Succinct Interactive Oracle Proofs: Applications and Limitations

Shafik Nassar
Technion, Israel Institute of Technology
shafiknassar@cs.technion.ac.il

Ron D. Rothblum [*]
Technion, Israel Institute of Technology
rothblum@cs.technion.ac.il

August 23, 2022

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* IOP*s*, 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.

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

Contents

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 $\mathsf{P} \overset{?}{=} \mathsf{NP}$ question, results such as the $\mathsf{IP} = \mathsf{PSPACE}$ 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 general[2] 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?

1.1 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 2.3 and 2.4). 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 1.1 (Informally Stated, see Theorem 5.1)**.** *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.*

[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.

The proof of Theorem 1.1 is a relatively simple extension of the classical "notarized envelpoes" 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 1.1, 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 1.1 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 1.1 (Informally Stated, see Theorem 5.3)**.** *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 $(1+\gamma)\cdot m + n^{\beta} \cdot \mathrm{poly}(\lambda)$, where n is the instance length, m is the witness length and λ is the security parameter.*

Corollary 1.1 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 1.1 in two ways: first, they achieve an even shorter communication complexity of $m + \mathrm{poly}(\lambda)$ and second, the result holds for *any*[4] NP relation, whereas Corollary 1.1 is restricted to bounded-space relations.

The fact that the transformation from Theorem 1.1 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 $\mathcal{R} = \{(x,w) : x \in \mathcal{L} \wedge (x,w) \in \mathcal{R}_{\mathsf{SAT}}\}$, 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 NP $\not\subseteq$ 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 1.2 (Informally stated, see Theorem 4.1)**.** *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 $\left(2^{qc}\cdot \mathrm{poly}(n, 2^{k\log(cc)})\right)$-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

[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.

a second Turing machine, which uses s-space, can continue the computation, viewing c as its (read-only) input tape (see Definition 4.1 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 1.2, we argue that the existence of succinct IOPs for all of NP would have unexpected, and (arguably) unlikely, repercussions.

For example, consider the relation $\mathcal{R}_{\mathsf{CSAT}}$, consisting of all satisfiable Boolean circuits and their corresponding satisfying assignment. Given Theorem 1.2, the existence of a succinct IOP for $\mathcal{R}_{\mathsf{CSAT}}$ 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 $\mathrm{poly}(n, m)$ time but from there can do arbitrary $\mathrm{poly}(m)$-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 $\mathrm{poly}(m)$-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 $\left(2^{o(m)} \cdot \mathrm{poly}(n)\right)$-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 1.2. *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* $\mathrm{poly}(m)$ *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 1.3. *Assuming Conjecture 1.2 we have:*

- *With $t(n, m) = \mathrm{poly}(n)$, there is no succinct* IOP *for $\mathcal{R}_{\mathsf{CSAT}}$ with a constant number of rounds and $O(\log n)$ query complexity.*
- *With $t(n, m) = 2^{\mathrm{polylog}(m)} \cdot \mathrm{poly}(n)$, there is no succinct* IOP *for $\mathcal{R}_{\mathsf{CSAT}}$ with a* $\mathrm{polylog}(m)$ *rounds and* $\mathrm{polylog}(m) + O(\log n)$ *query complexity.*
- *With $t(n, m) = 2^{o(m)} \cdot \mathrm{poly}(n)$, there is no succinct* IOP *for $\mathcal{R}_{\mathsf{CSAT}}$ 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 Håstad [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 a by small space algorithm. This result is incomparable to Theorem 1.2, 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 *poly-logarithmic* 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 Section 1.3.1 we discuss our compiler for zero-knowledge proofs from IOPs. In Section 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 1.1 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 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.

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 $\mathsf{IP} \subseteq \mathsf{PSPACE}$ 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 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 $2^{rc} \cdot \mathrm{poly}(n)$ 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 $2^{rc+qc} \cdot \mathrm{poly}(n)$ time complexity, where qc is the query complexity of the IOP. This yields the following lemma:

[6]See, e.g., [Gol08, Chapter 9]

Lemma 1.4 (Informally stated, see Lemma 4.4)**.** *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 $2^{rc+qc} \cdot \mathrm{poly}(n)$ 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 $\mathrm{poly}(cc)$ 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 1.5 (Informally stated, see Lemma 3.3)**.** *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 $rc \cdot \mathrm{poly}(cc)$ 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 1.4 and 1.5 to get a small space algorithm with probabilistic polynomial time preprocessing that decides $\mathcal{L}$. First, we apply Lemma 1.5 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 the 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.

[7]This assumption is not necessary to achieve the result, see Section 4.

