (shown not to exist), whereas the "real setting" is the ideal (Random Oracle Model) setting in which the scheme uses a random-oracle. (Indeed our perspective here is opposite to the one in [13], where the random-oracle is considered "imaginary" and its implementations by function ensembles are considered "real").

in the simulation, the length of the trapdoor is polynomial, but this polynomial is determined and fixed only after the proof system is specified).

We conclude that in order to use diagonalization as above, we should have a (WI) proof system that is capable of handling any "NP-statement" (and not merely statements in any a-priori fixed NP-set). Put in other words, we need a *single* proof system that can be used to provide proofs for *any* set S in $\mathcal{NP}$ such that the running time and communication needed for verifying that $x \in S$ is bounded by a fixed (i.e., *single*) polynomial in $|x|$, which does *not* depend on the set S. In particular, it may be the case that $S \in \mathrm{Ntime}(p)$, where $p(\cdot)$ is a polynomial that is *larger* than the fixed polynomial bounding the verifier's complexity. We stress that, in contrast, typically when the phrase "proof systems for $\mathcal{NP}$" is used, the intended meaning is that every set $S \in \mathcal{NP}$ has a different proof system (and the complexity of verifying that $x \in S$ is bounded by an S*-dependent polynomial* in $|x|$).

1.2 The notion of universal arguments

For sake of simplicity, we define and present proof systems only for the *universal set* $S_{\mathcal{U}}$ defined such that the tuple (M, x, t) is in $S_{\mathcal{U}}$ if M is a non-deterministic machine that accepts x within t steps.[2] This suffices for handling any NP-set via a single protocol, because every $\mathcal{NP}$-set S is linear-time reducible to $S_{\mathcal{U}}$ (e.g., via the mapping $x \mapsto (M_S, x, 2^{|x|})$, where M_S is any fixed non-deterministic polynomial-time machine that decides S). Thus, a proof system for $S_{\mathcal{U}}$ allows us to handle all "NP-statements" (in a uniform manner): for any $S \in \mathcal{NP}$, when wishing to verify the assertion "x in S", the verifier should just use the proof system of $S_{\mathcal{U}}$ on input $(M_S, x, 2^{|x|})$, where M_S is as above. (In fact, $S_{\mathcal{U}}$ is $\mathcal{NE}$-complete, by an analogous linear-time reduction).[3]

We consider also the natural witness-relation for $S_{\mathcal{U}}$, denoted $R_{\mathcal{U}}$: the pair $((M, x, t), w)$ is in $R_{\mathcal{U}}$ if M (viewed here as a two-input deterministic machine) accepts (x, w) within t steps. Loosely speaking, a universal argument system (or a universal argument system for $S_{\mathcal{U}}$) is a two-party protocol (P, V), for common inputs of the form (M, x, t), that satisfies the following:

Efficient verification: The *total* time spent by the (probabilistic) verifier V is polynomial in length of the common input (i.e., polynomial in $|(M, x, t)| = O(|M| + |x| + \log t)$). In particular, all messages exchanged in the protocol have length that is so bounded.

Completeness by a relatively-efficient prover: For every $((M, x, t), w)$ in $R_{\mathcal{U}}$, on common input (M, x, t), when P is given auxiliary input w, it always convinces V. Furthermore, the total time spent by P in this case is bounded by a fixed polynomial in ($|M|$ and) $T_M(x, w) \leq t$, where $T_M(x, w)$ is the number of steps taken by M on input (x, w).

Computational Soundness: For every polynomial-size circuit family $\{C_n\}_{n \in \mathsf{N}}$ and every $(M, x, t) \in \{0,1\}^n \setminus S_{\mathcal{U}}$, the probability that, on common input (M, x, t), the ("cheating") circuit C_n succeeds in fooling V (into accepting (M, x, t)) is negligible (as a function of n).

(The actual definition appears in Section 2.)

[2] One nice feature of $S_{\mathcal{U}}$ is that it comes with a natural measure of complexity of instances: the complexity of (M, x, t) is the actual time it takes M to accept x (when using the best sequence of non-deterministic choices). (Similarly, the complexity of (M, x, t) coupled with the witness w is the actual time it takes M to accept x when using w as the sequence of non-deterministic choices.) Such a complexity measure is pivotal to the refined formulation of the prover complexity condition.

[3] That is, for $S \in \mathrm{Ntime}(e)$, where $e(n) = 2^{cn}$ for some constant c, we use the reduction $x \mapsto (M_S, x, 2^{c|x|})$. Furthermore, every set in $\mathcal{NEXP}$ is polynomial-time (but not linear-time) reducible to $S_{\mathcal{U}}$.

Relation to prior notions. Universal-arguments are related but different from both CS-proofs (as defined by Micali [23]) and arguments (as defined by Brassard, Chaum and Crepeau [12]). Specifically:

1. The efficient-verification condition is identical in all definitions (except that arguments are typically defined only for sets in $\mathcal{NP}$).
2. The foregoing "completeness by a relatively-efficient prover" condition follows the instance-based paradigm of CS-proofs (but provides the prover with an auxiliary input).
3. The foregoing computational-soundness condition is exactly as in argument systems (and is typically weaker than the one in CS-proofs).

Thus, in a sense, universal-arguments are a hybrid of arguments and CS-proofs. Indeed, universal-arguments are weaker than CS-proofs, but the point is that we will be able to construct a universal-argument based on a weaker assumption than the ones that seem necessary for constructing CS-proofs (see Footnote 4).

We comment that computational-soundness seems unavoidable in any proof system for $S_{\mathcal{U}}$ that satisfies the efficient-verification condition (even just "uniformly for all $\mathcal{NP}$"). In contrast, statistical soundness (coupled with the standard notion of efficient-verification) would have implied that $S_{\mathcal{U}}$ (or "just" all $\mathcal{NP}$) is in $\mathrm{DSPACE}(p)$, for some fixed polynomial p. (The reason is that the total communication in such a protocol must be upper-bounded by a fixed polynomial, p, and that an optimal prover strategy can be implemented in space p.)

On natural applications of universal-arguments. A strange-looking aspect of universal-arguments is that, on some YES-instances, the designated prover may run more time than allowed to cheating provers (i.e., a fixed polynomial in $T_M(.,.)$ may be larger than an arbitrary polynomial in the length of the common input).[4] However, in typical application (such as ours), the designated prover will never be invoked on such inputs (i.e., requiring it to run for time that is super-polynomial in the length of the common input). Actually, we shall only use the fact that a universal-argument system guarantees the following behavior with respect to any polynomial p (which may be selected after the system is specified):

1. For every $(y, w) \in R_{\mathcal{U}}$ such that $y = (M, x, t)$ and $t \leq p(|x|)$, it holds that (given y and w) the designated prover convinces the verifier to accept y, and the total time used by the prover is $\mathrm{poly}(|M| + t) = \mathrm{poly}(|y|)$. Specifically, there exists a fixed polynomial q_0 that is associated with the universal-argument system such that the running time of the designated prover on the foregoing input is $q_0(|M| + t) \leq q_0(|M| + p(|x|)) < (q_0 \circ p)(|y|)$.
2. For every polynomial q and every sufficiently long $y = (M, x, t) \notin S_{\mathcal{U}}$, it holds that no $q(|y|)$-size circuit can convince the verifier to accept y.

[4]This phenomena does not occur in arguments [12] and in CS-proofs [23]: Arguments were defined only for individual sets in $\mathcal{NP}$, and so the issue never arises. In the case of CS-proofs, the definition of computational-soundness relates to cheating provers of size exponential in the security parameter, which is typically set to be linear in the length of the common input [23]. Thus, the cheating provers are always allowed more running time than the designated prover (since its running-time is always at most exponential in the length of the common input). However, it seems that allowing the adversaries time that is exponential in the security parameter requires using intractability assumptions that refer to exponential (or sub-exponential) circuits.

Thus, the complexity bound considered in the soundness condition (of Item 2) may exceed the complexity of the designated prover when handling YES-inputs of the type mentioned in Item 1. Needless to say, the foregoing items cover all NP-sets (where a set $S \in \mathcal{NP}$ is handled by selecting the polynomial p such that $S \in \mathrm{Ntime}(p)$ holds).

1.3 The construction of universal arguments

By adapting the construction of Kilian [21], one can show that the existence of *strong* collision-resistant hashing functions implies the existence of universal arguments (and even CS-proofs for $S_{\mathcal{U}}$; cf. Micali [23]). By strong collision-resistant hashing we mean families of functions for which collisions are hard to find even by using *subexponential-size* circuits. The goal, achieved in this paper, is to construct universal arguments based only on standard collision-resistant hashing; that is, families of functions for which collisions are hard to find by *polynomial-size* circuits. That is, we obtain:

Theorem 1.1 (our main result): *The existence of* (standard) *collision-resistant hashing functions implies the existence of universal arguments. Furthermore, these proof systems are of the public-coin*[5] *type and use a constant number of rounds.*

Our construction of universal arguments adapts Kilian's construction [21] in a quite straightforward manner. Our contribution is in the analysis of this construction. Unlike in previous analysis (as in [21] and [23]), when establishing computational-soundness via contradiction, we cannot afford to derive a collision-forming circuit of size that is polynomial in the worst-case time-complexity of the designated prover (because the designated prover may have (worst-case) complexity that is super-polynomial (in the input length)).[6] We need to derive collision-forming circuit of size that is polynomial in the input length. Indeed, doing so allows us to use standard collision-resistant hashing (rather than strong ones).

The analysis is further complicated by our desire to establish a "proof of knowledge" property, which is needed for our main application (discussed next).

1.4 Application to zero-knowledge arguments

Barak's construction [6] of non-black-box zero-knowledge arguments (for any set in $\mathcal{NP}$) uses a witness indistinguishable (WI) argument of knowledge for $R_{\mathcal{U}}$.[7] In his protocol, the prover uses this WI argument (of knowledge) to prove that it knows either an NP-witness for the original common input or a program that fits the verifier functionality (as reflected in the challenge-respond exchange that follows). Thus, as a first step, we need to transform our universal argument (of knowledge) into a corresponding WI universal argument (of knowledge). The transformation essentially follows Barak's transformation [6], but then we encounter a second place where Barak uses a super-polynomial hardness assumption: Barak uses a collision-resistant hashing function to

[5] A.k.a, Arthur–Merlin systems (cf. [3]).

[6] Specifically, Kilian's construction [21] uses a PCP system, and the contradiction hypothesis is shown to yield a collision-forming circuit that is always bigger than the length of the corresponding PCP-oracle (which, in the case of $S_{\mathcal{U}}$, is exponential in the input length). Instead, we show how to obtain a collision-forming circuit that is smaller than the length of the corresponding PCP-oracle.

[7] In fact, Barak uses a CS-proof (of knowledge) for $R_{\mathcal{U}}^f \subset R_{\mathcal{U}}$, where f is any "nice" super-polynomial function (e.g., $f(n) = n^{\log_2 n}$) and $((M, x, t), w)$ is in $R_{\mathcal{U}}^f$ only if $t \leq f(|x|)$. He constructs such CS-proof assuming the existence of hashing functions that are resilient with respect to f-size circuits (rather than subexponential hardness which would have been required for CS-proof for $R_{\mathcal{U}}$).

hash "$S_{\mathcal{U}}$-witnesses" (into fix-length strings), where the length of these witnesses is bounded by some super-polynomial function (but not by any polynomial). Consequently, a collision on such long strings only yields violation of a super-polynomial collision-resistant assumption. To avoid super-polynomial hardness assumptions, we hash these witnesses by combining "tree-hashing" (as in Kilian's construction [21]) with an error-correcting code. Specifically, first the witness string is encoded using an error-correcting code, and then the "tree-hashing" is applied to the result. Thus, if two different strings are so hashed to the same value, then we can form a collision with respect to the basic hashing function (used in the "tree-hashing") by considering a uniformly selected leaf (which is quite likely to be assigned different values under an error-correction coding of different strings). Combining the foregoing, we obtain:

Theorem 1.2 (our main application): *The existence of* (standard) *collision-resistant hashing functions implies the existence of* (non-black-box) *zero-knowledge arguments for any set in $\mathcal{NP}$ such that these protocols have the following additional properties:*

1. *The protocol has a constant number of rounds and uses only public-coins;*
2. *The simulator runs in strict* (rather than expected) *probabilistic polynomial-time;*
3. *The protocol remains zero-knowledge when, say, n^2 copies are executed concurrently.*

Recall that, assuming $\mathcal{NP} \not\subseteq \mathcal{BPP}$, each of the three extra properties requires a non-black-box simulator (and that such protocols were presented for the first time in [6]).[8] Thus, Theorem 1.2 establishes the main result of Barak's work [6] *under a weaker assumption*: We only assume the existence of hashing functions that are resilient with respect to polynomial-size circuits (rather than with respect to circuits of some super-polynomial size).

1.5 Organization

In Section 2 we define universal arguments and in Section 3 we show how to construct them (using any collision-resistant hash functions). The application to the construction of non-black-box zero-knowledge arguments is presented in Section 4. Appendix B contains a revised treatment of the notion of non-oblivious commitment scheme, which may be of independent interest (because it augments the initial treatment provided in [16, §4.9.2.1] and corrected in [17, Sec. C.3.3]).

2 The Definition of Universal Arguments

Let us start with some general notions. For an integer n, we denote the set $\{1, ..., n\}$ by $[n]$. We denote by $\mu : \mathsf{N} \to [0,1]$ an unspecified negligible function; that is, for every positive polynomial p and all sufficiently large n, it holds that $\mu(n) < 1/p(n)$. We say that an event occurs with overwhelmingly high probability if it occurs with probability at least $1 - \mu(n)$, where n is the relevant security parameter. For a pair of (interactive) strategies, denoted (P, V), we denote by $(P(w), V)(y)$ the output of V when interacting with $P(w)$ on common input y, where $P(w)$ denotes the functionality of P when given auxiliary input w.

In continuation to the discussion in Section 1.2, we now define universal argument systems (for $S_{\mathcal{U}}$). Recall that $S_{\mathcal{U}} = \{(M, x, t) : \exists w \text{ s.t. } ((M, x, t), w) \in R_{\mathcal{U}}\}$, where $((M, x, t), w) \in R_{\mathcal{U}}$ if M

[8] Indeed, see [6, 7] for a discussion of various results regarding the impossibility of achieving the above via a black-box simulator. We stress that the current discussion refers to protocols of negligible soundness error.

accepts (x,w) within t steps. Let $T_M(x,w)$ denote the number of steps made by M on input (x,w); indeed, if $((M,x,t),w) \in R_{\mathcal{U}}$ then $T_M(x,w) \le t$. Recall that $|(M,x,t)| = O(|M|+|x|+\log t)$; that is, t is given in binary. In the following definition, we incorporate a (weak) "proof of knowledge" property (which was mentioned in Sections 1.3 and 1.4, but not in Section 1.2).

