





Succinct Interactive Oracle Proofs: Applications and Limitations

Shafik Nassar^(✉)  and Ron D. Rothblum 

Technion, Haifa, Israel

{shafiknassar, rothblum}@cs.technion.ac.il

Abstract. *Interactive Oracle Proofs* (IOPs) are a new type of proof-system that combines key properties of interactive proofs and PCPs: IOPs enable a verifier to be convinced of the correctness of a statement by interacting with an untrusted prover while reading just a few bits of the messages sent by the prover. IOPs have become very prominent in the design of efficient proof-systems in recent years.

In this work we study *succinct IOPs*, which are IOPs in which the communication complexity is polynomial (or even linear) in the original witness. While there are strong impossibility results for the existence of succinct PCPs (i.e., PCPs whose length is polynomial in the witness), it is known that the rich class of NP relations that are decidable in small space have succinct IOPs. In this work we show both new applications, and limitations, for succinct IOPs:

- First, using one-way functions, we show how to compile IOPs into zero-knowledge *proofs*, while nearly preserving the proof length. This complements a recent line of work, initiated by Ben Sasson *et al.* (TCC, 2016B), who compile IOPs into super-succinct zero-knowledge *arguments*.

Applying the compiler to the state-of-the-art succinct IOPs yields zero-knowledge proofs for bounded-space NP relations, with communication that is nearly equal to the original witness length. This yields the shortest known zero-knowledge proofs from the minimal assumption of one-way functions.

- Second, we give a barrier for obtaining succinct IOPs for more general NP relations. In particular, we show that if a language has a succinct IOP, then it can be decided in *space* that is proportionate only to the witness length, after a bounded-time probabilistic preprocessing. We use this result to show that under a simple and plausible (but to the best of our knowledge, new) complexity-theoretic conjecture, there is no succinct IOP for CSAT.

1 Introduction

The study of proof-systems has played an incredibly influential role in the development of theoretical computer science at large and complexity theory and cryptography in particular. Some of the most important results, concepts and

open problems in this field revolve around efficient proof-systems. These include the $P \stackrel{?}{=} NP$ question, results such as the $IP = PSPACE$ and PCP theorems, and the notion of *zero-knowledge proofs*.

Interactive Oracle Proofs (IOP) [BCS16,RRR21], are a recently proposed type of proof-system that is playing an important role in the development of highly efficient, even practical, proofs. An IOP can be viewed as an interactive analogue of a PCP, that is, an interactive protocol in which the prover can send long messages, but the verifier only reads a few bits from each of the prover's messages. A recent exciting line of research initiated by Ben Sasson *et al.* [BCS16] (following [Kil92,Mic00]) compiles highly efficient IOPs into highly efficient zero-knowledge argument-systems that are now also being developed and deployed in practice.

One of the intriguing aspects of IOPs is that, by leveraging interaction, they allow us to circumvent some inherent efficiency barriers of PCPs (in which the interaction is just a single message). In particular, it has been known for over a decade [KR08,FS11] that SAT (the Boolean satisfiability problem) does *not* have a PCP whose length is polynomial *only* in the length of the original satisfying assignment (assuming the polynomial hierarchy does not collapse). In contrast, Kalai and Raz [KR08] showed that SAT (and more generally any bounded depth NP relation) does have a succinct IOP.¹ A more recent work of Ron-Zewi and Rothblum [RR20] gives such a succinct IOP for SAT, and more generally any bounded-space relation, in which the communication approaches the length of the unencoded witness.

In this work, we aim to better understand the limitations, and applications, of succinct IOPs. In particular we would like to understand the following two questions:

1. The results of [KR08,RR20] give succinct IOPs for either bounded-space or bounded-depth relations. But what about general² NP relations? For example, does the *circuit satisfiability problem* (CSAT) have a succinct IOP, or is the limitation to small depth/space in [KR08,RR20] inherent?
2. So far the applicability of succinct IOPs has been limited. This seems to mainly be due to the fact that the main bottleneck in the compilers of IOPs to efficient arguments is not the communication complexity.³ This begs the question of what other applications can succinct IOPs be used for?

¹ Actually, [KR08] consider the model of *interactive PCP*, which in modern terminology, is a special-case of an IOP.

² Note that even though SAT is NP-complete, a succinct IOP for SAT does not automatically yield a succinct IOP for every NP relation, because the Cook-Levin theorem produces a formula whose length is polynomial in the complexity of the original NP relation, rather than just its witness.

³ The key bottlenecks seem to be the prover's runtime and the communication complexity of the resulting argument-system. The IOP's communication complexity is a lower bound on prover runtime. The argument's communication complexity only has a logarithmic dependence on the IOPs communication.

1.1 Our Results

1.1.1 Succinct Zero-Knowledge Proofs from Succinct IOPs

As our first main result, we show how to compile IOPs into zero-knowledge *proofs*, under the minimal [OW93] assumption of one-way functions. We consider IOPs which consist of two phases: there is an *interaction phase* where the prover sends messages and the verifier only replies with random coins (without reading anything), then there is a *local computation phase* where the verifier queries the prover messages and applies a *decision predicate* on those query values (see Definitions 3 and 4). Our compiler transforms IOPs into zero-knowledge proofs in a way that preserves the communication complexity up to an additive factor which depends on the size of the verifier’s decision predicate (as well as the security parameter).

Theorem 1 (Informally Stated, see Theorem 5). *Suppose the language \mathcal{L} has an IOP with communication complexity cc and where the verifier’s decision predicate has complexity γ . If one-way functions exist, then \mathcal{L} has a zero-knowledge proof of length $cc + \text{poly}(\gamma, \lambda)$, where λ is the security parameter. The zero-knowledge proof has perfect completeness and negligible soundness error.*

The proof of Theorem 1 is a relatively simple extension of the classical “notarized envelopes” technique of Ben-Or *et al.* [BGG+88]. Indeed, our main contribution is in observing that this technique can be adapted to the IOP setting in a manner that very nearly preserves the communication complexity.

Using the compiler of Theorem 1, we are able to derive the shortest known zero-knowledge proofs that are based on one-way functions. In particular, the aforementioned work of Ron-Zewi and Rothblum [RR20] gives an IOP construction with proof length that approaches the witness length for any bounded space NP relation. This class of relations includes a large variety of natural NP relations such as SAT, k -Clique, k -Coloring, etc. We show that their IOP has very low verifier decision complexity. Applying the transformation of Theorem 1 to it we obtain a zero-knowledge proof for any bounded space NP relation, where the communication complexity in this zero-knowledge proof approaches the witness length.

Corollary 1 (Informally Stated, see Theorem 7). *Let \mathcal{R} be an NP relation that can be computed in polynomial time and bounded polynomial space. If one-way functions exist, then for any constants $\beta, \gamma \in (0, 1)$, there exists a (public-coin) zero-knowledge proof for \mathcal{R} with perfect completeness, negligible soundness error and proof length $(1 + \gamma) \cdot m + n^\beta \cdot \text{poly}(\lambda)$, where n is the instance length, m is the witness length and λ is the security parameter.*

Corollary 1 constitutes the shortest known general-purpose zero-knowledge proofs under the minimal assumption of one-way functions. Prior to our work, the shortest zero-knowledge proofs, that were based on one-way functions, had communication complexity $\tilde{O}(m)$ [IKOS09, GKR15]. In contrast, under the much stronger assumption of *fully-homomorphic encryption*, Gentry *et al.* [GGI+15]

constructed zero-knowledge proofs that are better than those of Corollary 1 in two ways: first, they achieve an even shorter communication complexity of $m + \text{poly}(\lambda)$ and second, the result holds for *any*⁴ NP relation, whereas Corollary 1 is restricted to bounded-space relations.

The fact that the transformation from Theorem 1 can potentially work on other succinct IOP constructions, further motivates the study of succinct IOPs, their capabilities and limitations.

1.1.2 Limitations of Succinct IOPs

Given the succinct IOP constructions of [KR08,RR20], as well as known limitations of standard interactive proofs, it is natural to wonder whether the restriction of the [KR08,RR20] succinct IOPs to bounded depth/space relations is inherent. That is, does a succinct IOP for a given relation imply that the relation can be decided in small space?

The immediate, albeit highly unsatisfactory, answer to the above question is (most likely) negative: the class BPP has a trivial succinct IOP (in which the prover sends nothing and the verifier decides by itself) but is conjectured not to be contained in any fixed-polynomial space. So perhaps succinct IOPs are limited to relations computable in small-space *or* for which the corresponding language is in BPP? Unfortunately, the answer is again (likely) negative: consider the NP relation $\mathcal{R} = \{(x, w) : x \in \mathcal{L} \wedge (x, w) \in \mathcal{R}_{\text{SAT}}\}$, where \mathcal{L} is some P-complete problem.⁵ Using [RR20], it is clear that \mathcal{R} has a succinct IOP despite the fact that it is unlikely to be solvable in small space and the corresponding language is unlikely to be in BPP (assuming $\text{NP} \not\subseteq \text{BPP}$).

We show that the above example essentially serves as the limit of succinct IOPs. This negative result is based on a complexity-theoretic conjecture, which, while to the best of our knowledge is new, seems quite plausible.

In more detail, we prove that if an NP relation $\mathcal{R}_{\mathcal{L}}$, corresponding to a language \mathcal{L} (i.e., $\mathcal{L} = \{x : \exists w, (x, w) \in \mathcal{R}_{\mathcal{L}}\}$), has a succinct IOP, then there exists a small space algorithm *with probabilistic bounded time preprocessing* that can decide \mathcal{L} .

Theorem 2 (Informally stated, see Theorem 4). *If a language \mathcal{L} has a k -round IOP with communication complexity cc and query complexity qc , then \mathcal{L} can be decided by a $O(cc + k \log cc)$ -space algorithm with probabilistic $(2^{qc} \cdot \text{poly}(n, 2^{k \log(cc)}))$ -time preprocessing.*

By a s -space algorithm with t -time (probabilistic) preprocessing, we mean a (probabilistic) Turing machine that first runs in time t and outputs some intermediate state c (of size at most t). From there on a second Turing machine,

⁴ For this result, [GGI+15] need full-fledged (rather than leveled) fully-homomorphic encryption, which are known only assuming a circular-security assumption on LWE (see, e.g., [Bra19]) or via indistinguishability obfuscation [CLTV15].

⁵ As usual, by P-completeness we refer to log-space reductions. Such languages are conjectured not to be solvable in small space, see [Sma14] for further discussion.

which uses s -space, can continue the computation, viewing c as its (read-only) input tape (see Definition 12 for the formal definition). We emphasize that the restriction on the second machine is only that it runs in s space (and in particular can run for 2^s time).

Infeasibility of Succinct IOP for NP. Using Theorem 2, we argue that the existence of succinct IOPs for all of NP would have unexpected, and (arguably) unlikely, repercussions.

