PCPs and Instance Compression from a Cryptographic Lens

Liron Bronfman* Ron D. Rothblum*

June 21, 2021

Abstract

Modern cryptography fundamentally relies on the assumption that the adversary trying to break the scheme is computationally bounded. This assumption lets us construct cryptographic protocols and primitives that are known to be impossible otherwise. In this work we explore the effect of bounding the adversary's power in other information theoretic proof-systems and show how to use this assumption to bypass impossibility results.

We first consider the question of constructing *succinct* PCPs. These are PCPs whose length is polynomial only in the length of the original NP witness (in contrast to standard PCPs whose length is proportional to the non-deterministic verification time). Unfortunately, succinct PCPs are known to be impossible to construct under standard complexity assumptions. Assuming the sub-exponential hardness of the learning with errors (LWE) problem, we construct *succinct probabilistically checkable arguments* or PCAs (Zimand 2001, Kalai and Raz 2009), which are PCPs in which soundness is guaranteed against efficiently generated false proofs. Our PCA construction is for every NP relation that can be verified by a small-depth circuit (e.g., SAT, clique, TSP, etc.) and in contrast to prior work is *publicly verifiable* and has *constant* query complexity. Curiously, we also show, as a proof-of-concept, that such publicly-verifiable PCAs can be used to derive *hardness of approximation* results.

Second, we consider the notion of *Instance Compression* (Harnik and Naor, 2006). An instance compression scheme lets one compress, for example, a CNF formula φ on m variables and $n \gg m$ clauses to a new formula φ' with only $\mathrm{poly}(m)$ clauses, so that φ is satisfiable if and only if φ' is satisfiable. Instance compression has been shown to be closely related to succinct PCPs and is similarly highly unlikely to exist. We introduce a *computational analog* of instance compression in which we require that if φ is unsatisfiable then φ' is *effectively* unsatisfiable, in the sense that it is computationally infeasible to find a satisfying assignment for φ' (although such an assignment may exist). Assuming the same sub-exponential LWE assumption, we construct such computational instance compression schemes for every bounded-depth NP relation. As an application, this lets one compress k formulas $\phi_1, \ldots, \phi_k$ into a single short formula ϕ that is *effectively* satisfiable if and only if at least one of the original formulas was satisfiable.

*Technion. Email: {br,rothblum}@cs.technion.ac.il. Supported in part by a Milgrom family grant, by the Israeli Science Foundation (Grants No. 1262/18 and 2137/19), and the Technion Hiroshi Fujiwara cyber security research center and Israel cyber directorate.

Contents

1 Introduction

The question of *instance compression*, introduced by Harnik and Naor [HN10], asks whether it is possible to compresses long instances of an NP relation down to their witness length, which can be significantly shorter, while preserving the "correctness" of the statement. In more detail, an instance compression scheme for an NP relation $\mathcal{R}$ is a polytime computable function f, such that for every instance x, of length n, with corresponding witness length $m \ll n$, it holds that (1) the compressed instance $f(x)$ has length $|f(x)| = \text{poly}(m)$, and (2) x is a true statement (i.e., exists w such that $(x, w) \in \mathcal{R}$) if and only if $f(x)$ is a true statement. The definition can be extended to allow for f to map instances of $\mathcal{R}$ to a different NP relation $\mathcal{R}'$ and to allow for a poly-logarithmic dependence on the instance length n.

For example, consider the relation SAT of all satisfiable Boolean formulas (in conjunctive normal form). An instance compression scheme for SAT can compress a long formula φ over m variables with $n \gg m$ clauses to a new formula φ' with only $\text{poly}(m)$ clauses, such that φ is satisfiable if and only if φ' is satisfiable.[1]

Harnik and Naor as well as followup works [FS11, Dru15, BDFH09], have shown that instance compression scheme can be quite useful. Some of their diverse applications include natural ones such as efficiently storing instances in order to solve them later (see [HN10]), or as a preprocessing step for solving algorithmic problems (see [BDFH09]), as well as more surprising applications such as establishing the existence of a variety of fundamental cryptographic primitives (see [HN10] for details).

Unfortunately, Fortnow and Santhanam [FS11] showed that, under standard complexity assumptions (namely, that the polynomial hierarchy does not collapse), instance compression cannot exist for some NP languages. More specifically, they showed that the existence of instance compression for SAT implies that $\text{NP} \subseteq \text{coNP/poly}$ (and therefore the polynomial hierarchy collapses).

To establish their result, Fortnow and Santhanam showed that instance compression is closely related (and, in a sense, *equivalent*) to the notion of *succinct probabilistically checkable proofs* (succinct PCPs) [KR08, FS11]. Recall that a PCP is a special format for writing proofs that can be verified by only reading a few of the bits from the proof. The PCP theorem [ALM+98] shows that every NP language has a constant-query PCP whose length is polynomial in the non-deterministic verification time of the language.[2] In contrast, a PCP is *succinct* if its length is polynomial only in the *witness length*. For example, the PCP theorem guarantees that SAT on m variables and n clauses has a PCP of length $\text{poly}(n, m)$, whereas a *succinct* PCP would have length roughly $\text{poly}(m)$.

Fortnow and Santhanam showed that the existence of a succinct PCP for SAT implies that SAT also has an instance compression scheme. In particular, this implies that SAT cannot have a succinct PCP, unless the polynomial hierarchy collapses.

This leaves us at the unfortunate state of affairs that (under a widely believed assumption) these two natural objects (i.e., instance compression schemes and succinct PCPs) do not exist. In this work we show how to bypass these negative results by taking a cryptographic perspective and considering *computational analogs* of PCPs and instance compression. We further show that some of the applications of instance compression and succinct PCPs remain valid also for these relaxed

[1] We denote the number of variables by m and the number of clauses by n (rather than vice-versa, as typically done) to better align with the notation for general NP relations, in which the instance length is denoted by n and the witness length by m.

[2] The dependence of the length of the PCP on the non-deterministic verification time has since been improved to *quasi-linear* [BS08, Din07]

notions.

1.1 Our Results

To get around the barriers posed by [FS11], we consider *probabilistically checkable arguments* (PCAs), which are similar to PCPs except that soundness is only guaranteed against proof-strings that are *efficiently generated*. The notion of PCAs was first considered by Zimand [Zim01] in the context of constructing a lightweight version of the PCP theorem and by Kalai and Raz [KR09] with the exact same motivation as in our work: constructing PCPs whose length is proportional to the witness size.

We also introduce *computational instance compression* (CIC), a relaxation of instance compression in which for false statements x it may be the case that the compressed statement $f(x)$ is true, but it should be *computationally infeasible* to find a witness for $f(x)$.

We first discuss our results for PCAs (in Section 1.1.1) and then the results for computational instance compression (in Section 1.1.2).

1.1.1 Probabilistically Checkable Arguments

As noted above, PCAs are a natural computational analog of PCPs. Similarly to a PCP verifier, a PCA verifier is a probabilistic machine that can only read a few bits from the proof, and is required to accept correct proofs (with high probability). However, while a PCP verifier is guaranteed to reject *any* proof for a false statement (with high probability), a PCA verifier is only required to reject false proofs generated by *computationally bounded* malicious provers. In other words, there can exist accepting proofs for false statements, but it should be *intractable* to find them. Since an accepting proof can be non-uniformly hardwired into the adversary, PCAs are defined in the *common reference string* model. In this model, a common reference string (CRS) is sampled before the beginning of the protocol, and is available both to the PCA prover and the verifier. In analogy to PCPs, we say that a PCA is succinct if the proof length is proportional to the witness length.[3]

Kalai and Raz [KR09] constructed succinct PCAs for a large class of NP relations. However, their construction suffers from two significant caveats. First, their PCA is only *privately-verifiable*. That is, the PCA verifier is given a secret trapdoor to the CRS which can be used for verification. We emphasize that for soundness to hold in this model it is imperative that the prover does not know the trapdoor. The fact that the PCA is only privately-verifiable limits its applicability. In particular, only the *designated verifier* that has the trapdoor can verify the proof. In contrast, we consider *publicly-verifiable* PCAs. In other words, the verifier does *not* get a trapdoor to the CRS, and *anyone* can verify the PCA proof, given the corresponding CRS and by making a few queries (see Definition 3.1 for the formal definition). As shall be discussed further below, public-verifiability is crucial for some of the applications of PCAs.

The second major caveat of the [KR09] construction is that the query complexity, while small, is super-constant. More specifically, while typical PCP verifiers only need to make a constant number

[3] It is well known that PCPs, or any other interactive proof for that matter, cannot be *shorter* than the witness length [GH98, GVW02]. In contrast, under extremely strong cryptographic assumptions we *can* construct PCAs (with non-trivial query complexity) that are *shorter* than the witness length. We refer to such PCAs (i.e., whose proof length is shorter than the witness) as *super-succinct*. Since our focus is on standard cryptographic assumptions, we leave the study of super-succinct PCAs for future work. See further discussion at the end of the current subsection.

of queries in order to obtain constant soundness error, the PCA verifier of [KR09] needs to make $\text{poly}(\log n, \lambda)$ queries, where n is the instance length and λ is the computational security parameter.

In this work we resolve both of these caveats. Assuming the sub-exponential hardness of the learning with errors problem (LWE), we construct a *publicly-verifiable* PCA for a rich subclass of NP relations, with *constant* query complexity.

Theorem 1.1 (Informally stated, see Theorem 3.3)**.** *Assume the sub-exponential hardness of* LWE*. Then, for every* NP *relation* $\mathcal{R}$*, for which membership can be decided in logspace uniform* NC*, there exists a succinct constant-query publicly-verifiable* PCA *with constant soundness error.*

We remark that the subclass of NP relations for which we construct PCAs is identical to the subclass considered by Kalai and Raz [KR09]. It should be noted that this class seems quite rich and in particular contains many well-known NP relations, such as SAT, k-Clique, 3Col, etc.[4]

Our proof of Theorem 1.1 follows the approach of [KR09] while replacing a privately-verifiable non-interactive argument that they use (based on [KR08, GKR15]) with a new *publicly-verifiable* non-interactive argument of Jawale et. al [JKKZ20]. Actually making this approach work, while obtaining constant query complexity, turns out to be non-trivial. See Section 1.2 for details.

PCAs and Hardness of Approximation. As an additional contribution, we also show that publicly-verifiable PCAs can be used to obtain *hardness of approximation results*. We find this surprising since, typically, the NP-hardness of approximation problems is *equivalent* to the existence of PCPs.[5]

While we only give a proof-of-concept construction, we believe that this aspect of PCAs could have important implications for hardness of approximation. We note that results establishing hardness of approximation based on cryptography (or more generally, average-case hardness assumptions) have been demonstrated in the past. For example, lattice based cryptography inherently relies on hardness of approximation (of geometric properties of integer lattices), whereas "local cryptography" is tightly related to approximating CSPs (see in particular the recent work of Applebaum [App17] and references therein). Nevertheless, we find the approach for using PCAs for hardness of approximation more generic and hope that it may allow to bypass known limitations of PCPs. See Section 1.3 for details.

Beyond the Witness Length. It is natural to ask if the witness length is indeed a barrier for PCAs. In particular, a so-called "SNARG for NP" (see Section 1.4) achieves length that is *shorter* than the witness, albeit with relatively poor query complexity, as the SNARG verifier has to read its entire proof. We believe that our techniques can possibly also be useful in the context of converting SNARGs to be highly-efficient PCAs but leave the study of such "super-succinct" PCAs to future work.

1.1.2 Computational Instance Compression

In order to bypass the impossibility result of Fortnow and Santhanam for instance compression, we introduce a new computational variant, which we call *computational instance compression* (CIC).

[4]The Cook-Levin theorem implies that every NP language has an NP relation that can be verified by a bounded depth circuit (in fact, a CNF). However, this transformation blows up the witness size. Here we refer to the *natural* NP relations for these well-known problems.

[5]For example, the NP completeness of, e.g., GapSAT, is equivalent to the PCP theorem.

In a nutshell, a CIC scheme also compresses the instance down to the witness length, but we require that if the instance x is false, then it should be *computationally infeasible* (but not necessarily impossible) to find a witness for the compressed instance $f(x)$. It may be useful to contrast CICs with cryptographic hashing. Such hash functions can similarly be used to compress an instance x but the result is a "cryptographic string" that mainly serves as a commitment that can later be opened by someone who knows x. In contrast the result of compression by a CIC is a new instance x' which can be used independently of the original input x, while preserving its *effective correctness*.

In more detail, let $\mathcal{R}$ and $\mathcal{R}'$ be NP-relations, where the instance length of $\mathcal{R}$ is denoted by n and the witness length by m. A *computational instance compression* (CIC) scheme from $\mathcal{R}$ to $\mathcal{R}'$ is an efficiently computable function f such that $|f(x)| = \text{poly}(m, \log n)$ with the following requirements:

1. If x is true then $x' = f(x)$ is true.
2. If x is false, then it is infeasible for a polynomial-size adversary, given $x' = f(x)$, to find a witness w' such that $(x', w') \in \mathcal{R}'$.
3. In contrast, there is a polynomial-time algorithm that given a true statement x, and a corresponding witness w (i.e., $(x, w) \in \mathcal{R}$), finds a witness w' for $x' = f(x)$.

(Our actual definition of CICs is in the common reference string model - that is, all algorithms are further given a common reference string. For simplicity we ignore this aspect here.)

The notion of CIC is indeed suitable for the main application of instance compression: efficient storage of long instances: The compressed output x' is a short instance of $\mathcal{R}'$ which preserves the *effective* correctness of the original input x.

At first glance, it may not be entirely clear why we insist that it is possible to efficiently find a witness for the compressed instance given a witness for the original witness (i.e., requirement 3 above). However, this requirement is indeed crucial for many applications. For example, for efficient storage, if one has a witness for the original instance then the third requirement lets one preserve this witness for the compressed instance. Moreover, if we omit this requirement then there is a trivial construction that satisfies the first two requirements (in a non-interesting way): to compress an instance x, simply ignore x, and output some x' which is a hard instance of $\mathcal{L}'$ (i.e., an instance which is computationally indistinguishable from $x'' \notin \mathcal{L}'$). Such an instance x' is true but a satisfying witness cannot be efficiently found.

In this paper we construct CICs for the same subclass of NP relations as in Theorem 1.1 and under the same assumption - the sub-exponential hardness of LWE.

Theorem 1.2 (Informally stated, see Theorem 3.5)**.** *Assume the sub-exponential hardness of* LWE*. Then, for every* NP *relation* $\mathcal{R}$*, for which membership can be decided in logspace uniform* NC*, there exists a* CIC *scheme from* $\mathcal{R}$ *to* SAT*.*

As explained above, the class of source NP-relations, from which we can compress, includes many natural relations. As for the target relation, the choice of SAT is somewhat arbitrary (but convenient). In particular, we can replace SAT with any NP relation $\mathcal{R}'$ for which (x, w), an instance-witness pair for SAT can be efficiently translated to an instance-witness pair (x', w') for $\mathcal{R}'$, and vice-versa. This includes most natural NP complete problems that we are aware of.

An Application: OR-SAT. Consider the following problem: we are given as input k different formulas $\varphi_1, \ldots, \varphi_k$ and want to generate a new formula φ' such that φ' is satisfiable if and only if at least one of the φ_i's is satisfiable.

This problem, referred to by Harnik and Naor [HN10] as OR-compression, can be viewed as a special case of instance compression where the source relation $\text{OR-SAT} = \left\{\left((\varphi_1, \ldots, \varphi_k), w\right) : \exists i \in [k], (\varphi_i, w) \in \text{SAT}\right\}$ consists of k formulas out of which at least one is satisfiable, and the target relation is simply SAT. In their aforementioned work, Fortnow and Santhanam [FS11] also rule out the existence of OR-instance compression for SAT, assuming the polynomial hierarchy does not collapse.

We bypass this impossibility result and obtain a *computational* OR instance compression scheme. In such a scheme the compressed formula is *effectively* sound if and only if at least one of the original formulas was sound. Indeed, leveraging the fact that OR-SAT has witnesses of length that is independent of k (namely a witness that satisfies at least one of the k formulas), the following corollary is immediate from Theorem 1.2.

Corollary 1.3. *Assume the sub-exponential hardness of* LWE. *Then, there exists a* CIC *scheme from* OR-SAT *to* SAT.

We find Corollary 1.3 particularly appealing. It gives a method to reduce k formulas to a single formula while preserving the *effective* satisfiability.

On AND-compression. It is natural to ask whether a similar statement to Corollary 1.3 holds when replacing OR with AND - namely, the resulting formula should be satisfiable if and only if *all* of the source formulas are satisfiable. This is called AND-instance compression, and has been shown not to exist in the information theoretic setting, assuming the polynomial hierarchy does not collapse [Dru15, Del16].

Unfortunately, since the witnesses in the case of AND do grow linearly with k, Theorem 1.1 does not seem to imply anything meaningful in this case. We leave the study of AND-computational instance compression for future work (this question seems closely related to that of *non-interactive batch verification*, see Section 1.4).

1.2 Technical Overview

The main technical contribution of this work is a construction of a publicly-verifiable constant-query succinct PCA. We give an overview of this construction in Section 1.2.1. Then, in Section 1.2.2, building on techniques from [FS11], we show how to use our PCA construction to build CICs.

1.2.1 Publicly-Verifiable PCAs from Publicly-Verifiable PSNARGs

We construct our publicly-verifiable PCAs by using a recent publicly-verifiable Succinct Non-interactive ARGument (SNARG) constructed by Jawale et al. [JKKZ20]. A SNARG, as its name suggests, is a non-interactive argument-system (in the common reference string model) in which the verifier runs in time that is sublinear in the computation.[6]

[6] Actually somewhat different definitions of the notion exist in the literature. Some works require only short communication but allow a long verification time, and other works insist on strictly poly-logarithmic verification time.

Usually, SNARGs are considered in the context of NP languages, and the succinctness requirement implies, in particular, that the proof string be shorter than the length of the NP witness. In contrast, we will focus on non-interactive arguments for languages in P. To avoid confusion we refer to such arguments as PSNARGs. Note that since we insist on sublinear verification, it is (highly) non-trivial to construct PSNARGs (even for problems in P).

Moreover, in the following we will sometimes refer to PSNARGs for an NP relation $\mathcal{R}$. What we mean by this is that we view the membership problem in $\mathcal{R}$ (i.e., given an instance *and* a witness - check if they satisfy the relation) as a problem in P. Therefore, the key difference between a SNARG or a PSNARG for an NP relation $\mathcal{R}$ is that in the latter the verifier gets access to the instance *and* the witness, whereas in the former it gets only the instance.

The starting point for our construction is a recent publicly-verifiable PSNARG for logspace-uniform NC due to Jawale et. al [JKKZ20], based on a secure application of Fiat-Shamir transform [FS86] to the [GKR15] doubly-efficient interactive proof-system. This result relies on the sub-exponential hardness of LWE (via [CCH+19, PS19]).

With this result in hand, consider an NP relation $\mathcal{R}$ for which membership can be decided in logspace-uniform NC. Given an instance x of $\mathcal{R}$, consider a proof string that consists of the witness w for x appended with a PSNARG proof π showing that $(x, w) \in \mathcal{R}$.