Definition 2.1 (universal argument): *A* universal-argument system *is a pair of strategies, denoted (P,V), that satisfies the following properties:*

Efficient Verification: *There exists a polynomial p such that for any $y=(M,x,t)$, the total time spent by the* (probabilistic) *verifier strategy V, on common input y, is at most $p(|y|)$. In particular, all messages exchanged in the protocol have length smaller than $p(|y|)$.*

Completeness via a relatively-efficient prover: *For every $((M,x,t),w)$ in $R_{\mathcal{U}}$,*

$$\Pr[(P(w),V)(M,x,t)=1] \;=\; 1.$$

Furthermore, there exists a polynomial p such that the total time spent by $P(w)$, on common input (M,x,t), is at most $p(|M|+T_M(x,w)) \le p(|M|+t)$.

Computational Soundness: *For every polynomial-size circuit family $\{\widetilde{P}_n\}_{n\in\mathsf{N}}$, and every $(M,x,t) \in \{0,1\}^n \setminus S_{\mathcal{U}}$,*

$$\Pr[(\widetilde{P}_n,V)(M,x,t)=1] \;<\; \mu(n)$$

where $\mu:\mathsf{N}\to[0,1]$ is a negligible function.

A weak proof-of-knowledge property: *For every positive polynomial p there exists a positive polynomial p' and a probabilistic polynomial-time oracle machine E such that the following holds:*[9] *for every polynomial-size circuit family $\{\widetilde{P}_n\}_{n\in\mathsf{N}}$, and every sufficiently long $y=(M,x,t)\in\{0,1\}^*$ if $\Pr[(\widetilde{P}_n,V)(y)=1] > 1/p(|y|)$ then*

$$\Pr_r\left[\begin{array}{c} \exists w=w_1\cdots w_t \in R_{\mathcal{U}}(y) \\ \quad\forall i\in[t] \quad E_r^{\widetilde{P}_n}(y,i)=w_i \end{array}\right] > \frac{1}{p'(|y|)}$$

where $R_{\mathcal{U}}(y) \stackrel{\rm def}{=} \{w:(y,w)\in R_{\mathcal{U}}\}$ and $E_r^{\widetilde{P}_n}(.,.)$ denotes the function defined by fixing the random-tape of E to equal r, and providing the resulting E_r with oracle access to $\widetilde{P}_n$. The oracle machine E is called a (knowledge) extractor.

A few comments regarding the weak proof-of-knowledge property are in place. First, note that the condition "$\forall i\in[t]$ it holds that $E_r^{\widetilde{P}_n}(y,i)=w_i$" means that $E_r^{\widetilde{P}_n}(y,.)$ is an *implicit representation* of the string $w=w_1\cdots w_t$ (i.e., any specific bit of w is obtained by instantiating the second input to $E_r^{\widetilde{P}_n}(y,.)$ accordingly). If $\Pr[(\widetilde{P}_n,V)(y)=1] > 1/p(|y|)$ then at least an $1/p'(|y|)$ fraction of the possible r's yield such implicit representations of some string in $R_{\mathcal{U}}(y)$, but these strings are not necessarily equal (i.e., different r's may yield different strings in $R_{\mathcal{U}}(y)$). Implicit (rather than explicit) representation is required here because we want the extractor to run in polynomial-time, whereas the length of the strings in $R_{\mathcal{U}}(y)$ may not be bounded by any polynomial (in $|y|$). Finally, we note that the weak proof-of-knowledge property is indeed weaker than the standard definition of a proof of knowledge (cf. [9], [16, Sec. 4.7] and Footnote 9), but it suffices for the applications that we have in mind.

[9]Indeed, the polynomial p' as well as the (polynomial) running-time of E may depend on the polynomial p (which determines the noticeable threshold probability).

3 The Construction of Universal Arguments

As mentioned in Section 1.3, by adapting of the construction of Kilian [21], one can easily show that the existence of *strong* collision-resistant hashing functions implies the existence of universal arguments (and even CS-proofs for $S_{\mathcal{U}}$; cf. Micali [23]). Here we show how a similar adaptation, when using only *standard* collision-resistant hashing functions, yields universal arguments. Our focus is on demonstrating the computational soundness of this construction, which should now be established under a weaker assumption than the one used in [21, 23].

3.1 Motivation

In order to explain the difficulty and its resolution, let us recall the basic construction of Kilian [21] (used also by Micali [23]), as adapted to our setting.

Our starting point is a $\mathcal{PCP}[\text{poly}, \text{poly}]$ system for $S_{\mathcal{U}} \in \mathcal{NEXP}$, which is used in the universal-argument system as follows. The verifier starts by sending the prover a hashing function. The prover constructs a PCP-proof/oracle (corresponding to the common input and its own auxiliary input), places the bits of this oracle at the leaves of a polynomial-depth full binary tree, and places in each internal node the hash-value obtained by applying the hashing function to the labels of its children. The prover sends the *label of the root* to the verifier, which responses by sending a random tape of the type used by the PCP-verifier. Both parties determine the queries corresponding to this tape, and the prover responds with the values of the corresponding leaves along with the labels of the vertices along the paths from these leaves to the root (as well as the labels of the siblings of these vertices). The verifier checks that this sequence of labels matches the corresponding applications of the hashing function, and also emulates the PCP-verifier. Ignoring (for a moment) the issue of prover's complexity, the problem we consider next is that of establishing computational-soundness.

The naive approach is to consider what how the prover responds to each of the possible random-tapes sent to it. If the prover answers consistently (i.e., with leaf-labels that depend only on the leaf location), then we obtain a pcp-oracle and soundness follows by the soundness of the PCP scheme. On the other hand, inconsistent labels for the same leaf yield a (hashing) collision somewhere along the path to the root. However, in order to find such a collision, we must spend time proportional to the size of the tree, which yields contradiction only in the case that the hashing function is supposed to withstand adversaries that use that much time. Note that the size of the tree is picked by the (adversarial) prover, and since we wish to handle $S_{\mathcal{U}}$ (or merely "only" all of $\mathcal{NP}$) we do not have an *a priori* polynomial bound on the size of the tree that the prover is allowed to use (because no such bound exists for the designated prover). But in such a case, if the tree is exponential (or even merely super-polynomial) in the security parameter, then we derive contradiction only when using hashing functions of sub-exponential (respectively, super-polynomial) security.

In contrast to the foregoing naive approach, the approach taken here is to consider each leaf separately rather than all leaves together. That is, the naive analysis distinguishes the case that the prover answers inconsistently on *some* leaf from the case it answer consistently on *all* leaves. Instead, we consider each leave separately, and distinguishes the case that the prover answers inconsistently on *this leaf* from the case it answer consistently on *this leaf*. Loosely speaking, we call a leaf good if the prover answers consistently on it, and observe that if a big fraction of the leaves are good then soundness follows by the soundness of the PCP scheme (regardless of the contents of other leaves). On the other hand, if sufficiently many leaves are not good, then we obtain a collision by picking a random leave (hoping that it is not good) and obtaining inconsistent labels for it. This requires being able to uniformly select a random-tape that makes the pcp-verifier make the corresponding query, a property which is fortunately enjoyed by the relevant PCP systems.

We warn that the above is merely a rough description of the main idea in our analysis. Furthermore, in order to establish the proof-of-knowledge property of our construction, we need to rely on an analogous property of the PCP system (which again happens to be satisfied by the relevant PCP systems).

3.2 The PCP system in use

We first recall the basic definition of a PCP system. Loosely speaking, a probabilistically checkable proof (PCP) system consists of a probabilistic polynomial-time verifier having access to an oracle which represents a proof in redundant form. Typically, the verifier accesses only few of the oracle bits, and these bit positions are determined by the outcome of the verifier's coin tosses. It is required that if the assertion holds then the verifier always accepts (i.e., when given access to an adequate oracle); whereas, if the assertion is false then the verifier must reject with high probability (as specified in an adequate bound), no matter which oracle is used. The basic definition of the PCP setting is given in Item (1) below. Typically, the complexity measures introduced in Item (2) are of key importance, but this is not the case in the current work.

Definition 3.1 (PCP – basic definition):

1. *A* probabilistic checkable proof system (pcp) with error bound $\epsilon : \mathsf{N} \to [0,1]$ for a set S *is a probabilistic polynomial-time oracle machine* (called verifier)*, denoted V, satisfying*
   - Completeness*: For every $x \in S$ there exists an oracle π_x such that V, on input x and access to oracle π_x, always accepts x.*
   - Soundness*: For every $x \notin S$ and every oracle π, machine V, on input x and access to oracle π, rejects x with probability at least $1 - \epsilon(|x|)$.*
2. *Let r and q be integer functions. The complexity class $\mathcal{PCP}_\epsilon[r(\cdot), q(\cdot)]$ consists of sets having a pcp system with error bound ϵ in which the verifier, on any input of length n, makes at most $r(n)$ coin tosses and at most $q(n)$ oracle queries.*

Note that if S has a pcp system with error bound ϵ, then $S \in \mathcal{PCP}_\epsilon[p(\cdot), p(\cdot)]$, for some polynomial p. Here we will only care that $S_\mathcal{U} \in \mathcal{NE}$ has a pcp system with an exponentially decreasing error bound (i.e., $\epsilon(n) = 2^{-n}$). Instead of caring about the refine complexity measures (of Item 2), we will care about the following *additional properties* which are satisfied by some pcp systems, where only some of these properties were explicitly considered before (see discussion below).

Definition 3.2 (PCP – auxiliary properties): *Let V be a pcp verifier with error $\epsilon : \mathsf{N} \to [0,1]$ for a set $S \in \mathcal{NEXP}$, and let R be a corresponding witness relation. That is, if $S \in$* Ntime$(t(\cdot))$*, then we refer to a* polynomial-time decidable *relation R satisfying $x \in S$ if and only if there exists w of length at most $t(|x|)$ such that $(x, w) \in R$. We consider the following auxiliary properties:*

Relatively-efficient oracle-construction: *This property holds if there exists a polynomial-time algorithm P such that, given any $(x, w) \in R$, algorithm P outputs an oracle π_x that makes V always accept (i.e., as in the completeness condition).*

Non-adaptive verifier: *This property holds if the verifier's queries are determined based only on the input and its internal coin tosses, independently of the answers given to previous queries. That is, V can be decomposed into a pair of algorithms, Q and D, such that on input x and random-tape r, the verifier makes the query sequence $Q(x, r, 1), Q(x, r, 2), \ldots, Q(x, r, p(|x|))$,* *obtains the answers $b_1, ..., b_{p(|x|)}$, and decides by according to $D(x, r, b_1 \cdots b_{p(|x|)})$, where p is some fixed polynomial.*

Efficient reverse-sampling: *This property holds if there exists a probabilistic polynomial-time algorithm S such that, given any string x and integers i and j, algorithm S outputs a uniformly distributed r that satisfies $Q(x, r, i) = j$, where Q is as in the previous item.*

A proof-of-knowledge property: *This property holds if there exists a probabilistic polynomial-time oracle machine E such that the following holds:*[10] *for every x and π, if $\Pr[V^\pi(x) = 1] > \epsilon(|x|)$ then there exists $w = w_1 \cdots w_t$ such that $(x, w) \in R$ and $\Pr[E^\pi(x, i) = w_i] > 2/3$ holds for every i.*

Non-adaptive pcp verifiers were explicitly considered in several works, and in fact in some sources PCP is defined in terms of non-adaptive verifiers. (Needless to say, almost all pcp systems use non-adaptive verifiers.) The oracle-construction and proof-of-knowledge properties are implicit in some works, and are known to hold for the many pcp systems (although we are not aware of a text that contains a proof of this fact). To the best of our knowledge, the reverse-sampling property was not considered before. Nevertheless, it can be verified that any $S \in \mathcal{NEXP}$ has a pcp system that satisfies all the foregoing properties.

Theorem 3.3 *For every $S \in \mathcal{NEXP}$ and for every $\epsilon : \mathsf{N} \to [0, 1]$ such that $\epsilon(n) > 2^{-\mathrm{poly}(n)}$, there exist a pcp system with error ϵ for S such that this pcp satisfies the four properties listed in Definition 3.2.*

Proof sketch: For $S \in \mathrm{Ntime}(t(\cdot))$, we consider the $\mathcal{PCP}_{1/2}[O(\log t(\cdot)), \mathrm{poly}(\cdot)]$ system presented by Babai *et. al.* [5] (i.e., the starting point of Arora *et. al.* [2, 1]). (We stress that this pcp system, unlike the one of Feige *et. al.* [14], uses oracles of length polynomial in t.) This pcp system is non-adaptive and is well-known to satisfies the oracle-construction property. It is also known (alas less well-known) that this pcp system satisfies the proof-of-knowledge property. Finally, it is easy to see that this pcp system (as any reasonable pcp system we know of) also satisfies the reverse-sampling property.[11] *Further details regarding the proof of all the foregoing facts can be found in Appendix A.* Thus, we obtain a pcp system with error $1/2$ (for S) that satisfies all the auxiliary properties listed in Definition 3.2. To obtain the desired error of ϵ, we apply straightforward error-reduction, while noting that this process does not affect the oracle and so the resulting (error-reduced) pcp preserves all the auxiliary properties. ■

3.3 The actual construction

The construction is an adaptation of Kilian's construction [21] (used also by Micali [23]). Using Theorem 3.3, we start with a pcp system with error $\epsilon(n) = 2^{-n}$ for $S_\mathcal{U}$ that satisfies the auxiliary properties in Definition 3.2. Actually, the corresponding witness relation will not be $R_\mathcal{U}$ as defined in Section 1.2, but rather a minor modification of it, denoted $R'_\mathcal{U}$: the pair $((M, x, t), (w, 1^{t'}))$ is in $R'_\mathcal{U}$ if M accepts (x, w) in $t' \leq t$ steps. (The purpose of the modification is to obtain a relation that

[10]For negligible ϵ (as used below), this proof-of-knowledge property is stronger than the *standard* proof-of-knowledge property (as in [9] and [16, Sec. 4.7.1]). Indeed, the proof-of-knowledge property in Definition 3.2 is analogous to the definition of a *strong* proof-of-knowledge (as in [16, Sec. 4.7.6]).