For example, consider the relation $\mathcal{R}_{\text{CSAT}}$, consisting of all satisfiable Boolean circuits and their corresponding satisfying assignment. Given Theorem 2, the existence of a succinct IOP for $\mathcal{R}_{\text{CSAT}}$ with, say, constant rounds and logarithmic query complexity, would mean that the satisfiability of a circuit of size n on m input bits, can be decided by an algorithm that first runs in time $\text{poly}(n, m)$ time but from there can do arbitrary $\text{poly}(m)$ -space computations. We find the existence of such an algorithm unlikely and in particular point out that the straightforward decision procedure for CSAT enumerates all possible assignments, which takes space m , but also needs to check that each assignment satisfies the given circuit, which, in general, seems to require additional space n , which our $\text{poly}(m)$ -space algorithm does not have at this point. In other words, the straightforward algorithm needs to evaluate the circuit for each one of the candidate assignments, whereas our preprocessing model only allows for a polynomial number of evaluations (which happen a priori). Taking things a little further, we conjecture that even probabilistic quasi-polynomial time preprocessing would not be sufficient, and taking things to an extreme, it is (arguably) unlikely that $(2^{o(m)} \cdot \text{poly}(n))$ -time preprocessing is sufficient. A more elaborate discussion can be found in the full version [NR22]. A parameterized version of our conjecture is stated below:

Conjecture 1. For a function class \mathcal{T} , the conjecture states that CSAT for circuits of size n over m input bits cannot be solved by an algorithm that uses $\text{poly}(m)$ space and $t(n, m)$ -time probabilistic preprocessing, for any $t \in \mathcal{T}$.

We get stronger bounds on succinct IOPs as we make the function class larger. Three interesting regimes are stated in the following corollary (ordered from the weakest bound):

Corollary 2. *Assuming Conjecture 1 we have:*

- With $t(n, m) = \text{poly}(n)$, there is no succinct IOP for $\mathcal{R}_{\text{CSAT}}$ with a constant number of rounds and $O(\log n)$ query complexity.
- With $t(n, m) = 2^{\text{polylog}(m)} \cdot \text{poly}(n)$, there is no succinct IOP for $\mathcal{R}_{\text{CSAT}}$ with a $\text{polylog}(m)$ rounds and $\text{polylog}(m) + O(\log n)$ query complexity.
- With $t(n, m) = 2^{o(m)} \cdot \text{poly}(n)$, there is no succinct IOP for $\mathcal{R}_{\text{CSAT}}$ with a $o(\frac{m}{\log m})$ rounds and $o(m) + O(\log n)$ query complexity.

1.2 Related Works

Lower Bounds for IPs and IOPs. Goldreich and Hastad [GH98] showed how to transform IPs to probabilistic algorithms that run in time exponential in the

bits sent by the prover. Goldreich *et al.* [GVW02] showed limits on the communication complexity of interactive proofs. In particular, their results show that it is unlikely that NP-complete languages have interactive proofs with communication that is significantly shorter than the witness length. Berman *et al.* [BDRV18] showed that extremely short zero-knowledge proofs imply the existence of public-key encryption. Chiesa and Yogev [CY20] show that if a language has an IOP with very good soundness relative to its query complexity then it can be decided by a small space algorithm. This result is incomparable to Theorem 2, which shows that languages which have IOPs with short *communication* can be computed in small space (with preprocessing).

Minimizing Communication in Zero-Knowledge Proofs. Significant effort has been put into minimizing the proof length of zero-knowledge. Under the assumption of one-way functions, Ishai *et al.* [IKOS09] constructed “constant-rate” zero-knowledge proofs for all NP relations, i.e., the proof length is constant in the size of the circuit that computes the relation. For AC_0 circuits, [IKOS09] presents a zero-knowledge proof that is quasi-linear in the *witness length* and [KR08] presents a zero-knowledge proof that is polynomial in the witness length for constant depth formulas. Goldwasser *et al.* [GKR15] significantly improved the latter and showed a similar result for all of (log-space uniform) NC, again under the minimal assumption of one-way functions. As previously mentioned, using fully-homomorphic encryption, Gentry *et al.* [GGI+15] constructed zero-knowledge proofs with communication that is larger than the witness by only a small additive factor.

Another approach to minimize the proof length is to relax the notion of soundness and settle for computationally-sound proof systems, known as *arguments*. Kilian [Kil92] and Micali [Mic00] constructed extremely efficient zero-knowledge argument systems in which the communication is merely *polylogarithmic* in the witness size. Improving the latter protocol has been the focus of a major line of research in recent years. However, we stress that in this work, we focus on proof systems with statistical soundness - that is, soundness is guaranteed even against computationally unbounded cheating provers.

1.3 Our Techniques

First, in Sect. 1.3.1 we discuss our compiler for zero-knowledge proofs from IOPs. In Sect. 1.3.2, we discuss our techniques for compiling IOPs to small space algorithms with bounded time preprocessing.

1.3.1 ZKPs from IOPs

Notarized Envelopes. The zero-knowledge proof of Theorem 1 is constructed using the “notarized envelopes” technique, first introduced in [BGG+88]. We start with a high-level overview of their compiler from interactive proofs to zero-knowledge proofs. The compiler, which is applicable to any public-coin interactive proof, proceeds by emulating the original protocol but instead of having the prover send its messages in the clear (which would likely violate zero-knowledge),

the prover sends (statistically binding) commitments to its messages. Leveraging the fact that the protocol is *public-coin*, the verifier does not have to read the prover messages before responding with its own random coins and so the interaction can progress.

At the end of the interaction phase, the verifier would like to actually read the messages that the prover committed to and to compute some predicate on the transcript. The key observation is that the latter is an NP statement: since the verifier is a polynomial-time machine and the computation in the end is deterministic (since the coins have been tossed already), then the commitments to the prover messages and the verifier's randomness define an NP statement, with the NP witness being the decommitments of those messages; given those decommitments, it is straightforward to decide the predicate.

At this point [BGG+88] use the fundamental zero-knowledge proof for NP [GMW86], so that the prover does not have to actually decommit (and reveal its messages) in order to prove the correctness of the NP statement. Rather can convince the verifier in *zero-knowledge* that had it indeed revealed the messages, the verifier would have accepted.

Locally Notarized Envelopes. The overhead of using the notarized envelopes technique depends on the overhead that the commitments introduce as well as the cost of the zero-knowledge proof for the final NP statement. When applied to a traditional interactive proof, this overhead depends on the total length of communication from the prover to the verifier.

For IOPs though, the overhead can be much smaller; the IOP verifier is not interested in the entire transcript but instead only in the locations of its queries. Therefore, if the prover uses a commitment that allows for a local decommitment for each bit, then, intuitively at least, the size of the NP statement should depend only on the number of queries (and the security parameter).

In the computationally-sound setting such a succinct commitment with local openings is obtained by using a Merkle tree. In our context however, we need a *statistically binding* commitment. To minimize the overhead of the commitment and achieve the desired locality, we use a pseudo-random function (PRF) as a stream cipher. In more detail, the prover first commits to the PRF seed and then uses the PRF as a pseudorandom one-time pad to all of the messages. This yields a length preserving commitment scheme that is also statistically binding, where the overhead is merely the additional commitment to the PRF seed. Although this commitment scheme does *not* support local openings, it allows us to define an NP statement that depends only on the (short) seed and the desired bits which the verifier would like to query.

IOP Compactness. The size of the NP statement depends also on the size of the computation that the verifier performs once it receives the queries. Naively, we can say that the size of the mentioned verifier computation is bounded by the running time of the verifier, and thus the NP statement is polynomially related to the running time of the verifier. However, we distinguish between the total running time of the verifier and the size of the computation it performs offline after receiving the answers to its queries. We refer to the size of the

offline computation as the “compactness” of the IOP (verifier). We show that the additive overhead of using the locally notarized envelopes technique with IOPs is polynomial in the compactness of the IOP and the security parameter, thus presenting a potentially efficient transformation from IOPs to ZKPs.

We also analyze an IOP for bounded space NP relations from a previous work [RR20] and show that it is indeed very compact, thus achieving a very efficient ZKP for bounded space NP relations as per Corollary 1.

1.3.2 Infeasibility of Succinct IOPs

To show lower bounds for succinct IOPs, it is tempting to utilize existing lower bounds for either interactive proofs or PCPs. Trying to do so we run into the following difficulty. First, observe that CSAT has a very succinct interactive proof - the prover simply sends the witness! Thus, we cannot employ generic lower bounds for interactive proofs (such as $IP \subseteq PSPACE$ and similar extensions). Likewise, we cannot use the known lower bounds for succinct PCPs since, for example, the main lower bound that is known, due to Fortnow and Santhanam [FS11], also rules out a PCP for SAT, whereas by [KR08,RR20] we know that SAT has a succinct IOP.

Nevertheless, our approach is inspired by Fortnow and Santhanam [FS11], who showed that a succinct PCP for SAT collapses the polynomial hierarchy. Their proof goes through an interesting intermediate step: they show that a succinct PCP for any language implies a special kind of reduction for that language, called *instance compression*, and then prove that if SAT is instance compressible then the polynomial hierarchy collapses.

Interactive Proofs and Small Space Algorithms. There is a well established relationship between interactive proofs and small space algorithms, which stems from the fact that the optimal prover strategy can be computed in space that is proportional to the length of the transcript of the interactive proof, if given oracle access to the verifier’s decision procedure.⁶ This way, we can compute the probability that the verifier accepts against the optimal prover strategy and decide accordingly. Notice that this reduction does not require the interactive proof to have perfect completeness, but rather only a gap between completeness and soundness.

So the problem now boils down to bounding the space needed to emulate the verifier’s decision procedure. The problem is that the verifier’s decision procedure can take space that is polynomial in the instance length. This means that if we look at the overall space used, it may very well be dominated by the verifier’s decision procedure (making the succinctness of the proof irrelevant).

IOPs and Small Space Algorithms. When it comes to IOPs, we can leverage the fact that the verifier queries a small number of bits from the prover messages. For simplicity, we first consider a *non-adaptive* IOP which means that after the interaction is over, the verifier generates a predicate and a set of query locations (both of which depend only on the instance and the random coins) and outputs

⁶ See, e.g., [Gol08, Chapter 9].

the evaluation of the predicate on the query values. If we can compute the predicates and query locations for all possible random coins of the verifier, then the verifier's decision procedure can be emulated by simple oracle access to those queries and predicates - which does not add any substantial amount of space to our computation. Let's analyze the time it takes to generate all such predicates and queries: assuming the verifier uses rc random coins, we need to iterate over 2^{rc} possibilities and emulate the verifier on each of them. This takes $2^{rc} \cdot \text{poly}(n)$ time. Extending this approach to general *adaptive* IOPs is not difficult: the predicates and query sets are simply replaced by *decision trees*. Computing each decision tree now requires us to iterate over all possible query values, so we get a total of $2^{rc+qc} \cdot \text{poly}(n)$ time complexity, where qc is the query complexity of the IOP. This yields the following lemma:

Lemma 1 (Informally stated, see Lemma 4). *If a language \mathcal{L} can be decided by a public coin IOP where the query complexity is qc and the randomness complexity is rc . Then all of the verifier's decision trees can be computed in $2^{rc+qc} \cdot \text{poly}(n)$ time.*

This approach presents another problem: computing all possible decision trees requires time that is exponential in the randomness complexity. Note that the exponential dependence on the query complexity does not bother us for two reasons: the number of queries in common IOP constructions tends to be small and, moreover, for non-adaptive IOPs this factor vanishes.