The proof string (w, π) is very short but it is still not locally checkable. Thus, a natural idea is for the prover to further append a standard PCP proof π' attesting that (w, π) are such that π is an accepting PSNARG proof for $(x, w) \in \mathcal{R}$. Intuitively we seem to have made progress since the PCP proof length corresponds to the verification time, which is short due to the efficient PSNARG verification.

Before proceeding, a small correction is in order. Since a PCP verifier needs to read its entire main input (which in our case is all of w and π) what we will actually use is a PCP *of proximity* (PCPP). In the current context it suffices to think of a PCP of proximity as a PCP in which the verifier only needs to read a few bits from the input (as well as the proof), as long as the input is encoded under an error-correcting code.

Thus, as our first attempt, consider a PCA proof-string that consists of $(E(w, \pi), \pi')$, where E is a suitable error-correcting code (i.e., with polynomial or even linear block length). The PCA verifier emulates the PCPP verifier while using $(E(x), E(w, \pi))$ as the input oracle and π' as the proof oracle (note that the verifier can generate $E(x)$ by itself).

While natural, this construction runs into several problems that we have to resolve and are discussed next.

Adaptivity. The first problem we encounter with the construction is that of adaptivity. Namely, a malicious PCA prover can choose a witness w^*, which serves as part of the input for the PSNARG, *after* seeing the sampled CRS. In particular, the PSNARG of [JKKZ20], is non-adaptive: soundness only holds if the input is fixed *before* the CRS is given to the prover.

Thankfully, there is a relatively standard and simple solution to this problem via complexity leveraging. Namely, we use complexity leveraging to transform sub-exponentially secure PSNARGs to ones which are adaptively sound in the choice of the witness. In a nutshell this is done by increasing the security parameter as to ensure that the soundness error of the PSNARG is so small that we can afford to take a union bound over all possible witnesses. Crucially, the overhead incurred by this transformation is a fixed polynomial only in the witness length m (rather than the instance size n).

Having resolved the adaptivity issue, we turn to a more pressing problem that arises when examining the PCA proof length more closely.

The PCA Proof Length. The proof-string consists of (an encoding of) w which has length m, and π which has length $\text{poly}(\log(n), \lambda)$, where λ is the security parameter. In addition, the PCA contains π', a PCPP proof that the PSNARG verifier would accept. While the PSNARG verifier runs in time that is sublinear in the verification time of $\mathcal{R}$, at very least the verifier needs to read its input (x, w) which has length $n + m$. This means that the length of the PCPP proof string π' is at least $\text{poly}(n)$, whereas we were aiming for a polynomial dependence only on m.

Holographic Proofs to the Rescue? As a first idea one might try to get around this by leveraging the fact that the [JKKZ20] verifier is *holographic*. A PSNARG is said to be *holographic* if the verifier can run in time that it is *sublinear* in the input, as long as it is given oracle access to an *encoding* of the input. The verifier in the protocol [JKKZ20] is indeed holographic, with respect to the low degree extension code.[7] Indeed, given oracle access to the low degree extension of the input, the PSNARG verifier of [JKKZ20] runs in time $\text{polylog}(n)$.

One would hope that this observation suffices to reduce the PCPP length to $\text{poly}(m, \log(n))$. Unfortunately, this is not the case. The difficulty that we encounter now is that PCPPs are not designed for computations involving an oracle. In particular, it is known that the PCP theorem does not relativize [AIV92, For94] and it is unclear how to construct a PCPP in which the input is given as an oracle (without materializing the input).

Sublinear Verification via Crypto. To cope with this difficulty we take, yet again, a cryptographic approach. In particular, we propose a simple PSNARG construction in which the PSNARG verifier truly runs in sublinear time, after a suitable pre-processing step. Our construction is inspired by, and closely related to, ideas arising in the context of *memory delegation* [CKLR11, KP16].

The sublinear PSNARG is obtained by having the verifier first, as a pre-processing step, generate the low degree extension of its input and hash the low degree extension using a Merkle tree. At this point all that the verifier needs to remember is the root of the hash tree. In the online phase, the prover provides a standard PSNARG proof, but in addition also provides authentication paths for all of the verifier's queries to the input. This results in a pre-processing sublinear PSNARG construction.

We note that the fact that this solution works relies on two critical facts. First, the fact that the PSNARG prover knows, already when sending the proof, which locations of encoding of the input the holographic verifier is going to query. Indeed, this follows from the fact that the PSNARG is publicly-verifiable (and we could not apply similar reasoning, e.g., to the [KR09] construction). Second, we need that it be possible for the verifier to fully materialize the low degree extension of the input in $\text{poly}(n)$ time. This is actually non-trivial since the low degree extension has length that is polynomial in the input only for *some* parameter settings. In particular, using the parameter setting of [JKKZ20] results in *sub-exponential* length (due to their use of extremely large finite

[7] Recall that [JKKZ20] is constructed by applying the Fiat-Shamir transform to the [GKR15] protocol, which is known to be holographic. The holographic property is preserved by the Fiat-Shamir transform because, since [JKKZ20] are only aiming for non-adaptive soundness, they do not include the instance as an input to the Fiat-Shamir hash function.

fields). To solve this we rely on the followup work of Holmgren et al. [HLR21] who extend the [JKKZ20] protocol to work also for small fields and in particular, when the block length of the low degree extension is polynomial.

Using this observation, we can reduce the PSNARG verification time, and therefore the PCPP length to be sublinear. Having done so, the entire PCA proof length becomes polynomial only in the witness length, as desired.

Relation to Fiat-Shamir of Kilian's Protocol. Given that the [JKKZ20, HLR21] protocols rely on a secure application of the Fiat-Shamir transform, and we are applying this transform based on a Merkle hash of the instance, our construction bears some resemblance to a secure application of Fiat-Shamir to Kilian's [Kil92] celebrated protocol. Since the question of securely applying Fiat-Shamir to Kilian's protocol (as envisioned by Micali [Mic00]) in the standard-model is wide open (with mainly negative evidence thusfar [BBH+19]) this may seem surprising. In a nutshell the key difference that we leverage is the fact that the verifier has a *trusted* Merkle hash of the instance. We refer the reader to Appendix A for a more detailed discussion.

1.2.2 CICs from Succinct Publicly-Verifiable PCAs

Fortnow and Santhanam [FS11] show how to convert a succinct PCP into an instance compression scheme for SAT. We follow their approach in order to convert our succinct PCA construction into a CIC for SAT.

Let $\mathcal{V}$ be a PCA verifier for SAT, and φ be an instance (i.e., formula) of SAT. Fixing φ, the verifier $\mathcal{V}$ tosses random coins, reads $q = O(1)$ bits from its proof π and runs some predicate over the answers that it read in order to decide whether to accept or reject.

The compressed formula φ' is generated in the following manner: The variables for φ' correspond to the bits of the PCA string π. To construct the clauses of φ', we enumerate over all possible random strings of $\mathcal{V}$, and for each such random string ρ, we consider the residual decision predicate $\mathcal{V}_{\varphi,\rho}$ of $\mathcal{V}$ when both the random string and input formula φ are fixed. Assuming that the verifier makes non-adaptive queries (this is indeed the case for our PCA construction and is also true without loss of generality whenever the query complexity is constant), each random string is associated with a sequence of q indices from the proof and the predicate $\mathcal{V}_{\varphi,\rho} : \{0,1\}^q \to \{0,1\}$.

Note that each $\mathcal{V}_{\varphi,\rho}$ can be represented as a CNF on 2^q clauses. We let our compressed formula $\varphi' = \bigwedge_\rho \mathcal{V}_{\varphi,\rho}$ simply be the conjunction of all of these clauses. The length of φ' is exponential in the randomness complexity of $\mathcal{V}$. Since our $\mathcal{V}$ has $O(\log m + \log\log n)$ randomness complexity, the output formula has length $|\varphi'| = \mathrm{poly}(m, \log n)$ as desired.[8]

For $\varphi \in \mathrm{SAT}$, by the completeness of the PCA, there exists a proof π satisfying all predicates, meaning $(\varphi', \pi) \in \mathcal{R}_{\mathrm{SAT}}$. Thus, φ' is satisfiable (as per the first CIC requirement), and using the witness w for φ, it is possible to efficiently find $w' = \pi$ a witness for φ' (as per the third CIC requirement). For $\varphi \notin \mathrm{SAT}$, by the soundness of the PCA, it is computationally infeasible to find a proof string π^* satisfying over half of the predicates $\{\mathcal{V}_{\varphi,\rho}\}$. This means that it is intractable to find a satisfying assignment for the compressed formula φ', as per the second CIC requirement.

[8] We note that even if our verifier had larger randomness complexity, it would have still been possible to build a CIC from a succinct-PCA, similarly to [FS11].

1.3 Applications to Hardness of Approximation and Future Directions

PCPs have had an incredible impact on the field of hardness of approximation. The seminal work of Feige et. al [FGL+96] showed how to use PCPs to derive hardness of approximation results, and this result has blossomed into a major technique in the field, leading to optimal inapproximabilty results for many important problems (see, e.g., [Hås01]). This is achieved by constructing suitable PCPs which enable us to show the NP-completeness of gap versions of natural optimization problems.

We show that the weaker notion of a PCA can, in principle, also yield hardness of approximation results. Moving forward, we believe that this observation has significant potential for bypassing limitations of PCPs in the context of hardness of approximation.

The main difference between using PCPs and PCAs to show hardness of approximation is that we will only be able to derive hardness of *search* versions of the gap problems, and the hardness will be based on a Cook reduction rather than a Karp reduction. Nevertheless, these results suffice to demonstrate that the underlying optimization problem is hard to approximate.

As a proof-of-concept we next sketch how to use PCAs to rule out a polynomial-time approximation algorithm for MaxSAT. Namely, an algorithm that, given a formula φ finds an assignment that satisfies *approximately* the maximal number of satisfiable clauses in φ. We emphasize that the conclusion is not surprising since it follows from the PCP theorem that GapSAT (deciding whether a formula is satisfiable or if any assignment satisfies at most a constant fraction of the clauses) is NP-complete. What we find surprising is that the notion of a PCA suffices to establish this result. We note that for this result we do not need *succinct* PCAs, but believe that the fact that such PCAs exist may be useful for establishing new hardness of approximation results, e.g., in the context of fine-grained complexity (see more below). We also emphasize that this result crucially relies on the PCA being *publicly verifiable* - we do not know how to establish any hardness of approximation results based on privately verifiable PCAs (such as those of [KR09]) - see further discussion below.

Given a CNF formula $\varphi : \{0,1\}^m \to \{0,1\}$ over n clauses, and an assignment $z \in \{0,1\}^m$, let $|\varphi(z)|$ denote the number of clauses satisfied by z in φ. Recall that in the MaxSAT problem, the goal is to find an assignment z that maximizes $|\varphi(z)|$. Consider the following approximation problem:

Definition 1.4 (Approximate MaxSAT$_\epsilon$)**.** *In the* Approximate MaxSAT$_\epsilon$ *problem, the input is a* CNF *formula* φ*. Denote* $\alpha = \max_{z\in\{0,1\}^m} \frac{|\varphi(z)|}{n}$ *the maximally satisfiable fraction of clauses of* φ*. The goal is to output an assignment* z*, such that* $\frac{|\varphi(z)|}{n} \geq (1-\epsilon)\cdot\alpha$*.*

Theorem 1.5. *Assume* P $\neq$ NP*. If there exists a publicly-verifiable constant-query* PCA *for* SAT*, then there exists* $\epsilon > 0$ *for which there does not exist a polynomial-time algorithm solving Approximate* MaxSAT$_\epsilon$*.*

Since Theorem 1.5 is only a proof-of-concept, we do not make any attempt to optimize the constant ϵ.

Proof Sketch of Theorem 1.5. Assume for contradiction that there exists $\mathcal{A}$, a polynomial time algorithm solving Approximate MaxSAT$_\epsilon$, for a constant $\epsilon > 0$ specified below. We show a polynomial-time decision procedure for SAT.

Let $\mathcal{V}$ be a constant-query PCA verifier for SAT, and φ an instance of SAT over m variables and n clauses. With φ fixed, and a fixed CRS sampled from the generator, the verifier $\mathcal{V}$ samples a random string ρ and runs a predicate $\mathcal{V}_{\varphi,\rho} : \{0,1\}^q \to \{0,1\}$, over the $q = O(1)$ values it reads from

its proof π. The predicate $\mathcal{V}_{\varphi,\rho}$ can be represented as CNF formula with 2^q clauses. We denote by $\varphi' = \bigwedge_{\rho} \mathcal{V}_{\varphi,\rho}$ the conjunction of all these formulas. Denote by n' and m' the number of clauses and variables in φ', respectively.

If $\varphi \in$ SAT, then by the completeness of the PCA, there exists a proof satisfying every predicate $\{\mathcal{V}_{\varphi,\rho}\}_\rho$. This means that φ' is satisfiable, i.e., has $\alpha = \max_{z\in\{0,1\}^{m'}} \frac{|\varphi'(z)|}{n'} = 1$. Therefore, by the definition of $\mathcal{A}$, for $z = \mathcal{A}(\varphi')$ it holds that $\frac{|\varphi'(z)|}{n'} \geq (1-\epsilon)\cdot\alpha = 1-\epsilon$.

If $\varphi \notin$ SAT, then by the computational soundness of the PCA, no polynomial-time algorithm can find a proof satisfying half of the verifier's predicates $\{\mathcal{V}_{\varphi,\rho}\}_\rho$. For each such rejecting predicate, at least one of its 2^q clauses has to evaluate to False. Therefore, for $\epsilon = \frac{1}{2^{q+1}}$, it holds that no polynomial-time algorithm can find an assignment satisfying a $(1-\epsilon)$-fraction of the clauses in φ'. Consequently, if for some $\varphi \notin$ SAT the algorithm $\mathcal{A}$ outputs $z = \mathcal{A}(\varphi')$ for which $\frac{|\varphi'(z)|}{n'} \geq 1-\epsilon$, then $\mathcal{A}$ breaks the computational soundness of the PCA for instance φ, which is a contradiction.

Therefore, for every $\varphi \notin$ SAT and $z = \mathcal{A}(\varphi')$ it holds that $\frac{|\varphi'(z)|}{n'} < 1-\epsilon$. In this case, it is possible to decide SAT in polynomial time. This is done in the following manner: Given a CNF formula φ, compute φ' as above, run the algorithm $z = \mathcal{A}(\varphi')$, and consider the value $\beta = \frac{|\varphi'(z)|}{n'}$.

- If $\beta \geq 1-\epsilon$, output $\varphi \in$ SAT. As discussed above, if $\varphi \in$ SAT then $\varphi' \in$ SAT, and $\beta \geq 1-\epsilon$. Thus, $\varphi \in$ SAT is accepted.
- If $\beta < 1-\epsilon$, output $\varphi \notin$ SAT. If $\varphi \notin$ SAT and $\mathcal{A}$ doesn't break the PCA, then $\beta < 1-\epsilon$. Thus, $\varphi \notin$ SAT is rejected.

Therefore, it is possible to decide SAT in polynomial time, contradicting the assumption that P $\neq$ NP.

□

We discuss two important points regarding the reduction above: The first, is that unlike the hardness of approximation results using PCPs, the reduction using PCAs is a Cook reduction rather than a Karp reduction. The second point is about the importance of the public-verifiability of the PCA (mentioned briefly above). The reduction above does not work with privately-verifiable PCAs, in which the verification relies on a secret trapdoor τ for the CRS. Specifically, generating the formula φ' is problematic: Hard-wiring τ to φ' prevents it from being a "hard" instance, as breaking the privately-verifiable PCA is trivial given the trapdoor. On the other hand, it is not clear how to compute β when omitting τ from φ', as the PCA proof is only verifiable given its secret key. Thus, public-verifiability seems essential to the reduction above.

Hardness of approximation in Fine-Grained Complexity: A potential use-case. The field of fine-grained complexity aims to understand the precise complexity of fundamental problems that are already known to be solvable in polynomial time: for example, whether one can compute the edit distance of two strings in sub-quadratic time. Negative results here are usually shown by assuming a precise hardness for a core problem, and using so-called "fine-grained reductions" to prove lower bounds for the problem at hand [Wil15].

A recent line of work in this field [ARW17, CGL+19] uses proof systems, such as PCPs and interactive proofs, to show the fine-grained *inapproximability* of problems in P. We believe succinct PCAs may have applications in this field as well. To demonstrate a possible use case, consider the

3SUM problem, in which we are given a list of integers $a_1, ..., a_n$, each over polylog(n) bits, and need to decide whether there exist $i, j, k \in [n]$ such that $a_i + a_j + a_k = 0$. This problem can be solved in time $\tilde{O}(n^2)$, but it is conjectured that there are no substantially faster algorithms.

Observe that while it seems that 3SUM may be hard to solve in time $O(n)$, it is easy to *verify* given an NP witness - namely the indices i, j and k. Applying, e.g., the [BS08, Din07] short PCP yields a constant-query PCP for 3SUM of length $\widetilde{O}(n)$. In contrast, since the NP witness has *logarithmic* length, our succinct PCA for 3SUM (which follows from Theorem 1.1) has *poly-logarithmic* length and constant-query complexity. We believe that this use of PCAs has potential for deriving new inapproximability results in the context of fine-grained complexity.

1.4 Additional Related Works

Succinct Non-interactive Arguments. SNARGs for NP have drawn considerable attention recently (see, e.g., [Tha] for a recent survey). We note that a SNARG for NP, with say poly-logarithmic proof length can be viewed as an (extremely short) PCA with poly-logarithmic query complexity (i.e., the verifier reads the entire proof).

Unfortunately, all known SNARG constructions rely on assumptions that are not-falsifiable [Nao03] (e.g., knowledge-of-exponent or heuristic instantiations of Fiat-Shamir) and this is known to be inherent for blackbox constructions (achieving adaptive soundness) [GW11].

Nevertheless, there has recently been significant progress on constructing SNARGs for languages in P. For example, the PSNARG of [JKKZ20] that we use in our construction. We also note that a publicly-verifiable PSNARGs for *all* of P was constructed by Kalai, Paneth and Yang [KPY19] based on a new, but falsifiable, assumption on groups with bilinear maps. However, since the [KPY20] construction does not appear to be *holographic*, it does not seem to be directly usable in our construction.

In a very recent and exciting independent work, Choudhuri et. al [CJJ21] constructed a PSNARG for all of P, based on the LWE assumption. If their construction can be shown (or modified) to be holographic (using sufficiently small fields) then it could potentially be used to extend our PCA and CIC constructions to all of NP and to relax the assumption to plain LWE rather than sub-exponential LWE.