[11]This property follows from the structure of the standard pcp systems. In our case, the system consists of a *sum-check* (a la Lund *et. al.* [22]), and a *low-degree test*. In both tests, the queries are selected in a very simple manner, and what is complex (at least in the case of low-degree tests) is the analysis of the test.

is decidable in polynomial-time, as required in Definition 3.2.) Let V_{pcp} denote the aforementioned pcp system (or rather its verifier), and $P_{\text{pcp}}, Q_{\text{pcp}}, D_{\text{pcp}}, S_{\text{pcp}}, E_{\text{pcp}}$ denote the auxiliary algorithms (or machines) guaranteed by Definition 3.2 (i.e., P_{pcp} is the oracle-constructing procedure, Q_{pcp} determines the verifier's queries, D_{pcp} describes the verifier's final decision, S_{pcp} provides reverse-sampling, and E_{pcp} is the "witness extractor" guaranteed in the proof-of-knowledge property).

A second ingredient used in the construction is a family of collision-resistant hashing functions. That is, a collection of (uniformly polynomial-time computable) functions $\{h_\alpha : \{0,1\}^* \to \{0,1\}^{|\alpha|}\}$ such that for every (non-uniform) family of polynomial-size circuits $\{C_n\}_{n\in\mathsf{N}}$

$$\Pr_{\alpha\in\{0,1\}^n}[C_n(\alpha) = (x,y) \text{ s.t. } x \neq y \text{ and } h_\alpha(x) = h_\alpha(y)] \quad = \quad \mu(n)$$

where μ is a negligible function.

Construction 3.4 (a universal argument for $S_{\mathcal{U}}$):

Common input: *$y = (M, x, t)$, supposedly in $S_{\mathcal{U}}$. Let $n \stackrel{\text{def}}{=} |y|$.*

Auxiliary input to the prover: *w such that supposedly $(y, w) \in R_{\mathcal{U}}$ holds.*

First verifier step (V1): *Uniformly select $\alpha \in \{0,1\}^n$, and send it to the prover.*

First prover step (P1): *When describing the prover's actions, we assume that $(y, w) \in R_{\mathcal{U}}$.*

1. Preliminary action by the prover: *The prover invokes M on input (x, w), and obtains $t' = t_M(x, w)$. Assuming that $(y, w) \in R_{\mathcal{U}}$ and letting $w' = (w, 1^{t'})$, the prover obtains an $R'_{\mathcal{U}}$-witness; that is, $(y, w') \in R'_{\mathcal{U}}$.*
2. Oracle-construction: *Invoking P_{pcp} on (y, w'), the prover obtains $\pi_y = P_{\text{pcp}}(y, w')$.*
3. Construction of a hashing tree: *Letting $d \stackrel{\text{def}}{=} \lceil \log_2 |\pi_y| \rceil$, the prover constructs a binary tree of depth d and associates its nodes with binary strings of length at most d such that the root is associated with the empty string, and an internal node associated with γ has children associated with $\gamma 0$ and $\gamma 1$. Using the oracle π_y (just constructed) and the hashing function h_α (sent by the verifier), the prover* labels *the nodes of this tree as follows:*
   - *The* label of a leaf *associated with $\gamma \in \{0,1\}^d$ is the value of π_y at position γ; that is, this label, denoted ℓ_γ, equals the answer of oracle π_y to the query γ.*
   - *The* label of an internal node *associated with $\gamma \in \cup_{i=0}^{d-1}\{0,1\}^i$ is the value obtained by applying h_α to the string $\ell_{\gamma 0}\ell_{\gamma 1}$. This label is denoted ℓ_γ.*

   Thus, the label of the node associated with $\gamma \in \cup_{i=0}^{d}\{0,1\}^i$ is denoted ℓ_γ.
4. The actual message sent *by the prover is the depth of the tree and the label of its root. That is, the prover sends the pair (d, ℓ_λ) to the verifier.*

Second verifier step (V2): *The verifier selects uniformly a random-tape r for the pcp system, and sends r to the prover.*

Second prover step (P2): *The prover provides the corresponding* (pcp) *answers, augmented by* proofs of consistency *of these answers with the label of the root as provided in Step* (P1).

1. Determining the queries: *Invoking Q_{pcp}, the prover determines the sequence of queries that the pcp system makes on random-tape r. That is, for $i = 1, ..., m$, it computes $q_i = Q_{\text{pcp}}(y, r, i)$, where $m \stackrel{\text{def}}{=}$ poly(n) is the number of queries made by the system.*

2. The message sent: *for $i = 1, ..., m$ and $j = 0, ..., d-1$, the prover sends the pair $(\ell_{\gamma_{i,j}0}, \ell_{\gamma_{i,j}1})$, where $\gamma_{i,j}$ is the j-bit long prefix of q_i.* (Note that this message contains, for every $i \in [m]$, the value of ℓ_{q_i} as well as information that enables the authentication of this value with respect to ℓ_λ.)

Verifier final decision – Step (V3): *The verifier checks that the answers provided by the prover would have been accepted by the pcp-verifier, and that the corresponding proofs of consistency (with the label of the root) are valid. That is, denoting by ℓ'_γ the label provided by the prover for the node associated with γ, the verifier accepts if and only if all the following checks pass:*

1. *Invoking D_{pcp}, the verifier checks whether, on input y and random-tape r, the pcp-verifier would have accepted the answer sequence $\ell'_{q_1}, ..., \ell'_{q_m}$. That is, it checks whether $D_{\text{pcp}}(y, r, \ell'_{q_1} \cdots \ell'_{q_m}) = 1$.*
2. *Check whether the labels provided are consistent with the label of the root of the tree as sent in Step P1. That is, for $i = 1, ..., m$ and $j = 0, ..., d-1$, check whether $\ell'_{\gamma_{i,j}} = h_\alpha(\ell'_{\gamma_{i,j}0}\ell'_{\gamma_{i,j}1})$, where $\gamma_{i,j}$ is the j-bit long prefix of q_i and $\ell'_\lambda \stackrel{\text{def}}{=} \ell_\lambda$.*

We denote the foregoing verifier and prover strategies by V and P, respectively.

We highlight the fact that Construction 3.4 uses a constant number of rounds and is of the public-coin type. Furthermore, Construction 3.4 satisfies the first two requirements of Definition 2.1; that is, the verifier's strategy is implementable in probabilistic polynomial-time, and completeness holds with respect to a prover strategy that (when given $y = (M, x, t)$ and w as above) runs in time polynomial in $T_M(x, w)$. We thus focus on establishing the two last requirements of Definition 2.1. In fact, computational soundness follows from the weak proof-of-knowledge property, because if some adversary can convince the verifier to accept y with non-negligible probability then the extractor (given oracle access to that adversary) outputs a valid witness for membership of y in $S_\mathcal{U}$ (which implies that y is indeed in $S_\mathcal{U}$). Thus, it suffices to establish the latter.

3.4 Establishing the weak proof-of-knowledge property

This subsection contains the main technical contribution of the current section. The novel aspect in the analysis is the use of a "local definition of a conflict" (i.e., considering conflicting values for individual oracle-bit rather than conflicting values for the entire oracle), and the use of reverse-sampling for deriving (in polynomial-time) hashing-collisions when given a conflict on any bit-position in the oracle.

Lemma 3.5 *Construction 3.4 satisfies the weak proof-of-knowledge property of Definition 2.1, provided that the family $\{h_\alpha\}$ is indeed collision-resistant.*

Combining Lemma 3.5 with the foregoing discussion, we establish Theorem 1.1.

Proof: Fixing any polynomial p, we present a probabilistic polynomial-time knowledge-extractor that extracts witnesses from any feasible prover strategy that makes V accept with probability above the threshold specified by p. Specifically, for any family of (deterministic) polynomial-size circuits representing a possible cheating prover strategy and for all sufficiently long y's, if the prover convinces V to accept y with probability at least $1/p(|y|)$ then, with noticeable probability (i.e., $1/p'(|y|)$), the knowledge-extractor (given oracle access to the strategy) outputs the bits of a corresponding witness.

We fix an arbitrary family, $\{\widetilde{P}_n\}_{n\in\mathsf{N}}$, of (deterministic) polynomial-size circuits representing a possible cheating prover strategy, and a generic n and $y \in \{0,1\}^n$ such that $\Pr[(\widetilde{P}_n, V)(y) = 1] > \varepsilon \stackrel{\text{def}}{=} 1/p(n)$. We consider a few key notions regarding the interaction of $\widetilde{P}_n$ and the designated verifier V on common input y. First we consider notions that refer to a specific interaction (corresponding to a fixed sequence of verifier coins, which consists of a pair of choices (α, r) that the verifier takes in Steps (V1) and (V2), respectively):

- The i^{th} query in such interaction is $q_i = Q_{\text{pcp}}(y, r, i)$, where r is the Step (V2) message.
- The i^{th} answer supplied by (the prover) $\widetilde{P}_n$ is the label (i.e., ℓ'_{q_i}) that the prover has provided (in Step (P2)) for the leaf (associated with) the i^{th} query (i.e., $q_i = Q_{\text{pcp}}(y, r, i)$). The corresponding authentication is the corresponding sequence of pairs $\langle(\ell'_{\gamma_{i,j}0}, \ell'_{\gamma_{i,j}1}) : j = 0, ..., d-1\rangle$, where $\gamma_{i,j}$ is the j-bit long prefix of q_i.

  Note that the various parts of the prover's message in Step (P2) are a function of α and r, and thus the notation ℓ'_γ is actually a shorthand for $\ell'_\gamma(\alpha, r)$.
- The i^{th} answer supplied by $\widetilde{P}_n$ is said to be proper if the corresponding authentication passes the verifier's test (in Step (V3)); that is, $\ell'_{\gamma_{i,j}} = h_\alpha(\ell'_{\gamma_{i,j}0}\ell'_{\gamma_{i,j}1})$ holds, for $j = 0, ..., d-1$ (where $\gamma_{i,j}$ is as above and $\ell'_\lambda \stackrel{\text{def}}{=} \ell_\lambda$).

Next, we consider the probability distribution induced by the verifier's coins. Note that these coins consist of the pair of choices (α, r) that the verifier takes in Steps (V1) and (V2), respectively. Fixing any $\alpha \in \{0,1\}^n$, we consider the conditional probability, denoted $p_{y,\alpha}$, that the verifier accepts y when choosing α is Step (V1). Clearly, for at least a $\varepsilon/2$ fraction of the possible α's it holds that $p_{y,\alpha} \geq \varepsilon/2$. We fix any such α for the rest of the discussion. We now consider notions that refer to the residual probability space induced by a uniformly distributed $r \in \{0,1\}^{\text{poly}(n)}$ (as selected by the verifier in Step (V2)).

- For a query value $q \in \{0,1\}^d$, a query index $i \in [m]$, a possible answer $\sigma \in \{0,1\}$, and a parameter $\delta \in [0,1]$, we say that σ is δ-strong for (i,q) if, conditioned on the i^{th} query being q, the probability that $\widetilde{P}_n$ *properly answers* the i^{th} query with σ is at least δ. That is,

  $$\Pr_r[\ell'_{q_i} = \sigma \text{ is proper} \,|\, q_i = Q(y, r, i)] \;\geq\; \delta\,,$$

  where $\ell'_{q_i} = \ell'_{Q(y,r,i)}(\alpha, r)$ as well as its being proper are determined based on α and r.

  When i and q are understood from the context, we just say that σ is a δ-strong answer.
- We say that a query $q \in \{0,1\}^d$ has δ-conflicting answers if there exist i and j (possibly $i = j$) such that 0 is δ-strong for (i,q) and 1 is δ-strong for (j,q).

We stress that throughout the rest of the analysis we consider a fixed $\alpha \in \{0,1\}^n$ and a uniformly distributed $r \in \{0,1\}^{\text{poly}(n)}$.

Our goal is to show that if $p_{y,\alpha} \geq \varepsilon/2$, then we can extract a witness for y by using $\widetilde{P}_n$. We first show that using $\widetilde{P}_n$, we can reconstruct an adequate PCP-oracle that convinces V_{pcp} (with probability at least $\text{poly}(\varepsilon)$), although he strategy of $\widetilde{P}_n$ need not be consistent with any such oracle. That is, although *a priori* the strategy of $\widetilde{P}_n$ may answer queries in an inconsistent fashion, we shall show that in order to be convincing the answers of $\widetilde{P}_n$ must be "essentially" consistent.

As a preparation to the oracle reconstruction procedure, we first show (in Claim 3.5.1) that answers that are not adequately strong are quite rare (because they are useless for convincing V). Indeed, the oracle reconstruction will be based on adequately strong answers (i.e., answers that appear frequently in interactions). Next, we show (in Claim 3.5.2) that reconstructing the oracle based on strong answers is essentially well-defined, because queries that have conflicting answers (which are both adequately strong) occur rarely in the interaction.

Claim 3.5.1 *The probability that the verifier accepts while receiving only δ-strong answers is at least $p_{y,\alpha} - m\delta$.*

(Recall that m is the number of queries asked by the PCP verifier V_{pcp}.) Thus, picking $\delta = p_{y,\alpha}/2m$, we may focus on the case that all the prover's answers are δ-strong.

Proof: The key observation is that whenever the verifier accepts, all answers are proper. Intuitively, answers that are not δ-strong (i.e., are rarely proper) are unlikely to appear in such interactions. Specifically, we just upper-bound the probability that, for some $i \in [m]$, the answer $\ell'_{Q(y,r,i)}$ is proper but not δ-strong for $(i, Q(y,r,i))$, where r is uniformly distributed. Fixing any i and any possible value of q_i, by definition (of being proper but not δ-strong), the probability that the answer ℓ'_{q_i} is proper but not δ-strong for (i, q_i) is smaller than δ. Averaging over the possible values of q_i (as emerging from $Q(y,\cdot,i)$) and taking a union bound over all $i \in [m]$, the claim follows. $\square$

Claim 3.5.2 *There exist a probabilistic polynomial-time oracle machine that, for any δ, given α and oracle access to $\widetilde{P}_n$, finds collisions with respect to h_α with success probability that polynomially related to δ/m and to the probability that the verifier makes a query that has δ-conflicting answers. That is, let η_α denote the probability that* (after choosing α in Step (V1)) *the verifier makes a query that has δ-conflicting answers. Then, the probability of finding a collision* (i.e., $z' \neq z''$ such that $h_\alpha(z') = h_\alpha(z'')$) *is at least $\eta_\alpha \delta^2/m^3$.*

Thus, on a typical α (or rather on all but a negligible fraction of the α's) and for $\delta > 1/\text{poly}(n)$, the quantity η_α must be negligible, because otherwise we derive a contradiction to the collision-resistant hypothesis of the family $\{h_\alpha\}$. Consequently, for $\delta = p_{y,\alpha}/2m > 1/\text{poly}(n)$, we may focus on the case that the prover's answers are not $(\delta/2)$-conflicting.