Randomness Reduction for IOPs. To address the problem of exponential dependence on the randomness complexity, we present a transformation that reduces the randomness complexity of IOPs, at the expense of having a single long verifier message at the beginning of the interaction.

We remark that a similar type of randomness reduction is known in many contexts, such as communication complexity [New91], property testing [GS10, GR18], interactive proofs [AG21], and likely many other settings as well. Nevertheless we point out the following key feature of our transformation, which to the best of our knowledge is novel: we reduce the randomness complexity during the interaction so that it does not even depend logarithmically on the input size. This is crucial in our context since a logarithmic dependence on the input size, would translate into a polynomial dependence in the reduction.

The technique, as usual in such randomness reductions is subsampling. In more detail, for a public-coin IOP, in each round, the verifier simply chooses a random string from some set U and sends it to the prover. The randomness complexity required to uniformly choose a string from U is $rc = \log |U|$. Imagine if a proper subset $S \subset U$ was chosen a priori and made known to both the prover and the verifier, such that the verifier now samples a uniform random string from S instead of U . This reduces the randomness complexity to $\log |S|$.

But does any subset S preserve the completeness and soundness properties? For perfect completeness, the answer is yes; any string that the verifier chooses would make it accept (a "yes" instance), therefore any subset S would preserve that property. Preserving soundness, on the other hand, is more challenging;

there is a prover strategy and a fraction of random strings that would make the verifier accept a “no” instance, and if S contains only such strings, then the soundness property is not preserved.

Nevertheless, we show that if the verifier generates the set S at random by choosing $\text{poly}(cc)$ strings (where cc is the prover-to-verifier communication complexity) from U , then the soundness property is preserved with overwhelming probability. The result is informally stated below.

Lemma 2 (Informally stated, see Lemma 3). *If \mathcal{L} has a public-coin IOP where the prover sends cc bits in total and the verifier uses rc random coins, then \mathcal{L} has a public-coin IOP where the verifier first sends a random string of length $rc \cdot \text{poly}(cc)$ and all subsequent verifier random messages are of length $O(\log cc)$. If the original IOP has perfect completeness, then so does the resulting IOP.*

At first glance, it may seem that we have not gained anything. After all, sampling a multi-set S from U requires more randomness than sampling a single string from U . However, we observe that this reduction moves up most of the randomness to the first round, while reducing the randomness complexity in all subsequent rounds.

Small Space with Probabilistic Preprocessing. Assume \mathcal{L} has an IOP and assume for simplicity that the IOP has perfect completeness.⁷ We sketch how we can combine Lemmas 2 and 1 to get a small space algorithm with probabilistic polynomial time preprocessing that decides \mathcal{L} . First, we apply Lemma 2 on the IOP and get an IOP where each verifier message, except the first one, has length $O(\log cc)$. The preprocessing algorithm starts by sampling the first verifier message S , and then computes all of the decision trees conditioned on S being the first verifier message. We note that, overall, the decision tree can be encoded using a string of length $2^{qc+k \log cc}$.

Denote by k the number of rounds in the IOP. Given all of the decision trees, the bounded space algorithm can emulate the optimal prover strategy in $O(cc + qc + k \cdot \log(cc))$ space, since the length of the remaining transcript is $O(cc + k \cdot \log(cc))$ and the queries to the decision trees can be computed in $O(qc + k \log cc)$ space. The algorithm then returns 1 if there exists a strategy that makes the verifier always accept.

Since we assume that the original IOP has perfect completeness, then so does the new IOP and specifically, for any $x \in \mathcal{L}$ and any sampled message S , there exists a prover strategy that makes the verifier accept.

We move on to analyzing soundness. By Lemma 2, the IOP produced by the randomness reduction in step is sound, so we can assume it has some constant soundness error $\varepsilon > 0$. Let x be a “no” instance. Soundness error ε implies that at most ε fraction of the possibilities for the first message might result in a “doomed state” (i.e., the verifier accepts with probability 1 after that first message). Therefore, with probability at least $1 - \varepsilon$ over the sampled S , for any residual prover strategy, there exists a strictly positive probability (over the

⁷ This assumption is not necessary to achieve the result, see Sect. 4.

verifier's randomness) that the verifier rejects. This means that with probability $1 - \varepsilon$, the small-space algorithm would not find a prover strategy that always makes the verifier accept and therefore would reject the instance x . This yields the desired small-space algorithm with probabilistic preprocessing as stated in Theorem 2.

1.4 Organization

Section 2 includes the preliminaries, definitions and notations. In Sect. 3, we formally state randomness reduction for IOPs. In Sect. 4, we prove that any language that has an IOP can be decided in space that is proportionate to the communication complexity after some bounded-time preprocessing. In Sect. 5, we present the compiler from IOPs to zero-knowledge proofs and apply it to the IOP of [RR20]. In addition, the full version [NR22], contains further discussion on Conjecture 1 and a proof sketch that the IOP of [RR20] is indeed compact (as per Definition 5).

2 Preliminaries

For any positive integer $n \in \mathbb{N}$, we denote by $[n]$ the set of integers $\{1, \dots, n\}$.

2.1 Basic Complexity Notations and Definitions

2.1.1 NP Relations

For a language $\mathcal{L} \in \text{NP}$, we denote by $R_{\mathcal{L}}$ an NP relation of \mathcal{L} . The relation $R_{\mathcal{L}}$ consists of pairs (x, w) such that $x \in \mathcal{L}$ and w is a witness that allows one to verify that x is indeed in \mathcal{L} in polynomial time. It holds that $x \in \mathcal{L}$ if and only if $\exists w$ such that $(x, w) \in R_{\mathcal{L}}$. We denote by n the instance size $|x|$, and by m the witness size $|w|$. Throughout this work, we implicitly assume that $m \leq n$.

We extend the definition of NP relations and languages to general relations and their respective languages.

Definition 1 (Languages and Relations). *Let \mathcal{R} be a binary relation and assume that there exists a function $m(n)$ such that if the first element has length n , then the second element has length $m = m(n)$. We call m the witness length and n the instance length of the relation. We define $\mathcal{L}(\mathcal{R}) = \{x : \exists y \in \{0, 1\}^{m(|x|)} \text{ s.t. } (x, y) \in \mathcal{R}\}$ and we call $\mathcal{L}(\mathcal{R})$ the language of the relation \mathcal{R} .*

2.1.2 Circuit-SAT

In the *Circuit-SAT* (CSAT) problem, the instance is a Boolean circuit and we say that it is in the language if there exists an assignment that satisfies that circuit. We define the natural relation $\mathcal{R}_{\text{CSAT}}$ as the satisfiable Boolean circuits and their satisfying assignments.

2.2 Interactive Proofs and Oracle Proofs

We use the definition and notations of *interactive machines* from [Gol04].

Definition 2 (*Interactive proof*). A pair of interactive machines $(\mathcal{P}, \mathcal{V})$ is called an interactive proof system for a language \mathcal{L} if \mathcal{V} is a probabilistic polynomial-time machine and the following conditions hold:

- **Completeness:** For every $x \in \mathcal{L}$,

$$\Pr [\langle \mathcal{P}, \mathcal{V} \rangle (x) = 1] = 1.$$

- **Soundness:** For every $x \notin \mathcal{L}$,

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

When the language \mathcal{L} is an NP language and $x \in \mathcal{L}$, it is standard to give the prover as an input a witness w such that $(x, w) \in R_{\mathcal{L}}$. In this case, the completeness and soundness requirements are stated as follows:

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

$$\Pr [\langle \mathcal{P}(w), \mathcal{V} \rangle (x) = 1] = 1.$$

- **Soundness:** For every $x \notin \mathcal{L}$ and $w^* \in \{0, 1\}^*$,

$$\Pr [\langle \mathcal{P}^*(w^*), \mathcal{V} \rangle (x) = 1] \leq \frac{1}{2}.$$

Definition 3 (*Interactive Oracle Proof*). A public-coin interactive oracle proof (IOP) for a language \mathcal{L} is an interactive protocol between a prover \mathcal{P} and a probabilistic polynomial-time machine \mathcal{V} . On a common input x of length $|x| = n$, the protocol consists of two phases:

1. **Interaction:** \mathcal{P} and \mathcal{V} interact for $k(n)$ rounds in the following manner: in round i , \mathcal{P} sends an oracle message π_i and \mathcal{V} replies with a random string r_i . Denote $r = r_1 \dots r_k$ and $\pi = \pi_1 \dots \pi_k$.
2. **Query and Computation:** \mathcal{V} makes bounded number of queries to the oracles sent by the prover and accepts and rejects accordingly.

We require:

- **Completeness:** If $x \in \mathcal{L}$ then

$$\Pr [\langle \mathcal{P}, \mathcal{V} \rangle (x) = 1] = 1.$$

- **Soundness:** If $x \notin \mathcal{L}$ then for every prover \mathcal{P}^* it holds that

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

Parameters of IOP. We call $cc := |\pi|$ and $rc := |r|$ the *communication complexity* and *randomness complexity* of the IOP, respectively. The bound on the number of queries is denoted by qc and called the *query complexity* of the IOP. The round complexity is the total number of rounds $k = k(n)$ in the interaction phase.

Non-adaptive IOPs. Most IOP construction have the useful property of being *non-adaptive*, that is, the query locations do not depend on the answers to the previous queries. Formally:

Definition 4 (Non-adaptive IOP). *An IOP is called non-adaptive if the query and computation phase can be split into the two following phases:*

1. **Local Computation:** \mathcal{V} deterministically (based on r and x) produces a vector $\mathbf{ql}_{x,r} \in [|\pi|]^{qc}$ of qc queries and a circuit that evaluates a predicate $\phi_{x,r} : \{0, 1\}^{qc} \rightarrow \{0, 1\}$.
2. **Evaluation:** \mathcal{V} queries π on the locations denoted by $\mathbf{ql}_{x,r}$ and plugs the values into the predicate and outputs $\phi_{x,r}(\pi[\mathbf{ql}_{x,r}])$.