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 1.5, 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 verifier's randomness) that the verifier rejects. This means that with probability $1 - \varepsilon$, the small-space algorithm would not a 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 1.2.

1.4 Organization

Section 2 includes the preliminaries, definitions and notations. In Section 3, we formally state randomness reduction for IOPs. In Section 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 Section 5, we present the compiler from IOPs to zero-knowledge proofs and apply it to the IOP of [RR20]. In Appendix A, we elaborate more on Conjecture 1.2. Finally, Appendix B contains a sketch of the proof that the IOP of [RR20] is indeed compact (as per Definition 2.5).

2 Preliminaries

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

2.1 Basic Complexity Notations and Definitions

2.1.1 NP relations

For a language $\mathcal{L} \in \mathsf{NP}$, 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 2.1 (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* 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 $\mathcal{R}_{\mathsf{CSAT}}$ 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 2.2. *(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 $x \in \mathcal{L}$,*

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

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

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

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 $(x, w) \in R_{\mathcal{L}}$,*

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

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

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

Definition 2.3. *(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 $r = r_1...r_k$ and $\pi = \pi_1...\pi_k$.*
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*

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

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

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

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 2.4 (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 $\vec{ql}_{x,r} \in [|\pi|]^{qc}$ of qc queries and a circuit that evaluates a predicate $\phi_{x,r} : \{0,1\}^{qc} \to \{0,1\}$.*
2. ***Evaluation:*** *$\mathcal{V}$ queries π on the locations denoted by $\vec{ql}_{x,r}$ and plugs the values into the predicate and outputs $\phi_{x,r}\left(\pi\left[\vec{ql}_{x,r}\right]\right)$.*

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 2.5. *(α-uniform γ-compact IOP) For any time-constructible [8] 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 2.6. *Let $\mathcal{L} \in$ 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* $\text{poly}(m)$.

2.3 Computational Indistinguishability

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

Definition 2.7. *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 $D_n \overset{c}{\approx} E_n$, 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 2.8 (Concatenation)**.** *Let H, H', G and G' be any efficiently sampleable distribution ensembles such that $H \overset{c}{\approx} H'$ and $G \overset{c}{\approx} G'$. Then $(H, G) \overset{c}{\approx} (H', G')$.*

Fact 2.9 (Triangle Inequality)**.** *Let H_1, H_2, H_3 be any distribution ensembles. If $H_1 \overset{c}{\approx} H_2$ and $H_2 \overset{c}{\approx} H_3$ then $H_1 \overset{c}{\approx} H_3$.*

Fact 2.10 (Computational Data-Processing Inequality)**.** *Let H and H' be any distribution ensembles and A be any PPT algorithm. If $H \overset{c}{\approx} H'$ then $A(H) \overset{c}{\approx} A(H')$.*

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

[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.

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 2.11 (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:*

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

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*
$$\Pr_{\substack{r \leftarrow Gen\left(1^\lambda\right) \\ (c,d) \leftarrow Com\left(1^\lambda, r, y\right)}} \left[Ver\left(1^\lambda, r, c, y, d\right) = 1\right] = 1.$$
- ***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: $\left\{Com_c\left(1^\lambda, r, y_1\right)\right\}_{\lambda \in \mathbb{N}} \overset{c}{\approx} \left\{Com_c\left(1^\lambda, r, y_2\right)\right\}_{\lambda \in \mathbb{N}}$.*
- ***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.,*
$$\Pr_{r \leftarrow Gen(1^\lambda)} \left[\exists y_1, y_2, d_1, d_2 \in \{0,1\}^* : \bigwedge_{i \in \{1,2\}} Ver\left(1^\lambda, r, c^*, y_i, d_i\right) = 1\right] < 2^{-\lambda}.$$

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 $\{0,1\}^n \to \{0,1\}$. In what follows, by a truly random function, we mean a function that is sampled uniformly at random from $\mathcal{F}_n$.

Definition 2.12. *(Pseudorandom Function) A function $F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}$ 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*

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

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

Pseudorandom One-Time Pad. We mat refer to any function $f : \{0,1\}^n \to \{0,1\}$ as a string. For any collection of indices $i_1 < i_2 < ... < i_q$, we use the notation $f[I] := f(i_1) \circ ... \circ f(i_q)$ where $I = \{i_1, ..., i_q\}$. 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 $F_S\left[[\ell]\right] \oplus y$ 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 $F_S(1), ..., F_S(\ell)$ to encrypt y_1 and $F_S(\ell+1), ..., F_S(2 \cdot \ell)$ 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 2.13. *Let $F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}$ be a PRF. For any set $I \subseteq \{0,1\}^n$, the distribution ensemble $\left\{s \xleftarrow{\$} \{0,1\}^\lambda : F_s(I)\right\}_{\lambda \in \mathbb{N}}$ 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 2.7. 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 $\langle \mathcal{P}(w), \mathcal{V}^*(z) \rangle \left(x, 1^\lambda\right)$. 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 2.14. *(Zero-knowledge proofs) Let $(\mathcal{P}, \mathcal{V})$ be an interactive proof system for some language $\mathcal{L} \in \mathsf{NP}$ 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 $z \in \{0,1\}^{\mathrm{poly}(|x|)}$, the following distribution ensembles are computationally indistinguishable:*