Batch Verification. Given an NP language $\mathcal{L}$ it is interesting to ask what is the complexity of verifying that k different instances $x_1, \ldots, x_k$ all belong to $\mathcal{L}$. This question seems closely related to the question of AND-compression in which one wants to compress $x_1, \ldots, x_k$ into a short instance x' that is true if and only if $x_1, \ldots, x_k \in \mathcal{L}$. In the statistical setting general purpose batch verification is known for any language in UP (i.e., NP relations in which the YES instances have a unique witness) [RRR16, RRR18, RR20b] and for *statistical zero-knowledge proofs* [KRR+20, KRV21]. In the cryptographic setting, *non-interactive* batch verification is known for language in P [BHK17], albeit in the designated-verifier model. The aforementioned work of [CJJ21] constructs *publicly-verifiable* batch verification for NP. Lastly, we note the related notion of *folding schemes* which allows one to perform a type of AND-compression, by interacting with the prover in order to perform the compression itself [KST21].

Interactive PCPs and IOPs. Kalai and Raz [KR08] introduced the model of *interactive* PCP*s* as a different way to circumvent the impossibility result for succinct PCPs. In an interactive PCP, the

prover first sends a PCP proof string but afterwards the prover and verifier engage in a standard (public-coin) interactive proof in order to verify the PCP. This model can also be seen as a special case of the more recent notion of *interactive oracle proof* (IOP) [BCS16, RRR16]. Kalai and Raz constructed interactive PCPs whose length is polynomial in the witness length. More recently, Ron-Zewi and Rothblum [RR20a] constructed interactive PCPs whose length is only *additively* longer than the witness length.

1.5 Organization

We start with preliminaries in Section 2. Then, in Section 3 we describe our main results. Our succinct PCA construction is given in Section 4 and the CIC construction (and equivalence to succinct PCAs) is shown in Section 5.

2 Preliminaries

A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is *negligible* if for every $c \in \mathbb{N}$ it holds that $\epsilon(\lambda) = O(\lambda^{-c})$. The *relative distance* between strings $x, y \in \Sigma^\ell$ over alphabet Σ is $\Delta(x,y) = \frac{|\{i : x_i \neq y_i\}|}{\ell}$. The *relative distance* between a string $x \in \Sigma^\ell$ and a non-empty set $S \subseteq \Sigma^\ell$ is $\Delta(x,S) = \min_{y \in S}\big(\Delta(x,y)\big)$.

2.1 Proof Systems

2.1.1 Argument Systems

We start by defining argument systems. We restrict our attention to *non-interactive* arguments for languages in P.

Definition 2.1 (Non-interactive argument system)**.** *A* non-interactive argument system *for $\mathcal{L} \in \mathsf{P}$, with soundness error $s : \mathbb{N} \to [0,1]$ and adversary size $T : \mathbb{N} \to \mathbb{N}$, is a triplet of probabilistic polynomial time algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ such that:*

1. ***Completeness:*** *For every $x \in \mathcal{L}$ and every security parameter λ, it holds that*

$$\Pr_{\mathsf{CRS} \leftarrow \mathcal{G}(1^{|x|}, 1^\lambda)} \big[\mathcal{V}(x, \mathsf{CRS}, \pi) = 1\big] = 1,$$

   for $\pi = \mathcal{P}(x, \mathsf{CRS})$.

2. ***Computational soundness:*** *For every $x \notin \mathcal{L}$, every security parameter λ, and every malicious prover $\mathcal{P}^*$ of size $\leq T(\lambda)$, it holds that*

$$\Pr_{\mathsf{CRS} \leftarrow \mathcal{G}(1^{|x|}, 1^\lambda)} \big[\mathcal{V}(x, \mathsf{CRS}, \pi^*) = 1\big] \leq s(\lambda),$$

   for $\pi^* = \mathcal{P}^*(x, \mathsf{CRS})$.

The string generated by $\mathcal{G}$ is called a *common reference string* (CRS). The message π sent by $\mathcal{P}$ called the *proof*.

An argument system is said to be *public-coin* if the CRS generated by $\mathcal{G}$ is simply a common *random* string, and $\mathcal{V}$ is deterministic.

By default, unless specified otherwise, an argument system is secure against adversaries of size $T = \text{poly}(\lambda)$, with $s = \mathsf{negl}(\lambda)$.

We now define a stronger notion of soundness for argument systems, in which the malicious prover gets to choose the input for the verifier *after* the CRS has been sampled.

Definition 2.2 (Adaptive computational soundness)**.** *A non-interactive argument $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ for language $\mathcal{L} \in \mathcal{P}$ is said to be* (T, s)-adaptively sound *if for every instance length n, security parameter λ, and every malicious prover $\mathcal{P}^*$ of size $\leq T(\lambda)$, it holds that*

$$\Pr_{\mathsf{CRS} \leftarrow \mathcal{G}(n, 1^\lambda)} \left[\mathcal{V}(\mathsf{CRS}, x^*, \pi^*) = 1 \ \wedge \ x^* \notin \mathcal{L} \right] \leq s(\lambda)$$

for $(x^*, \pi^*) = \mathcal{P}^*(\mathsf{CRS})$, where x^* is of length n.

2.1.2 Holographic Argument Systems

We now define *holographic* argument systems. In such a system, the verifier can run in sublinear time as long as it can make a few queries to the *low degree extension* of its input.

Let $\mathbb{F}$ be a field, $\mathbb{H}$ a subfield of $\mathbb{F}$ of size $|\mathbb{H}| \geq \log(|\mathbb{F}|)$, and $\ell \in \mathbb{N}$. Let $x : \mathbb{H}^\ell \to \mathbb{F}$. The *low degree extension* of x, denoted $\mathsf{LDE}_{\mathbb{F},\mathbb{H},\ell}(x)$, is the unique $(|\mathbb{H}| - 1)$-individual degree polynomial $\hat{x} : \mathbb{F}^\ell \to \mathbb{F}$ such that $\hat{x} \mid_{\mathbb{H}^\ell} \equiv x$. For $t = (t_1, ..., t_\ell) \in \mathbb{F}^\ell$, the value $\hat{x}(t)$ can be computed in the following manner:

$$\hat{x}(t) = \sum_{b_1, ..., b_\ell \in \mathbb{H}} \widehat{EQ}(t_1, ..., t_\ell; b_1, ..., b_\ell) \cdot x(b_1, ..., b_\ell),$$

where $EQ : \mathbb{H}^\ell \times \mathbb{H}^\ell \to \{0, 1\}$ is the function defined as

$$EQ(t_1, ..., t_\ell; b_1, ..., b_\ell) = 1 \iff \forall i : t_i = b_i.$$

Moreover, as shown in [JKKZ20, Section 6.1], the low degree extension $\widehat{EQ} : \mathbb{F}^\ell \times \mathbb{F}^\ell \to \mathbb{F}$ of EQ can be computed in the following manner:

$$\widehat{EQ}(t_1, ..., t_\ell; b_1, ..., b_\ell) = \prod_{i=1}^{\ell} \sum_{\alpha \in \mathbb{H}} \prod_{\beta \in \mathbb{H} \setminus \{\alpha\}} \frac{(\beta - t_i) \cdot (\beta - b_i)}{(\beta - \alpha)^2}.$$

Proposition 2.3 ([RRR16, Proposition 3.8])**.** *Let $\mathbb{H}$ and $\mathbb{F}$ be field ensembles such that $\mathbb{F}$ is an extension field of $\mathbb{H}$ and $|\mathbb{H}| \geq \log(|\mathbb{F}|)$. There exist procedures* $\mathsf{LDECompose}$ *and* $\mathsf{LDEDeompose}$ *as follows:*

- $\mathsf{LDECompose}$*: Given oracle access to $\hat{x}_1, ..., \hat{x}_k$ over $\mathbb{F}$, and a point $i \in \mathbb{F}^\ell$, for $\ell = \log(k) + \max_{j \in [k]} \log_{|\mathbb{H}|}(|x_j|)$, the procedure* $\mathsf{LDECompose}$ *makes a single query to each $\hat{x}_j$, runs in time $k \cdot \text{poly}(|\mathbb{H}|, \sum_{j=1}^k \log_{|\mathbb{H}|}(|x_j|), \log(k))$, and outputs $\widehat{(x_1, ..., x_k)}(i)$.*

- $\mathsf{LDEDeompose}$*: Given oracle access to $\widehat{(x_1, ..., x_k)}$, some $j \in [k]$, and a point $i \in \mathbb{F}^{\log_{|\mathbb{H}|}(|x_j|)}$, the procedure* $\mathsf{LDEDeompose}$ *makes a single oracle query, runs in time $\text{poly}(|\mathbb{H}|, \log(k), \sum_{j=1}^k \log_{|\mathbb{H}|}(|x_j|))$ and outputs $\hat{x}_j(i)$.*

Definition 2.4 (Holographic argument system)**.** *Let C be a code. Let $(\mathcal{P}, \mathcal{V})$ be an argument system, and let $q : \mathbb{N} \to \mathbb{N}$. The verifier $\mathcal{V}$ is said to be* q-query C-holographic *in its input x, if given oracle access to $C(x)$ (*without *direct access to x), security parameter λ, and transcript τ from the interaction with $\mathcal{P}$, the computation $\mathcal{V}^{C(x)}(\tau, 1^\lambda)$ can be divided into two steps:*

1. *$\mathcal{V}_1(\tau, 1^\lambda)$ runs a $\mathrm{poly}(|\tau|, \lambda)$-time computation to verify the transcript τ.*
2. *$\mathcal{V}_2^{C(x)}(\tau, 1^\lambda)$ parses the transcript τ to get I, a set of query indices of size $q = q(\lambda)$, and Z a set of field elements of size q, and accepts if and only if $C(x) |_I = Z$.*

We consider the complexity of a holographic verifier $\mathcal{V}$ to be its runtime given the oracle to $C(x)$.

Let $\mathbb{H}$ be a field, $\mathbb{F}$ an extension field of $\mathbb{H}$ of size $|\mathbb{H}| \geq \log(|\mathbb{F}|)$, and $\ell \in \mathbb{N}$. The q-query $\mathsf{LDE}_{\mathbb{F},\mathbb{H},\ell}$-holographic verifier can be implemented in the standard model by directly computing the value of $\hat{x}$ at the queried indices, which results in an additional $q \cdot |\mathbb{H}|^\ell \cdot \mathrm{poly}(|\mathbb{H}|, \ell)$ overhead in the running time (see, e.g. [Rot09, Claim 3.2.2]).

2.1.3 Probabilistically Checkable Proofs

A *probabilistically checkable proof* (PCP) is a special format for writing a proof that can be verified by only reading a few bits.

Definition 2.5 (Probabilistically checkable proof)**.** *A* probabilistically checkable proof (PCP) *for language $\mathcal{L} \in \mathsf{NTIME}(T)$ consists of a $\mathrm{poly}(T)$ prover $\mathcal{P}$, that gets as input the instance x as well as a witness w, and a $\mathrm{polylog}(T)$ time oracle machine $\mathcal{V}$, that receives x as input as well as an oracle to a proof string π. For every x the following holds:*

1. ***Completeness:*** *If $x \in \mathcal{L}$ and w is a corresponding witness, for $\pi = \mathcal{P}(x, w)$ it holds that*

$$\Pr[\mathcal{V}^\pi(x) = 1] = 1.$$

2. ***Soundness:*** *If $x \notin \mathcal{L}$, for every oracle π^* it holds that*

$$\Pr[\mathcal{V}^{\pi^*}(x) = 1] \leq \frac{1}{2}.$$

The length of π as a function of $|x|$ and $|w|$ is called the proof length*. In order to verify its oracle, the verifier $\mathcal{V}$ tosses $r = r(|x|)$ random coins, and makes $q = q(|x|)$ queries to π. The functions r and q are called the* randomness complexity *and* query complexity*, respectively.*

We note that unlike some PCP definitions in the literature, we require that the PCP prover run in polynomial time given the witness (but standard constructions do satisfy this requirement).

The celebrated PCP theorem [ALM+98] establishes the expressive power of PCPs.

Theorem 2.6 ([ALM+98])**.** *Every $\mathcal{L} \in \mathsf{NP}$ has a PCP with constant query complexity and logarithmic randomness complexity.*

A PCP for $\mathcal{L} \in \mathsf{NP}$ is said to be *succinct* [KR08, FS11] if there exists a polynomial p (which may depend on $\mathcal{L}$), such that for every $(x, w) \in R_\mathcal{L}$ it holds that $|\pi| = p(|w|, \log|x|)$, for $\pi = \mathcal{P}(x, w)$.

2.1.4 Probabilistically Checkable Proofs of Proximity

Similarly to a PCP, a *probabilistically checkable proof of proximity* (PCPP) [BGH+05, DR06] is a special format of writing proofs in which the verifier only has to read a few bits from the proof. Unlike PCPs however, a PCPP verifier only has to read a few bits from its *input*, and is therefore only required to reject inputs that are far from its language. Following [BGH+06], we define PCPPs for pair languages.

Pair languages. A language $\mathcal{L}$ is said to be a *pair language* if $\mathcal{L} \subseteq \{0,1\}^* \times \{0,1\}^*$. Given instance x, the *projection of* $\mathcal{L}$ *on* x is the set $\mathcal{L}(x) = \{y : (x,y) \in \mathcal{L}\}$.

Definition 2.7 (Probabilistically checkable proof of proximity)**.** *A* probabilistically checkable proof of proximity *(*PCPP*) for a pair-language* $\mathcal{L} \in \mathsf{DTIME}(T)$ *consists of a* $\mathrm{poly}(T)$ *time prover* $\mathcal{P}$ *that gets as input a pair* (x,y) *as well as a witness* w*, and a probabilistic* $\mathrm{polylog}(T)$ *time oracle machine* $\mathcal{V}$ *that receives* x *as an explicit input, oracle access to implicit input* y*, and oracle access to a proof string* π*. The verifier also receives (explicitly) a proximity parameter* $\delta > 0$*. For every proximity parameter* $\delta \in [0,1]$ *and input* (x,y) *it holds that:*

1. ***Completeness:*** *If* $(x,y) \in \mathcal{L}$*, for* $\pi = \mathcal{P}(x,y)$ *it holds that*

$$\Pr[\mathcal{V}^{y,\pi}(x, |y|, |\pi|, \delta) = 1] = 1.$$

2. ***Soundness:*** *If* $\Delta(y, \mathcal{L}(x)) \geq \delta$*, for every oracle* π^* *it holds that*

$$\Pr[\mathcal{V}^{y,\pi^*}(x, |y|, |\pi^*|, \delta) = 1] < \frac{1}{2}.$$

The length of π *as a function of* $|x|$ *and* $|y|$ *is called the* proof length*. In order to verify its oracles, the verifier* $\mathcal{V}$ *tosses* $r = r(|x|,|y|,\delta)$ *random coins, and makes* $q = q(|x|,|y|,\delta)$ *queries to* y *and* π*. The functions* r *and* q *are called the* randomness complexity *and* query complexity*, respectively.*

We omit the lengths $|y|$ and $|\pi|$ when these are clear from the context.

Note that our definition of PCPPs differs from the standard definition in the literature, as we require that the PCPP prover run in polynomial time, but as above most constructions satisfy this requirement. In particular:

Theorem 2.8 ([BGH+05])**.** *Let* $\mathcal{L}$ *be a pair language, with instances* (x,y)*, computable in time* $T = T(|x|,|y|)$*, and let* $\delta \in [0,1]$*. There exists a* PCPP *for* $\mathcal{L}$ *with respect to proximity parameter* δ*, with query complexity* $O(1/\delta)$*, randomness complexity* $O(\log \frac{T}{\delta})$*, and proof length* $\mathrm{poly}(T)$*. The verifier runs in time* $\mathrm{polylog}(T, \frac{1}{\delta})$ *and the prover runs in time* $\mathrm{poly}(T)$*.*

2.2 Cryptographic Tools

2.2.1 Collision Resistant Hash Functions

Definition 2.9 (Collision resistant hash function ensemble)**.** *A hash function ensemble* $\mathcal{H} = \{\mathcal{H}_\lambda\}_\lambda$*, for* $\mathcal{H}_\lambda = \left\{h : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda\right\}_h$*, is said to be* (T,s)-collision resistant *(*CRHF*), for functions*

$T : \mathbb{N} \to \mathbb{N}$ and $s : \mathbb{N} \to [0,1]$, if there exists a probabilistic polynomial time algorithm $\mathcal{H}.Gen$ that on input 1^λ outputs an efficiently computable function $h \in \mathcal{H}_\lambda$, such that for every adversary $\mathcal{A}$ of size $\leq T(\lambda)$ it holds that

$$\Pr_{h \leftarrow \mathcal{H}.Gen(1^\lambda)} \left[h(x) = h(x') \right] \leq s(\lambda),$$

for $(x, x') = \mathcal{A}(h)$.

By default, we assume a CRHF is secure against $T = \mathrm{poly}(\lambda)$ size adversaries, with $s = \mathsf{negl}(\lambda)$.

2.2.2 Merkle Trees

Merkle trees [Mer87] allow one to commitment to a long string, with succinct openings of specific blocks.

Definition 2.10 (Merkle commitment scheme)**.** *Let $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda \in \mathbb{N}}$ be a (T, s)-collision hash function ensemble.*

A Merkle tree commitment scheme is a triplet of algorithms $(\mathsf{MC}, \mathsf{MO}, \mathsf{MV})$ which fulfill the following requirements:

- *The commitment algorithm $\mathsf{MC}(h, m)$ takes as input a hash function $h : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ sampled from $\mathcal{H}_\lambda$, and a message $m \in \{0,1\}^{2^d \cdot \lambda}$.*

  The algorithm parses m as $(\ell_1^d \| ... \| \ell_{2^d}^d)$, where each $\ell_i^d \in \{0,1\}^\lambda$, and for every $j \in [d]$ and every $i \in [2^{j-1}]$ computes $\ell_i^{j-1} = h(\ell_{2i-1}^j \| \ell_{2i}^j)$. The algorithm outputs $\mathsf{root} = \ell_1^0$.

- *The opening algorithm $\mathsf{MO}(h, m, i)$ takes as input the hash function h, a message $m \in \{0,1\}^{2^d \cdot \lambda}$, and an index $i \in [2^d]$.*

  It outputs $\mathsf{open} = \{(c_j, \mathsf{sib}_j, p_j)\}_{j=0}^d$, where each $c_j, \mathsf{sib}_j \in \{0,1\}^\lambda$, $p_j \in \{\mathsf{left}, \mathsf{right}\}$, $c_0 = \mathsf{MC}(h, m)$, and c_d is the i-th block of m. For every $j \in \{1, ..., d\}$, it holds that if $p_j = \mathsf{left}$ then $c_{j-1} = h(\mathsf{sib}_j, c_j)$ and otherwise, for $p_j = \mathsf{right}$, it holds that $c_{j-1} = h(c_j, \mathsf{sib}_j)$.

  For $\mathsf{open} = \mathsf{MO}(h, m, i)$, we denote by $\mathsf{open}\,|_i$ the i-th block of m as described by open, i.e., for $\mathsf{open} = \{(c_j, \mathsf{sib}_j, p_j)\}_{j=0}^d$ it holds that $\mathsf{open}\,|_i = c_d$.