Compactness. By definition, the size of the predicate (meaning the circuit that evaluates the predicate) produced by the non-adaptive IOP verifier is bounded by the running time of the verifier. However, many concrete IOP constructions have a predicate that is much shorter than the total running time of the verifier, and we leverage that property in order to construct succinct proofs. The following definition captures this property:

Definition 5 (α -uniform γ -compact IOP). *For any time-constructible⁸ functions $\alpha(n), \gamma(n)$, we say that the IOP is α -uniform γ -compact if for every input $x \in \{0, 1\}^n$ and all random coins r , the size of the circuit that evaluates the predicate $\phi_{x,r}$ is $O(\gamma(n))$. Furthermore, the circuit can be produced in time $O(\alpha(n))$ given x, r . For simplicity, if $\alpha = \text{poly}(n)$, we say that the IOP is γ -compact.*

We use “the size of ϕ ” and “the size of the circuit that evaluates ϕ ” interchangeably, and denote that value by $|\phi|$.

IOPs for Relations. Given a binary relation \mathcal{R} , we define an IOP for \mathcal{R} as an IOP that decides $\mathcal{L}(\mathcal{R})$, where the prover additionally receives the witness as a private input. In these constructions, the prover is usually required to be efficient, since it has the witness as an input.

Succinct IOPs. An IOP for an NP relation $\mathcal{R}_{\mathcal{L}}$ is called *succinct* if the communication complexity (which can be thought of as the proof length) is polynomial in the witness size, that is $cc = \text{poly}(m)$. Formally:

Definition 6. *Let $\mathcal{L} \in \text{NP}$ be a language and $\mathcal{R}_{\mathcal{L}}$ be a corresponding NP relation with instance size n and witness size m . An IOP for $\mathcal{R}_{\mathcal{L}}$ is called a succinct IOP if the communication complexity is $\text{poly}(m)$.*

⁸ A function $f(n)$ is *time-constructible* if given there exists a Turing machine that given 1^n outputs the binary representation of $f(n)$ in $O(f(n))$ time.

2.3 Computational Indistinguishability

We say that a function $f : \mathbb{N} \rightarrow \mathbb{R}$ is negligible if for every polynomial $p(\cdot)$ it holds that $f(n) = O(1/p(n))$.

Definition 7. Let $\{D_n\}_{n \in \mathbb{N}}, \{E_n\}_{n \in \mathbb{N}}$ be two distribution ensembles indexed by a security parameter n . We say that the ensembles are computationally indistinguishable, denoted $D_n \approx E_n$, if for any (non-uniform) probabilistic polynomial time algorithm A , it holds that following quantity is a negligible function in n :

$$\left| \Pr_{x \leftarrow D_n} [A(x) = 1] - \Pr_{x \leftarrow E_n} [A(x) = 1] \right|.$$

We use some basic properties of computational indistinguishability such as the following:

Fact 1 (Concatenation). Let H, H', G and G' be any efficiently sampleable distribution ensembles such that $H \approx H'$ and $G \approx G'$. Then $(H, G) \approx (H', G')$.

Fact 2 (Triangle Inequality). Let H_1, H_2, H_3 be any distribution ensembles. If $H_1 \approx H_2$ and $H_2 \approx H_3$ then $H_1 \approx H_3$.

Fact 3 (Computational Data-Processing Inequality). Let H and H' be any distribution ensembles and A be any PPT algorithm. If $H \approx H'$ then $A(H) \approx A(H')$.

2.4 Cryptographic Primitives

2.4.1 Commitment Scheme in the CRS Model

A commitment scheme in the common reference string model is a tuple of probabilistic polynomial time algorithms (Gen, Com, Ver) where Gen outputs a common random string $r \in \{0, 1\}^*$. The commit algorithm Com takes a message to be committed and the random string r and produces a commitment c and a decommitment string d . The verification algorithm takes the commitment c , the decommitment string d , an alleged committed value y and the random string r and outputs 1 if and only if it is “convinced” that c is indeed a commitment of y . We require the commitment to be computationally hiding and perfectly binding with overwhelming probability over the common random string r . All of the algorithms also take a security parameter $\lambda \in \mathbb{N}$ (in unary representation). We formally define the commitment scheme:

Definition 8 (Commitment Scheme). A commitment scheme in the common reference string model is a tuple of probabilistic polynomial time algorithms (Gen, Com, Ver) that with the following semantics:

- $r \leftarrow Gen(1^\lambda)$ where r is referred to as the common reference string.
- For any string $y \in \{0, 1\}^*$: $(c, d) \leftarrow Com(1^\lambda, r, y)$.
- For any strings $c, d, y \in \{0, 1\}^*$: $\{0, 1\} \leftarrow Ver(1^\lambda, r, c, y, d)$.

The scheme must satisfy the following requirements:

- **Correctness:** *Ver* always accepts in an honest execution, i.e., for any string y and any security λ

$$\Pr_{\substack{r \leftarrow \text{Gen}(1^\lambda) \\ (c,d) \leftarrow \text{Com}(1^\lambda,r,y)}} [\text{Ver}(1^\lambda, r, c, y, d) = 1] = 1.$$

- **Hiding:** For any two strings $y_1, y_2 \in \{0,1\}^*$ and any common reference string r , the distribution of the commitment of y_1 and y_2 are computationally indistinguishable, i.e., if we denote by Com_c only the commitment part of Com then: $\{\text{Com}_c(1^\lambda, r, y_1)\}_{\lambda \in \mathbb{N}} \stackrel{\approx}{\sim} \{\text{Com}_c(1^\lambda, r, y_2)\}_{\lambda \in \mathbb{N}}$.
- **Binding:** For any λ , with probability at least $1 - 2^{-\lambda}$ over the common reference string, any commitment c^* has at most one value y that can be accepted by Ver , i.e.,

$$\Pr_{r \leftarrow \text{Gen}(1^\lambda)} \left[\exists y_1, y_2, d_1, d_2 \in \{0,1\}^* : \bigwedge_{i \in \{1,2\}} \text{Ver}(1^\lambda, r, c^*, y_i, d_i) = 1 \right] < 2^{-\lambda}.$$

In this work, we refer to commitment schemes in the CRS model simply as commitment schemes. We note that due to [Nao91], we have such a commitment scheme under the standard cryptographic assumption of one-way functions.

2.4.2 Pseudorandom Function

We denote by \mathcal{F}_n the set of all functions $\{0,1\}^n \rightarrow \{0,1\}$. In what follows, by a truly random function, we mean a function that is sampled uniformly at random from \mathcal{F}_n .

Definition 9 (*Pseudorandom Function*). A function $F : \{0,1\}^\lambda \times \{0,1\}^n \rightarrow \{0,1\}$ is a pseudorandom function if for any probabilistic polynomial time oracle machine A , every polynomial $p(\cdot)$ and all sufficiently large λ it holds that

$$\left| \Pr_{s \xleftarrow{\$} \{0,1\}^\lambda} \left[A^{F(s,\cdot)}(1^n) = 1 \right] - \Pr_{U \xleftarrow{\$} \mathcal{F}_n} \left[A^{U(\cdot)}(1^n) = 1 \right] \right| < \frac{1}{p(\lambda)}.$$

For convenience, we denote $F(s, x) = F_s(x)$.

Pseudorandom One-Time Pad. We may refer to any function $f : \{0,1\}^n \rightarrow \{0,1\}$ as a string. For any collection of indices $i_1 < i_2 < \dots < i_q$, we use the notation $f[I] := f(i_1) \circ \dots \circ f(i_q)$ where $I = \{i_1, \dots, i_q\}$. When it is clear from context, we may write $f(x)$ for any integer $x \geq 0$ and it should be understood as applying f on the binary representation of x . In this work, we will use pseudorandom functions to encrypt messages, so for any PRF F and string $y \in \{0,1\}^\ell$ (where $\ell \leq 2^n$), the expression $F_S[[\ell]] \oplus y$ represents encrypting y with a pseudorandom

one-time pad obtained from F with seed S . A PRF allows us to implement a stream cipher, e.g. if we want to encrypt the messages $y_1, y_2 \in \{0, 1\}^\ell$ then we can use the values $F_S(1), \dots, F_S(\ell)$ to encrypt y_1 and $F_S(\ell + 1), \dots, F_S(2 \cdot \ell)$ to encrypt y_2 .

The following fact implies that the pseudorandom one-time pad reveals no information about the encrypted string (to any computationally bounded adversary):

Fact 4. *Let $F : \{0, 1\}^\lambda \times \{0, 1\}^n \rightarrow \{0, 1\}$ be a PRF. For any set $I \subseteq \{0, 1\}^n$, the distribution ensemble $\{s \stackrel{\$}{\leftarrow} \{0, 1\}^\lambda : F_s(I)\}_{\lambda \in \mathbb{N}}$ is computationally indistinguishable from uniform random strings of length $|I|$.*

As mentioned above, this property immediately implies that for any string y , the encryption of y using F with key/seed S yields a pseudorandom string.

2.5 Zero-Knowledge Proofs

In this section, we formally define *zero-knowledge proofs* (ZKP) for NP languages. In this paper, we focus on *computational* zero-knowledge. This notion of ZKP relies on computational indistinguishability between distribution ensembles as per Definition 7. These ensembles are indexed by a security parameter λ which is passed to the prover and verifier of the ZKP as an explicit input (in unary representation). We also require that the zero-knowledge property holds even if the verifier is given some auxiliary information. Loosely speaking, this means that any (malicious) verifier does not learn anything from the interaction with the honest prover \mathcal{P} even if the verifier is given some additional a priori information. For any verifier \mathcal{V}^* , input $x \in \{0, 1\}^*$ and auxiliary input $z \in \{0, 1\}^*$ (that might depend on x), we denote by $View_{\mathcal{V}^*(z)}^{\mathcal{P}(w)}(x, \lambda)$ the *view* of \mathcal{V}^* in the interaction $\langle \mathcal{P}(w), \mathcal{V}^*(z) \rangle(x, 1^\lambda)$. The view consists of the random coins tossed by \mathcal{V}^* and the messages it received from the prover (alongside the inputs x, z and λ).

Definition 10 (*Zero-knowledge proofs*). *Let $(\mathcal{P}, \mathcal{V})$ be an interactive proof system for some language $\mathcal{L} \in \text{NP}$ with security parameter λ . The proof-system $(\mathcal{P}, \mathcal{V})$ is computational zero-knowledge w.r.t. auxiliary input if for every polynomial-time interactive machine \mathcal{V}^* there exists a probabilistic polynomial-time machine Sim^* , called the simulator, such that for all $x \in \mathcal{L}$ and any auxiliary input $z \in \{0, 1\}^{\text{poly}(|x|)}$, the following distribution ensembles are computationally indistinguishable:*

- $\left\{ View_{\mathcal{V}^*(z)}^{\mathcal{P}(w)}(x, \lambda) \right\}_{\lambda \in \mathbb{N}}$ where $(x, w) \in R_{\mathcal{L}}$.
- $\left\{ Sim^*(z, x, 1^\lambda) \right\}_{\lambda \in \mathbb{N}}$.