Proof: We uniformly select $r \in \{0,1\}^{\text{poly}(n)}$ and $i \in [m]$, hoping that $q_i = Q(y,r,i)$ is δ-conflicting (which is the case with probability at least η_α/m). Uniformly selecting $i', i'' \in [m]$, and invoking the reverse-sampling algorithm S_{pcp} on inputs (y, i', q_i) and (y, i'', q_i), respectively, we obtain uniformly distributed r' and r'' that satisfy $q_i = Q(y, r', i')$ and $q_i = Q(y, r'', i'')$. We now invoke $\widetilde{P}_n$ twice, feeding it with α and r' (resp. α and r'') in the first (resp., second) invocation. Assuming that q_i is δ-conflicting, with probability at least $(\delta/m)^2$, both answers to q_i will be proper but with opposite values. Thus, with probability at least $(\eta_\alpha/m) \cdot (\delta/m)^2$, we have obtained two different *proper* answers to the same query q_i. In such a case, the authentication information corresponding to these two (proper) answers yield a collision under h_α, because of the following considerations:

- Both answers to query q_i are authenticated with respect to the same pair (d, ℓ_λ) that was sent by $\widetilde{P}_n$ in step (P1). Thus, both these different values correspond to the leaf associated with the string $q_i \in \{0,1\}^d$.
- Each of the different values for the same leaf (associated with q_i) is authenticated with respect to the same value of the root (i.e., ℓ_λ). This mean that a collision under h_α must occur somewhere along the path from the leaf to the root. Details follow.

Recall that when invoked with input α and r' (resp., α and r''), the circuit $\widetilde{P}_n$ provides authenticating information corresponding to the leaf q_i. This information takes the form of a sequence of pairs $\langle(\ell'_{\gamma_{i,j}0}, \ell'_{\gamma_{i,j}1}) : j = 0, ..., d-1\rangle$ (resp., $\langle(\ell''_{\gamma_{i,j}0}, \ell''_{\gamma_{i,j}1}) : j = 0, ..., d-1\rangle$), where $\gamma_{i,j}$ denotes the j-bit long prefix of q_i. Note that $\ell'_{q_i} \neq \ell''_{q_i}$ (i.e., the answers to q_i are different) while $\ell'_\lambda = \ell_\lambda = \ell''_\lambda$ (since both sequences refer to the same root value ℓ_λ). Thus, there exists a $j \in \{0, ..., d-1\}$ such that $\ell'_{\gamma_{i,j}} = \ell''_{\gamma_{i,j}}$ and $\ell'_{\gamma_{i,j+1}} \neq \ell''_{\gamma_{i,j+1}}$. It follows that $h_\alpha(\ell'_{\gamma_{i,j}0}\ell'_{\gamma_{i,j}1}) = \ell'_{\gamma_{i,j}} = \ell''_{\gamma_{i,j}} = h_\alpha(\ell''_{\gamma_{i,j}0}\ell''_{\gamma_{i,j}1})$ but $\ell'_{\gamma_{i,j}0}\ell'_{\gamma_{i,j}1} \neq \ell'_{\gamma_{i,j}0}\ell'_{\gamma_{i,j}1}$.

The claim follows. □

Suppose for a moment, that (for $\delta = p_{y,\alpha}/2m$) *all* the prover's answers are δ-strong but none is $(\delta/2)$-conflicting. Then, we can use the prover's answers in order to construct (and not merely claim the existence of) an oracle for the pcp system that makes $V_{\rm pcp}$ accept with probability at least $p_{y,\alpha}/2$. Specifically, let the $q^{\rm th}$ bit of the oracle be σ if and only if there exists an i such that σ is δ-strong for (i, q). This setting of the oracle bits can be decided in probabilistic polynomial-time by using the reverse-sampling algorithm $S_{\rm pcp}$ to generate multiple samples of interactions in which these specific oracle bits are queried. Specifically, to determine the $q^{\rm th}$ bit we generate, for every $i \in [m]$, multiple samples of interactions in which the $i^{\rm th}$ oracle query equals q, and determine the answer by using the gap provided by the hypothesis that for some i there is an answer that is δ-strong for (i, q), whereas (by the non-conflicting hypothesis) for every j the opposite answer is not $(\delta/2)$-strong for (j, q).

Recall that the foregoing outline relies on the simplifying assumption by which *all* the prover's answers are δ-strong but none is $(\delta/2)$-conflicting. In general, some queries may either have no strong answers or be conflicting (although these cases will occur rarely). In such a case, the procedure may fail to recover the corresponding entries in the pcp-oracle, but this will not matter much (because with sufficient high probability the pcp verifier will not query these badly-recovered locations).

The oracle-recovery procedure: We present a probabilistic polynomial-time oracle machine that, on input (y, α) and $q \in \{0,1\}^d$ and oracle access to the prover $\widetilde{P}_n$, outputs a candidate for the $q^{\rm th}$ bit of a pcp-oracle. The procedure operates as follows, where $T \stackrel{\rm def}{=} {\rm poly}(n/\delta)$ and $\delta = \varepsilon/4m$:

1. For $i = 1, ..., m$ and $j = 1, ..., T$, invoke $S_{\rm pcp}$ on input (y, i, q) and obtain $r_{i,j}$.

2. For $i = 1, ..., m$ and $j = 1, ..., T$, invoke $\widetilde{P}_n$ feeding it with α and $r_{i,j}$, and if the $i^{\rm th}$ answer is proper then *record* (i, j) *as supporting this answer value.*

3. If for some $i \in [m]$, there are $(2\delta/3) \cdot T$ records for the form $(i, \cdot)$ supporting the value $\sigma \in \{0,1\}$ then define σ as a *candidate*. That is, σ is a candidate if there exists an i and at least $(2\delta/3) \cdot T$ different j's such that the $i^{\rm th}$ answer of $\widetilde{P}_n(\alpha, r_{i,j})$ is proper and has value σ.

4. If a single value of $\sigma \in \{0,1\}$ is defined as a candidate then set the $q^{\rm th}$ bit accordingly. (Otherwise, do whatever you please.)

We call the query q good if it does not have $(\delta/2)$-conflicting answers and there exists an $i \in [m]$ and a bit value that is δ-strong for (i, q). For a good query, with overwhelmingly high probability, the foregoing procedure will define the latter value as a unique candidate. (The expected number of $(i, \cdot)$-supports for the strong value is at least $\delta \cdot T$, whereas for the opposite value the expected number of $(i', \cdot)$-supports is less than $(\delta/2) \cdot T$, for every i'.) Let use denote the pcp-oracle induced by the above procedure by π.

Claim 3.5.3 *Let $\delta = \varepsilon/4m$ and recall that $p_{y,\alpha} \geq \varepsilon/2$. Suppose that the probability that V makes a query that has $(\delta/2)$-conflicting answers is at most $p_{y,\alpha}/4$. Then, with probability at least $1-2^{-n}$ taken over the reconstruction of π, the probability that ${V_{\rm pcp}}^{\pi}(y)$ accepts is lower bounded by $p_{y,\alpha}/4$.*

Proof: Combining the hypothesis (regarding $(\delta/2)$-conflicting answers) with Claim 3.5.1, we conclude that with probability at least $(p_{y,\alpha} - m\delta) - (p_{y,\alpha}/4) \geq p_{y,\alpha}/4$ the verifier (of the interactive argument) accepts while making only good queries and receiving only δ-strong answers. However, in this case, with probability at least $1 - 2^{-n}$, the corresponding bits of π will be set to equal the answers provided by $\widetilde{P}_n$ (because, for a good query, with overwhelmingly high probability, the answer that is δ-strong will be the only candidate found by the oracle-recovery procedure). Since the (interactive argument) verifier is accepting after invoking $V_{\rm pcp}$ on the answers it obtained, it follows that in this case ${V_{\rm pcp}}^{\pi}$ accepts too. □

Recall that by Claim 3.5.2 (and the hypothesis that the family of hashing functions is indeed collision-resistant), it holds that with probability at least $(\varepsilon/2) - (\varepsilon/4)$ over the random choices of α the hypotheses of Claim 3.5.3 hold (i.e., $p_{y,\alpha} \geq \varepsilon/2$ and the probability that V makes a query that has $(\varepsilon/8m)$-conflicting answers is at most $p_{y,\alpha}/4$). We call such an α useful. Applying Claim 3.5.3, it follows that for any useful α, with probability at least $1 - 2^{-n}$, the foregoing oracle-recovery procedure defines an oracle π such that ${V_{\rm pcp}}^{\pi}(y)$ accepts with probability at least $\varepsilon/8$.

The weak proof-of-knowledge property (of the interactive argument) now follows from the corresponding property of the pcp system. Specifically, we combine the pcp extractor with the foregoing oracle-recovery procedure, and obtain the desired extractor (detailed next).

Extractor for the argument system: On input (y,i), where $y = (M,x,t)$ and $i \in [t]$, and access to a prover strategy $\widetilde{P}_n$, the extractor operates as follows (using $\delta = \varepsilon/4m$):

1. Uniformly select $\alpha \in \{0,1\}^n$, hoping that α is useful (which hold with probability at least $\varepsilon/4$). Fix α for the rest of the discussion.

2. Uniformly select coins ω for the oracle-recovery procedure, and fix ω for the rest of the discussion. Note that the oracle-recovery procedure (implicitly) provides oracle access to a pcp-oracle π, which is determined by ($\widetilde{P}_n$, y, α, and) ω.

   We call ω α-good if the foregoing oracle-recovery procedure defines an oracle π such that ${V_{\rm pcp}}^{\pi}(y)$ accepts with probability at least $\varepsilon/8$. Recall that, by Claim 3.5.3, if α is good then, with probability at least $1 - 2^{-n}$, a random ω is α-good.

3. Invoke $E_{\rm pcp}(y,i)$ providing it with oracle access to π. This means that each time $E_{\rm pcp}(y,i)$ makes a query q, we invoke the oracle-recovery procedure on input (y,q) (and with α and ω as fixed above), and obtain the $q^{\rm th}$ bit of π, which we return as answer to $E_{\rm pcp}(y,i)$. When $E_{\rm pcp}(y,i)$ provides an answer (supposedly the $i^{\rm th}$ bit of a suitable witness w for y), we just output this answer.

Recall that if α is useful and ω is α-useful then with probability at least $2/3$ (over the coins of $E_{\rm pcp}$), the output of $E_{\rm pcp}$ (and thus of our extractor) will be correct (i.e., will yield the desired bit of a fixed witness for y). This follows by the proof-of-knowledge property of the pcp system, which refers to convincing $V_{\rm pcp}$ with probability at least 2^{-n}, while here $V_{\rm pcp}$ is convinced with probability at least $\varepsilon/8 > 2^{-n}$. Using suitable amplification, we can obtain (for each bit in the witness) the correct answer with probability at least $1 - 2^{-2n}$ (over the coins of $E_{\rm pcp}$), pending again on α and ω being useful. Denoting the coins of the amplified $E_{\rm pcp}$ by ρ, we infer that for at least a fraction $1 - t \cdot 2^{-2n} \geq 1 - 2^{-n}$ of the possible ρ's, the amplified $E_{\rm pcp}$ provides correct answers *for each of the possible t bit locations.* We call such ρ's (α, ω)-useful.

Let us denote the foregoing extractor by E. The running-time of E is dominated by the running-time of the oracle-recovery procedure, whereas the latter is dominated by the poly(n/ε) invocations of $\widetilde{P}_n$ (during the oracle-recovery procedure). Using $\varepsilon = 1/p(n)$, it follows that E runs in polynomial-time (specifically, the running time is polynomial in n and $p(n)$). The random choices of E correspond to the above three steps; that is, they consist of α, ω and ρ. Whenever they are all useful (i.e., α is useful, ω is α-useful, and ρ is (α, ω)-useful), the extractor E recovers correctly each of the bits of a suitable witness (for y). The event in the condition (i.e., α, ω and ρ being adequately useful) occurs with probability at least $(\varepsilon/4) \cdot (1 - 2^{-n}) \cdot (1 - 2^{-n}) > \varepsilon/5 = 1/5p(n)$. Letting $p'(n) = 5p(n)$, the lemma follows. ■

4 Application to Zero-Knowledge Arguments

Using Theorem 1.1, we prove Theorem 1.2 in two steps:

1. Using any constant-round public-coin universal-argument, we derive one that is witness-indistinguishable. Here we follow the paradigm of "encrypted interactions" (introduced in [10] and also used in Barak's paper [6]). The construction and its analysis, which appear in Section 4.1, differ from prior versions that appeared (or were outlined) in [8, 7, 17].

2. Using the result of the first step, we modify Barak's main construction [6] of a zero-knowledge argument (for any $S \in \mathcal{NP}$) such that it can be analyzed based on standard collision-resistant hashing. Specifically, rather than using any collision-resistant hashing (for the very first message in his protocol), we use tree-hashing (as in Step (P1) of Construction 3.4) composed with an error-correcting code. The construction and its analysis appear in Section 4.2.

The reasons for the various modifications will be discussed in the corresponding subsections. But before turning to the constructions, we stress that the notion of witness-indistinguishability (which is typically applied to proofs/argument systems for specific sets in $\mathcal{NP}$) needs to be redefined when applied to universal-arguments. Specifically, this property should apply to any "fragment" of $R_{\mathcal{U}}$ that can be recognized in time that is polynomial in the length of the common input (where the polynomial is fixed after the universal-argument system is specified). That is, we refer to the following definition.

Definition 4.1 (witness-indistinguishable universal argument): *A universal-argument system, (P, V), is called* witness-indistinguishable *if, for every polynomial p, every polynomial-size circuit family $\{V_n^*\}_{n \in \mathsf{N}}$, and every three sequences $\langle y_n = (M_n, x_n, t_n) : n \in \mathsf{N}\rangle$, $\langle w_n^1 : n \in \mathsf{N}\rangle$ and $\langle w_n^2 : n \in \mathsf{N}\rangle$ such that $|y_n| = n$, $t_n \leq p(|x_n|)$, and $(y_n, w_n^1), (y_n, w_n^2) \in R_{\mathcal{U}}$, the probability ensembles $\{\langle P(w_n^1), V^*\rangle(y_n)\}_{n \in \mathsf{N}}$ and $\{\langle P(w_n^2), V^*\rangle(y_n)\}_{n \in \mathsf{N}}$ are computationally indistinguishable, where $\langle P(w), V^*\rangle(y)$ denotes the output of V^* when interacting with $P(w)$ on common input y.*

Note the analogy to the discussion at the end of Section 1.2 (regarding "natural applications of universal-arguments").