- *The verification algorithm $\mathsf{MV}(h, \mathsf{root}, i, \mathsf{open})$ takes as input the hash function h, the commitment root, an index $i \in [2^d]$, and openings open. It parses $\mathsf{open} = \{(c_j, \mathsf{sib}_j, p_j)\}_{j=0}^d$, where $c_j, \mathsf{sib}_j \in \{0,1\}^\lambda$ and $p_j \in \{\mathsf{left}, \mathsf{right}\}$.*

  The algorithm checks that $\{p_j\}_{j=0}^d$ correspond to the i-th block. From $j = d$ to $j = 1$ it checks that if $p_j = \mathsf{left}$ then $c_{j-1} = h(\mathsf{sib}_j \| c_j)$, and if $p_j = \mathsf{right}$ then $c_{j-1} = h(c_j \| \mathsf{sib}_j)$. Lastly, it checks that $c_0 = \mathsf{root}$. It outputs 1 if all of the checks above pass, otherwise it outputs 0.

Given $I = \{i_1, ..., i_q\} \subseteq [2^d]$ we define the output of $\mathsf{MO}(h, m, I)$ to be $\mathsf{open} = \mathsf{MO}(h, m, i_1) \| ... \| \mathsf{MO}(h, m, i_q)$, and $\mathsf{MV}(h, \mathsf{root}, I, \mathsf{open}) = \bigwedge_{i \in I} \mathsf{MV}(h, \mathsf{root}, i, \mathsf{open})$.

Claim 2.11 ([Mer87])**.** *Assume there exists a (T, s)-CRHF ensemble $\mathcal{H} = \{\mathcal{H}_\lambda\}_\lambda$. Then, the Merkle tree commitment scheme $(\mathsf{MC}, \mathsf{MO}, \mathsf{MV})$ defined above has the following properties:*

1. ***Correctness:*** *For every security parameter $\lambda \in \mathbb{N}$, every message $m \in \{0,1\}^{2^d \cdot \lambda}$, and every $i \in [2^d]$, it holds that*

$$\Pr_{h \leftarrow \mathcal{H}.Gen(1^\lambda)} \left[\mathsf{MV}(h, \mathsf{root}, i, \mathsf{open}) = 1 \right] = 1,$$

where $\mathsf{root} = \mathsf{MC}(h, m)$ *and* $\mathsf{open} = \mathsf{MO}(h, m, i)$.

2. ***Computational binding:*** *For every security parameter $\lambda \in \mathbb{N}$, every message $m \in \{0,1\}^{2^d \cdot \lambda}$, every $i \in [2^d]$, and every adversary $\mathcal{A}$ of size $\leq T(\lambda)$, it holds that*

$$\Pr_{h \leftarrow \mathcal{H}.Gen(1^\lambda)} \left[\mathsf{MV}(h, \mathsf{root}, i, \mathsf{open}^*) = 1 \ \wedge \ \mathsf{open}^*|_i \neq m|_i \right] \leq s(\lambda),$$

where $\mathsf{open}^* = \mathcal{A}(h, \mathsf{root}, i)$, $\mathsf{root} = \mathsf{MC}(h, m)$, $\mathsf{open}^*|_i$ *is the block corresponding to i revealed in* open^*, *and $m|_i$ is the i-th block of m.*

3. ***Succinctness:*** *For every security parameter $\lambda \in \mathbb{N}$, every $m \in \{0,1\}^{2^d \cdot \lambda}$, and every $i \in [2^d]$, it holds that* $|\mathsf{root}|, |\mathsf{open}| = O(\lambda \cdot \log(|m|))$, *for* $\mathsf{root} = \mathsf{MC}(h, m)$ *and* $\mathsf{open} = \mathsf{MO}(h, m, i)$.

3 Our Results

In this section we state our main results. First, in Section 3.1 we define the notion of *publicly-verifiable probabilistically checkable arguments* (PCAs). We then state a theorem showing the existence of *succinct* publicly-verifiable PCAs for a large subclass of NP, assuming the sub-exponential hardness of LWE. Then, in Section 3.2 we introduce the notion of *computational instance compression* (CICs). Assuming the sub-exponential hardness of LWE, we show the existence of CIC for a large subclass of NP.

3.1 Probabilistically Checkable Arguments

Much like a PCP, a *probabilistically checkable argument* (PCA) is a special format for writing a proof in which the verifier only has to read a few bits from the proof. Unlike PCPs, in which *every* alleged proof for a false statement is rejected with high probability, for PCAs, accepting proofs may exist, but we require that it is *intractable* to find them. This relaxation enables us to build PCAs of length polynomial in the length of the *witness*, rather than the non-deterministic verification time, a result which is unlikely to be possible for PCPs [FS11]. In contrast to PCPs, PCAs are constructed with respect to a *common reference string* (CRS), which can be accessed both by the prover and the verifier.

PCAs were originally introduced by Kalai and Raz [KR09]. We consider a natural restriction of their definition that mandates that the PCA is *publicly-verifiable* - that is, given the PCA proof-string π (and the CRS) *anyone* can verify the proof by making only a few queries to π.

Definition 3.1 (Publicly verifiable PCA)**.** *A* publicly-verifiable probabilistically checkable argument *(*PCA*) for an* NP *relation $\mathcal{R}$, with adversary size $T : \mathbb{N} \to \mathbb{N}$ and soundness error $s : \mathbb{N} \to [0,1]$, is a triplet of* $\mathrm{poly}(n, m, \lambda)$*-time algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$, with deterministic $\mathcal{P}$ and probabilistic $\mathcal{G}$ and $\mathcal{V}$, such that for every instance length n and witness length m the following holds:*

- ***Completeness:*** *For every $x \in \{0,1\}^n$ and $w \in \{0,1\}^m$, such that $(x,w) \in \mathcal{R}$, every $\lambda \in \mathbb{N}$, and every $\mathsf{CRS} \leftarrow \mathcal{G}(1^n, 1^m, 1^\lambda)$ it holds that*

$$\Pr\left[\mathcal{V}^{\pi}(x, \mathsf{CRS}) = 1\right] = 1, \tag{1}$$

where $\pi = \mathcal{P}(x, w, \mathsf{CRS})$, and the probability in Eq. (1) is only on $\mathcal{V}$'s coin tosses.

- ***Computational soundness:*** *For every $x \in \{0,1\}^n$ s.t. $\mathcal{R}(x) = \emptyset$, every $\lambda \in \mathbb{N}$, and every $\mathcal{P}^*$ of size $\leq T(\lambda)$, with all but $s(\lambda)$ probability over the choice of $\mathsf{CRS} \leftarrow \mathcal{G}(1^n, 1^m, 1^\lambda)$ it holds that*

$$\left[\mathcal{V}^{\pi^*}(x, \mathsf{CRS}) = 1\right] \leq \frac{1}{2}, \tag{2}$$

where $\pi^* = \mathcal{P}^*(x, \mathsf{CRS})$, and again the probability in Eq. (2) is only on $\mathcal{V}$'s coin tosses.

The length of π, as a function of n, m, and λ is called the proof length*. In order to verify its oracle, the verifier $\mathcal{V}$ tosses $r = r(n, m, \lambda)$ random coins, and makes $q = q(n, m, \lambda)$ queries to π. The functions r and q are called the* randomness complexity *and* query complexity*, respectively.*

We reiterate that in contrast to the definition of [KR09], our PCA verifier is *not* given a trapdoor to the CRS. In this work we only discuss publicly-verifiable PCAs, and in what follows whenever we say PCA we refer to the publicly-verifiable variant. Also, by default we assume that PCAs are secure against malicious $T = \text{poly}(\lambda)$ size provers, with soundness error $s = \mathsf{negl}(\lambda)$.

Note that in Definition 3.1 we distinguish between the randomness of $\mathcal{G}$ (used to generate the CRS) and that of $\mathcal{V}$ (used to check the proof). This separation is done since we would like the CRS to be "good" with overwhelming probability, but since the verifier only makes a small number of queries, can only guaranteed that its checks will fail (in case $x \notin \mathcal{L}$) with constant probability. In particular, this distinction allows us to easily reduce the soundness error of $\mathcal{V}$ arbitrarily, and without increasing the proof length. We do so in the natural way - by simply having the verifier generate more query sets.

Fact 3.2. *Let $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ be a PCA for relation $\mathcal{R}$, with adversary size T and soundness error s. Denote by $\mathcal{V}_k$ the verifier which runs the verifier k times and accepts if and only if every run accepts. Then, for every $n, m, \lambda \in \mathbb{N}$, every $x \in \{0,1\}^n$ such that $\mathcal{R}(x) = \emptyset$, and $\mathcal{P}^*$ of size $\leq T(\lambda)$, with all but $s(\lambda)$ probability over the choice of $\mathsf{CRS} \leftarrow \mathcal{G}(1^n, 1^m, 1^\lambda)$ it holds that*

$$\Pr\left[\mathcal{V}_k^{\pi^*}(x, \mathsf{CRS}) = 1\right] \leq \frac{1}{2^k},$$

where $\pi^* = \mathcal{P}^*(x, \mathsf{CRS})$.

Succinct PCAs. A PCA for an NP relation $\mathcal{R}$, which is decidable in some time $t = t(n, m) \geq n$, is said to be *succinct* if the PCA proof is of length $\text{poly}(m, \lambda, \log(t))$, where poly refers to a fixed *universal* polynomial (that does not depend on the relation $\mathcal{R}$).

Our first main result shows the existence of succinct publicly-verifiable PCAs for a large subclass of NP.

Theorem 3.3. *Assume the sub-exponential hardness of* LWE*. Let $\mathcal{R}$ be an NP relation where membership can be decided in logspace-uniform NC, given the instance of length n and witness of length m. Then, there exists a succinct constant-query publicly-verifiable PCA for $\mathcal{R}$. Furthermore, the verifier runs runs in time $\text{poly}(n, \log(m), \lambda)$ and has randomness complexity $O(\log(m) + \log\log(n) + \log(\lambda))$.*

(Note that the furthermore clause is non-trivial as the requirement for the PCA verifier runtime is $\text{poly}(n, m, \lambda)$, where as we achieve $\text{poly}(n, \log(m), \lambda)$.)

The proof of Theorem 3.3 is given in Section 4.

3.2 Computational Instance Compression

Instance compression [HN10] is a mechanism for compressing a long instance with a short witness, down to (roughly) the witness length. Formally, an *instance compression from* NP*-relation* $\mathcal{R}$ *to* NP*-relation* $\mathcal{R}'$ is a poly-time computable function f, such that for every x of length n with corresponding witness length m, the compressed value $x' = f(x)$ is an instance of $\mathcal{R}'$ with length and witness length $\text{poly}(m, \log(n))$, and $\mathcal{R}(x) = \emptyset \iff \mathcal{R}'(x') = \emptyset$.

Unfortunately, instance compression for SAT and many other NP-complete problems does not exist, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, in which case the polynomial hierarchy collapses [FS11]. In this work we consider a relaxation of instance compression which we call *computational instance compression* (CIC). A CIC consists of three polynomial-time algorithms: a *generator* $\mathcal{G}$, an *instance compressor* $\mathcal{IC}$, and a *witness transformer* $\mathcal{WT}$. Loosely speaking, we require that:

- The *generator* $\mathcal{G}$ which, given the security parameter and instance and witness lengths, generates a CRS.
- If $\mathcal{R}(x) \neq \emptyset$, then the (deterministic) *instance compressor* $\mathcal{IC}$ outputs $x' = \mathcal{IC}(x, \mathsf{CRS})$ such that $\mathcal{R}'(x') \neq \emptyset$. Moreover, given a witness w for x, the (deterministic) *witness transformer* $\mathcal{WT}$ produces a corresponding witness w' for x'.
- If $\mathcal{R}(x) = \emptyset$, we require that it is computationally intractable to find a witness w' for $x' = \mathcal{IC}(x, \mathsf{CRS})$ (even though it may be that such witnesses exist).

Definition 3.4 (Computational instance compression)**.** *Let $\mathcal{R}$ and $\mathcal{R}'$ be* NP*-relations. Denote by n the instance length for $\mathcal{R}$, and by m the corresponding witness length. A* computational instance compression *(*CIC*) scheme from $\mathcal{R}$ to $\mathcal{R}'$, with adversary size $T : \mathbb{N} \to \mathbb{N}$ and soundness error $s : \mathbb{N} \to [0, 1]$, is a triplet of $\text{poly}(n, m, \lambda)$-time algorithms $(\mathcal{G}, \mathcal{IC}, \mathcal{WT})$, for probabilistic $\mathcal{G}$ and deterministic $\mathcal{IC}$ and $\mathcal{WT}$, such that:*

- ***Completeness:*** *For $x \in \{0,1\}^n$ and $w \in \{0,1\}^m$ such that $(x, w) \in \mathcal{R}$, and every λ, it holds that*
$$\Pr_{\mathsf{CRS} \leftarrow \mathcal{G}(1^n, 1^m, 1^\lambda)} \left[\left(\mathcal{IC}(x, \mathsf{CRS}), \mathcal{WT}(x, w, \mathsf{CRS}) \right) \in \mathcal{R}' \right] = 1.$$
- ***Computational soundness:*** *For every $x \in \{0,1\}^n$ with $\mathcal{R}(x) = \emptyset$, every $\lambda \in \mathbb{N}$, and every adversary $\mathcal{A}$ of size $\leq T(\lambda)$ it holds that*
$$\Pr_{\mathsf{CRS} \leftarrow \mathcal{G}(1^n, 1^m, 1^\lambda)} \left[(\mathcal{IC}(x, \mathsf{CRS}), w^*) \in \mathcal{R}' \right] \leq s(\lambda),$$
where $w^* = \mathcal{A}(x, \mathsf{CRS})$.

By default, we assume a CIC scheme is secure against adversaries of size $T = \text{poly}(\lambda)$, with $s = \mathsf{negl}(\lambda)$.

Discussion. We briefly discuss some important points regarding the definition of CICs:

1. The first point that we highlight, is that for x a NO instance, it could very well be the case that x' is a YES instance (indeed, this is virtually a certainty in our construction). Whereas standard instance compression requires that x' be a NO instance, the CIC soundness requirement states that if x' is indeed a YES instance, then finding a witness w' for x' is intractable.

2. One may also consider a strengthening of CIC in which the compression algorithm, applied to a NO instance, outputs x' that is *computationally indistinguishable* from a false instance. We refer to this notion as a *strong* CIC and note that it indeed implies the weaker notion, since if x' is indistinguishable from a NO instance, then it is intractable to find a witness w' for x'. We leave the study of strong CICs to future work.

3. The last point is regarding the importance of the witness transform algorithm $\mathcal{WT}$. One might wonder whether the requirement that such an efficient transformation exists is significant. In particular, one can consider a weaker notion of CIC in which it is not required that a witness w' for x' can be efficiently found from (x, w), i.e., forgoing the use of $\mathcal{WT}$.

   This weaker definition seems substantially less useful to us. In particular, it does not suffice for a main application of instance compression - storing instances more efficiently, in order to reveal their solution at a later time. Using classical instance compression, finding a witness w' for x' proves that there exists a witness w for x. As for CIC, without $\mathcal{WT}$, it's not clear how to find a witness w' for x', even given a witness w for x.

   Moreover, we note that this weaker definition can be trivially realized for languages $\mathcal{L}'$ that are hard on average as follows: the instance compressor ignores x and simply outputs a "hard" false instance x' (i.e., a YES instance x' that is computationally indistinguishable from a NO instance x''). This trivial compressor satisfies the property that (1) a YES instance is mapped to a YES instance, whereas (2) for a NO instance x it is computationally intractable to find a witness w' for x' (as this would distinguish x' from x'').

Our second main result is a construction of CICs from (sub-exponential) LWE.

Theorem 3.5. *Assume the sub-exponential hardness of* LWE*. Then, every* NP *relation* $\mathcal{R}$*, for which membership can be decided in logspace-uniform* NC*, is computationally instance compressible to* SAT*.*

The proof of Theorem 3.5 is given in Section 5.1. We remark that the proof of Theorem 3.5 follows in a relatively straightforward manner from Theorem 3.3. In particular, Lemma 5.1 states that if a language $\mathcal{L}$ has a PCA, then $\mathcal{L}$ is computationally compressible to SAT. Combining Lemma 5.1 with Theorem 3.3 yields Theorem 3.5. A complementary result is given in Section 5.2, showing that if $\mathcal{L}$ is computationally compressible to SAT then $\mathcal{L}$ has a succinct PCA.

4 Probabilistically Checkable Arguments

In this section we construct PCAs for all NP relations that are decidable by (logspace uniform) NC circuits. These include many natural NP relations such as SAT, k-Clique, 3Col, etc. The hardness

assumption for our construction is the sub-exponential hardness of the LWE problem (see, e.g., [Reg10]).

This section is organized as follows:

1. First, we build a PSNARG which is both holographic and partially-adaptively sound, a technical tool for our PCA construction. The definition and construction of this PSNARG is done in Section 4.1.

2. We then use the holographic and partially-adaptively sound PSNARG to prove Theorem 3.3. This is done in Section 4.2.

4.1 A Technical Tool — Holographic and Partially-Adaptive PSNARGs

A technical tool we use in our construction is a PSNARG. In short, it is a non-interactive argument for a relation $\mathcal{R} \in \mathsf{P}$, which is succinct in the first coordinate of its input.

Definition 4.1 (Partial-succinctness)**.** *A non-interactive argument-system for a* relation $\mathcal{R} \in \mathsf{P}$ *is said to be* partially-succinct *if for every instance* (x_1, x_2) *of* $\mathcal{R}$ *with* $|x_1| = n$*, and* $|x_2| = m$*, and every security parameter* $\lambda \in \mathbb{N}$*, the communication complexity of the argument-system is* $\mathrm{poly}(m, \log n, \lambda)$*. That is, polylogarithmic in* n*, the length of the first component.*

Definition 4.2 (PSNARG)**.** *A* PSNARG *is a partially-succinct public-coin non-interactive argument-system for a relation* $\mathcal{R} \in \mathsf{P}$*.*

In this section we construct PSNARGs geared toward our PCA construction. Since it is somewhat confusing, we should first clarify the difference between a PSNARG and a regular SNARG. A regular SNARG is defined with respect to an NP relation $\mathcal{R}$ and the assumption is that the prover gets the witness but the verifier does not. In contrast, a PSNARG is also defined with respect to a relation that is decidable in P (thus $\mathcal{R}$ is an NP relation but we focus on the task of deciding membership in $\mathcal{R}$ given also the witness). In a PSNARG the verifier gets both the instance *and* the witness. Verifying that the pair satisfies the relation is non-trivial since we aim for the verification to be highly efficient.

Given instance x, our goal for the PCA construction in the next section is to have the prover send (1) a witness w, (2) a PSNARG proving that $(x, w) \in \mathcal{R}$, (3) and a PCPP proving that, had the verifier read the PSNARG, it would have accepted. The prover generates its proof *after* seeing the CRS. To maintain soundness, it means that for $x \notin \mathcal{L}$, a polytime malicious prover should be incapable of finding a false witness w^* and a PSNARG proof π^* which convince the PSNARG verifier. Therefore, we need the PSNARG verifier to be *adaptively* sound, at least in the witness w.

Additionally, since the PCPP proves the the computation of the PSNARG, it holds that the length of the PCPP is proportional to the runtime of the PSNARG verifier. Non-holographic verifiers have to, at the very least, read their input x, which would result in a proof of length proportional to $|x|$. Therefore, it is essential that the PSNARG verifier be holographic to minimize its runtime.