Remark 1. W.l.o.g., we can assume that the malicious verifier \mathcal{V}^* is deterministic, since coin tosses can be passed as auxiliary input.

Throughout this manuscript, we refer to computational zero-knowledge proofs w.r.t. auxiliary input simply as *zero-knowledge proofs*. When the security parameter λ is clear from context, we may omit it from the notation.

2.6 Hoeffding's Inequality

Hoeffding's inequality [Hoe63] is a classical concentration inequality which is widely used in theoretical computer science.

Theorem 3. *Let X_1, \dots, X_n be real random variables bounded by the interval $[0, 1]$ and denote their sum by $S := X_1 + \dots + X_n$. If X_1, \dots, X_n are independent then for any $\varepsilon > 0$ it holds that*

$$\Pr [|S - \mathbb{E}[S]| > \varepsilon] < 2e^{-2\frac{\varepsilon^2}{n}}.$$

3 Randomness Reduction

In this section, we show how to reduce the randomness used by an IOP verifier. While we view this result as being of independent interest, we note that this randomness reduction will also be useful later on in Sect. 4 when we convert IOPs to bounded-space algorithms with probabilistic time preprocessing.

The procedure that we introduce achieves a relaxed notion of randomness reduction: the verifier can use a large (but still polynomial) amount of randomness only in the first round of interaction, then uses only $O(\log cc)$ random bits in each subsequent round, where cc is the communication complexity of the original IOP. Alternatively, we can view this as if before the interaction begins, a trusted setup samples a uniform random string and then the prover and verifier run an IOP, where they both have explicit access to that random string. Moreover, as shown in the full version [NR22], this common random string can also be fixed as a non-uniform advice.

The following definition captures this special type of IOP:

Definition 11. *A protocol $(\mathcal{P}, \mathcal{V})$ is a IOP in the common reference string model (CRS IOP) for a language $\mathcal{L} \subseteq \{0, 1\}^*$ with CRS-error μ if $(\mathcal{P}, \mathcal{V})$ is an IOP as per Definition 3 with the following modifications:*

- **Additional Input:** *In addition to the instance $x \in \{0, 1\}^n$, both the prover and the verifier get an input $\rho \in \{0, 1\}^*$.*
- **Completeness:** *For any $x \in \mathcal{L}$, with probability $1 - \mu$ over $\rho \xleftarrow{\$} \{0, 1\}^*$, \mathcal{P} makes \mathcal{V} accept with probability at least $1 - \varepsilon_c$ over the verifier's coin tosses:*

$$\Pr_{\rho \in \{0, 1\}^*} \left[\Pr [\langle \mathcal{P}, \mathcal{V} \rangle(\rho, x) = 1] \geq 1 - \varepsilon_c \right] \geq 1 - \mu.$$

- **Non-adaptive Soundness:** *For any $x \notin \mathcal{L}$ and every prover strategy \mathcal{P}^* , with probability $1 - \mu$ over $\rho \xleftarrow{\$} \{0, 1\}^*$, \mathcal{P}^* makes \mathcal{V} accept with probability at most ε_s over the verifier's coin tosses. Formally, for every $x \notin \mathcal{L}$ and \mathcal{P}^**

$$\Pr_{\rho \in \{0, 1\}^*} \left[\Pr [\langle \mathcal{P}^*, \mathcal{V} \rangle(\rho, x) = 1] \leq \varepsilon_s \right] \geq 1 - \mu.$$

We emphasize that the external probability $1 - \mu$ is only over the choice of CRS ρ whereas the internal probabilistic statement is over all of the verifier's other coin tosses.

Remark 2. We say that the CRS IOP has *perfect completeness* if for any $x \in \mathcal{L}$ and any $\rho \in \{0, 1\}^*$ it holds that $\Pr[\langle \mathcal{P}, \mathcal{V} \rangle(\rho, x) = 1] = 1$.

The following lemma shows how to transform any IOP into a CRS IOP with small randomness complexity:

Lemma 3 (*Randomness Reduction for IOPs*). *Let \mathcal{L} be a language that has a k -round public-coin IOP with constant completeness and soundness errors $\varepsilon_c, \varepsilon_s$ and communication complexity cc . For any λ and constant ϵ_0 be some constant. Then \mathcal{L} has a CRS IOP with CRS-error $2^{-\lambda}$, completeness error $\varepsilon_c + \epsilon_0$ and soundness error $\varepsilon_s + \epsilon_0$. The CRS length is $\text{poly}(cc, rc, \lambda)$, the randomness complexity is $O(k \cdot \log(cc \cdot \lambda))$, and the query complexity, communication complexity and number of rounds are the same as in the original IOP. Furthermore, if the original IOP has perfect completeness then the CRS IOP has perfect completeness.*

The idea that underlies the proof of Lemma 3 is to use the CRS to shrink the probability space and then argue that the resulting space is a good representative of the original space. To formalize this idea, we make use of the *game tree of a proof system* defined by Goldreich and Håstad [GH98]. The CRS defines an approximation of the tree, and both parties interact with respect to the approximated tree. Due to page constraints, the proof is deferred to the full version [NR22].

4 Limitations of Succinct IOPs

In this section, we show limitations on the expressive power of succinct IOPs. Here we consider general, *adaptive* IOPs (which only strengthens the negative result).

Loosely speaking, we show that if a language has an IOP then it can be decided by a bounded-space algorithm with bounded-time preprocessing. The amount of space used is closely related to the communication complexity of the IOP. When applying this result to a succinct IOP for an NP relation $\mathcal{R}_{\mathcal{L}}$ (as per Definition 6) with instance length n and witness length m we get a $\text{poly}(m)$ -space algorithm with $\text{poly}(n)$ -time preprocessing that decides the language \mathcal{L} .

We start by formally defining what it means for a relation to be decidable by a bounded-space algorithm with a bounded-time preprocessing:

Definition 12. *Let \mathcal{R} be a relation with instance length n and witness length m and $\mathcal{L} = \mathcal{L}(\mathcal{R})$ the corresponding language (see Definition 1). We say that \mathcal{R} can be decided in $s(n, m)$ -space with $t(n, m)$ preprocessing with soundness error ε_s and completeness error ε_c if there exists a $t(n, m)$ -time probabilistic algorithm \mathcal{A}_1 and a $s(n, m)$ -space (deterministic) algorithm \mathcal{A}_2 such that:*

- If $x \in \mathcal{L}$ then $\Pr_{y \leftarrow \mathcal{A}_1(x)} [\mathcal{A}_2(y) = 1] \geq 1 - \varepsilon_c$.
- If $x \notin \mathcal{L}$ then $\Pr_{y \leftarrow \mathcal{A}_1(x)} [\mathcal{A}_2(y) = 1] \leq \varepsilon_s$.

In what follows we refer to \mathcal{A}_1 as the *preprocessor* and \mathcal{A}_2 as the *decider*. We also note that, as usual, the soundness error can be reduced by repetition, while increasing the time (and space) complexities accordingly. Observe that we can reduce it to negligible simply by repeating the preprocessing a polynomial number of times, while almost preserving the space used by the decider.

The main result shown in this section is the following theorem:

Theorem 4. *If a language \mathcal{L} has a k -round IOP with communication complexity cc , query complexity qc and verifier run-time T_V , then \mathcal{L} can be decided in $O(cc + k \cdot \log cc)$ -space with $(2^{qc+O(k \cdot \log cc)} \cdot T_V)$ preprocessing with completeness and soundness errors 2^{-cc} . Furthermore, if the IOP has perfect completeness, then the algorithm has perfect completeness as well.*

As an immediate corollary, we obtain the following:

Corollary 3. *Let $\mathcal{R}_{\mathcal{L}}$ be an NP relation with instance size n and witness size m . If $\mathcal{R}_{\mathcal{L}}$ has a succinct constant-round IOP with perfect completeness and $O(\log n)$ query complexity then \mathcal{L} can be decided in $\text{poly}(m)$ -space with $\text{poly}(n)$ preprocessing. Similarly, if $\mathcal{R}_{\mathcal{L}}$ has a succinct $o(\frac{m}{\log m})$ -round IOP with perfect completeness and $o(m) + O(\log n)$ query complexity then \mathcal{L} can be decided in $\text{poly}(m)$ -space with $2^{o(m)} \cdot \text{poly}(n)$ preprocessing. The soundness error in both algorithms is $2^{-\text{poly}(m)}$.*

4.1 Handling Small Randomness

Given an IOP and an input x , each string of random coins r defines a *decision tree* of the verifier, which dictates which queries to make and what value to output. The idea is to encode all of these decision trees, and generate a large string that represents all of them. This string can be used to implement the verifier in very small space, simply by reading a few locations of the string to determine what queries to make (to prover messages) and decide the output. This resulting verifier runs in very small space, and the final step is to convert this resulting IOP to a small-space algorithm using the following fact⁹:

Fact 5. *For any IP (P, V) there exists an algorithm \mathcal{A}_1 s.t. $\mathcal{A}_1(x) = \Pr[(P, V)(x) = 1]$ for any $x \in \{0, 1\}^*$. In addition, \mathcal{A}_1 runs in $O(cc + rc + S)$ space, where cc is the communication complexity, rc is the randomness complexity and S is the verifier’s space complexity.*

Hence, going back to the terminology of Definition 12, the preprocessor is responsible for generating the string and the decider is simply an algorithm for computing the probability that the verifier accepts and deciding according to that probability. The following lemma captures the main property of the preprocessor:

⁹ See, e.g., [Gol08, Chapter 9].

Lemma 4. *Let $(\mathcal{P}, \mathcal{V})$ be an IOP with communication complexity cc , randomness complexity rc , query complexity qc and verifier run-time $T_{\mathcal{V}} > \log cc$. Then there exists a function $f : \{0, 1\}^n \rightarrow \{0, 1\}^{O(2^{rc+qc} \cdot \log cc)}$ that can be computed in $O(T_{\mathcal{V}} \cdot 2^{rc+qc})$ time s.t. for any x , the query and computation phase of \mathcal{V} can be implemented in $O(rc + qc + \log cc)$ space given $f(x)$.*

Proof (of Lemma 4). Let $(\mathcal{P}, \mathcal{V})$ be an IOP as stated in the lemma. Denote by k the number of communication rounds in the IOP, and for all $i \in [k]$ let cc_i and rc_i be the length of the prover message and verifier message in round i , respectively.