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

Remark 2.15. *W.l.o.g., we can assume that the malicious verifier 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 2.1. *Let $X_1, ..., X_n$ be real random variables bounded by the interval $[0,1]$ and denote their sum by $S := X_1 + ... + X_n$. If $X_1, ..., X_n$ are independent then for any $\varepsilon > 0$ it holds that*

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

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 Section 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 3.1. *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 2.3 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:*

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

- ***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 $\mathcal{P}^*$*

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

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 3.2. *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 $\Pr\left[\langle\mathcal{P},\mathcal{V}\rangle(\rho,x)=1\right]=1$.*

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

Lemma 3.3. *(Randomness Reduction for* IOP*s) 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 $\mathrm{poly}(cc, rc, \lambda)$, the randomness complexity is $O\big(k\cdot\log(cc\cdot\lambda)\big)$, 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 3.3 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. We extend the original definition of [GH98], which was for interactive proofs, to public-coin IOPs:

Definition 3.4. *Let $(\mathcal{P},\mathcal{V})$ be an* IOP *for some language $\mathcal{L}$ and $x \in \{0,1\}^n$ be some instance for $\mathcal{L}$. The game tree T_x is a leveled tree with $2k+1$ levels, where k is the number of interaction rounds of the* IOP *on instance x, and each node in T_x corresponds to a prefix of a possible interaction transcript. The root, defined to be at level 0, represents the empty transcript $\perp$. For each $i=0,1,...,k-1$, we call nodes in level $2i$* prover nodes *and nodes in level $2i+1$* verifier nodes*. The edges going from prover nodes correspond to possible prover messages given the transcript so far, and the edges going from verifier nodes correspond to the randomness that $\mathcal{V}$ sends in the current round of interaction. The leaves correspond to a full transcript, after which the verifier is ready to decide.*

Any (partial or full) transcript τ corresponds to a path in T_x (or a node) and vice versa, therefore we use "transcript" and "node" interchangeably. We associate with any such τ a value $X_\tau \in [0,1]$ which is defined recursively as follows: given a full transcript τ, the value (of the corresponding leaf) X_τ is 1 if $\mathcal{V}$ accepts given τ as the transcript [9] *and 0 otherwise. For a partial transcript τ that corresponds to a verifier node, X_τ is defined as the average of all of its children in the tree. For a partial transcript τ that corresponds to a prover node, X_τ is defined as the maximum of all of its children in the tree. The value of T_x, denoted as $val(T_x)$, is defined as $X_\perp$, the value of the root.*

Remark 3.5. *The value of T_x is an upper bound on the probability that $\mathcal{V}$ accepts when interacting with any prover, and the bound is achieved by an optimal prover that always chooses the message that maximizes $\mathcal{V}$'s acceptance probability. Consequently, the soundness error of the* IOP *is equal to* $\max_{x \notin \mathcal{L}} val(T_x)$.

With the notion of a game tree in hand, we are ready to prove Lemma 3.3.

3.1 Proof of Lemma 3.3

Let $(\mathcal{P}, \mathcal{V})$ be an IOP for some language $\mathcal{L} \subseteq \{0,1\}^*$ with perfect completeness and soundness error $\frac{1}{2}$. Denote the number of rounds by k and the communication complexity by cc. For any $i = 1, ..., k$, denote by rc_i the number of random bits sent by $\mathcal{V}$ in round i and by cc_i the length of $\mathcal{P}$'s message in that same round.

Our goal is to construct a CRS IOP $(\mathcal{P}', \mathcal{V}')$ for $\mathcal{L}$ with small randomness complexity, by using the CRS to sample a subset of the edges going out of verifier nodes. Let $t \in \mathbb{N}$ be a parameter to be set later. This parameter will correspond to the number of edges that we sample from verifier nodes in each level (round). The larger t is, the closer the approximated tree is to the original tree, but also the larger the CRS and the randomness complexity.