For a large subset of relations $\mathcal{R} \in \mathsf{P}$, we build a PSNARG for $\mathcal{R}$ which is both holographic, and adaptively sound in the second component.

Definition 4.3 (Partial-adaptive soundness)**.** *Let* $\mathcal{R} \subseteq \{0,1\}^n \times \{0,1\}^m$ *be a relation. A* PSNARG $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *for membership in* $\mathcal{R}$ *is said to be* (T, s)-partially-adaptively sound *if for every* $x_1 \in$

$\{0,1\}^n$, *every* $\lambda \in \mathbb{N}$, *and every* $\mathcal{P}^*$ *of size* $\leq T(\lambda)$ *it holds that*

$$\Pr_{\mathsf{CRS} \leftarrow \mathcal{G}(1^n,1^m,1^\lambda)} \left[\mathcal{V}\big((x_1, x_2^*), \mathsf{CRS}, \pi^*\big) = 1 \ \wedge \ (x_1, x_2^*) \notin \mathcal{R} \right] \leq s(\lambda)$$

for $(x_2^*, \pi^*) = \mathcal{P}^*(\mathsf{CRS})$ *with* x_2^* *of length* m.

We rely on the recent work of [JKKZ20] (and an improvement due to [HLR21]), who construct PSNARGs for all relations in logspace uniform NC. We note that this construction, as-is, provides only *non-adaptive* soundness, but this will be easy to fix (at a moderate cost).

Theorem 4.4 ([JKKZ20, HLR21])**.** *Assume the sub-exponential hardness of* LWE.

Let $C : \{0,1\}^N \to \{0,1\}$ *be a log-space uniform circuit of size* $S = S(N)$ *and depth* $D = D(N)$ *for inputs of length* N.[9] *Let* $\mathbb{H}$ *be a field of size* $|\mathbb{H}| = O(\log(N))$, $\mathbb{F}$ *be an extension field of* $\mathbb{H}$ *of size* $|\mathbb{F}| = O(\log^2(N))$, *and* $\ell = \frac{\log(N)}{\log\log(N)}$.

Then, there exists a non-interactive argument for the language $\{x \colon C(x) = 0\}$ *in the* CRS *model. This non-interactive argument has the following efficiency guarantees: for security parameter* λ, *the prover runs in time* $\mathrm{poly}(\lambda, S)$, *the verifier runs in time* $(D+N) \cdot \mathrm{poly}(\lambda, \log S)$, *and the communication complexity is* $D \cdot \mathrm{poly}(\lambda, \log S)$.

Furthermore, the non-interactive argument has a soundness error $2^{-\lambda^\epsilon}$ *against adversaries of size* $2^{\lambda^{\epsilon'}}$, *for some constants* $\epsilon, \epsilon' > 0$, *and is* $\mathsf{LDE}_{\mathbb{F},\mathbb{H},\ell}$*-holographic with* $\mathrm{poly}(\lambda)$*-queries to its input (as per Definition 2.4).*

The furthermore clause does not hold for the [JKKZ20] construction, but follows from an improvement in [HLR21] and some simple observations, detailed next.

First, while [JKKZ20] only show polynomial-hardness, it is straightforward to show that their PSNARG has sub-exponential soundness assuming the sub-exponential hardness of LWE.

Second, the holographic property is due to the the following reason: The [JKKZ20] protocol is constructed by applying the Fiat-Shamir[10] transform [FS86] to the GKR protocol [GKR15]. The GKR protocol is shown to be holographic in [GKR15]. This feature is preserved by the Fiat-Shamir transform because [JKKZ20] do *not* feed the main input as an input to the Fiat-Shamir hash function[11].

Third, in our construction we instantiate GKR with a small field, of size $\mathrm{polylog}(N) = \mathrm{polylog}(n, m)$, in order to allow our PCA verifier to generate a full representation of the low degree extension of the input. To do so we take $\mathrm{poly}(\lambda)$ parallel repetitions of the base GKR protocol over the small field, and compile the resulting protocol using Fiat-Shamir. We rely here on the work of [HLR21] who show this can be done (using list recoverable codes). We remark that in this variant (in contrast to vanilla GKR), the number of queries by the PSNARG verifier to the low-degree extension of its

[9] In Theorem 4.4 the input length is denoted by N, whereas elsewhere in this manuscript we denote the input length by n. This is done because later on we will view the input here as consisting of two parts, one of length n and the other of length m (i.e., $N = n + m$).

[10] The Fiat-Shamir transform is a generic method for reducing the number of rounds of arbitrary public-coin protocols. In a nutshell, instead of getting the random coins from the public-coin verifier, the prover generates the coins by applying a complex hash function over the transcript generated thus far.

[11] Typically when using the Fiat-Shamir transform the main input of the verifier is also used as part of the input to the hash function. This is done in order to obtain *adaptive* soundness. Since the techniques of [JKKZ20, HLR21] do not provide adaptive soundness anyhow, they do not use the main input when applying the hash, which means that their proof-system remains holographic.

input is $\text{poly}(\lambda)$.

Next, we show how to generically use complexity leveraging to transform a sub-exponentially secure PSNARG for a relation $\mathcal{R} \in \mathsf{P}$ (such as the one from Theorem 4.4) into one that is partially-adaptively sound.

Lemma 4.5. *Let $\mathcal{R} \in \mathsf{P}$ be a relation, and $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ be a PSNARG for $\mathcal{R}$ with sub-exponential soundness error. Then, $\mathcal{R}$ has a PSNARG which is partially-adaptively sound.*

Proof. Let (x_1, x_2) be an instance of $\mathcal{R}$, of length $|x_1| = n$, $|x_2| = m$, and let λ be the security parameter. By the sub-exponential computational soundness of $\mathcal{V}$, there exists a constant ϵ such that if $(x_1, x_2) \notin \mathcal{R}$, then for every $\mathcal{P}^*$ of size $\leq \text{poly}(\lambda)$ it holds that

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathcal{G}(1^n, 1^m, 1^\lambda) \\ \pi^* \leftarrow \mathcal{P}^*((x_1,x_2),\mathsf{CRS})}} \left[\mathcal{V}\big((x_1, x_2), \mathsf{CRS}, \pi^*\big) = 1 \right] \leq 2^{-\lambda^\epsilon}.$$

Therefore, for a choice of $\lambda' = (\lambda + m)^{1/\epsilon}$, for every $\mathcal{P}^*$ of size $\leq T(\lambda')$ it holds that

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathcal{G}(1^n, 1^m, 1^{\lambda'}) \\ \pi^* \leftarrow \mathcal{P}^*((x_1,x_2),\mathsf{CRS})}} \left[\mathcal{V}\big((x_1, x_2), \mathsf{CRS}, \pi^*\big) = 1 \right] \leq 2^{-(\lambda')^\epsilon} = 2^{-(\lambda+m)^{\epsilon \cdot 1/\epsilon}} = 2^{-\lambda - m}.$$

Let x_1 and $\mathcal{P}^*$ a malicious adversary of size $\leq T(\lambda')$. Given CRS, if $\mathcal{P}^*$ chooses x_2^* adaptively, such that $(x_1, x_2^*) \notin \mathcal{R}$, by the law of total probability it holds that

$$\begin{aligned}
\Pr_{\substack{\mathsf{CRS} \leftarrow \mathcal{G}(1^n, 1^m, 1^{\lambda'}) \\ x_2^*, \pi^* \leftarrow \mathcal{P}^*(x_1, \mathsf{CRS})}} \left[\mathcal{V}\big((x_1, x_2^*), \mathsf{CRS}, \pi^*\big) = 1 \right] &= \sum_{x_2 \in \{0,1\}^m} \Pr_{\substack{\mathsf{CRS} \leftarrow \mathcal{G}(1^n, 1^m, 1^{\lambda'}) \\ x_2^*, \pi^* \leftarrow \mathcal{P}^*(x_1, \mathsf{CRS})}} \left[\begin{array}{c} \big(\mathcal{V}\big((x_1, x_2^*), \mathsf{CRS}, \pi^*\big) = 1\big) \\ \wedge \big(x_2^* = x_2\big) \end{array} \right] \\
&\leq \sum_{x_2 \in \{0,1\}^m} \Pr_{\substack{\mathsf{CRS} \leftarrow \mathcal{G}(1^n, 1^m, 1^{\lambda'}) \\ \pi^* \leftarrow \mathcal{P}^*(x, \mathsf{CRS})}} \left[\mathcal{V}\big((x_1, x_2), \mathsf{CRS}, \pi^*\big) = 1 \right] \\
&\leq 2^m \cdot 2^{(-\lambda')^\epsilon} \\
&= 2^{-\lambda}.
\end{aligned}$$

Since λ' is a fixed polynomial in λ and m, the overhead in communication complexity is polynomial in λ and m. □

Combining Theorem 4.4 and Lemma 4.5 we obtain:

Corollary 4.6. *Let $\mathcal{R}$ be a relation which is decidable by a family of logspace-uniform circuits of size $S = S(n, m)$ and depth $D = D(n, m)$, for instance (x_1, x_2) with lengths $|x_1| = n$, $|x_2| = m$. Let $\mathbb{H}$ be a field of size $|\mathbb{H}| = O(\log(n + m))$, $\mathbb{F}$ be an extension field of size $|\mathbb{F}| = O(\log^2(n + m))$, and $\ell = \frac{\log(n+m)}{\log\log(n+m)}$.*

Assuming the sub-exponential hardness of LWE*, there exists a PSNARG for $\mathcal{R}$ in which the communication complexity is $D \cdot \text{poly}(\lambda, \log S)$, the prover runs in time $\text{poly}(\lambda, S)$, and the verifier runs in time $(D + m) \cdot \text{poly}(\lambda, \log S)$, for λ a fixed polynomial of m. Moreover, the verifier is $\mathsf{LDE}_{\mathbb{F},\mathbb{H},\ell}$-holographic in its input (x_1, x_2), and partially-adaptively sound.*

We note that while the starting point for our partially-adaptive holographic PSNARG construction is the PSNARG of [JKKZ20, HLR21], there exists another construction of publicly-verifiable PSNARGs for P, due to Kalai, Paneth and Yang [KPY19]. This PSNARG holds for all of P, rather than only NC as in [JKKZ20], under a new assumption on groups with bi-linear maps. The PSNARG of [KPY19] is known to be adaptively sound. If it can also be shown to be holographic, then it can be used for our PCA construction, thus giving succinct PCAs for all of NP, under the same cryptographic assumption as in [KPY19].

An additional technical tool we need for our PCA construction is *collision resistant hash functions* (CRHFs, see Definition 2.9).

Theorem 4.7 ([GGH11])**.** *Assuming the hardness of* LWE*, there exist collision-resistant hash function ensembles.*

4.2 Proof of Theorem 3.3: Succinct PCAs from LWE

To prove Theorem 3.3 we start with a lemma showing how to construct (public-coin) PCAs, from holographic partially-adaptive PSNARGs and collision-resistant hash functions.

Lemma 4.8. *Let $\mathcal{R}$ be an* NP*-relation. We use n to denote the instance length and m to denote the witness length.*

Let $\mathbb{H}$ be a field of size $|\mathbb{H}| = O(\log(n+m))$, and $\mathbb{F}$ an extension field of $\mathbb{H}$ of size $|\mathbb{F}| = O\big(\log^2(n+m)\big)$. Assume that field operations over $\mathbb{F}$ and $\mathbb{H}$ can be performed in time $\mathrm{polylog}(n,m)$. Denote by $\ell = \frac{\log(n+m)}{\log\log(n+m)}$, $\ell_n = \frac{\log(n)}{\log\log(n+m)}$, $\ell_m = \frac{\log(m)}{\log\log(n+m)}$.

Assume there exists a public-coin PSNARG *$(\mathcal{G}_{\mathsf{PSNARG}}, \mathcal{P}_{\mathsf{PSNARG}}, \mathcal{V}_{\mathsf{PSNARG}})$ for $\mathcal{R}$ which is holographic in (x,w), partially-adaptively sound, has communication complexity $\mathsf{cc} = \mathsf{cc}(n,m,\lambda)$, verifier complexity $\mathsf{vc} = \mathsf{vc}(n,m,\lambda)$, and prover complexity $\mathsf{pc} = \mathsf{pc}(n,m,\lambda)$. Assume also, that there exist* CRHF*.*

Then, there exists a public-coin constant-query PCA *for $\mathcal{L}$ with proof length $\mathrm{poly}(m, \log(n), \lambda, \mathsf{cc}, \mathsf{vc})$. Furthermore, the* PCA *verifier runs in time $\mathrm{poly}(n, \log(m), \lambda, \log \mathsf{vc})$ and has randomness complexity $O(\log\log(n) + \log(m) + \log(\lambda) + \log(\mathsf{vc}))$, and the prover runs in time $\mathrm{poly}(n, m, \lambda, \mathsf{pc}, \mathsf{vc})$.*

Theorem 3.3 follows immediately by combining Corollary 4.6, Theorem 4.7, and Lemma 4.8. Therefore, the remainder of this section is devoted to the proof of Lemma 4.8.

Let $\mathcal{R} \in \mathsf{P}$ be an NP-relation. Let $(\mathcal{G}_{\mathsf{PSNARG}}, \mathcal{P}_{\mathsf{PSNARG}}, \mathcal{V}_{\mathsf{PSNARG}})$ be the holographic, partially-adaptively sound PSNARG for $\mathcal{R}$.

Note that $|\mathbb{H}|^{\ell} \geq n+m$, $|\mathbb{H}|^{\ell_n} \geq n$, $|\mathbb{H}|^{\ell_m} \geq m$, and that $|\mathbb{F}|^{\ell} = O\big((n+m)^2\big)$, $|\mathbb{F}|^{\ell_n} = O(n^2)$, and $|\mathbb{F}|^{\ell_m} = O(m^2)$.

The proof will take the following steps:

1. First, we construct $\mathcal{V}'_{\mathsf{PSNARG}}$, a modified PSNARG verifier for $\mathcal{R}$, which runs in time $\mathrm{poly}(m, \log(n), \lambda)$ (i.e., polylogarithmic in the input length), given a Merkle commitment to $\hat{x}$, the low degree extension of the input x. This is done in Section 4.2.1.
2. We construct a PCA for $\mathcal{L}$ (based on $\mathcal{V}'_{\mathsf{PSNARG}}$). This is done in Section 4.2.2.
3. Lastly, we prove the complexity, completeness, and soundness of the PCA. This is done in Section 4.2.3.

4.2.1 A Sublinear PSNARG Verifier

Our eventual goal is to compose a PCP on top of the computation performed by the PSNARG verifier. Since the length of the PCP is proportional to the size of this computation, we would like for the size of the verifier's computation to depend only (poly-)logarithmically on the input length n.

At first glance, this task might seem trivial as a holographic PSNARG verifier runs in polylogarithmic time, given oracle access to $\hat{x}$, the low degree extension of its input x. The main problem lies with composing a PCP on top of the computation of an oracle machine. Current PCP techniques do not allow for $\text{poly}(T)$-long proofs of time T machines with oracles of length much larger than T, and instead require that the input of the machine be explicit. In this case, a PSNARG verifier which receives $\hat{x}$ or even x explicitly would have to at least read its input, which would result in a running time proportional to n, rather than $\log(n)$. This results in a proof of length $\text{poly}(n)$, which we cannot afford.

In this subsection we use cryptographic tools (namely Merkle hashing, see Definition 2.10) to resolve this problem. The following Lemma achieves a PSNARG verifier which runs in (poly)logarithmic time in n.

Lemma 4.9. *Let $\mathcal{R}$ be an NP-relation. Denote by n the instance length and by m the witness length for $\mathcal{R}$. Let $\mathcal{V}_{\mathsf{PSNARG}}$ be the $\text{poly}(\lambda)$-query $\mathsf{LDE}_{\mathbb{F},\mathbb{H},\ell}$-holographic and partially-adaptively sound PSNARG verifier for $\mathcal{R}$. Let $\mathcal{H} = \{\mathcal{H}_\lambda\}_\lambda$ be a CRHF ensemble.*

Then, there exists a PSNARG verifier $\mathcal{V}'_{\mathsf{PSNARG}}$ for $\mathcal{R}$ which runs in time $\text{poly}(m, \log(n), \lambda)$, given root, the Merkle root of $\hat{x}$ (with respect to a random $h \in \mathcal{H}$).

Proof of Lemma 4.9. Since the verifier $\mathcal{V}_{\mathsf{PSNARG}}$ is q-query LDE-holographic, as discussed in Definition 2.4 for $q = \text{poly}(\lambda)$, we divide its implementation into two steps:

First step, $\mathcal{V}^1_{\mathsf{PSNARG}}(\mathsf{CRS}, \pi)$. This step emulates the holographic verifier $\mathcal{V}_{\mathsf{PSNARG}}$, other than checking the holographic queries to $\widehat{(x,w)}$. The computation performed by $\mathcal{V}^1_{\mathsf{PSNARG}}$ takes time vc.

Second step, $\mathcal{V}^2_{\mathsf{PSNARG}}(x, w, \pi)$. The second step implements the query to $\widehat{(x,w)}$ in the standard model. It parses the proof π to get a set of indices $I = (i_1, ..., i_q) \subseteq \mathbb{F}^\ell$ and corresponding field elements $Z = (z_1, ..., z_q) \subseteq \mathbb{F}$, and accepts if and only if $\widehat{(x,w)} \mid_I = Z$.

Using Proposition 2.3, we implement the holographic query of $\mathcal{V}^2_{\mathsf{PSNARG}}$ by running LDECompose, a procedure which allows us to break each query to $\widehat{(x,w)}$ into two separate queries, one to $\hat{x}$ and one to $\hat{w}$. We denote by I_x and I_w the sets of queries made by LDECompose to $\hat{x}$ and $\hat{w}$, respectively. In other words, $\mathcal{V}^2_{\mathsf{PSNARG}}$ checks that $\mathsf{LDECompose}\big(\hat{x} \mid_{I_x}, \hat{w} \mid_{I_w}\big) = Z$. As stated in Definition 2.4, this step takes time $q \cdot |\mathbb{H}|^\ell \cdot \text{poly}(|\mathbb{H}|, \ell) = \text{poly}(n, m, \lambda)$, which is too long for our needs.

Succinct second step, $\mathcal{V}^{2'}_{\mathsf{PSNARG}}(h, \mathsf{root}, w, \pi, \mathsf{open})$. The procedure turns to the prover to provide the values $\hat{x} \mid_{I_x}$, which we denote by $Z_x \triangleq \hat{x} \mid_{I_x}$. These values are fixed by the CRS, the input (x, w), and the proof π, and therefore known to the PSNARG prover when writing the proof. In order to prevent the prover from providing incorrect values, we ask that the prover provide Z_x by

revealing local openings of $\hat{x}$ using root, a commitment of $\hat{x}$ with respect to the collision resistant hash function h, which we assume is known in advance to both parties.