Encoding a Computation Path: On input $x \in \{0, 1\}^n$ and randomness $r \in \{0, 1\}^{rc}$, the verifier \mathcal{V} performs a local deterministic computation that depends on x , r and the values of the queries it makes. Fixing x and r , a function $f_r(x)$ computes a decision tree for the possible executions of the verifier by feeding it every possible value of each query it makes (one by one, since the verifier might be adaptive). Since there are qc binary queries, then the decision tree is a binary tree with depth qc , where each internal node contains location of the next query and each leaf contains the verifier's output given the values of the queries. Each query location can be represented using $\log cc$ bits, so the tree can be encoded using a string of length $O(2^{qc} \cdot \log cc)$. Each leaf in the tree (along with the path leading to it) takes at most $T_{\mathcal{V}}$ time to produce. Therefore, it can be produced in $O(2^{qc} \cdot T_{\mathcal{V}})$ time. In total, the function $f(x)$ computes $f_r(x)$ for each $r \in \{0, 1\}^{rc}$, so the size of the string that contains all of the decision trees is $O(2^{rc+qc} \cdot \log cc)$, and it can be computed in $O(2^{rc+qc} \cdot T_{\mathcal{V}})$ time.

Space-Efficient Construction of the Verifier: Given this string, we can build an IOP verifier \mathcal{V}_s that interacts the prover the same way that \mathcal{V} does. However, in the query and computation phase, \mathcal{V}_s does not have to perform any actual computation. Instead, \mathcal{V}_s looks at $f(x)$ whenever it makes a query to the prover messages and finally outputs the value of the leaf it reaches in the decision tree. The total space used by \mathcal{V}_s is the space required to make the queries to $f(x)$ and prover messages π_1, \dots, π_k , which is $O(rc + qc + \log cc)$.

Correctness: For any x , it is easy to see that for any random coins r and any prover messages π , both \mathcal{V} and \mathcal{V}_s make the same decision: the decision tree of \mathcal{V} is encoded in $f(x)$, and \mathcal{V}_s simply behaves according to the instructions in $f(x)$. ■

4.2 Handling Larger Randomness

Looking at Lemma 4, we note that the time it takes to compute $f(x)$ grows exponentially with the randomness complexity of the IOP, so for an IOP with $\omega(\log n)$ randomness complexity, the running time jumps to super polynomial. In order to solve this problem, we use the randomness reduction stated in Lemma 3. After getting the input x , we let the preprocessor sample a random CRS ρ , and then apply Lemma 4 on both of x and ρ and the IOP that takes them as an input. By setting $\lambda = cc$ in Lemma 3, we get the following properties:

1. The preprocessor that computes $f(x)$ would run in $2^{qc} \cdot T_{\mathcal{V}} \cdot \text{poly}(cc)^k$ time.
2. The string $f(x)$ would have length $2^{qc} \cdot \text{poly}(cc)^k$.
3. If the original IOP has probability $p < 1$ of accepting an input x , then with probability $1 - 2^{-cc}$ of the CRS, the new IOP has probability $p - \epsilon_0 < p' < p + \epsilon_0$ of accepting x . If $p = 1$ then the new IOP accepts x with probability 1 as well.

We are now ready to prove Theorem 4 using Lemma 4 and Fact 5.

Proof (of Theorem 4). Let $(\mathcal{P}, \mathcal{V})$ be an IOP for the language \mathcal{L} as stated in the theorem, and assume without loss of generality that the completeness and soundness errors are 0.3. By applying Lemma 3 with $\lambda = cc$ and $\epsilon_0 = 0.1$, there exists a CRS IOP $(\mathcal{P}', \mathcal{V}')$ for \mathcal{L} with CRS-error 2^{-cc} and completeness and soundness errors 0.4. On any input x , the preprocessor \mathcal{A}_1 generates a random CRS which we denote by ρ . By Lemma 4, there exists a function $f : \{0, 1\}^{n+|\rho|} \rightarrow \{0, 1\}^{O(2^{qc} \cdot \text{poly}(cc)^k)}$ s.t. the query and computation phase of \mathcal{V}' with the fixed ρ can be implemented by an oracle machine \mathcal{V}_s that has oracle access to $f(x, \rho)$ and runs in $O(qc + k \cdot \log cc)$ space. The preprocessor \mathcal{A}_1 computes and outputs this $f(x)$. With probability $1 - 2^{-cc}$ over ρ , it holds that if $x \in \mathcal{L}$ then $\Pr[(\mathcal{P}', \mathcal{V}')(\rho, x) = 1] \geq 0.6$ and if $x \notin \mathcal{L}$ then $\Pr[(\mathcal{P}', \mathcal{V}')(\rho, x) = 1] \leq 0.4$. By Fact 5, the value $\Pr[(\mathcal{P}', \mathcal{V}')(\rho, x) = 1]$ can be computed in $O(qc + cc + k \log cc) = O(cc + k \log cc)$ space. The decider algorithm \mathcal{A}_2 simply computes $p = \Pr[(\mathcal{P}', \mathcal{V}')(\rho, x) = 1]$ and accepts if and only if $p \geq 0.6$. ■

5 Succinct Zero-Knowledge Proofs from OWF

In this section we show how to construct succinct zero-knowledge proofs from succinct IOPs. Intuitively, the idea is to run an “encrypted” version of the communication phase of the IOP, where the prover sends a commitment of each oracle, instead of the oracle itself, and the verifier replies with the usual random coins. At the end of the interaction, rather than having the prover “reveal” the queries that the verifier asks, the prover proves in zero-knowledge that *if* it would have revealed the queries then the original IOP verifier would have accepted.

The technique of committing to messages in a public-coin protocol and then proving in zero-knowledge that revealing them would make the verifier accept dates back to [BGG+88]. This technique is called “notarized envelopes”, where we can think that each bit of the messages is put in a secure envelope and then a “notary” proves something about the contents of those envelopes without actually opening them. But here, we leverage the fact that the verifier only cares about the values of its queries rather than the entire transcript. Therefore, the statement which the prover has to prove in zero-knowledge is much smaller than the IOP transcript and, in fact, depends mainly on the complexity of the IOP verifier.

Leveraging the small query complexity of the IOP is inspired by [Kil92, Mic00]: there, the notarized envelopes technique is applied to PCPs to achieve very efficient *computationally sound interactive proofs*, i.e., protocols that are only sound against computationally-bounded cheating provers. In Sect. 5.1, we modify this approach in two ways: we apply the notarized envelopes to IOPs instead of PCPs and we achieve *unconditional soundness* instead of computational soundness. We can think of this as a *transformation* from IOPs to ZKPs. Our contribution is implementing this transformation in a way that preserves the original communication complexity of the IOP up to an additive overhead. More precisely, the additive overhead depends on the security parameter and the complexity of the IOP verifier (or more precisely, its “compactness”, see Definition 5). Furthermore, we do so under the minimal [OW93] cryptographic assumption of one-way functions.

In Sect. 5.2, we apply the transformation to the succinct IOP of [RR20]. This yields a succinct zero-knowledge proof for a rich sub-class of NP relations. Namely, for bounded-space relations, we construct zero-knowledge proofs with communication complexity that is arbitrarily close to the *witness* length - with the small additive overhead we mentioned earlier.

5.1 Communication Preserving ZKP

In this subsection, we prove the following general theorem that shows how to construct a ZKP from an IOP while nearly preserving the communication complexity, i.e., the transformation discussed above:

Theorem 5. *Assume the existence of one-way functions. Let $\mathcal{L} \in \text{NP}$. If $(P_{\text{IOP}}, V_{\text{IOP}})$ is a T_V -uniform γ -compact IOP for \mathcal{L} , with soundness error ε_{IOP} , communication complexity $cc = cc(n)$ and query complexity $qc = qc(n)$ where the prover runs in T_P time given the witness for x , then \mathcal{L} has a public-coin zero-knowledge proof with soundness error $\varepsilon_{\text{IOP}} + 2^{-\lambda}$ and proof length $cc + \text{poly}(\lambda, \gamma, \log cc)$, where $\lambda > 0$ is the security parameter. Furthermore, the running time of the ZKP verifier is $T_V + \text{poly}(\lambda, \gamma, \log cc)$ and the running time of the prover is $T_P + \text{poly}(\lambda, \gamma)$.*

5.1.1 The Transformation

The existence of one-way functions implies the existence of the following cryptographic tools:

- A pseudorandom function $F : \{0, 1\}^\lambda \times \{0, 1\}^* \rightarrow \{0, 1\}$ [GGM86, HILL99].
- A commitment scheme (Gen, Com, Ver) [Nao91, HILL99] as defined in Sect. 2.4.
- For any security parameter $\lambda > 0$, a public-coin ZKP with an efficient prover for any language $L \in \text{NP}$ with perfect completeness, soundness error $2^{-\lambda}$ and proof length $\text{poly}(\lambda, n)$ [GMW86].

We describe how to use those components to transform $(P_{\text{IOP}}, V_{\text{IOP}})$ to a pair of interactive machines (P, V) .

Let x be the input. For $\langle P_{\text{IOP}}(w), V_{\text{IOP}} \rangle(x)$, denote by k the number of rounds and by π_i (resp. r_i) the prover message (resp. verifier public-coins) sent in round i . Recall that at the end of the interaction phase of the IOP, the verifier V_{IOP} has an oracle access to the prover messages $\pi = (\pi_1, \dots, \pi_k)$ and full access to its own random coins $r = (r_1, \dots, r_k)$. In the local computation phase, V_{IOP} produces a predicate $\phi_{x,r} : \{0, 1\}^{qc} \rightarrow \{0, 1\}$ and query locations $Q_{x,r} \in [cc]^{qc}$ and it accepts if and only if $\phi_{x,r}(\pi[Q_{x,r}]) = 1$.

As discussed above, rather than sending π_i , the prover P sends a commitment α_i to the message π_i . The commitment is computed as follows: first, P commits to a PRF seed S , then uses F_S to encrypt each π_i using F_S as a pseudorandom one-time pad. The prover P then convinces V , in zero-knowledge, that if the query locations of the messages were revealed then V_{IOP} would have accepted. In particular, P proves that it knows some seed S such that the decryption of α w.r.t. F_S in the query locations specified by $Q_{x,r}$ would satisfy the predicate $\phi_{x,r}$.