4.1 Constructing witness-indistinguishable universal-arguments

Our starting point is any constant-round, public-coin universal-argument (for $S_{\mathcal{U}}$), denoted $(P_{\rm ua}, V_{\rm ua})$. For sake of simplicity, we assume (without loss of generality) that, on any n-bit long common input, each message sent by either parties has length $m = \mathrm{poly}(n)$. Using the public-coin clause this means that the protocol proceeds in rounds, where in each round the verifier selects uniformly an m-bit string, and the prover responds with an m-bit string determined based on its inputs and the messages it has received so far. We denote by c the (constant) number of such rounds; in case of Construction 3.4, $c = 2$.

A second ingredient used in the construction is a (constant-round, public-coin) *non-oblivious statistically-binding commitment scheme*. Loosely speaking, such a scheme allows a sender to "commit" to a value such that the value remains hidden from the receiver and still the sender is "committed" to this value. Furthermore, "statistically-binding" means that, with high probability, if the commitment phase is concluded successfully then there exists at most one value that can be later revealed as a proper decommitment, whereas "non-oblivious" means that the sender actually knows a proper decommitment of the committed value. For further discussion see Appendix B, where we also show that such a scheme can be constructed based on the existence of one-way functions. We denote this commitment scheme by $\mathtt{C}$, and the corresponding decommitment verification by $\mathtt{D}$. Furthermore, we denote by $(s, c) \leftarrow \mathtt{C}(v)$ an execution of $\mathtt{C}$ in which the sender enters the value v, the receiver obtains the commitment value c, and the sender obtains corresponding decommitment information d satisfying $\mathtt{D}((v, d), c) = 1$. Recall that statistically-binding means that, with overwhelmingly high probability, the protocol yields a commitment c such that if for any v, d, v', d' it holds that $\mathtt{D}((v, d), c) = 1$ and $\mathtt{D}((v', d'), c) = 1$ then it must be that $v = v'$.

A third (and last) ingredient used in the construction is a (constant-round, public-coin) zero-knowledge proof *of constant soundness error* for some NP-complete set. Such proof system can be constructed based on the existence of one-way functions (see, e.g., [16, Chap. 4, Exec. 20]).[12] Let us denote such a system by $(P_{\rm zk}, V_{\rm zk})$.

The construction presented next is based on an "encrypted" emulation of the execution of the universal-argument system $(P_{\rm ua}, V_{\rm ua})$. That is, the prover in our protocol responds to the verifier's messages by sending commitments to the messages that $P_{\rm ua}$ would have sent. This "encrypted" form of the prover's message does not impair the new verifier which merely emulates $V_{\rm ua}$, because $V_{\rm ua}$ is of the public-coin type. At the end of the emulation phase, the prover provides a zero-knowledge proof that the committed values correspond to a transcript that $V_{\rm ua}$ would have accepted. The zero-knowledge property of the foregoing protocol is quite intuitive, and so we focus on its weak proof-of-knowledge property. Indeed, the fact that the commitment scheme is non-oblivious is used for extracting the corresponding transcript, but the acceptability of this transcript is only guaranteed when the new prover convinces the new verifier with constant probability (which originates in the constant soundness error of the zero-knowledge proof). To handle any non-negligible acceptance probability, we repeat the foregoing protocol for a super-logarithmic number of times, where these repetitions are performed in parallel (so to maintain a constant number of rounds). These repetitions do not necessarily preserve the zero-knowledge property of the basic protocol, but they preserve its witness indistinguishability property. The resulting protocol is described next.

Construction 4.2 (a witness-indistinguishable universal-argument):

Common input: $y = (M, x, t)$*, supposedly in* $S_{\mathcal{U}}$*. Let* $n \stackrel{\rm def}{=} |y|$*.*

Auxiliary input to the prover: w *such that supposedly* $(y, w) \in R_{\mathcal{U}}$ *holds.*

[12]This construct replaces the *strong witness-indistinguishable* proof system *of negligible soundness error* assumed in [8]. Unfortunately, in contrary to prior misconceptions (cf. [16, Sec. 4.6]), such protocols are not known to exist [17, Apdx. C.3]. In fact, in our construction, we may use a *strong witness-indistinguishable* proof system *of constant soundness error* (for some NP-complete set), but we do not know of such a protocol that is not zero-knowledge as well.

Part 1: encrypted emulations of $(P_{\text{ua}}, V_{\text{ua}})$. *The parties perform n parallel emulations of the $(P_{\text{ua}}, V_{\text{ua}})$ protocol, where each emulation is performed in a partially encrypted manner. Specifically, the verifier generates random messages exactly as V_{ua}, but the prover answers with commitments to the corresponding responses of P_{ua}. That is, for $i = 1, ..., c$, the parties emulate in parallel n copies of the i^{th} round of $(P_{\text{ua}}, V_{\text{ua}})$ as follows:*

1. *The verifier selects uniformly at random $r_i^1, ..., r_i^n \in \{0,1\}^m$, and sends these strings to the prover.*
2. *The prover determines P_{ua}'s answers, and responds with commitment to them. That is, for $j = 1, ..., n$, the prover first determines $a_i^j \leftarrow P_{\text{ua}}(y, w; r_1^j, ..., r_i^j)$. Next, the parties invoke n parallel executions of $\mathtt{C}$, where the prover plays the sender and the verifier plays the receiver, such that the j^{th} copy yields the output pair $(s_i^j, e_i^j) \leftarrow \mathtt{C}(a_i^j)$. If the verifier detects improper termination in any of these executions then it halts and rejects.*

Part 2: proving that V_{ua} accepts in the encrypted emulations. *The parties invoke the proof system $(P_{\text{zk}}, V_{\text{zk}})$, where the prover's goal is proving that the transcripts $(r_1^j, e_1^j, ..., r_c^j, e_c^j)$ generated in Part 1 corresponds to encryptions of accepting $(P_{\text{ua}}, V_{\text{ua}})$ transcripts. That is, the parties run n parallel copies of $(P_{\text{zk}}, V_{\text{zk}})$ such that, in the j^{th} copy, the NP-statement being proved refers to the input $(y, r_1, e_1, ..., r_c, e_c)$ and asserts that there exists $((a_1^j, s_1^j), ..., (a_c^j, s_c^j))$ such that*

1. *for $i = 1, ..., c$, it holds that $\mathtt{D}((a_i^j, s_i^j), e_i^j) = 1$.*
2. *$V_{\text{ua}}(y; r_1^j, a_1^j, ..., r_c^j, a_c^j) = 1$.*

Needless to say, the prover executes this copy of P_{zk} using the NP-witness $((a_1^j, s_1^j), ..., (a_c^j, s_c^j))$, where this sequence of pairs is as determined by it in Part 1.

Note that the length of the NP-statement being proven (as well as the length of the corresponding NP-witness) is bounded by a fixed polynomial in $n + m$ (and thus by a fixed polynomial in n).

We denote the above verifier and prover strategies by V and P, respectively.

Clearly, Construction 4.2 is constant-round, public-coin, and satisfies the first two requirements of Definition 2.1; that is, the verifier's strategy is implementable in probabilistic polynomial-time, and completeness holds with respect to a prover strategy that (given $y = (M, x, t)$ and w as above) runs in time polynomial in $T_M(x, w)$. To establish that Construction 4.2 is a witness-indistinguishable universal-argument (as per Definition 4.1), it remains to prove the following two properties of Construction 4.2:

1. The weak proof-of-knowledge property, which in turn implies also the computational soundness property.
2. The witness-indistinguishability property.

We mention that (unlike in [8, Lem. 4.2]) the following proofs do not take advantage of the fact that the basic ingredients are constant-round protocols. We start with the witness-indistinguishability property, because its proof is significantly simpler.

Lemma 4.3 *Construction 4.2 is witness-indistinguishable* (as per Definition 4.1), *provided that $(P_{\text{zk}}, V_{\text{zk}})$ is zero-knowledge and that $\mathtt{C}$ is computationally-hiding.*

Proof: We view Construction 4.2 as the result of n parallel executions of a basic protocol, which consists of performing a single "encrypted" execution of $(P_{\text{ua}}, V_{\text{ua}})$ followed by a zero-knowledge proof that the corresponding "encrypted" transcript encodes an accepting transcript of $(P_{\text{ua}}, V_{\text{ua}})$. Intuitively, this basic protocol is zero-knowledge, because it can be simulated by generating dummy commitments and invoking the simulator of $(P_{\text{zk}}, V_{\text{zk}})$ on these commitments. The computational indistinguishability of this simulation from the real execution is argued as follows:

- As a mental experiment, we consider a hybrid distribution in which the simulator of $(P_{\text{zk}}, V_{\text{zk}})$ is invoked on an "encrypted" transcript of $(P_{\text{ua}}, V_{\text{ua}})$.
- The computationally-hiding property of the commitment scheme implies that the output of our simulator (which invokes the simulator of $(P_{\text{zk}}, V_{\text{zk}})$ on dummy commitments) is computationally indistinguishable from the foregoing hybrid distribution.
- The hybrid distribution is computationally indistinguishable from the real execution of the basic protocol (by our hypothesis regarding the simulator of $(P_{\text{zk}}, V_{\text{zk}})$).

Having established the zero-knowledge property of the basic protocol, we conclude that the basic protocol is witness-indistinguishable (in the sense of Definition 4.1). Furthermore, when confining our attention to common and auxiliary inputs that are admissible as per Definition 4.1, it is the case that P runs in polynomial time (when given adequate auxiliary inputs). Thus, the witness-indistinguishability of P (on such inputs) is preserved under parallel composition (cf., e.g., [16, Sec. 4.6.2])). The lemma follows. ■

Lemma 4.4 *Construction 4.2 satisfies the weak proof-of-knowledge property of Definition 2.1, provided that so does* $(P_{\text{ua}}, V_{\text{ua}})$ *and that* $\mathtt{C}$ *is statistically-binding and non-oblivious.*

Proof: Here we decompose Construction 4.2 in a different way, considering first Part 1 as a whole and then Part 2 as a whole. We start with an overview of the proof. Loosely speaking, the non-oblivious property of the commitment scheme guarantees that we can extract the cleartext version of all encrypted transcripts (of $(P_{\text{ua}}, V_{\text{ua}})$), but at this point it is unclear whether any of these transcripts is accepting. Using the statistically-binding property of the commitment scheme we distinguish the case that some of these transcripts are accepting from the case that none of them is accepting. Using the (constant error) soundness of $(P_{\text{zk}}, V_{\text{zk}})$, we infer that the second case (which refers to multiple failures) happens with negligible probability, and so we may ignore it. Thus, either V detects an improper execution of the commitment scheme or we are able to extract an accepting transcript of $(P_{\text{ua}}, V_{\text{ua}})$. It follows that if V is convinced with some noticeable probability p, then for some $j \in [n]$, with probability at least p/n (or so), the j^{th} transcript is accepting and extractable. At this point the lemma follows by invoking the weak proof-of-knowledge property of $(P_{\text{ua}}, V_{\text{ua}})$. We now turn to the actual proof.

Fixing an arbitrary prover strategy for Construction 4.2 we shall derive a related strategy for the underlying universal-argument system such that the circuit complexity (resp., the success probability) of the resulting strategy will be related to the circuit complexity (resp., the success probability) of the original strategy. Specifically, let us consider an arbitrary family, $\{\widetilde{P}_n\}_{n\in\mathbb{N}}$, of (deterministic) polynomial-size circuits representing a possible prover strategy in the system (P, V). Fixing a generic n and $y \in \{0,1\}^n$, we let $p_y \stackrel{\text{def}}{=} \Pr[(\widetilde{P}_n, V)(y) = 1]$. Using $\widetilde{P}_n$, we shall construct a corresponding prover strategy $\widetilde{P_{\text{ua}}}$ that makes V_{ua} accept y with probability $\Omega((p_y - \mu(n))/n)$. Thus, once we are done constructing $\widetilde{P_{\text{ua}}}$, the weak proof-of-knowledge property of Construction 4.2 will follow from the weak proof-of-knowledge property of the underlying universal-argument system.

We thus turn to constructing $\widetilde{P_{\text{ua}}}$, which starts by selecting uniformly $j \in [n]$ and trying to extract the transcript of the j^{th} encrypted interaction performed by $\widetilde{P}_n$ in Part 1. Specifically, after selecting j, the strategy $\widetilde{P_{\text{ua}}}$ operates in c iterations, where in each iteration it obtains from $\widetilde{P}_n$ a sequence of encrypted message and extracts from it a message that it sends to V_{ua}. That is, for $i = 1, ..., c$:

1. $\widetilde{P_{\text{ua}}}$ obtains from the (real) verifier V_{ua} a (uniformly distributed) string, denoted $r_i \in \{0,1\}^m$.
2. $\widetilde{P_{\text{ua}}}$ selects uniformly $r_i^1, ..., r_i^{j-1}, r_i^{j+1}, ..., r_i^n \in \{0,1\}^m$ and feeds $r_i^1, ..., r_i^{j-1}, r_i, r_i^{j+1}, ..., r_i^n$ to $\widetilde{P}_n$, obtaining a residual sender for the current round of executions of the non-oblivious commitment scheme. Using this residual sender and the knowledge extractor guaranteed for the non-oblivious commitment scheme, $\widetilde{P_{\text{ua}}}$ extracts the cleartext answer that corresponds to the j^{th} copy of the current executions of the non-oblivious commitment scheme. (Note that the non-oblivious/proof-of-knowledge property is preserved when the protocol is executed multiple times in parallel.)
3. $\widetilde{P_{\text{ua}}}$ sends the aforementioned answer to the (real) verifier V_{ua}.