In more detail, $\mathcal{V}^{2'}_{\mathsf{PSNARG}}(h, \mathsf{root}, w, \pi, \mathsf{open})$ does the following:

1. Receives as input h, a collision resistant hash function, $\mathsf{root} = \mathsf{MC}(h, \hat{x})$ the commitment to the LDE $\hat{x}$, the witness w, the PSNARG proof π, and open, the Merkle openings to Z_x, the supposed value of $\hat{x}$ at coordinates I_x.
2. Checks that $\mathsf{MV}(h, \mathsf{root}, I_x, \mathsf{open}) = 1$.
3. Parses open to get the value Z_x, which supposedly maintains that $Z_x = \hat{x} \mid_{I_x}$. It then runs $\mathsf{LDECompose}$, with the oracle query to $\hat{x} \mid_{I_x}$ being replaced by Z_x, and the oracle access to $\hat{w}$ being implemented in the standard model, and checks that the result is equal to Z as described in π. In other words, the check is $\mathsf{LDECompose}(Z_x,\ \hat{w} \mid_{I_w}) = Z$.
4. Accepts if and only if all previous checks passed.

The second step of $\mathcal{V}^{2'}_{\mathsf{PSNARG}}$ takes time $q \cdot \log(n) \cdot \mathrm{poly}(\lambda) = \mathrm{poly}(m, \log(n), \lambda)$. In the third step of $\mathcal{V}^{2'}_{\mathsf{PSNARG}}$, the computation of $\hat{w} \mid_{I_w}$ takes time $q \cdot |\mathbb{H}|^{\ell_m} \cdot \mathrm{poly}(|\mathbb{H}|, \ell_m) = \mathrm{poly}(m, \log(n), \lambda)$, and the computation of $\mathsf{LDECompose}$ takes time $q \cdot \mathrm{poly}(|\mathbb{H}|, \ell_n + \ell_m) = \mathrm{poly}(m, \log(n), \lambda)$. Therefore, $\mathcal{V}^{2'}_{\mathsf{PSNARG}}$ runs in time $\mathrm{poly}(m, \log(n), \lambda)$, and can thus be proven to accept by a PCP of length $\mathrm{poly}(m, \log(n), \lambda)$.

In conclusion, our verifier is defined by

$$\mathcal{V}'_{\mathsf{PSNARG}}(\mathsf{CRS}, h, \mathsf{root}, w, \pi, \mathsf{open}) = 1 \iff \mathcal{V}^{1}_{\mathsf{PSNARG}}(\mathsf{CRS}, \pi) = 1 \wedge \mathcal{V}^{2'}_{\mathsf{PSNARG}}(h, \mathsf{root}, w, \pi, \mathsf{open}) = 1$$

and runs in time $\mathsf{vc} + \mathrm{poly}(m, \log(n), \lambda)$.

Claim 4.10 (Completeness)**.** *There exists a deterministic* $\mathrm{poly}(n, m, \lambda)$*-time prover* $\mathcal{P}'_{\mathsf{PSNARG}}$ *such that for every* $(x, w) \in \mathcal{R}$*, of lengths* $|x| = n,\ |w| = m$*, it holds that*

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathcal{G}_{\mathsf{PSNARG}}(1^n, 1^m, 1^\lambda) \\ h \leftarrow \mathcal{H}.Gen(1^\lambda)}} \left[\mathcal{V}'_{\mathsf{PSNARG}}(\mathsf{CRS}, h, \mathsf{root}, w, \pi, \mathsf{open}) = 1 \right] = 1,$$

where $\mathsf{root} = \mathsf{MC}(h, \hat{x})$ *and* $(\pi, \mathsf{open}) = \mathcal{P}'_{\mathsf{PSNARG}}(x, w, \mathsf{CRS}, h)$.

Proof. Given (x, w, CRS, h) the prover $\mathcal{P}'_{\mathsf{PSNARG}}$ does the following: It computes $\pi = \mathcal{P}_{\mathsf{PSNARG}}(x, w, \mathsf{CRS})$ and parses π to find the set of indices I for queries in $\widehat{(x, w)}$, and field elements Z. Using $\mathsf{LDECompose}$ and I, it finds I_x, the set indices for queries to $\hat{x}$. It then computes $\mathsf{open} = \mathsf{MO}(h, \hat{x}, I_x)$, and sends the proof (π, open) to $\mathcal{V}'_{\mathsf{PSNARG}}$.

Upon receiving (π, open), the verifier $\mathcal{V}'_{\mathsf{PSNARG}}$ runs $\mathcal{V}^{1}_{\mathsf{PSNARG}}(\mathsf{CRS}, \pi)$ and $\mathcal{V}^{2'}_{\mathsf{PSNARG}}(h, \mathsf{root}, w, \pi, \mathsf{open})$. For the first step, by the completeness of $\mathcal{V}_{\mathsf{PSNARG}}$ and the fact that π was generated by the honest prover $\mathcal{P}_{\mathsf{PSNARG}}$, it holds that $\mathcal{V}^{1}_{\mathsf{PSNARG}}(\mathsf{CRS}, \pi)$ accepts. For the second step, it holds that open is the honest opening to $Z_x = \hat{x} \mid_{I_x}$, which, by the correctness property of the Merkle commitment scheme, implies that $\mathsf{MV}(h, \mathsf{root}, I_x, \mathsf{open}) = 1$. By the definition of $\mathsf{LDECompose}$ it also holds that $\mathsf{LDECompose}(Z_x, \hat{w} \mid_{I_w}) = \mathsf{LDECompose}(\hat{x} \mid_{I_x}, \hat{w} \mid_{I_w}) = Z$, which means $\mathcal{V}^{2'}_{\mathsf{PSNARG}}$ accepts.

Since both steps accept it holds that $\mathcal{V}'_{\mathsf{PSNARG}}$ accepts with probability 1. □

Claim 4.11 (Partial-adaptive soundness)**.** *For every $x \in \{0,1\}^n$, and every $\mathcal{P}^*$ of size $\text{poly}(\lambda)$, it holds that*

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathcal{G}_{\mathsf{PSNARG}}(n,m,1^\lambda)\\ h\leftarrow\mathcal{H}.Gen(1^\lambda)}}\left[\mathcal{V}'_{\mathsf{PSNARG}}(\mathsf{CRS},h,\mathsf{root},w^*,\pi^*,\mathsf{open}^*)=1\wedge(x,w^*)\notin\mathcal{R}\right]\leq\mathsf{negl}(\lambda),$$

where $\mathsf{root}=\mathsf{MC}(h,\hat{x})$, $(w^*,\pi^*,\mathsf{open}^*)=\mathcal{P}^*(\mathsf{CRS},h)$, *and* $|w^*|=m$.

Proof. Assume for contradiction there exists x, a prover $\mathcal{P}^*$ of size $\text{poly}(\lambda)$, and a polynomial p such that

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathcal{G}_{\mathsf{PSNARG}}(n,m,1^\lambda)\\ h\leftarrow\mathcal{H}.Gen(1^\lambda)}}\left[\mathcal{V}'_{\mathsf{PSNARG}}(\mathsf{CRS},h,\mathsf{root},w^*,\pi^*,\mathsf{open}^*)=1\wedge(x,w^*)\notin\mathcal{R}\right]\geq\frac{1}{p(\lambda)},$$

where $\mathsf{root}=\mathsf{MC}(h,\hat{x})$, $(w^*,\pi^*,\mathsf{open}^*)=\mathcal{P}^*(\mathsf{CRS},h)$, and $|w^*|=m$. By the definition of $\mathcal{V}'_{\mathsf{PSNARG}}$ it holds that $\mathcal{V}^1_{\mathsf{PSNARG}}(\mathsf{CRS},\pi^*)=1$ and $\mathcal{V}^{2'}_{\mathsf{PSNARG}}(h,\mathsf{root},w^*,\pi^*,\mathsf{open}^*)=1$.

Consider the case where the holographic claim described in π^* is true. That is, the prover $\mathcal{P}^*$ finds w^* and π^* such that $\widehat{(x,w^*)}\mid_{I^*}=Z^*$, for indices I^* and field elements Z^* described in π^*. Since $\mathcal{V}^2_{\mathsf{PSNARG}}$ only checks the validity of the holographic claim, and in this case the claim is true, it holds that $\mathcal{V}^2_{\mathsf{PSNARG}}\big((x,w^*),\pi^*\big)$ accepts. Since both $\mathcal{V}^1_{\mathsf{PSNARG}}$ and $\mathcal{V}^2_{\mathsf{PSNARG}}$ accept it holds that $\mathcal{V}_{\mathsf{PSNARG}}\big((x,w^*),\mathsf{CRS},\pi^*\big)$ accepts, despite the fact that $(x,w^*)\notin\mathcal{R}$. This contradicts the partial-adaptive soundness of $\mathcal{V}_{\mathsf{PSNARG}}$.

Therefore, consider the case where the holographic claim described in π^* is false. That is, $\widehat{(x,w^*)}\mid_{I^*}\neq Z^*$. Denote by I^*_x and I^*_w the query indices made by $\mathsf{LDECompose}$ to $\hat{x}$ and $\hat{w}$, respectively, when composing for the queries described in I^*. Namely, $\mathsf{LDECompose}(\hat{x}\mid_{I^*_x},\hat{w}\mid_{I^*_w})=\widehat{(x,w^*)}\mid_{I^*}\neq Z^*$. Denote by Z^*_x the alleged value of $\hat{x}\mid_{I^*_x}$ revealed in open^*. Since $\mathcal{V}^{2'}_{\mathsf{PSNARG}}$ accepts it must be that the revealed value passed the holographic check, i.e., $\mathsf{LDECompose}(Z^*_x,\hat{w}\mid_{I^*_w})=Z^*$. Thus, on the one hand $\mathsf{LDECompose}(\hat{x}\mid_{I^*_x},\hat{w}\mid_{I^*_w})\neq Z^*$, while on the other hand $\mathsf{LDECompose}(Z^*_x,\hat{w}\mid_{I^*_w})=Z^*$. This implies that $\hat{x}\mid_{I^*_x}\neq Z^*_x$. Since $\mathcal{V}^{2'}_{\mathsf{PSNARG}}$ accepts, it holds that $\mathsf{MV}(h,\mathsf{root},I^*_x,\mathsf{open}^*)=1$, for $\mathsf{root}=\mathsf{MC}(h,\hat{x})$ the honest commitment to $\hat{x}$ which we assume is given to $\mathcal{V}'_{\mathsf{PSNARG}}$. This means that open^* is a valid opening to a value which is different from the committed value of $\hat{x}$, thus contradicting the binding property of the Merkle commitment scheme.

In either case, the $\text{poly}(\lambda)$-size adversary $\mathcal{P}^*$ breaks the cryptographic tools with non-negligible probability. Therefore, no such adversary exists, and $\mathcal{V}'_{\mathsf{PSNARG}}$ is partially-adaptively sound. □

This completes the proof of Lemma 4.9.

□

4.2.2 The Succinct PCA

Before defining the PCA scheme, we first define an auxiliary pair language $\mathcal{L}'$ which we will use in our construction. Let E be an efficiently encodable and decodable error correcting code ensemble with relative distance $\delta>0$. Denote by E^{-1} the (poly-time) decoding algorithm for E. The language $\mathcal{L}'$ is defined as follows:

$$\mathcal{L}'=\left\{\Big((\mathsf{CRS},h,\mathsf{root}),E\big(w\|\pi_1\|\mathsf{open}\big)\Big):\mathcal{V}'_{\mathsf{PSNARG}}(\mathsf{CRS},h,\mathsf{root},w,\pi_1,\mathsf{open})=1\right\}.$$

In other words, the language $\mathcal{L}'$ is the set of pairs of the following structure:

1. The first component consists of inputs known to $\mathcal{V}'_{\mathsf{PSNARG}}$ explicitly, namely, the common reference string CRS, a collision resistant hash function h, and root a Merkle root which we assume is given to $\mathcal{V}'_{\mathsf{PSNARG}}$.

2. The second component consists of inputs given to $\mathcal{V}'_{\mathsf{PSNARG}}$ by the prover, namely, the witness w, the PSNARG proof π_1, and the opening open. All of these are encoded by the error correcting code E.

The language consists of such pairs which make $\mathcal{V}'_{\mathsf{PSNARG}}$ accept. Note that $\mathcal{L}' \in \mathsf{P}$, since E is efficient and the verifier $\mathcal{V}'_{\mathsf{PSNARG}}$ runs in deterministic time $\mathsf{vc} + \mathrm{poly}(m, \log(n), \lambda)$ given its inputs of size $\mathrm{poly}(m, \log(n), \lambda)$.

Note that we can't use a regular PCP for $\mathcal{L}'$ in our PCA construction, as a PCP verifier reads its entire input, which in particular would read all of w, thus making the PCA construction trivial. We therefore use a PCPP (see Definition 2.7), in which the verifier only has to read a few bits from the oracle to its implicit input. In this case, the PCPP verifier can treat the second component of an instance of $\mathcal{L}'$ as the implicit input, thus only having to read a few bits from the second component.

The PCPP soundness requirement states that if the implicit input is *far* from the set of valid implicit inputs, then the PCPP verifier will reject with constant probability. Note that the set of explicit inputs $\mathcal{L}'_1 = \{x \mid \exists y : (x, y) \in \mathcal{L}'\}$ can potentially be all of Σ^*, as there can exist accepting PSNARG proofs and Merkle openings for every claim. The soundness of the PSNARG and the binding property of the Merkle commitment make it so that finding a valid implicit input is intractable. However, these properties do not rule out the feasibility of finding a string which is *close* to a valid implicit input. Therefore, the second component of $\mathcal{L}'$ is encoded under E, in order ensure that if the adversary found a string which is close to a valid implicit input, that string can be decoded to break the PSNARG or the Merkle commitment.

Since $\mathcal{L}' \in \mathsf{P}$, by Theorem 2.8, for the choice of proximity parameter $\frac{\delta}{2}$, there exists a PCPP for $\mathcal{L}'$ with constant query complexity, logarithmic randomness complexity, and polynomial proof length. Denote by $\mathcal{P}_{\mathsf{PCPP}}$ and $\mathcal{V}_{\mathsf{PCPP}}$ the PCPP prover and verifier for $\mathcal{L}'$, respectively.

We are now ready to present the PCA. The common reference string for the PCA consists of a CRS for the PSNARG, and a collision resistant hash function. Let x be an instance of $\mathcal{R}$. The first half of the PCA proof consists of $E(w \| \pi_1 \| \mathsf{open})$. The second half of the PCA proof is π_2, the PCPP proof for the fact that $\big((\mathsf{CRS}, h, \mathsf{root}), E(w \| \pi_1 \| \mathsf{open})\big) \in \mathcal{L}'$.

Formally, the PCA scheme $\big(\mathcal{G}(1^n, 1^m, 1^\lambda), \mathcal{P}(x, w, \mathsf{CRS}), \mathcal{V}^\pi(x, \mathsf{CRS})\big)$ is defined as follows:

The CRS generator $\mathcal{G}(1^n, 1^m, 1^\lambda)$. The generator $\mathcal{G}(1^n, 1^m, 1^\lambda)$ outputs $\mathsf{CRS}' = (\mathsf{CRS}, h)$ where $\mathsf{CRS} \leftarrow \mathcal{G}_{\mathsf{PSNARG}}(1^n, 1^m, 1^\lambda)$ and $h \leftarrow \mathcal{H}.Gen(1^\lambda)$.

The prover $\mathcal{P}\big(x, w, (\mathsf{CRS}, h)\big)$. Given common reference string (CRS, h), input $x \in \{0,1\}^n$, and corresponding witness $w \in \{0,1\}^m$, the prover $\mathcal{P}\big(x, w, (\mathsf{CRS}, h)\big)$ sends

$$\pi = \Big(E\big(w \| \pi_1 \| \mathsf{open}\big)\Big) \parallel \pi_2$$

where:

- E is the error correcting code mentioned above.

- π_1 is a PSNARG proof for the claim that $(x,w) \in \mathcal{R}$, for the given CRS. Formally, $\pi_1 = \mathcal{P}_{\mathsf{PSNARG}}((x,w),\mathsf{CRS})$.

  The PSNARG proof π_1 specifies a set of indices $I \subseteq \mathbb{F}^\ell$ and a set of values $Z \subseteq \mathbb{F}$, with the associated claim that $\widehat{(x,w)}\,|_I = Z$.

- $\mathsf{open} = \mathsf{MO}(h,\hat{x},I_x)$ is the Merkle tree local opening, with respect to hash function h, revealing the values of $Z_x = \hat{x}\,|_{I_x}$, for I_x the queries to $\hat{x}$ made by $\mathsf{LDECompose}$ for the index set I.

- π_2 is a PCPP for $\mathcal{L}'$, namely, $\pi_2 = \mathcal{P}_{\mathsf{PCPP}}\big(\mathsf{CRS},h,\mathsf{root},E(w\|\pi_1\|\mathsf{open})\big)$.

The verifier $\mathcal{V}^\pi\big(x,(\mathsf{CRS},h)\big)$. Given common reference string (CRS,h), input $x \in \{0,1\}^n$, and oracle access to the proof π, the verifier $\mathcal{V}^\pi\big(x,(\mathsf{CRS},h)\big)$ does the following:

1. Compute $\hat{x}$, and $\mathsf{root} = \mathsf{MC}(h,\hat{x})$.
2. Parse π as $\pi = E(w\|\pi_1\|\mathsf{open})\|\pi_2$, where $|w| = m$, $|\pi_1| = \mathsf{cc}$, $|\mathsf{open}| = \log(n)\cdot\mathrm{poly}(\lambda)$, and $|\pi_2| = \widetilde{O}(\mathsf{vc}) + \mathrm{poly}(m,\log(n),\lambda)$.
3. Run $\mathcal{V}_{\mathsf{PCPP}}$, the PCPP verifier for the language $\mathcal{L}'$, with explicit input $(\mathsf{CRS},h,\mathsf{root})$, implicit input $E(w\|\pi_1\|\mathsf{open})$, proof π_2, and proximity parameter δ. The PCA verifier $\mathcal{V}$ accepts if and only if $\mathcal{V}_{\mathsf{PCPP}}$ accepts. That is, if and only if $\mathcal{V}_{\mathsf{PCPP}}^{E(w\|\pi_1\|\mathsf{open}),\pi_2}\big((\mathsf{CRS},h,\mathsf{root}),\frac{\delta}{2}\big) = 1$.

4.2.3 Analysis of the PCA

Complexity: For the communication complexity, since E is a constant rate code, the proof is of length

$$|\pi| = O\Big(|w| + |\pi_1| + |\mathsf{open}|\Big) + |\pi_2|.$$

It holds that $|w| = m$, $|\pi_1| = \mathsf{cc}$, $|\mathsf{open}| = \lambda\cdot O(\log(n))$, meaning the first part of π is of length $O\big(m + \mathsf{cc} + \lambda\cdot O(\log(n))\big)$, for cc the communication complexity of the PSNARG. Since deciding $\mathcal{L}'$ takes time $\mathsf{vc} + \mathrm{poly}(m,\log(n),\lambda)$, for vc the verifier complexity of the PSNARG, by Theorem 2.8 it holds that $|\pi_2| = \mathrm{poly}(m,\log(n),\lambda,\mathsf{vc})$. In conclusion, the entire proof is of length $|\pi| = \mathrm{poly}(m,\log(n),\lambda,\mathsf{cc},\mathsf{vc})$.