For that purpose, we define the language \mathcal{L}' , consisting of tuples (ϕ, ρ, Q, y, c) , where $\phi : \{0, 1\}^{qc} \rightarrow \{0, 1\}$ is a predicate, $\rho \in \{0, 1\}^{\text{poly}(\lambda)}$ is a (common reference) string, $Q \subseteq [cc]$ is a set of qc query locations, $y \in \{0, 1\}^{qc}$ is a vector of *encrypted* query values (supposedly taken from the transcript) and $c \in \{0, 1\}^{\text{poly}(\lambda)}$ is the commitment of some seed. The tuple is in the language if and only if there exists a seed $S \in \{0, 1\}^\lambda$ that can be revealed from c and ρ such that the decryption of y w.r.t. S and Q satisfies the predicate. For simplicity, we assume that the security parameter λ can be inferred from ρ and furthermore, is polynomially related to $|\rho|$ (which is indeed the case in [Nao91]). For simplicity, we refer to F_S as a string and use the notation $F_S[Q]$ to access the Q coordinates from F_S . Formally:

$$\mathcal{L}' = \left\{ (\phi, \rho, Q, y, c) : \exists d, S \text{ s.t. } Ver(1^\lambda, \rho, c, S, d) = 1 \text{ and } \phi(y \oplus F_S[Q]) = 1 \right\}.$$

The length of an instance (ϕ, ρ, Q, y, c) is $|\phi| + \text{poly}(\lambda) + qc \cdot O(\log cc)$. The language \mathcal{L}' is clearly in NP since given S and d (which have $\text{poly}(\lambda)$ length), a verifier can verify that $Ver(1^\lambda, \rho, c, S, d) = 1$ in $\text{poly}(\lambda)$ time, compute $y \oplus F_S[Q]$ in $\text{poly}(\lambda, qc, \log cc)$ time and verify $\phi(y \oplus F_S[Q]) = 1$ in $O(|\phi|)$ time. Therefore, \mathcal{L}' has a ZKP with an efficient prover which we denote by $(P_{\mathcal{L}'}, V_{\mathcal{L}'})$. Thus, the final step of the protocol is to have P and V emulate $(P_{\mathcal{L}'}, V_{\mathcal{L}'})$ to prove (in zero-knowledge) that the tuple is in \mathcal{L}' . Note that P can perform this step efficiently since it has the NP witness S, d . The protocol is presented in Fig. 1.

We prove the protocol is a zero-knowledge proof in two steps: first we prove that it is an interactive proof for \mathcal{L} with the desired complexity properties and then we prove that it is computational zero-knowledge w.r.t. auxiliary input. This is captured by the following two lemmas:

Lemma 5. *The protocol in Fig. 1 is a public-coin interactive proof for \mathcal{L} . The proof length is $cc + \text{poly}(\lambda, \gamma, \log cc)$, the soundness error is $\epsilon_{\text{IOP}} + 2^{-\lambda}$, the verifier runs in time $T + \text{poly}(\lambda, \gamma, \log cc)$ and the prover runs in time $T_P + \text{poly}(\lambda, \gamma)$.*

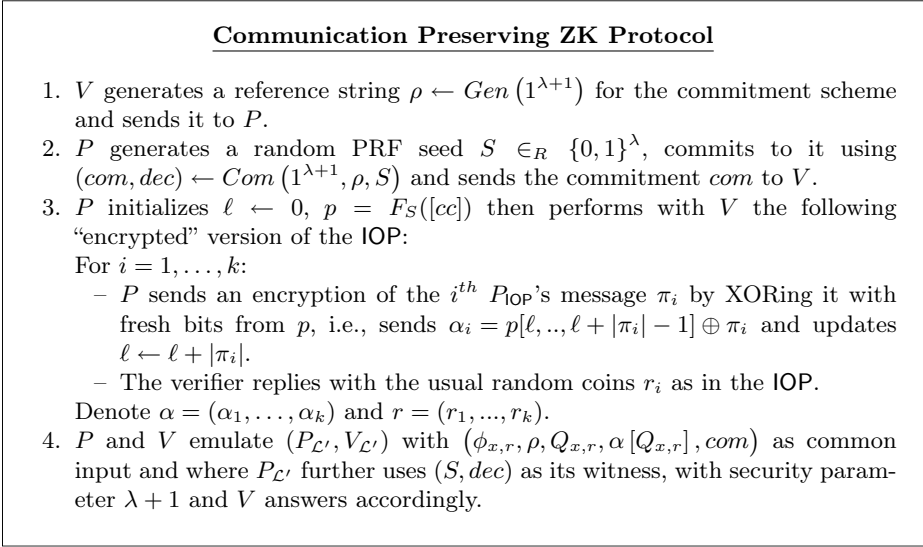


Fig. 1. The ZK protocol from Theorem 5

Lemma 6. *The protocol in Fig. 1 is computational zero-knowledge w.r.t. auxiliary input.*

Due to page constraints, the proof is deferred to the full version [NR22].

5.1.2 Proof of Lemma 6

We now move on to proving that the protocol (P, V) of Fig. 1 is computational zero-knowledge w.r.t. auxiliary input.

By definition of zero-knowledge, we need to show a simulator for the interaction (P, V^*) , where V^* is an arbitrary (malicious) verifier. The idea is to simulate the IOP phase by replacing the honest prover messages with truly random messages and simulate the ZKP phase using the simulator for $(P_{\mathcal{L}'}, V_{\mathcal{L}'})$.

In the following discussion, for readability, we often omit the security parameter λ from the notation. However, formal claims and proofs that follow do mention the security parameter explicitly.

Let V^* be a polynomial time verifier as per Remark 1, and denote by $V_{\mathcal{L}'}$ the residual verifier strategy¹⁰ that V^* uses in step 4. Let $x \in \mathcal{L}$ be an instance, $z \in \{0,1\}^{\text{poly}(|x|)}$ be some auxiliary input and w be the NP witness for x . The view of V^* when interacting with the prover P on common input x and prover input w , denoted by $\text{View}_{V^*(z)}(x) := \text{View}_{V^*(z)}^{P(w)}(x)$, consists of x, z , the commitment com , the encrypted messages α and the view

¹⁰ This strategy might depend on the view of V^* up to that point, but this can be passed to $V_{\mathcal{L}'}$ as an auxiliary input as we show later on.

of $V_{\mathcal{L}'}^*$ in step 4, denoted by $View'(\phi_{x,r}, \rho, Q_{x,r}, \alpha[Q_{x,r}], com, z_{\mathcal{L}'}, (S, dec)) := View_{V_{\mathcal{L}'}, z_{\mathcal{L}'}}^{P_{\mathcal{L}'}, (S, dec)}(\phi_{x,r}, \rho, Q_{x,r}, \alpha[Q_{x,r}], com)$ where $z_{\mathcal{L}'} = (x, z, r, \alpha)$ is the auxiliary input for $V_{\mathcal{L}'}$ and S, dec are the witness for tuple being in \mathcal{L}' . Since $z_{\mathcal{L}'}$ contains α and the instance contains com that is, everything in the view of V^* up to that point, we can assume that

$$View_{V^*(z)}(x) = View'(\phi_{x,r}, \rho, Q_{x,r}, \alpha[Q_{x,r}], com, z_{\mathcal{L}'}).$$

By the zero-knowledge property of $(P_{\mathcal{L}'}, V_{\mathcal{L}'})$, there exists a simulator $Sim'_{V_{\mathcal{L}'}}^*$ that can simulate $View'$ with auxiliary input $z_{\mathcal{L}'}$. We observe that w.l.o.g., the input of $Sim'_{V_{\mathcal{L}'}}^*$ can simply consist of $(x, r, z, \alpha, \rho, com)$ - this is due to the fact that $Sim'_{V_{\mathcal{L}'}}^*$ can compute $\phi_{x,r}$ and $Q_{x,r}$ on its own and there is no need to pass the bits $\alpha[Q_{x,r}]$ twice. Recall that we assume $(P_{\text{OP}}, V_{\text{OP}})$ has perfect completeness, therefore for any verifier messages ρ, r and honestly generated prover messages com and α , it holds that $(\phi_{x,r}, \rho, Q_{x,r}, \alpha[Q_{x,r}], com) \in \mathcal{L}'$. Therefore, it holds that $Sim'_{V_{\mathcal{L}'}}^*(x, r, z, \alpha, \rho, com)$ and $View'(\phi_{x,r}, \rho, Q_{x,r}, \alpha[Q_{x,r}], com, z_{\mathcal{L}'})$ are computationally indistinguishable.

We now describe a simulator $Sim_{V^*}(x, z)$ that simulates $View_{V^*(z)}^{P(w)}(x)$. We start with a high-level description. Given as input x, z , the simulator emulates step 1 of the protocol from Fig. 1 exactly as V^* would, namely, it runs $V^*(x, z)$ to generate the CRS ρ . For step 2, the simulator commits to a string of zeros and “sends” the commitment to V^* , leveraging the hiding property of the commitment scheme. For step 3, the simulator “sends” random messages to V^* , this time leveraging the pseudorandomness of the PRF. Finally, for step 4, the simulator simply runs $Sim'_{V_{\mathcal{L}'}}^*$ on the instance that V^* sees - which includes the commitment to zeros and the subsequent random messages. Since the output of $Sim'_{V_{\mathcal{L}'}}^*$ includes its input, specifically the auxiliary input, then Sim_{V^*} can just output the output of $Sim'_{V_{\mathcal{L}'}}^*$. The simulator Sim_{V^*} is formally described in Fig. 2.

ZK Simulator for Theorem 5

Input: $x \in \mathcal{L}, z \in \{0, 1\}^*, 1^\lambda$

1. Generate a reference string $\rho \leftarrow V^*(x, z)$.
2. Compute a commitment to zeros $(c_0, d_0) \leftarrow Com(1^\lambda, \rho, 0^\lambda)$. “Send” c_0 to V^* .
3. For $i = 1, \dots, k$: Generate a random β_i and “send” it to V^* and get in response r_i .
4. Output $Sim'_{V_{\mathcal{L}'}}^*(x, r, z, \beta, \rho, c_0, 1^\lambda)$.

Fig. 2. The simulator Sim_{V^*} for Theorem 5

At first glance, the zero-knowledge property of $(P_{\mathcal{L}'}, V_{\mathcal{L}'})$ does not necessarily hold in this case since the “mock” instance on which we run $Sim'_{V_{\mathcal{L}'}}$ is (almost definitely) not a “yes” instance. However, we observe that this “mock” instance and a “yes” instance are computationally indistinguishable; the commitment to zeros is indistinguishable from that of a randomly generated seed due to the hiding property of the commitment scheme and the truly random messages are indistinguishable from the encrypted messages used in the protocol due to the pseudorandomness of the PRF. Therefore, we can apply the computational data processing inequality (as stated in Fact 3) to deduce that the outputs of $Sim'_{V_{\mathcal{L}'}}$ on both instances are indistinguishable, which are in turn indistinguishable from the view $V_{\mathcal{L}'}$. This yields the following proposition:

Proposition 1. *For all $x \in \mathcal{L}$ and auxiliary input $z \in \{0, 1\}^{\text{poly}(|x|)}$,*

$$\left\{ Sim_{V^*}(x, z, 1^\lambda) \right\}_{\lambda \in \mathbb{N}} \stackrel{c}{\approx} \left\{ View_{V^*}(x, \lambda) \right\}_{\lambda \in \mathbb{N}}.$$

Due to page constraints, the proof is deferred to the full version [NR22].