We now turn to the analysis of the size of $\widetilde{P_{\text{ua}}}$ and its success probability. These quantities are determined by the size of $\widetilde{P}_n$ and its success probability, denoted p_y. We may indeed focus on the case that p_y is a noticeable function of $n = |y|$ (i.e., $p_y > 1/\text{poly}(n)$). Recall that $\widetilde{P}_n$ executes $c+1$ sub-protocols, where the first c protocols are n parallel executions of the non-oblivious commitment scheme and the last protocol consists of n parallel executions of the (constant-error) zero-knowledge proof system. We first prove that, with probability at least $p_y/2$, the entire execution produces a transcript such that for every $i \in \{0, 1, ..., c\}$ given the partial transcript of the i previous executions the next execution is successful with probability at least $p_y/2c$. This follows from the following general claim.

Claim 4.4.1 *Let $G \subseteq \Omega^m$ and let $\rho = |S|/|\Omega|^m$. For $i \in [m]$, a sequence $(e_1, ..., e_m) \in \Omega^m$ is called i-good if $\Pr_{e\in\Omega}[(e_1, ..., e_{i-1}, e) \in G_i] \geq \frac{\rho}{2(m-1)}$, where $(e'_1, ..., e'_i) \in G_i$ if and only if there exist $e'_{i+1}, ..., e'_m \in \Omega$ such that $(e'_1, ..., e'_m) \in G$. Then, at least half of the sequences in G are i-good for every $i \in [m]$.*

Needless to say, the foregoing sequences correspond to the executions of the different $c+1$ protocols, and i-good sequences correspond to transcripts in which the given the partial transcript of the $i-1$ previous (successful) executions the next execution is successful with probability at least $p_y/2c$. Actually, we gave-up on execution transcripts that are not fully successful and yet for every $i \in [c+1]$ given the partial transcript of the $i-1$ previous executions the next execution is successful with probability at least $p_y/2c$.

Proof: Let $B_i \stackrel{\text{def}}{=} \{(e_1, ..., e_{i-1}) \in \Omega^{i-1} : \Pr_{e\in\Omega}[(e_1, ..., e_{i-1}, e) \in G_i] < \rho/2(m-1)\}$ be the set of all $(i-1)$-long prefixes of sequences that are not i-good. Note that $B_1 = \emptyset$, because $\Pr_{e\in\Omega}[e \in G_1] \geq \Pr_{\overline{e}\in\Omega^m}[\overline{e} \in G] = \rho$. Now, on one hand, every sequence in G that has no prefix in $\cup_{i=1}^m B_i$, is i-good for every i. On the other hand, the number of sequences in G that have a prefix in B_i is less than $\frac{\rho}{2(m-1)} \cdot |\Omega|^m$, because each such sequence has an i-long prefix $(e_1, ..., e_{i-1}, e_i) \in G_i$ whereas $\Pr_{e\in\Omega}[(e_1, ..., e_{i-1}, e) \in G_i] < \rho/2(m-1)$. The claim follows. □

By Claim 4.4.1, a $p_y/2$ fraction of all executions are fully successful and furthermore each partial transcript of these executions leads to success in the next execution with probability at least $p_y/2c$. Thus, in these executions each of the c rounds of non-oblivious commitments is successful with probability at least $p_y/2c$, and the same holds for the interactive proofs that take place in Part 2. The first fact implies that we can extract all values that were committed to in Part 1, by invoking $\widetilde{P}_n$ for $\mathrm{poly}(n)/(p_y/2c)$ times. Thus, we can bound the size of $\widetilde{P_{\mathrm{ua}}}$ by $\mathrm{poly}(n)/p_y$ times the size of $\widetilde{P}_n$. Combining the second fact, which refers to Part 2, with the constant soundness-error of each execution of the proof system, it follows that at least one of the n committed (and recovered) $(P_{\mathrm{ua}}, V_{\mathrm{ua}})$-transcripts must be accepting (because otherwise the probability that all these executions of the interactive proof are successful is at most $s^n \ll p_y/2c$, where $s < 1$ denotes the constant soundness-error of the proof system). Thus, with probability at least $(p_y/2) - \mu(n) > p_y/3$, the strategy $\widetilde{P_{\mathrm{ua}}}$ recovered all the corresponding $(P_{\mathrm{ua}}, V_{\mathrm{ua}})$-transcripts and at least one of these transcripts is accepting. Conditioned on the foregoing, with probability at least $1/n$, the strategy $\widetilde{P_{\mathrm{ua}}}$ selects j such that the j^{th} $(P_{\mathrm{ua}}, V_{\mathrm{ua}})$-transcript is accepting. We conclude that $\widetilde{P_{\mathrm{ua}}}$ convinces V_{ua} with probability at least $p_y/2cn$, and the lemma follows. ■

Remarks: We stress again that Construction 4.2 is different from the corresponding construction presented in [8]. It is also different from the patch described in [7, Apdx. A.4] and the one outlined in [17, Apdx. C.3.3]. An important difference is that the current proof of Lemma 4.4 does not rely on the fact that the protocols employed have a constant number of rounds. We also mention that the main analysis of Construction 4.2, which is provided in the proofs of Lemmas 4.3 and 4.4, proceeds by decomposing the construction in two different ways. A similar strategy is employed in Appendix B (see the proof of Claims B.2.1 and B.2.2).

4.2 Modifying Barak's zero-knowledge argument

Here our starting point is any (constant-round, public-coin) strong witness-indistinguishable universal-argument (for $S_{\mathcal{U}}$), denoted $(P_{\text{wi-ua}}, V_{\text{wi-ua}})$.

A second ingredient used in the construction is a tree-hashing scheme, denoted $\mathtt{TH}$, as used in Construction 3.4. Loosely speaking, such a scheme can be applied to arbitrary long strings and allows for the verification of the value of a particular bit in the string within time polynomial in the hash-value (and possibly polylogarithmic in the length of the string). We stress that the verification does not require presenting the entire string to which hashing was applied (but rather only auxiliary authentication information that is specific to that bit position). Recall that tree-hashing is constructed based on some "basic" hashing function (which maps $2n$-bit strings to n-bit strings), and that conflicting values assigned to any bit position in the tree-hashing yield a collision in the basic hashing. Specifically, when using a basic hashing function indexed by α, and applying the tree-hashing procedure to an m-bit long string z that is placed at the leaves of the ($\log_2 m$-deep) tree, we denote the resulting label of the root by $\mathtt{TH}_\alpha(z)$ and denote the corresponding sequence of m authenticators by $\mathtt{auth}_\alpha(z)$.[13]

In addition, we use a standard statistically-binding commitment scheme, denoted $\mathtt{C}$, and a binary error-correcting code of constant relative distance and polynomial-time encoding algorithm, denoted $\mathtt{ECC}$. (Note that so called "good" codes have this property and are known to exist, but we can actually use even weaker codes than postulated in the foregoing.) As in Construction B.2, we let $\mathtt{C}_s(z)$ denote the receiver's view of the commitment phase when the sender inputs the value z and uses randomness s.

[13] That is, $\mathtt{TH}_\alpha(z) = \ell_\lambda$, where ℓ_i is the i^{th} bit of z, and $\ell_\gamma = h_\alpha(\ell_{\gamma 0}\ell_{\gamma 1})$. Actually, here it is more natural to let $\mathtt{TH}_\alpha(z) = \ell_{0,0}$, where $\ell_{d,i}$ is the i^{th} bit of $z \in \{0,1\}^{2^d}$, and $\ell_{j,i} = h_\alpha(\ell_{j+1,2i}\ell_{j+1,2i+1})$. Similarly, the i^{th} sequence in $\mathtt{auth}_\alpha(z)$ is $(\ell_{d,2\lfloor i/2\rfloor}, \ell_{d,2\lfloor i/2\rfloor+1}, \ell_{d-1,2\lfloor i/4\rfloor}, \ell_{d-1,2\lfloor i/4\rfloor+1}, \ldots, \ell_{1,2\lfloor i/2^d\rfloor}, \ell_{1,2\lfloor i/2^d\rfloor+1})$.

The key idea in our modification of Barak's construction [6] is replacing an arbitrary hashing of strings by the following two-step (hashing) process:

1. Apply the error-correcting code to the input string.
2. Apply the tree-hashing to the resulting codeword.

The advantage of this two-step (hashing) process over standard hashing is that if two different strings are hashed to the same value then we can quickly obtain a collision in the basic hashing function (underlying the tree-hashing). We stress that this collision is found in time that is polynomial in the hash-value (which may be polylogarithmic in the length of the strings being hashed). The reason is that, with (positive) constant probability, a uniformly selected bit-position in the codeword will have different values in the two codewords, and in this case we obtain from the corresponding authentications a collision in the basic hashing. (The foregoing motivational discussion will be clarified by the proof of Lemma 4.6.)

Construction 4.5 (a zero-knowledge argument for $S \in \mathcal{NP}$ (with a corresponding witness relation R_S)):

Common input: *x, supposedly in S. Let $n \stackrel{\text{def}}{=} |x|$.*

Auxiliary input to the prover: *w such that supposedly $(x, w) \in R_S$ holds.*

Part 1: introducing a trapdoor for the simulation. *The prover commits to a dummy value that allows cheating in the case that this value equals a random value that is sent by the verifier after getting the said commitment. This is done as follows:*

1. *The verifier uniformly selects $\alpha \in \{0,1\}^n$* (i.e., a basic hash function)*, and sends it to the prover.*
2. *The prover sends a dummy commitment* (i.e., a commitment to the value 0^{2n})*; that is, it sends $c \stackrel{\text{def}}{=} \mathtt{C}_s(0^{2n})$ to the verifier, where s denotes the sender's randomness in this execution of the commitment scheme.*
3. *The verifier uniformly selects $r \in \{0,1\}^n$, and sends it to the prover.*

Cheating in Part 2 will become possible if and only if r to agree with c in the sense that one may present a circuit Π such that $\Pi(c) = r$ and $c = C_s(|\mathtt{ECC}(\Pi)|, \mathtt{TH}_\alpha(\mathtt{ECC}(\Pi)))$.

Part 2: effectively proving that $x \in S$. *Specifically, the prover will prove that he knows either a witness w for $x \in S$ (i.e., $(x, w) \in R_S$) or a circuit Π such that $\Pi(c) = r$ and $c = C_s(|\mathtt{ECC}(\Pi)|, \mathtt{TH}_\alpha(\mathtt{ECC}(\Pi)))$. This is done as follows:*

Loosely speaking, the parties invoke the proof system $(P_{\text{wi-ua}}, V_{\text{wi-ua}})$ on common input (x, α, c, r), where the prover intends to prove that it knows a tuple (w, m, η, γ, s) such that either $(x, w) \in R_S$ or (η, γ, s) encodes authentication and decommitment information for a circuit Π such that $\Pi(c) = r$, where the encoding clause means that it holds that $\eta = \mathtt{ECC}(\Pi) \in \{0,1\}^m$, $\gamma = \mathtt{auth}_\alpha(\eta)$ and $c = C_s(|\eta|, \mathtt{TH}_\alpha(\eta))$.[14] Actually, the parties reduce the above instance (x, α, c, r) to the triplet $y = (M'_S, (x, \alpha, c, r), 2^n)$, where $y \in \{0,1\}^{\text{poly}(n)}$ is supposedly in $S_{\mathcal{U}}$, and M'_S is such that $M'_S((x, \alpha, c, r), (w, m, \eta, \gamma, s)) \stackrel{\text{def}}{=} 1$ if and only if at least one of the following two conditions holds:

[14]The reason that we include (in the witness) the authentication information (i.e., $\mathtt{auth}(\mathtt{ECC}(\Pi))$) rather than Π itself will become clear in the proof of Lemma 4.6. Furthermore, for simplicity and clarity, we explicitly include in the witness both $\mathtt{ECC}(\Pi)$ and its length.

1. $(x, w) \in R_S$.
2. $m = |\eta|$, $c = \mathtt{C}_s(m, \mathtt{TH}_\alpha(\eta))$, $\gamma = \mathtt{auth}_\alpha(\eta)$, *and* $\Pi(c) = r$, *where* $\Pi \leftarrow \mathtt{ECC}^{-1}(\eta)$ *is a description of a circuit.*

When invoking $P_{\text{wi-ua}}$*, the prover provides it with the witness* $(w, m_0, \eta_0, \gamma_0, s_0)$*, where* $m_0 = n$ *and* $\eta_0 = \gamma_0 = s_0 \stackrel{\text{def}}{=} 0^n$ *are* (short) *dummy values.*

Note that the first condition can be evaluated in (fixed) *polynomial-time* (in $|x|$)*, whereas the complexity of evaluating the second condition is dominated by the running-time of* Π *on input* c*. Furthermore, if* $(x, w) \in R_S$ *then* $(y, (w, m_0, \eta_0, \gamma_0, s_0)) \in R_{\mathcal{U}}$ *and the running-time of* $P_{\text{wi-ua}}$ *on* $(y, (w, m_0, \eta_0, \gamma_0, s_0))$ *is a fixed polynomial in* $|x|$*. We stress that, in any case, the length of the statement being proven is bounded by a fixed polynomial in* $|x|$*.*

We denote the above verifier and prover strategies by V *and* P*, respectively.*

Clearly, Construction 4.5 is constant-round, public-coin, and employs a probabilistic polynomial-time verifier strategy. Furthermore, the designated prover satisfies the completeness property while running in polynomial-time, given x and w as above. Demonstrating that Construction 4.5 is zero-knowledge is done by following the ideas of [6]. We start with a rough sketch of this proof, and then turn to establish the computational-soundness property of Construction 4.5.

Construction 4.5 is zero-knowledge: We present a non-black-box simulator that, given the code of any feasible cheating verifier (represented by a polynomial-size circuit family $\{\widetilde{V}_n\}_{n \in \mathbb{N}}$), simulates the interaction of P with that verifier. Specifically, given $\widetilde{V}_n$, the simulator emulates Part 1 of the protocol, except that it sets $c \leftarrow \mathtt{C}_s(|\mathtt{ECC}(\widetilde{V}_n)|, \mathtt{TH}_\alpha(\mathtt{ECC}(\widetilde{V}_n)))$ instead of $c \leftarrow \mathtt{C}_s(0^{2n})$. Next, the simulator emulates Part 2 of the protocol by using the witness $(w_0, |\mathtt{ECC}(\widetilde{V}_n)|, \mathtt{ECC}(\widetilde{V}_n), \mathtt{auth}_\alpha(\mathtt{ECC}(\widetilde{V}_n)), s)$, where $w_0 = 0^n$ is a (short) dummy value, s was selected by the simulator when emulating Part 1, and $\widetilde{V}_n$ was given to it as (auxiliary) input.