For the prover complexity, the prover $\mathcal{P}$ first has to compute π_1 the PSNARG proof for $\mathcal{R}$, which takes time pc. It then computes $\hat{x}$, root, and open, which takes time $\mathrm{poly}(n,\log(m),\lambda)$. Lastly, it computes π_2 the PCPP proof for $\mathcal{L}'$ which, by Theorem 2.8, takes time $\mathrm{poly}(m,\log(n),\lambda,\mathsf{vc})$. Therefore, the prover runs in time $\mathrm{poly}(n,m,\lambda,\mathsf{pc},\mathsf{vc})$.

For the verifier complexity, the verifier $\mathcal{V}$ first computes the low-degree extension $\hat{x}$ and its Merkle commitment root, which takes time $\mathrm{poly}(n,\log(m),\lambda)$, by Claim 2.11. It then runs $\mathcal{V}_{\mathsf{PCPP}}$, which, by Theorem 2.8, takes time $\mathrm{polylog}\big(\mathsf{vc} + \mathrm{poly}(m,\log(n),\lambda)\big) = \mathrm{polylog}(m,\log(n),\lambda,\mathsf{vc})$. In total, $\mathcal{V}$ runs in time $\mathrm{poly}(n,\log(m),\lambda,\log\mathsf{vc})$.

For the randomness complexity, since $\mathcal{V}$ only uses randomness to run $\mathcal{V}_{\mathsf{PCPP}}$, it has randomness complexity $\log(\mathsf{vc} + \mathrm{poly}(m,\log(n),\lambda)) = O(\log(m) + \log\log(n) + \log(\lambda) + \log(\mathsf{vc}))$.

Lastly, for the query complexity, since $\mathcal{V}$ only queries its oracle when running $\mathcal{V}_{\mathsf{PCPP}}$ for a fixed proximity parameter, by Theorem 2.8, the query complexity is $O(1)$.

Completeness: Let $(x, w) \in \mathcal{R}$, where $|x| = n$ and $|w| = m$. By Claim 4.10, there exists a PSNARG prover strategy $\mathcal{P}'_{\mathsf{PSNARG}}$ such that for every $\mathsf{CRS} \leftarrow \mathcal{G}_{\mathsf{PSNARG}}(1^n, 1^m, 1^\lambda)$ and $h \leftarrow \mathcal{H}.Gen(1^\lambda)$ it holds that $\mathcal{V}'_{\mathsf{PSNARG}}$ accepts the proof given by $\mathcal{P}'_{\mathsf{PSNARG}}$. In other words, it holds that $\mathcal{V}'_{\mathsf{PSNARG}}(\mathsf{CRS}, h, \mathsf{root}, w, \pi_1, \mathsf{open}) = 1$, for $(\pi_1, \mathsf{open}) = \mathcal{P}'_{\mathsf{PSNARG}}(x, w, \mathsf{CRS}, h)$. Therefore, given $(\mathsf{CRS}, h) \leftarrow \mathcal{G}(1^n, 1^m, 1^\lambda)$, the PCA prover $\mathcal{P}$ first runs $(\pi_1, \mathsf{open}) = \mathcal{P}'_{\mathsf{PSNARG}}(x, w, \mathsf{CRS}, h)$. It then runs the PCPP prover for $\mathcal{L}'$ over the resulting instance, namely, it computes the PCPP $\pi_2 = \mathcal{P}_{\mathsf{PCPP}}(\mathsf{CRS}, h, \mathsf{root}, E(w \| \pi_1 \| \mathsf{open}))$. The prover sends an oracle to the proof $\pi = E(w \| \pi_1 \| \mathsf{open}) \| \pi_2$.

Given oracle access to $\pi = E(w \| \pi_1 \| \mathsf{open}) \| \pi_2$, the PCA verifier first computes $\mathsf{root} = \mathsf{MC}(h, \hat{x})$. It then runs the PCPP verifier for $\mathcal{L}'$. By Claim 4.10, the PCPP proof π_2 corresponds to a correct claim. Therefore, by the completeness of the PCPP verifier, it holds that $\mathcal{V}_{\mathsf{PCPP}}$ accepts input $\big((\mathsf{CRS}, h, \mathsf{root}), E(w \| \pi_1 \| \mathsf{open})\big)$ with proof π_2 with probability 1. Thus, the PCA verifier accepts input x with proof π.

Computational soundness: Let x of length $n \in \mathbb{N}$ with $\mathcal{R}(x) = \emptyset$, alleged witness length $m \in \mathbb{N}$, security parameter $\lambda \in \mathbb{N}$, and deterministic[12] malicious prover $\mathcal{P}^*$ of size $\mathrm{poly}(\lambda)$.

Let $(\mathsf{CRS}, h) \leftarrow \mathcal{G}(1^n, 1^m, 1^\lambda)$ and $\pi^* = \mathcal{P}^*(\mathsf{CRS}, h)$. Parse the proof as $\pi^* = \tau_1^* \| \tau_2^*$, for $|\tau_1^*| = O(m + \mathsf{cc} + \lambda \cdot O(\log(n)))$ and $|\tau_2^*| = \mathrm{poly}(m, \log(n), \lambda, \mathsf{vc})$, corresponding to the lengths of the components of an honest proof for input length n and witness length m.

Consider the case where $\Delta(\tau_1^*, \mathsf{Image}(E)) \geq \frac{\delta}{2}$. In this case, it holds that $\Delta(\tau_1^*, \mathcal{L}'(\mathsf{CRS}, h, \mathsf{root})) \geq \frac{\delta}{2}$, for $\mathsf{root} = \mathsf{MC}(h, \hat{x})$, as $\mathcal{L}'(\mathsf{CRS}, h, \mathsf{root}) \subseteq \mathsf{Image}(E)$. Therefore, by its soundness property, the PCPP verifier for $\mathcal{L}'$ with proximity parameter $\frac{\delta}{2}$ rejects input $\big((\mathsf{CRS}, h, \mathsf{root}), \tau_1\big)$, given the proof τ_2^*, with constant probability. By definition, this means that $\mathcal{V}$ rejects with constant probability as required.

Therefore, we assume without loss of generality that $\Delta(\tau_1^*, \mathsf{Image}(E)) < \frac{\delta}{2}$. In this case, the prefix τ_1^* can be decoded efficiently into $w^* \| \pi_1^* \| \mathsf{open}^* = E^{-1}(\tau_1^*)$.

A PCA proof π^* is said to be *convincing* with respect to (CRS, h) if $\pi^* = \tau_1^* \| \tau_2^*$ can be decoded as $w^* \| \pi_1^* \| \mathsf{open}^* = E^{-1}(\tau_1^*)$, such that at least one of the following happens:

1. $\mathcal{V}_{\mathsf{PSNARG}}((x, w^*), \mathsf{CRS}, \pi_1^*) = 1$. In other words, the proof π^* contains a false witness w^*, and an accepting PSNARG proof π_1^* for the statement that $(x, w^*) \in \mathcal{R}$ with respect to CRS, despite the statement being false (since $\mathcal{R}(x) = \emptyset$).

2. $\mathsf{MV}(h, \mathsf{root}, I_x^*, \mathsf{open}^*) = 1$ and $\hat{x} \,|_{I_x^*} \neq Z_x^*$, for $I_x^* \subseteq \mathbb{F}^{\ell_n}$ the set of queries to $\hat{x}$ described in π_1^*, and Z_x^*, the supposed value of $\hat{x} \,|_{I_x^*}$, revealed in open^*. In other words, the proof π^* contains Merkle openings to the indices I_x^* which pass the Merkle verification procedure, but specify values which are not consistent with the honest commitment values.

Based on the definition of convincing proofs discussed above, we define the set BAD, a set of common reference strings for which the adversary outputs a convincing proof. We'll argue that (1) the probability of sampling a PCA common reference string in BAD is negligible, and (2) assuming the sampled common reference string isn't in BAD, the PCA verifier has a soundness error of $\frac{1}{2}$.

Define the set BAD as follows:

$$\mathsf{BAD} = \{(\mathsf{CRS}, h) \colon \pi^* = \mathcal{P}^*(\mathsf{CRS}, h) \text{ is convincing w.r.t. } (\mathsf{CRS}, h)\}.$$

[12] A non-uniform adversary can be assumed to be deterministic without loss of generality. This is because the adversary can be hardwired with the best choice of randomness.

It holds that

$$\Pr_{(\mathsf{CRS},h)\leftarrow\mathcal{G}(1^n,1^m,1^\lambda)}\left[(\mathsf{CRS},h)\in\mathsf{BAD}\right]\leq\mathsf{negl}(\lambda)$$

since if this were not the case then with non-negligible probability the sampled $(\mathsf{CRS},h)\in\mathsf{BAD}$, which means $\pi^*=\mathcal{P}^*(\mathsf{CRS},h)$ would be convincing with non-negligible probability. In this case, at least one of the following holds:

1. The proof π^* contains w^* and π_1^*, an accepting PSNARG proof that "$(x,w^*)\in\mathcal{R}$" despite this being false. This means π^* breaks the partial-adaptive soundness of $\mathcal{V}_{\mathsf{PSNARG}}$.

2. The proof π^* contains a PSNARG proof π_1^* specifying indices I_x^*, and a valid opening open^* to these indices revealing Z_x^*, such that $\hat{x}\mid_{I_x^*}\neq Z_x^*$. This means that π^* breaks the binding property of the Merkle tree commitment.

Therefore, the poly(λ)-sized prover $\mathcal{P}^*$ can be used to break the partial-adaptive soundness of $\mathcal{V}_{\mathsf{PSNARG}}$, or the binding property of the Merkle tree commitment, with non-negligible probability, which is a contradiction. Thus, we conclude that $\Pr_{(\mathsf{CRS},h)\leftarrow\mathcal{G}(1^\lambda)}\left[(\mathsf{CRS},h)\in\mathsf{BAD}\right]\leq\mathsf{negl}(\lambda)$.

Next, we argue that conditioned on $(\mathsf{CRS},h)\notin\mathsf{BAD}$ it holds that $\Pr[\mathcal{V}^{\pi^*}\big(x,(\mathsf{CRS},h)\big)=1]\leq\frac{1}{2}$. Let us therefore fix $(\mathsf{CRS},h)\notin\mathsf{BAD}$. By Claim 4.11 and the choice of $(\mathsf{CRS},h)\notin\mathsf{BAD}$ it holds that $\mathcal{V}'_{\mathsf{PSNARG}}$ rejects its input. Therefore, the claim proven by π_2^* is false. In this case, by the soundness of the PCPP verifier for $\mathcal{L}'$ it holds that the probability of $\mathcal{V}_{\mathsf{PCPP}}$ accepting input $(\mathsf{CRS},h,\mathsf{root}),E(w^*\|\pi_1^*\|\mathsf{open}^*))$ with proof π_2^* is at most $\frac{1}{2}$. Thus, with all but $\mathsf{negl}(\lambda)$ probability over the choice of (CRS,h), the probability of $\mathcal{V}$ accepting x with proof π is at most $\frac{1}{2}$.

This completes the proof of Lemma 4.8.

An alternative PCA construction. In the construction above, the PCA proof consists of $\pi=E(w\|\pi_1\|\mathsf{open})\|\pi_2$, for w the witness, π_1 a PSNARG proof, open a Merkle opening, and π_2 a PCPP proof for $\mathcal{L}'$ defined above (Section 4.2.2).

We outline an alternative PCA construction, which has the advantage of being somewhat simpler, but unfortunately has super-constant query complexity. The construction does the following: Instead of considering $\mathcal{L}'$ as a language in P, consider $\mathcal{L}'$ as an NP relation. Formally, define

$$\mathcal{R}'=\left\{\Big((\mathsf{CRS},h,\mathsf{root}),E(w\|\pi_1\|\mathsf{open})\Big):\mathcal{V}'_{\mathsf{PSNARG}}(\mathsf{CRS},h,\mathsf{root},w,\pi_1,\mathsf{open})=1)\right\}$$

an NP-relation (which is exactly $\mathcal{L}'$). An instance in this relation consists of a CRS, a collision resistant hash function h, and a Merkle commitment root for $\hat{x}$. A witness for this relation consists of w a witness for x, a PSNARG proof π_1, and a Merkle opening open.

Given $(\mathsf{CRS},h)\leftarrow\mathcal{G}$, the PCA proof string for input x will be π, a proof of the membership of $(\mathsf{CRS},h,\mathsf{root})$ in the language corresponding to $\mathcal{R}'$.

We note that potentially, every string can belong in the language corresponding to $\mathcal{R}'$. This is because $\mathcal{V}'_{\mathsf{PSNARG}}$ is only computationally sound, i.e., for every $(\mathsf{CRS},h,\mathsf{root})$, there can exist $E(w^*\|\pi_1^*\|\mathsf{open}^*)$ which make $\mathcal{V}'_{\mathsf{PSNARG}}$ accept. Thus, a PCP proof for $\mathcal{R}'$ will not suffice, as such proofs are virtually guaranteed to exist and might be tractable to find.

Instead, we use a PCP *of knowledge* [BG08, BCGT13]. A PCP of knowledge states that if a prover manages to convince the verifier to accept an instance x, then the prover must know a witness w for x (see Definition 5.3 for a formal definition). In this case, if a polytime prover finds an accepting PCP of knowledge for a "false" instance $(\mathsf{CRS},h,\mathsf{root})$, then he must know a

witness $E(w^* \| \pi_1^* \| \mathsf{open}^*)$, which breaks the computational soundness property of $\mathcal{V}'_{\mathsf{PSNARG}}$. Thus, a construction which simply sends a PCP of knowledge for membership in $\mathcal{R}'$ makes for a sound PCA.

Unfortunately, to the best of our knowledge, existing PCPs of knowledge [BG08, BCGT13] have super-constant query complexity, which results in a PCA with super-constant query complexity. Therefore, in order to build constant-query PCAs, we resort to the (more complicated) construction described above based on PCPPs.

5 Computational Instance Compression

In this section we prove Theorem 3.5 by constructing a general purpose CIC scheme based on the subexponential hardness of LWE. As discussed in Section 3.2, in order to so so it suffices to show that PCAs yield CICs and then apply Theorem 3.3. We establish this connection in Section 5.1. Later, in Section 5.2 we show a complementary result: that CICs imply PCAs.

5.1 From PCAs to CICs

The following lemma shows how to use PCAs to construct CICs.

Lemma 5.1. *Let $\mathcal{R}$ be an* NP *relation with instance length n and witness length m. If $\mathcal{R}$ has a publicly-verifiable constant-query* PCA *with proof length* $\mathrm{poly}(m, \log(n), \lambda)$ *and randomness complexity $r = O(\log(m) + \log\log(n) + \log(\lambda))$, then there exists a computational instance compression scheme from $\mathcal{R}$ to* SAT.

The proof of Theorem 3.5 follows by combining Lemma 5.1 with Theorem 3.3. We remark that our proof of Lemma 5.1 is similar to the construction of standard instance compression from PCPs, as shown by Fortnow and Santhanam [FS11].

Proof of Lemma 5.1. Let $(\mathcal{G}_{\mathsf{PCA}}, \mathcal{P}_{\mathsf{PCA}}, \mathcal{V}_{\mathsf{PCA}})$ be a (non-adaptive) PCA for the NP relation $\mathcal{R}$, with query complexity $q = O(1)$ and randomness complexity $r = r(n, m, \lambda)$. The computational instance compression scheme $(\mathcal{G}, \mathcal{IC}, \mathcal{WT})$ is defined as follows:

The compression generator $\mathcal{G}(1^n, 1^m, 1^\lambda)$. Given instance length n, witness length m, and security parameter λ, the compression generator $\mathcal{G}(1^n, 1^m, 1^\lambda)$ simply outputs $\mathsf{CRS} \leftarrow \mathcal{G}_{\mathsf{PCA}}(1^n, 1^m, 1^\lambda)$.

The instance compressor $\mathcal{IC}(x, \mathsf{CRS})$. Given CRS and $x \in \{0,1\}^n$, the instance compression function $\mathcal{IC}(x, \mathsf{CRS})$ does the following:

1. The variables for the output formula φ correspond to bits of the PCA proof π.
2. For $\rho \in \{0,1\}^r$:
   (a) Compute the decision predicate of the verifier when the input is fixed to x and the random string is fixed to ρ. In other words, $\mathcal{V}_{x,\rho} : \{0,1\}^q \to \{0,1\}$, such that $\mathcal{V}_{x,\rho}(Q) = 1 \iff \mathcal{V}^{\pi}_{\mathsf{PCA}}(x, \mathsf{CRS}; \rho)$, where Q is the result of the queries to π.
   (b) Compute $\varphi_{x,\rho}$ a CNF representation of $\mathcal{V}_{x,\rho}$.
3. Output $\varphi = \bigwedge_{\rho \in \{0,1\}^r} \varphi_{x,\rho}$.

The witness transformer $\mathcal{WT}(x, w, \mathsf{CRS})$. Given CRS and $(x, w) \in \mathcal{R}$, the witness transform $\mathcal{WT}(x, w, \mathsf{CRS})$ simply outputs $w' = \mathcal{P}_{\mathsf{PCA}}(x, w, \mathsf{CRS})$.

Next, we prove that the above scheme satisfies all of the requirements for computational instance compression.

Complexity: First, the compressed formula $\varphi = \mathcal{IC}(x, \mathsf{CRS})$ is indeed a CNF formula and therefore an instance of SAT.

For the output length, observe that φ is obtained by taking the conjunction of 2^r sub-formulas. For every $\rho \in \{0,1\}^r$ it holds that $\mathcal{V}_{x,\rho}$ is a predicate over q variables, and can therefore be expressed by $\varphi_{x,\rho}$, a CNF formula of size $\tilde{O}(2^q)$. Therefore, the entire formula φ is of length $2^r \cdot \tilde{O}(2^q) = \mathrm{poly}(m, \log(n), \lambda)$, where we use here our specific setting of $r = O(\log(m) + \log\log(n) + \log(\lambda))$ and $q = O(1)$.

The generator runs in time $\mathrm{poly}(n, m, \lambda)$. The compression function $\mathcal{IC}$ runs in time $2^r \cdot \mathrm{poly}(n, m, \lambda) = \mathrm{poly}(n, m, \lambda)$. Lastly, the witness transform $\mathcal{WT}$ runs in time $\mathrm{poly}(n, m, \lambda)$.