Proof (of Lemma 6). Let $x \in \mathcal{L}$ and w be the corresponding witness. Fix some polynomial time verifier V^* and auxiliary input z . By Proposition 1, the output of the simulator Sim_{V^*} on input $(x, z, 1^\lambda)$ is computationally from the view $View_{V^*(z)}^{P(w)}(x, \lambda)$. In addition, Sim_{V^*} runs in polynomial time because it only generates random strings of polynomial size and runs the polynomial time algorithms V^* , Com and Sim'_{V^*} . This gives us the zero-knowledge property of the protocol. ■

In total, Lemma 5 and Lemma 6 complete the proof of Theorem 5.

5.2 Constructing Succinct ZKPs

Our next step is to use Theorem 5 to construct succinct zero-knowledge proofs for NP relations that can be verified in bounded space. We rely on the following result by [RR20]:

Theorem 6 (Extension of [RR20]). *Let $\mathcal{L} \in \text{NP}$ with corresponding relation $\mathcal{R}_{\mathcal{L}}$ in which the instances have length m and witnesses have length n , where $m \geq n$, and such that $\mathcal{R}_{\mathcal{L}}$ can be decided in time $\text{poly}(m + n)$ and space $s \geq \log m$. For any constants $\beta, \gamma \in (0, 1)$, there exists a $\beta^{-O(\frac{1}{\beta})}$ -round IOP for \mathcal{L} with soundness error $\frac{1}{2}$ and $(\gamma\beta)^{-O(\frac{1}{\beta})}$ query complexity. The communication consists of a first (deterministic) message sent by the prover of length $(1 + \gamma) \cdot m + \gamma \cdot n^\beta$ bits followed by $\text{poly}\left(n^\beta, (\gamma\beta)^{-\frac{1}{\beta}}, s\right)$ additional communication. In addition, the IOP is $\left(\tilde{O}(n) + \text{poly}\left(n^\beta, (\gamma\beta)^{-\frac{1}{\beta}}, s\right)\right)$ -uniform $\text{poly}\left(n^\beta, (\gamma\beta)^{-\frac{1}{\beta}}, s\right)$ -compact and the prover runs in $\text{poly}(n)$ time.*

Remark 3. The theorem statement in [RR20] does not include the compactness property. Nevertheless, it is relatively straightforward to show that the construction is indeed compact. In addition, [RR20] assumes that the instance length is polynomially related to the witness length and as a result, the length of the first message only depends on the witness length, whereas we do not make that assumption and therefore Theorem 6 introduces a small dependence on the instance length. In our context, we can afford this dependence since it will anyhow appear in our ZKP construction due to the compactness property. Given these differences, for completeness, Theorem 6 is proved in the full version [NR22].

The soundness error in Theorem 6 can be reduced by parallel repetition while observing that since the first prover message is deterministic, so it does not need to be repeated. In addition, if we choose a sufficiently small β and the space s in which we can decide $\mathcal{R}_{\mathcal{L}}$ is sufficiently small (but still polynomially related to n), then we get the following corollary:

Corollary 4. *There exists a fixed constant $\xi > 0$ such that the following holds. Let $\mathcal{L} \in \text{NP}$ with a corresponding relation $\mathcal{R}_{\mathcal{L}}$ in which the instances have length n and witnesses have length m such that $m \leq n$ and $\mathcal{R}_{\mathcal{L}}$ can be decided in $\text{poly}(n)$ time and n^{ξ} space. Then for any constants $\gamma \in (0, 1)$ and any function $\varepsilon = \varepsilon(m) \in (0, 1)$ there exists a constant β' such that for any $\beta \in (0, \beta')$ there exists an IOP for \mathcal{L} with communication complexity $(1 + \gamma) \cdot m + O(\log \frac{1}{\varepsilon}) \cdot \gamma \cdot n^{\beta}$, query complexity $O(\log \varepsilon)$ and soundness error ε . In addition, the IOP is $\tilde{O}(n)$ -uniform n^{β} -compact and the prover runs in $\text{poly}(n)$ time.*

Corollary 4 captures a rich class of NP relations (e.g. SAT or any other relation that can be decided in polynomial time and polylogarithmic space). We now apply Theorem 5 on the IOP from Corollary 4 and get a succinct ZKP for all languages in that class. This yields our main theorem:

Theorem 7. *There exists a fixed constant $\xi > 0$ such that the following holds. Let $\mathcal{R}_{\mathcal{L}}$ be an NP relation, in which the instances have length n and witnesses have length m such that $m \leq n$, that can be decided in $\text{poly}(n)$ time and n^{ξ} space. Assuming one-way functions exist, then for any constant $\gamma \in (0, 1)$ and security parameter $\lambda > 1$, there exists a constant β' such that for any $\beta \in (0, \beta')$ there exists a public-coin zero-knowledge proof for $\mathcal{R}_{\mathcal{L}}$ with $(1 + \gamma) \cdot m + \gamma \cdot n^{\beta} \cdot \text{poly}(\lambda)$ proof length, perfect completeness and soundness error $2^{-\lambda}$. Furthermore, the verifier runs in time $\tilde{O}(n) + n^{\beta} \cdot \text{poly}(\lambda)$ and the prover runs in $\text{poly}(n)$ time.*

Due to page constraints, the proof is deferred to the full version [NR22].

Acknowledgements. We thank Yuval Filmus and Yuval Ishai for helpful discussions. We also thank the anonymous reviewers of CRYPTO 2022 for their helpful comments.

Supported in part by the Israeli Science Foundation (Grants No. 1262/18 and 2137/19), by grants from the Technion Hiroshi Fujiwara cyber security research center

and Israel cyber directorate, and by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

- [AG21] Applebaum, B., Golombek, E.: On the randomness complexity of interactive proofs and statistical zero-knowledge proofs. In: Tessaro, S. (ed.) 2nd Conference on Information-Theoretic Cryptography, ITC 2021, 23–26 July 2021, Virtual Conference. LIPIcs, vol. 199, pp. 4:1–4:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
- [BCS16] Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A. (eds.) TCC 2016, Part II. LNCS, vol. 9986, pp. 31–60. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_2
- [BDRV18] Berman, I., Degwekar, A., Rothblum, R.D., Vasudevan, P.N.: From laconic zero-knowledge to public-key cryptography. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 674–697. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_23
- [BGG+88] Ben-Or, M., et al.: Everything provable is provable in zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 37–56. Springer, New York (1990). https://doi.org/10.1007/0-387-34799-2_4
- [Bra19] Brakerski, Z.: Fundamentals of fully homomorphic encryption. In: Goldreich, O. (ed.) Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali, pp. 543–563. ACM (2019)
- [CLTV15] Canetti, R., Lin, H., Tessaro, S., Vaikuntanathan, V.: Obfuscation of probabilistic circuits and applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 468–497. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_19
- [CY20] Chiesa, A., Yogev, E.: Barriers for succinct arguments in the random oracle model. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 47–76. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64378-2_3
- [FS11] Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.* **77**(1), 91–106 (2011)
- [GGI+15] Gentry, C., Groth, J., Ishai, Y., Peikert, C., Sahai, A., Smith, A.D.: Using fully homomorphic hybrid encryption to minimize non-interactive zero-knowledge proofs. *J. Cryptol.* **28**(4), 820–843 (2015)
- [GGM86] Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. *J. ACM* **33**(4), 792–807 (1986)
- [GH98] Goldreich, O., Hästad, J.: On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.* **67**(4), 205–214 (1998)
- [GKR15] Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for Muggles. *J. ACM* **62**(4), 27:1–27:64 (2015)
- [GMW86] Goldreich, O., Micali, S., Wigderson, A.: How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design (extended abstract). In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 171–185. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_11

- [Gol04] Goldreich, O.: The Foundations of Cryptography - Volume 2: Basic Applications. Cambridge University Press, Cambridge (2004)
- [Gol08] Goldreich, Oded: Computational Complexity - A Conceptual Perspective. Cambridge University Press, Cambridge (2008)
- [GR18] Gur, T., Rothblum, R.D.: Non-interactive proofs of proximity. *Comput. Complex.* **27**(1), 99–207 (2016). <https://doi.org/10.1007/s00037-016-0136-9>
- [GS10] Goldreich, O., Sheffet, O.: On the randomness complexity of property testing. *Comput. Complex.* **19**(1), 99–133 (2010)
- [GVW02] Goldreich, O., Vadhan, S.P., Wigderson, A.: On interactive proofs with a laconic prover. *Comput. Complex.* **11**(1–2), 1–53 (2002)
- [HILL99] Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. *SIAM J. Comput.* **28**(4), 1364–1396 (1999)
- [Hoe63] Hoeffding, W.: Probability inequalities for sums of bounded random variables. *J. Am. Stat. Assoc.* **58**(301), 13–30 (1963)
- [IKOS09] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.* **39**(3), 1121–1152 (2009)
- [Kil92] Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: Kosaraju, S.R., Fellows, M., Wigderson, A., Ellis, J.A. (eds.) Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 4–6 May 1992, Victoria, British Columbia, Canada, pp. 723–732. ACM (1992)
- [KR08] Kalai, Y.T., Raz, R.: Interactive PCP. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5126, pp. 536–547. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70583-3_44
- [Mic00] Micali, S.: Computationally sound proofs. *SIAM J. Comput.* **30**(4), 1253–1298 (2000)
- [Nao91] Naor, M.: Bit commitment using pseudorandomness. *J. Cryptol.* **4**(2), 151–158 (1991). <https://doi.org/10.1007/BF00196774>
- [New91] Newman, I.: Private vs. common random bits in communication complexity. *Inf. Process. Lett.* **39**(2), 67–71 (1991)
- [NR22] Nassar, S., Rothblum, R.D.: Succinct interactive oracle proofs: applications and limitations. *Cryptology ePrint Archive*, Paper 2022/281 (2022). <https://eprint.iacr.org/2022/281>
- [OW93] Ostrovsky, R., Wigderson, A.: One-way functions are essential for non-trivial zero-knowledge. In: Proceedings of the Second Israel Symposium on Theory of Computing Systems, ISTCS 1993, Natanya, Israel, 7–9 June 1993, pp. 3–17. IEEE Computer Society (1993)
- [RR20] Ron-Zewi, N., Rothblum, R.D.: Local proofs approaching the witness length [extended abstract]. In: 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, 16–19 November 2020, pp. 846–857. IEEE (2020)
- [RRR21] Reingold, O., Rothblum, G.N., Rothblum, R.D.: Constant-round interactive proofs for delegating computation. *SIAM J. Comput.* **50**(3) (2021)
- [Sma14] How do we know that $P \neq \text{LINSPEC}$ without knowing if one is a subset of the other? (2014). <https://mathoverflow.net/questions/40770/how-do-we-know-that-p-linspace-without-knowing-if-one-is-a-subset-of-the-othe>. Accessed 2 Feb 2022