Needless to say, given $\widetilde{V}_n$, the simulator computes $\eta \leftarrow \mathtt{ECC}(\widetilde{V}_n)$ as well as the tree-hash value $\mathtt{TH}_\alpha(\eta)$ and the corresponding sequence of authenticators $\mathtt{auth}_\alpha(\eta)$ in polynomial-time. It follows that, for every polynomial bounding the size of the verifier's strategy (i.e., $\widetilde{V}_n$), the simulator run in poly(n)-time. It is thus left to show that, for every polynomial bounding the size of the verifier's strategy (i.e., $\widetilde{V}_n$), the simulator's output (produced when given $\widetilde{V}_n$ as auxiliary input) is computationally indistinguishable from a real execution (as in Construction 4.5). The proof proceeds as follows:

- As a mental experiment, we consider a hybrid distribution in which Part 1 is performed as in the simulation (i.e., by committing to the tree-hashing of the encoding of $\widetilde{V}_n$) but Part 2 is performed as in the real execution (i.e., by using a witness w to the common input x).

- Combining the computationally-hiding property of the commitment scheme and the fact that P (when given w) runs in poly(n)-time, it follows that the hybrid distribution is computationally indistinguishable from the real execution of the protocol.

- The witness indistinguishability property of $P_{\text{wi-ua}}$ implies that the simulator's output (in which the witness $\widetilde{V}_n$ is used) is computationally indistinguishable from the hybrid distribution (in which the witness w is used). We stress that, when using the witness indistinguishability property of $P_{\text{wi-ua}}$, we refer to witnesses that are verifiable in fixed polynomial (in n) time (because the polynomial bounding the size of $\widetilde{V}_n$ has been fixed for the current discussion).

Thus, the zero-knowledge feature follows.

Construction 4.5 is computational-sound: We show that any feasible cheating strategy for the prover yields a feasible algorithm that form collisions with respect to the basic hashing family $\{h_\alpha : \{0,1\}^{2|\alpha|} \to \{0,1\}^{|\alpha|}\}_{\alpha \in \{0,1\}^*}$. The main idea is using the (weak) proof-of-knowledge property of $V_{\text{wi-ua}}$ in order to *implicitly* reconstruct an error-correcting codeword that encodes (different) valid witnesses that corresponding to different executions of Part 1. We stress that since there is no a-priori polynomial bound on the length of such witnesses, we cannot afford to *explicitly* reconstruct them (as done in [6], where only a contradiction to super-polynomial hardness is derived). However, implicit reconstruction of valid codewords will suffice, because different witnesses will be encoded by codewords that differ on a constant fraction of the bit-locations.

Lemma 4.6 *Construction 4.5 is computationally-sound* (w.r.t S)*, provided that the family* $\{h_\alpha\}$ *is indeed collision-resistant.*

Proof: Suppose towards the contradiction that there exists a feasible prover strategy that fools V with non-negligible probability (to accept inputs not in S). Specifically, let $\{\widetilde{P}_n\}_{n\in\mathsf{N}}$ be such a family, and p be a polynomial such that for infinitely many $x \notin S$ it holds that $p_x \stackrel{\text{def}}{=} \Pr[(\widetilde{P}_n, V)(x) = 1] > 1/p(|x|)$. Let us fix a generic n and $x \in \{0,1\}^n \setminus S$ such that $p_x > 1/p(n)$. For simplicity, we incorporate this x in $\widetilde{P}_n$. Using $\{\widetilde{P}_n\}_{n\in\mathsf{N}}$, we present a family of circuits $\{C_n\}_{n\in\mathsf{N}}$ that try to form collisions. On input $\alpha \in \{0,1\}^n$, the circuit C_n proceeds as follows:

1. Invoking $\widetilde{P}_n$, on input α, the circuit C_n obtains $c \leftarrow \widetilde{P}_n(\alpha)$. This is supposedly a commitment produced by the cheating prover (in the second step of Part 1 of the protocol).

2. Uniformly selecting $r \in \{0,1\}^n$ and feeding it to $\widetilde{P}_n$ yields a random residual prover $\widetilde{P}_n(\alpha, r)$ for Part 2 of the protocol. That is, $\widetilde{P_{\text{wi-ua}}} \stackrel{\text{def}}{=} \widetilde{P}_n(\alpha, r)$ is a prover strategy for the (witness-indistinguishable) universal-argument system $(P_{\text{wi-ua}}, V_{\text{wi-ua}})$.

   We shall focus on the case that $\widetilde{P_{\text{wi-ua}}}$ convinces $V_{\text{wi-ua}}$ to accept (x, α, c, r) with probability at least $p_x/4 > 1/4p(n)$. In this case, the (weak) proof-of-knowledge property guarantees that we can implicitly reconstruct a witness (w, m, η, γ, s) that satisfies the second condition in Part 2 (because $x \notin S$ makes it impossible to satisfy the first condition (i.e., the condition $(x, w) \in R_S$)). Recall that the second condition implies that, for some circuit Π such that $\Pi(c) = r$, it holds that $m = |\eta|$, $c = \mathtt{C}_s(m, \mathtt{TH}_\alpha(\eta))$, and $\gamma = \mathtt{auth}_\alpha(\eta)$. In the following two steps, we shall first use the foregoing (implicit) reconstruction procedure to obtain m, and next use it to obtain the i^{th} bit of η as well as the corresponding (authentication) segment of γ, for a uniformly selected $i \in [m]$.

3. Invoking the knowledge-extractor guaranteed for the system $(P_{\text{wi-ua}}, V_{\text{wi-ua}})$, while providing it with oracle access to $\widetilde{P_{\text{wi-ua}}}$, we reconstruct the values at bit-locations $n+1, ..., 2n$ in the witness (i.e., the integer m that indicates the length of a codeword).

   Recall that the values at bit-locations $2n+1, ..., 2n+m$ are an error-correcting encoding of Π, and the subsequent m strings consist of authentication information for the corresponding bits.

4. The circuit C_n uniformly selects $i \in [m]$. Invoking the knowledge-extractor again with oracle access to $\widetilde{P_{\text{wi-ua}}}$, it reconstructs the value of the i^{th} bit of the codeword as well as the values in the bit-locations that correspond to its authenticator.

5. We repeat Steps 2 and 4 with a new uniformly selected $r' \in \{0,1\}^n$ but with the same value of i as selected in Step 4. That is, analogously to Step 2, we first obtain a corresponding prover

strategy $\widetilde{P'_{\text{wi-ua}}} \stackrel{\text{def}}{=} \widetilde{P}_n(\alpha, r')$ (for the (witness-indistinguishable) universal-argument system $(P_{\text{wi-ua}}, V_{\text{wi-ua}})$). We need not repeat Step 3 because the statistically-binding property of C guarantees the uniqueness of m (obtained in Step 3). Analogously to Step 4, we invoke the knowledge-extractor with oracle access to $\widetilde{P'_{\text{wi-ua}}}$, and obtain the value of the i^{th} bit of this codeword as well as the values in the bit-locations that correspond to its authenticator.

the values in the bit-locations that correspond to the authenticator of the i^{th} bit in the codeword.

Recall that since $x \notin S$, the witness used by $\widetilde{P'_{\text{wi-ua}}}$ must encode a program Π' such that $\Pi'(c) = r'$. Since (with probability $1 - 2^{-n}$ it holds that) $r \neq r'$ (and $\Pi(c) = r$), it must be the case that $\Pi' \neq \Pi$ (whereas $|\mathtt{ECC}(\Pi)| = m = |\mathtt{ECC}(\Pi')|$). We hope that $\mathtt{ECC}(\Pi)$ and $\mathtt{ECC}(\Pi')$ differ on the i^{th} bit (which happens with constant probability), in which case we obtain authenticators to conflicting values (with respect to the same label of the root of the tree (where the uniqueness of this label is due to the statistically-binding property of C)).

6. The circuit C_n examines the authenticators obtained in Steps 4 and 5. If they authenticate conflicting values, then the circuit derives a collision under h_α (as in the proof of Claim 3.5.2).

The foregoing circuit family has polynomial-size (because each C_n is implementable by the same probabilistic polynomial-time oracle machine, which in turn is given access to the polynomial-size $\widetilde{P}_n$). We now turn to analyze the success probability of C_n.

Claim 4.6.1 *Given a uniformly distributed $\alpha \in \{0,1\}^n$, with probability at least $1/\text{poly}(n)$, the circuit C_n outputs a collision with respect to h_α.*

Proof: Recall that $\widetilde{P}_n$ makes V accept x with probability $p_x > 1/p(n)$, where the probability is taken over V's random choices in the two parts of Construction 4.5. Moreover, V's choices in Part 1 are (α, r), where α (resp., r) is uniformly selected in $\{0,1\}^n$.

We call α good if the probability that $\widetilde{P}_n$ makes V accept x, conditioned on V selecting α in the first step of Part 1, is at least $p_x/2$. Clearly, at least a $p_x/2$ fraction of the α's are good, and we focus on any such good α.

Similarly, we say that r is α-good if the probability that $\widetilde{P}_n$ makes V accept x, conditioned on V selecting α and r in Part 1, is at least $p_x/4$. We denote by $G = G_\alpha$ the set of α-good strings, and note that $|G| > (p_x/4) \cdot 2^n$ (since α is good). That is, for every $r \in G$, the residual prover $\widetilde{P}_n(\alpha, r)$ convinces $V_{\text{wi-ua}}$ with probability at least $p_x/4 > 1/4p(n)$, and therefore the knowledge-extractor (given oracle access to $\widetilde{P}_n(\alpha, r)$) implicitly extracts the corresponding witness with probability at least $q_n \stackrel{\text{def}}{=} 1/\text{poly}(n)$, where the polynomial depends on the polynomial p.

Observe that, with probability at least $(p_x/4)^2$, both r and r' selected in Steps 2 and 5 respectively reside in G. Conditioned on this event, we (implicitly) extract both the corresponding witnesses with probability at least q_n^2. We stress that the two witnesses must be of the same length by the virtue of their length being committed to in c (sent by the prover in the second step of Part 1). Finally, with constant probability, these witnesses differ in bit position i (selected in Step 4), in which case C_n succeeds in forming a collision under h_α.

We conclude that, for any good α, the circuit C_n succeeds in forming a collision under h_α with probability at least $(p_x/4)^2 \cdot q_n^2 \cdot \Omega(1) = 1/\text{poly}(n)$. Recalling that at least $p_x/2 = 1/\text{poly}(n)$ of the α's are good, the claim follows. □

Using Claim 4.6.1, we derive a contradiction to the hypothesis that $\{h_\alpha\}$ is collision-resistant. The lemma follows. ■

Achieving bounded concurrent zero-knowledge. We have shown that Construction 4.5 satisfies the first two extra properties asserted in Theorem 1.2. To establish the third extra property, we slightly modify the construction (analogously to the way this is done in [6]). Specifically, for a suitably chosen polynomial ℓ, in Part 1 we select r uniformly in $\{0,1\}^{\ell(n)+n}$ (rather than in $\{0,1\}^n$), and in Part 2 we relax the second condition (or case) such that now we require that there exists a string $z \in \{0,1\}^{\ell(n)}$ such that $\Pi(c,z) = r$ (rather than requiring that $\Pi(c) = r$). The extra string z allows to encode information from n^2 concurrent executions of the protocol (see details in [6]), and so enables the simulator strategy to insert a "trapdoor" to the second step of Part 1 of the current execution (and thus proceed in an "execution-by-execution" manner). When demonstrating computational-soundness, we merely observe that, for a uniformly distributed $r' \in \{0,1\}^{\ell(n)+n}$ (chosen in Step 5), it is unlikely that Π, which is (implicitly) recovered (in Step 4) obliviously of r', will satisfy $\Pi(c,z') = r'$ for some $z' \in \{0,1\}^{\ell(n)}$.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof Verification and Intractability of Approximation Problems. *JACM*, Vol. 45, pages 501–555, 1998. Preliminary version in *33rd FOCS*, 1992.

[2] S. Arora and S. Safra. Probabilistic Checkable Proofs: A New Characterization of NP. *JACM*, Vol. 45, pages 70–122, 1998. Preliminary version in *33rd FOCS*, 1992.

[3] L. Babai. Trading Group Theory for Randomness. In *17th STOC*, pages 421–429, 1985.

[4] L. Babai, L. Fortnow, and C. Lund. Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. *Computational Complexity*, Vol. 1, No. 1, pages 3–40, 1991. Preliminary version in *31st FOCS*, 1990.

[5] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking Computations in Polylogarithmic Time. In *23rd STOC*, pages 21–31, 1991.

[6] B. Barak. How to Go Beyond the Black-Box Simulation Barrier. In *42nd FOCS*, pages 106–115, 2001.

[7] B. Barak. Non-Black-Box Techniques in Cryptography. Ph.D Thesis, Weizmann Institute of Science, 2004.

[8] B. Barak and O. Goldreich. Universal Arguments and their Applications. *ECCC*, TR01-093, 2001. Extended abstract in the *17th IEEE Conference on Computational Complexity*, pages 194–203, 2002.

[9] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In *Crypto'92*, Springer-Verlag LNCS 740, pages 390–420.

[10] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali and P. Rogaway. Everything Provable is Provable in Zero-Knowledge. In *Crypto'88*, Springer-Verlag LNCS 403, pages 37–56, 1990.

[11] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability. In *20th STOC*, pages 113–131, 1988.

[12] G. Brassard, D. Chaum and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *JCSS*, Vol. 37, No. 2, pages 156–189, 1988. Preliminary version by Brassard and Crépeau in *27th FOCS*, 1986.

[13] R. Canetti, O. Goldreich and S. Halevi. The Random Oracle Methodology, Revisited. *JACM*, Vol. 51 (4), pages 557–594, July 2004. Preliminary version in *30th STOC*, 1998.

[14] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating Clique is almost NP-complete. *JACM*, Vol. 43, pages 268–292, 1996. Preliminary version in *32nd FOCS*, 1991.

[15] L. Fortnow, J. Rompel and M. Sipser. On the power of multi-prover interactive protocols. In *Proc. 3rd IEEE Symp. on Structure in Complexity Theory*, pages 156–161, 1988.

[16] O. Goldreich. *Foundation of Cryptography – Basic Tools*. Cambridge University Press, 2001.

[17] O. Goldreich. *Foundation of Cryptography – Basic Applications*. Cambridge University Press, 2004.

[18] O. Goldreich. *Computational Complexity: A Conceptual Perspective* Cambridge University Press, to appear.

[19] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *JACM*, Vol. 38, No. 1, pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.

[20] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SICOMP*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985.

[21] J. Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments. In *24th STOC*, pages 723–732, 1992.