Completeness: Let $(x, w) \in \mathcal{R}$. By the completeness of $\mathcal{V}_{\mathsf{PCA}}$, it holds that

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathcal{G}_{\mathsf{PCA}}(1^n, 1^m, 1^\lambda) \\ \rho \in_R \{0,1\}^r}} \left[\mathcal{V}^{\pi}_{\mathsf{PCA}}(\mathsf{CRS}, x; \rho) = 1 \right] = 1,$$

for $\pi = \mathcal{P}_{\mathsf{PCA}}(x, w, \mathsf{CRS})$. This means that for every $\rho \in \{0,1\}^r$, the predicate $\mathcal{V}_{x,\rho}$ run by $\mathcal{V}_{\mathsf{PCA}}$ is satisfied by π. Therefore, every one of the CNF formulas $\varphi_{x,\rho}$ is satisfied by π, which means that $\varphi = \bigwedge_{\rho \in \{0,1\}^r} \varphi_{x,\rho}$ is also satisfied by π. Thus, for the transformed witness $w' = \mathcal{WT}(x, w, \mathsf{CRS})$, which is simply $w' = \pi$, it holds that $(\varphi, w') \in \mathcal{R}_{SAT}$.

Computational soundness: Assume for contradiction there exists x with $\mathcal{R}(x) = \emptyset$, a (deterministic) adversary $\mathcal{A}$ of size $\mathrm{poly}(\lambda)$, and a polynomial p, such that for infinitely many λ's

$$\Pr_{\mathsf{CRS} \leftarrow \mathcal{G}(1^n, 1^m, 1^\lambda)} \left[(\varphi, w^*) \in \mathcal{R}_{SAT} \right] \geq \frac{1}{p(\lambda)},$$

for $\varphi = \mathcal{IC}(x, \mathsf{CRS})$ and $w^* = \mathcal{A}(x, \mathsf{CRS})$. Since φ corresponds to the conjunction of all predicates run by $\mathcal{V}_{\mathsf{PCA}}$ over its proof, and since $(\varphi, w^*) \in \mathcal{R}_{\mathrm{SAT}}$, it must be that w^* satisfies every predicate of $\mathcal{V}_{\mathsf{PCA}}$. Therefore, with non-negligible probability $\geq \frac{1}{p(\lambda)}$ over the choice of the CRS, the adversary $\mathcal{A}$ finds a proof $\pi^* = w^*$, which causes the verifier to accept with probability 1.

Thus, we have a contradiction to the soundness of $\mathcal{V}_{\mathsf{PCA}}$, which implies that no such adversary can exist.

□

5.2 From CICs to PCAs

In the previous section we proved that succinct PCAs imply the existence of CICs. Now we show that the converse is also true, albeit with super-constant query complexity.

Lemma 5.2. *If $\mathcal{L} \in \mathsf{NP}$ is computationally compressible to* SAT*, then there exists a succinct publicly-verifiable* PCA *for $\mathcal{L}$ with* $\mathrm{polylog}(m, \log n, \lambda)$ *query complexity.*

We note that the query complexity of this PCA verifier is super-constant. This is because in our construction below, we use the PCPs of [BG08, BCGT13] with the proof-of-knowledge property (defined below), which have polylogarithmic query complexity. We believe there exist constant-query PCPs with the proof-of-knowledge property (e.g., the original [ALM+98] construction), but unfortunately a reference does not seem to appear in the literature.

Proof of Lemma 5.2. In order to show that CICs imply PCAs, we will use PCPs with the *proof-of-knowledge* property.

Definition 5.3 (Proof-of-knowledge)**.** *Let $(\mathcal{P}, \mathcal{V})$ be a* PCP *for $\mathcal{L}$ as described in Definition 2.5. Denote the soundness error of $\mathcal{V}$ by ϵ.*

The verifier $\mathcal{V}$ is said to have the proof-of-knowledge *property if there exists a polynomial-time oracle machine $\mathcal{E}$ such that the following holds: For every x and π, if $\Pr[\mathcal{V}^{\pi}(x) = 1] \geq \epsilon(|x|)$, then there exists w such that $(x, w) \in \mathcal{R}_{\mathcal{L}}$ and $\Pr[\mathcal{E}^{\pi}(x, \frac{1}{\epsilon}) = w] \geq \frac{2}{3}$.*

The following theorem states the existence of such PCPs for NP.

Theorem 5.4 ([BG08], [BCGT13])**.** *Let $\mathcal{L} \in \mathsf{NP}$. There exists $(\mathcal{P}, \mathcal{V})$ a* PCP *for $\mathcal{L} \in \mathsf{NP}$ with negligible soundness error, polylogarithmic query complexity, polynomial proof length, and the* proof-of-knowledge *property.*

Let $(\mathcal{G}, \mathcal{IC}, \mathcal{WT})$ be the CIC scheme and let $(\mathcal{P}_{\mathsf{PCP}}, \mathcal{V}_{\mathsf{PCP}})$ be the PCP described in Theorem 5.4. The PCA $(\mathcal{G}_{\mathsf{PCA}}, \mathcal{P}_{\mathsf{PCA}}, \mathcal{V}_{\mathsf{PCA}})$ is described as follows:

The generator $\mathcal{G}_{\mathsf{PCA}}(1^n, 1^m, 1^\lambda)$. Given instance length n, witness length m, and security parameter λ, the generator $\mathcal{G}_{\mathsf{PCA}}(1^n, 1^m, 1^\lambda)$ outputs $\mathsf{CRS} \leftarrow \mathcal{G}(1^n, 1^m, 1^\lambda)$.

The prover $\mathcal{P}_{\mathsf{PCA}}(x, w, \mathsf{CRS})$. Given CRS, instance $x \in \{0,1\}^n$, and witness $w \in \{0,1\}^m$, the prover $\mathcal{P}_{\mathsf{PCA}}(x, w, \mathsf{CRS})$ does the following:

1. Compute $x' = \mathcal{IC}(x, \mathsf{CRS})$ and $w' = \mathcal{WT}(x, w, \mathsf{CRS})$.
2. Output $\pi = \mathcal{P}_{\mathsf{PCP}}(x', w')$.

The verifier $\mathcal{V}^{\pi}_{\mathsf{PCA}}(x, \mathsf{CRS})$. Given CRS and instance x of length n and witness length m, the verifier $\mathcal{V}^{\pi}_{\mathsf{PCA}}(x, \mathsf{CRS})$ does the following:

1. Compute $x' = \mathcal{IC}(x, \mathsf{CRS})$.
2. Interpret π as a PCP proof for the claim that $x' \in$ SAT, and accept if and only if $\mathcal{V}^{\pi}_{\mathsf{PCP}}(x')$ accepts.

Complexity: As x' and w' are both of length $\text{poly}(m, \log(n), \lambda)$ it holds that the PCP proof π is of length $\text{poly}(|x'|, |w'|) = \text{poly}(m, \log(n), \lambda)$, and therefore succinct.

The PCA prover $\mathcal{P}_{\mathsf{PCA}}$ runs the compression scheme and the witness transform, which take time $\text{poly}(n, m, \lambda)$. It then runs $\mathcal{P}_{\mathsf{PCP}}$ over a SAT instance of length $|x'| = \text{poly}(n, m, \lambda)$, and witness length $|w'| = \text{poly}(n, m, \lambda)$, which takes a total of $\text{poly}(n, m, \lambda)$.

The PCA verifier $\mathcal{V}_{\mathsf{PCA}}$ runs the compression scheme and then runs $\mathcal{V}_{\mathsf{PCP}}$ over an input of length $\text{poly}(m, \log n, \lambda)$, which takes time $\text{poly}(m, \log n, \lambda)$. As mentioned in Theorem 5.4, the query complexity of $\mathcal{V}_{\mathsf{PCA}}$ is $\text{polylog}(|x'|, |w'|) = \text{polylog}(m, \log n, \lambda)$.

Completeness: Let $x \in \mathcal{L}$. By the completeness of the of CIC, for every sampled $\mathsf{CRS} \leftarrow \mathcal{G}_{\mathsf{PCA}}(1^n, 1^m, 1^\lambda)$ it holds that $(x', w') = \big(\mathcal{IC}(x, \mathsf{CRS}), \mathcal{WT}(x, w, \mathsf{CRS})\big) \in \mathcal{R}_{SAT}$, meaning π is a valid PCP proof for $x' \in \text{SAT}$. Therefore,

$$\Pr_{\mathsf{CRS} \leftarrow \mathcal{G}_{\mathsf{PCA}}(1^n, 1^m, 1^\lambda)} \left[\mathcal{V}^{\pi}_{\mathsf{PCA}}(x, \mathsf{CRS}) = 1 \right] = 1,$$

for $\pi = \mathcal{P}_{\mathsf{PCA}}(x, w, \mathsf{CRS})$.

Computational soundness: Assume for contradiction there exists $x \notin \mathcal{L}$ of length n, alleged witness length m, and a deterministic malicious prover $\mathcal{P}^*_{\mathsf{PCA}}$ of size $\text{poly}(\lambda)$, such that

$$\Pr_{\mathsf{CRS} \leftarrow \mathcal{G}_{\mathsf{PCA}}(1^n, 1^m, 1^\lambda)} \left[\mathcal{V}^{\pi^*}_{\mathsf{PCA}}(x, \mathsf{CRS}) = 1 \right] = \Pr_{\mathsf{CRS} \leftarrow \mathcal{G}_{\mathsf{PCA}}(1^n, 1^m, 1^\lambda)} \left[\mathcal{V}^{\pi^*}_{\mathsf{PCP}}(x') = 1 \right] \geq \frac{1}{2},$$

for $x' = \mathcal{IC}(x, \mathsf{CRS})$, and $\pi^* = \mathcal{P}^*_{\mathsf{PCA}}(\mathsf{CRS})$. Let $\mathcal{E}$ be the extractor described in Theorem 5.4 for $\epsilon = \frac{1}{2}$. Consider $\mathcal{A}$, an adversary to break the CIC in the following manner:

1. Given CRS and compressed instance x', run $\pi^* = \mathcal{P}^*_{\mathsf{PCA}}(x', \mathsf{CRS})$.
2. Since π^* makes $\mathcal{V}_{\mathsf{PCP}}$ accept with probability $\geq \epsilon$, compute $w' \leftarrow \mathcal{E}^{\pi^*}(x', \frac{1}{\epsilon})$.
3. Output $w^* = w'$.

It holds that

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathcal{G}_{\mathsf{PCA}}(1^n, 1^m, 1^\lambda) \\ w^* \leftarrow \mathcal{A}(x, \mathsf{CRS})}} \left[(x', w^*) \in \mathcal{R}_{SAT} \right] = \Pr_{\substack{\mathsf{CRS} \leftarrow \mathcal{G}_{\mathsf{PCA}}(1^n, 1^m, 1^\lambda) \\ w' \leftarrow \mathcal{E}(x', 1^m)}} \left[(x', w') \in \mathcal{R}_{SAT} \right] \geq \frac{1}{2},$$

for $x' = \mathcal{IC}(x, \mathsf{CRS})$, which contradicts the soundness of the CIC.

□

Acknowledgements

We thank Justin Holmgren, Yuval Ishai, Alex Lombardi, and Omer Paneth for very useful discussions and comments.

References

[AIV92] Sanjeev Arora, Russel Impagliazzo, and Umesh Vazirani. Relativizing versus nonrelativizing techniques: the role of local checkability, 1992.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[App17] Benny Applebaum. Exponentially-hard gap-csp and local PRG via local hardcore functions. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 836–847. IEEE Computer Society, 2017.

[ARW17] Amir Abboud, Aviad Rubinstein, and R. Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 25–36. IEEE Computer Society, 2017.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 106–115. IEEE Computer Society, 2001.

[BBH+19] James Bartusek, Liron Bronfman, Justin Holmgren, Fermi Ma, and Ron D. Rothblum. On the (in)security of Kilian-based SNARGs. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 522–551. Springer, 2019.

[BCGT13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. On the concrete efficiency of probabilistically-checkable proofs. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 585–594. ACM, 2013.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, 2016.

[BDFH09] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008.

[BGH+05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Short PCPs verifiable in polylogarithmic time. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 120–134. IEEE Computer Society, 2005.

[BGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006.

[BHK17] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 474–482. ACM, 2017.

[BS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1082–1090. ACM, 2019.

[CGL+19] Lijie Chen, Shafi Goldwasser, Kaifeng Lyu, Guy N. Rothblum, and Aviad Rubinstein. Fine-grained complexity meets IP = PSPACE. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1–20. SIAM, 2019.

[CJJ21] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for P from LWE. Manuscript, 2021.

[CKLR11] Kai-Min Chung, Yael Tauman Kalai, Feng-Hao Liu, and Ran Raz. Memory delegation. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 151–168. Springer, 2011.

[Del16] Holger Dell. AND-compression of NP-complete problems: Streamlined proof and minor observations. *Algorithmica*, 75(2):403–423, 2016.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

[DR06] Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM J. Comput.*, 36(4):975–1024, 2006.

[Dru15] Andrew Drucker. New limits to classical and quantum instance compression. *SIAM J. Comput.*, 44(5):1443–1479, 2015.

[FGL+96] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996.

[For94] Lance Fortnow. The role of relativization in complexity theory. *Bull. EATCS*, 52:229–243, 1994.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[FS11] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011.

[GGH11] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, volume 6650 of *Lecture Notes in Computer Science*, pages 30–39. Springer, 2011.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4):205–214, 1998.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 102–113. IEEE Computer Society, 2003.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4):27:1–27:64, 2015.

[GVW02] Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Comput. Complex.*, 11(1-2):1–53, 2002.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 99–108. ACM, 2011.

[Hås01] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[HLR21] Justin Holmgren, Alex Lombardi, and Ron D. Rothblum. Fiat-Shamir via list-recoverable codes (or: Parallel repetition of GMW is not zero-knowledge). Cryptology ePrint Archive, Report 2021/286, 2021. https://eprint.iacr.org/2021/286.

[HN10] Danny Harnik and Moni Naor. On the compressibility of NP instances and cryptographic applications. *SIAM J. Comput.*, 39(5):1667–1713, 2010.

[JKKZ20] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Zhang. SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE. *IACR Cryptol. ePrint Arch.*, 2020:980, 2020.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 723–732. ACM, 1992.

[KP16] Yael Tauman Kalai and Omer Paneth. Delegating RAM computations. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 91–118, 2016.

[KPY19] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1115–1124. ACM, 2019.

[KPY20] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. Delegation with updatable unambiguous proofs and ppad-hardness. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 652–673. Springer, 2020.

[KR08] Yael Tauman Kalai and Ran Raz. Interactive PCP. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 536–547. Springer, 2008.

[KR09] Yael Tauman Kalai and Ran Raz. Probabilistically checkable arguments. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 143–159. Springer, 2009.

[KRR+20] Inbar Kaslasi, Guy N. Rothblum, Ron D. Rothblum, Adam Sealfon, and Prashant Nalini Vasudevan. Batch verification for statistical zero knowledge proofs. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 139–167. Springer, 2020.

[KRV21] Inbar Kaslasi, Ron D. Rothblum, and Prashant Nalini Vasudevan. Public-coin statistical zero-knowledge batch verification against malicious verifiers. *IACR Cryptol. ePrint Arch.*, 2021:233, 2021.

[KST21] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. Cryptology ePrint Archive, Report 2021/370, 2021. https://eprint.iacr.org/2021/370.

[Mer87] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, 1987.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 89–114. Springer, 2019.

[Reg10] Oded Regev. The learning with errors problem (invited survey). In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, USA, June 9-12, 2010*, pages 191–204. IEEE Computer Society, 2010.

[Rot09] Guy N. Rothblum. *Delegating computation reliably: paradigms and constructions*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2009.

[RR20a] Noga Ron-Zewi and Ron D. Rothblum. Local proofs approaching the witness length [extended abstract]. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 846–857. IEEE, 2020.

[RR20b] Guy N. Rothblum and Ron D. Rothblum. Batch verification and proofs of proximity with polylog overhead. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 108–138. Springer, 2020.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 49–62. ACM, 2016.

[RRR18] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Efficient batch verification for UP. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 22:1–22:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[Tha] Justin Thaler. http://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.html.

[Wil15] Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPIcs*, pages 17–29. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[Zim01] Marius Zimand. Probabilistically checkable proofs the easy way. *Electron. Colloquium Comput. Complex.*, 8(27), 2001.

A Securely Instantiating Fiat-Shamir of Kilian's Protocol

Our sublinear SNARG, described in Section 4.2.1, may seem (and is indeed) related to an application of Fiat-Shamir [FS86] to Kilian's protocol [Kil92]. At first glance this may seem perplexing since there is a line of research proving the (in)security of applying Fiat-Shamir over arguments [Bar01, GK03, BBH+19]. Of particular relevance is the result of Bartusek et al. who show that applying Fiat-Shamir to Kilian's protocol is not sound if one uses either the collision resistant hash function or PCP (or more precisely IOP) in a generic way [BBH+19].

To understand how we escape this paradigm, let us briefly recall Kilian's construction. In short, the 4-message protocol of Kilian [Kil92] for $\mathcal{L} \in \mathsf{NP}$ has the following structure:

1. The verifier $\mathcal{V}_{\mathsf{Kilian}}$ samples h, a collision-resistant hash function, and sends h to the prover.
2. Given input x, the prover $\mathcal{P}_{\mathsf{Kilian}}$ computes π, the PCP proving that $x \in \mathcal{L}$. It then computes root, the Merkle hash for π using h, and sends it to the verifier.
3. The verifier $\mathcal{V}_{\mathsf{Kilian}}$ sends the prover its random query indices for π.
4. The prover $\mathcal{P}_{\mathsf{Kilian}}$ sends open, the opening corresponding to the query indices, using the root and h.

   The verifier checks the consistency of open, and runs the PCP predicate with respect to the revealed values. It accepts if and only if both checks passed.

The main conceptual difference between applying Fiat-Shamir to the above protocol, vs. our PCA construction is that in the latter the verifier has a *trusted* hash of the input (which it generated by itself in a preprocessing step).

Consider therefore, the following setting: The verifier $\mathcal{V}_{\mathsf{Kilian}}$ is assumed to have root, the root corresponding to the *honest* proof π with respect to its input x. Specifically, for $x \in \mathcal{L}$ and w its witness, the verifier receives $\mathsf{root} = \mathsf{MC}(h, \pi)$, for $\pi = \mathcal{P}_{\mathsf{PCP}}(x, w)$. For $x \notin \mathcal{L}$, the verifier receives root the commitment of $\pi = \mathcal{P}_{\mathsf{PCP}}(x, 0^m)$, for m the alleged witness length for x.

For completeness, consider $(x, w) \in \mathcal{R}_{\mathcal{L}}$. The verifier has root, and generates indices I and sends them to the prover. The prover computes the opening open to indices I of π, which passes the Merkle verification procedure, and satisfies the PCP predicate, thus making the verifier accept.

For soundness, let $x \notin \mathcal{L}$ and $\mathcal{P}^*_{\mathsf{PCP}}$ a malicious prover. With high probability, querying $\pi = \mathcal{P}_{\mathsf{PCP}}(x, 0^m)$ will make the verifier reject. Therefore, the malicious prover $\mathcal{P}^*_{\mathsf{PCP}}$ cannot send the honest opening to π at I. Since the verifier already has root, the malicious prover can only cheat by searching for a false opening open^* to root which passes the Merkle verification procedure. This requires breaking the binding property of the Merkle commitment scheme, which means the prover

will fail, and thus the verifier will reject. Therefore, in this setting, applying the Fiat-Shamir transform to Kilian's protocol is indeed sound (but of course the assumption that the verifier has access to a trusted Merkle root is hard to justify in the usual context).