[22] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems. *JACM*, Vol. 39, No. 4, pages 859–868, 1992. Preliminary version in *31st FOCS*, 1990.

[23] S. Micali. Computationally Sound Proofs. *SICOMP*, Vol. 30 (4), pages 1253–1298, 2000. Preliminary version in *35th FOCS*, 1994.

Appendix A: auxiliary properties of popular PCP systems

We claim that the pcp system of Babai *et. al.* [5] satisfies all the auxiliary properties listed in Definition 3.2. As stated in the main text, it is clear that this pcp system is non-adaptive and thus we turn to establish the following remaining properties:[15]

Relatively-efficient oracle-construction: The oracle in this system consists of two parts: (1) a low-degree extension of a multi-variate function that encodes the original witness, and (2) various partial sums of the values of this polynomial. The low-degree polynomial can be constructed in time that is polynomial in the length of the *explicit* description of the aforementioned function, and the same holds with respect to each of the partial sums.

Efficient reverse-sampling: The queries in this pcp system are either evaluations of the aforementioned polynomial at various points or queries regarding some partial sums of such values. Specifically, the queries belong either to the low-degree test or to the sum-check, respectively. The queries of the low-degree test are points along a random line, while the queries of the sum-check are prefixes of a random point. Thus, given the location of the i-th query, it is easy to select uniformly a random tape that is consistent with this query.

A proof-of-knowledge property: The standard analysis of this pcp system actually establishes that if the verifier rejects with small probability then part of the oracle is close to a low-degree polynomial that extends a multi-variate function that encodes a valid witness. Thus, by standard self-correction, we may (probabilistically) recover each bit in this witness in polynomial-time. Repeating the process sufficiently many times (i.e., applying error-reduction to the foregoing recovery procedure), we obtain a polynomial-time oracle machine E such that for every x and π that satisfy $\Pr[V^\pi(x) = 1] > 1/2$ it follows that there exists $w = w_1 \cdots w_t$ such that $(x, w) \in R$ and $\Pr[E^\pi(x, i) = w_i] > 2/3$ holds for every i.

(As stated in the main text, to obtain the desired error of ϵ, we apply straightforward error-reduction, while noting that this process does not affect the oracle and so the resulting (error-reduced) pcp preserves all the auxiliary properties.) This completes the proof of Theorem 3.3.

We comment that the foregoing considerations are not specific to the pcp system of [5], but rather hold also with respect to many other pcp systems.

Appendix B: non-oblivious commitment schemes

We first recall the definition of non-oblivious commitment schemes, following [16, §4.9.2.1] and [17, Sec. C.3.3]. This definition augments the definition of a standard commitment scheme as presented in [16, §4.4.1.1]. The latter definition (i.e., [16, Def. 4.4.1]) refers to a scheme that is computationally-hiding and statistically-binding. Loosely speaking, such a commitment scheme is an efficient *two-phase* two-party protocol through which one party, called the *sender*, can commit itself to a *value* so the following two conflicting requirements are satisfied.

Hiding: At the end of the first phase, the other party, called the *receiver*, does not gain any knowledge of the sender's value. This requirement has to be satisfied even if the receiver tries to cheat. The computational version asserts that the receiver's view of interactions regarding any two values used by the sender are computational indistinguishable.

[15] Needless to say, we assume some familiarity with the work of Babai *et. al.* [5]. In particular, the high-level overview provided in [18, §9.3.2.2] suffices.

Binding: Given the transcript of the interaction in the first phase, there exists at most one value that the receiver may later (i.e., in the second phase) accept as a legal "opening" of the commitment. This requirement has to be satisfied even if the sender tries to cheat. The statistical version asserts that this property holds with overwhelmingly high probability, where the probability is taken solely over the receiver's randomness.

In addition, one should require that the protocol is *viable* in the sense that if both parties follow it then, at the end of the second phase, the receiver gets the value committed to by the sender. Note, however, that it may be the case that the sender cheats and the first phase is completed such that it is infeasible to present any legal "opening" of the commitment. Furthermore, this event may occur without the receiver detecting it. The definition of non-oblivious commitments is intended to address this concern. Moreover, it guarantees that the sender knows such an opening, unless it is detected cheating.

It is indeed time to define formally the notion of a *proper opening* of a commitment, which is also known as proper decommitment. Typically, one considers a *canonical decommitment*, which consist of all randomness used by the sender in the commit phase. In this case, the value v and the sender's randomness s may be considered a proper decommitment to the receiver's view of the commitment phase if it is consistent with that view (i.e., using these values along with the receiver's randomness yields the transcript of messages seen by the receiver). However, a more general notion of proper decommitment may be beneficial. Specifically, we refer to an arbitrary algorithm $\mathtt{D}$ that satisfies the following two conditions:

1. If the commitment phase is performed properly using the sender's value v and the outcome is a pair (s, c), where s is given to the sender and c is given to the receiver, then $\mathtt{D}((v, s), c) = 1$. This means that (v, s) is accepted as a proper decommitment to c.

2. If the receiver follows the commitment phase properly then, with overwhelmingly high probability, the receiver output c is such that there exists at most one value of v such that the set $\{s : \mathtt{D}((v, s), c) = 1\}$ is non-empty.

Needless to say, the aforementioned canonical decommitment gives rise to such an algorithm $\mathtt{D}$. In the sequel, we shall assume (without loss of generality) that the commitment c is contained in the information s given to the sender.

Definition B.1 (non-oblivious commitment schemes)*: A commitment scheme is called* non-oblivious *if the commit phase consists of two sub-phases such that the first sub-phase is an ordinary commitment phase and the second sub-phase is a proof-of-knowledge of the input and decommitment information that is consistent with the receiver's view of the first sub-phase. That is, the commitment scheme (S, R) decomposes into $S = (S_1, S_2)$ and $R = (R_1, R_2)$ such that (S_2, R_2) is a proof of knowledge system* (see [16, Sec. 4.7.1]) *for the relation $\{(c, (v, s)) : \mathtt{D}((v, s), c) = 1\}$ and this proof system is invoked on common input c and prover's auxiliary input (v, s), where (s, c) is the result of applying (S_1, R_1) to the value v.*

Constructing non-oblivious commitment schemes. Our construction follows the outline provided in [17, Sec. C.3.3].[16] The basic idea is augmenting a standard commitment scheme with an adequate zero-knowledge proof-of-knowledge, but the problem is that such a proof-of-knowledge

[16]We comment that the description in [17, Sec. C.3.3] only provides a relaxed version of the notion of non-oblivious commitment scheme. While this relaxed version suffices for our application (as well as for the applications in [17, Sec. C.3.3]), we consider it of interest to satisfy the non-relaxed version.

that is also constant-round and public-coin is not known. As observed in [16, §4.9.2.1], it suffices to use a *strong*-WI proof-of-knowledge, but again (see [17, Sec. C.3]) such a protocol is not known either. We stress that in both cases we have referred to protocols with negligible soundness error, and that corresponding protocols with constant soundness-error do exist. The solution suggested in [17, Sec. C.3.3] is using multiple commitments (via the standard scheme) and applying a *strong*-WI proof-of-knowledge of constant soundness-error to each of the different commitments. In this setting (of statistically independent inputs), the *strong*-WI property is preserved under parallel executions (cf. [17, Lem. C.3.1]). The following construction adapts this suggestion modulo a minor twist, which allows satisfying Definition B.1 (rather than a relaxed form of it).

Construction B.2 (a non-oblivious commitment scheme)*: Let* $\mathtt{C}$ *be a standard statistically-binding commitment scheme, and let* $\mathtt{C}_s(x)$ *denote the receiver's view of the commitment phase when the sender inputs the value* x *and uses randomness* s*. The following description refers to committing to the value* v *under the security parameter* n*, where* $|v| \leq \mathrm{poly}(n)$*.*

First commit sub-phase: *The parties invoke* $\mathtt{C}$*, in parallel, for* $2n-1$ *times. In all invocations the sender enters the value* v*, and in the* i^{th} *invocation the receiver obtains the value* $c_i = \mathtt{C}_{s_i}(v)$*, where* s_i *denotes the sender's randomness.*

These $2n-1$ *invocations yield a standard commitment scheme with a decommitment algorithm* $\mathtt{D}$ *that accepts the value* v *if it is supported by at least* n *proper decommitments with respect to the basic commitment scheme; that is,* $\mathtt{D}((v,\overline{s}'),\overline{c}) = 1$ *if* $\overline{c} = (c_1,...,c_{2n-1})$ *and* $\overline{s}' = (s'_1,...,s'_{2n-1})$ *such that* $|\{i : \mathtt{D}((v,s'_i),c_i)\}| \geq n$*.*

Second commit sub-phase: *The parties perform, in parallel,* $2n-1$ *copies of the following protocol. In the* i^{th} *copy, the sender proves in zero-knowledge that it knows the values* v *and* s_i *that were used in the corresponding copy of the first sub-phase. That is, the sender proves that it knows* (v,s_i) *such that* $c_i = \mathtt{C}_{s_i}(v)$*. Each of these proofs of knowledge has constant soundness error. If the receiver detects cheating in any of these executions then it aborts indicating that the sender is cheating. Otherwise, the receiver accepts the commitment* $(c_1,...,c_{2n-1})$*.*

A proper decommitment to the value v *with respect to a transcript* $\overline{c} = (c_1,...,c_{2n-1})$ *of the first sub-phase consists of a sequence of* $2n-1$ *strings* $(s'_1,...,s'_{2n-1})$ *such that* $|\{i : \mathtt{D}((v,s'_i),c_i)\}| \geq n$*.*

Construction B.2 inherits the statistical-binding property of the standard commitment $\mathtt{C}$ (since a proper decommitment in Construction B.2 requires a strict majority of proper decommitments of $\mathtt{C}$ to the same value). We first show that, although the zero-knowledge property may not be preserved in the parallel executions that take place in the second commit sub-phase, this sub-phase preserves the computational-hiding property of the standard commitment $\mathtt{C}$.

Claim B.2.1 *Construction B.2 is computationally-hiding.*

Proof: We view Construction B.2 as consisting of $2n-1$ parallel executions of a basic protocol in which the sender first commits to v and then proves (in zero-knowledge) that it knows a proper decommitment. This basic protocol is computationally-hiding, because it consists of producing a computationally-hiding commitment and running a zero-knowledge proof regarding this commitment.[17] Thus, it suffices to prove that any computationally-hiding commitment scheme

[17]Indeed, it is instructive to note that the zero-knowledge property implies the *strong*-WI property (see [16, §4.6.1.1]), whereas the latter property preserves the computational indistinguishability of $\mathtt{C}$-commitments (to different values).

preserves this property under parallel executions. Indeed, the proof proceeds by a hybrid argument and is very similar to the proof of Lemma C.3.1 in [17]. □

In order to simplify the proof of the non-oblivious property of Construction B.2, we assume that the proof-of-knowledge employed is such that the verifier tosses a single coin.[18] We note that such protocols (of constant soundness error) exist; see, e.g., [16, Chap.4, Exer. 28.1].

Claim B.2.2 *Construction B.2, when implemented using a proof-of-knowledge in which the verifier tosses a single coin, is non-oblivious.*

Proof: We fix an arbitrary execution of the first (commit) sub-phase, and consider a random execution of the second (commit) sub-phase, which amounts to $2n-1$ parallel executions of the proof-of-knowledge protocol. Fixing an arbitrary (deterministic) sender strategy, the underlying probability space consists of the $2n-1$ random (bit) choices made in the corresponding executions. We denote by p the probability that the receiver is convinced by all these proofs, where here the probability is taken uniformly over all the receiver's $2n-1$ choices. We call the i^{th} copy good if *for each choice of the verifier bit in this copy*, the receiver is convinced (in this copy) with probability at least $p/2n$, where the probability is taken over the receiver's choices in all the other $(2n-1)-1$ copies. The current claim follows by combining two facts:

1. If $p > 2^{-(n-1)}$ then at least n of the indices are good.

   Otherwise, we reach a contraction by upper-bounding the number of points in the probability space that cause the receiver to be convinced. Specifically, let $B \subseteq [2n-1]$ be the set of non-good indices, and let $C \subseteq \{0,1\}^{2n-1}$ denote the sub-space of all points (in the probability space) that cause the receiver to be convinced. Then, for every $i \in B$ there exists a bit σ_i such that the number of convincing points in which the i^{th} bit equals $1-\sigma_i$ is at most $(p/2n)\cdot 2^{2n-2}$. Observing that

   $$C = \{\alpha_1 \cdots \alpha_{2n-1} \in C : \exists i \in B \text{ s.t. } \alpha_i = 1-\sigma_i\} \cup \{\alpha_1 \cdots \alpha_{2n-1} \in C : (\forall i \in B)\ \alpha_i = \sigma_i\},$$

   it follows that $|C| \leq |B|\cdot(p/2n)\cdot 2^{2n-2} + 2^{2n-1-|B|}$. Thus, if $|B| \geq n$ then $p = \frac{|C|}{2^{2n-1}} \leq \frac{p}{2} + 2^{-n}$, which contradicts the hypothesis $p > 2^{-(n-1)}$.

2. If i is good then the corresponding knowledge (i.e., the pair (v, s_i)) can be extracted in $\text{poly}(n)/p$ steps.

   By making $O(n/p)$ trials, we can generate a convincing transcript for an execution of the i^{th} copy for each of the (two) possible choices of the verifier's random bit. This implies extraction (by definition of a proof-of-knowledge).

Thus, we can extract the relevant decommitment information (i.e., v as well as at least n out of the $2n-1$ corresponding s_i's) in time that is inversely proportional to the probability that the receiver is convinced in the second commit sub-phase. The claim follows. □

Recalling that all ingredients used in Construction B.2 can be implemented based on any one-way function, we get.

Theorem B.3 *The existence of one-way functions implies the existence of non-oblivious commitment schemes.*

[18] We believe that the argument can be extended to the general case (or at least to all standard proof-of-knowledge protocols). See related discussion in [9, Apdx. C.2].

We also mention that the properties of non-oblivious commitment schemes are preserved when many copies of the scheme are executed in parallel (or even concurrently under arbitrary scheduling). The preservation of the standard properties of a commitment scheme follows by a hybrid argument (as employed in the proof of Claim B.2.1). The preservation of the non-oblivious (or rather the proof-of-knowledge) property follows by focusing on each individual copy and considering an auxiliary sender that emulates all the other copies. (We stress that we do not claim here a reduction in the soundness error of the proof-of-knowledge property, but rather a preservation of the proof-of-knowledge property at the same level of soundness error.)