Let $x \in \{0,1\}^n$ be an instance for $\mathcal{L}$. We proceed to describe $(\mathcal{P}', \mathcal{V}')$ using an additional CRS string ρ. The two parties use ρ to sample k multisets $S_1, ..., S_k$, each of size t, where $S_i = \left\{ r_i^{(1)}, ..., r_i^{(t)} \right\}$ such that each $r_i^{(j)}$ is sampled uniformly and independently from $\{0,1\}^{rc_i}$. Then in each round $i = 1, ..., k$, the prover sends a message π_i as it usually does and $\mathcal{V}'$ chooses $r_i' \overset{\$}{\leftarrow} S_i$ uniformly at random (rather than sampling uniformly from $\{0,1\}^{rc_i}$). Note that $\mathcal{V}'$ can do so using only $\lceil \log t \rceil$ randomness in each round, as opposed to rc_i, and thus we get a total randomness complexity of $O(k \cdot \log t)$. Eventually, $\mathcal{V}'$ simulates the local computation phase of $\mathcal{V}$ on randomness $r = r_1' \circ ... \circ r_k'$. That is, it produces the corresponding query set $Q_{x,r}$ and decision predicate $\phi_{x,r}$. The verifier then accepts if and only if $\phi_{x,r}(\pi[Q_{x,r}]) = 1$.

We note that this construction trivially satisfies most of the properties required by Lemma 3.3, independently of the parameter t which we are yet to choose. We state those properties in the following proposition:

Proposition 3.6. *The protocol $(\mathcal{P}', \mathcal{V}')$ has the same the number of rounds, number of queries and communication complexity as $(\mathcal{P}, \mathcal{V})$. Furthermore, if $(\mathcal{P}, \mathcal{V})$ has perfect completeness then $(\mathcal{P}', \mathcal{V}')$ has perfect completeness.*

It remains to prove that if we choose a suitable t, we get the desired randomness complexity and non-adaptive soundness. Indeed, this is where the definition of a game tree (see Definition 3.4) comes into play.

Fix $x \notin \mathcal{L}$ and let $\rho \in \{0,1\}^*$ denote a random variables that represents the CRS. Let T_x be the tree of the original IOP on input x, i.e., the tree that represents the interaction $\langle \mathcal{P}, \mathcal{V} \rangle(x)$. For a fixed CRS $\rho = (S_1, \dots, S_k)$, denote by $T'_{x,\rho}$ the game tree that we get from pruning all edges that are inconsistent with $S_1, ..., S_k$. That is, for every verifier node, we remove the edges labeled by $\{0,1\}^{rc_i} \backslash S_i$ and their corresponding subtrees. Similarly, for any $i \in [k-1]$ and any fixed $S := S_1, ..., S_i$, denote by $T'_{x,S}$ the tree that is consistent with S (with no pruning after round i). Observe that, by definition, $T'_{x,\rho}$ is the interaction tree of $\langle \mathcal{P}', \mathcal{V}' \rangle(\rho, x)$, i.e., the interaction of $(\mathcal{P}', \mathcal{V}')$ on x with ρ as a CRS. Define T'_x to be the distribution over game trees obtained by sampling ρ at random and taking $T'_{x,\rho}$.

In what follows, whenever we say transcript we refer to either a full or partial transcript of the protocol. For each transcript τ that remains in the tree T'_x, we denote by Y_τ the value of its corresponding node

[9] Recall that the transcript includes all of the verifier's randomness, since it is a public-coin protocol

where the values of prover and verifier nodes are defined recursively in a similar fashion to X_τ (but with respect to the tree T'_x). Just as T'_x is a distribution over trees, the values Y_τ are all random variables with respect to the choice of ρ (in contrast to the values X_τ which are just fixed constants). For any ρ, the value $Y_\perp$ is equal to the probability (over the coins of $\mathcal{V}'$) that $\langle \mathcal{P}', \mathcal{V}' \rangle(\rho, x)$ accepts. We next turn to bounding this quantity, thus proving that the soundness error is preserved with overwhelming probability over ρ. For any

The following lemma gives a bound on the distance of X_τ and Y_τ when τ is a verifier node:

Lemma 3.7. *Fix a round $i \in [k]$ and multisets $S := S_1, ..., S_{i-1}$, and let τ be a verifier node for round i in the tree $T'_{x,S}$. Assume that there exist $\varepsilon' \geq 0$ and $\delta' \in \left[0, \frac{1}{2t}\right]$ such that for all $r_i \in \{0,1\}^{rc_i}$ it holds that*

$$\Pr_{S_{i+1},...,S_k}\left[|Y_{\tau r_i} - X_{\tau r_i}| > \varepsilon'\right] \leq \delta',$$

where τr_i is the prover node obtained by selecting the edge r_i from the node τ. Then, for any $\varepsilon > \varepsilon'$

$$\Pr_{S_i,...,S_k}\left[|Y_\tau - X_\tau| > \varepsilon\right] < t \cdot \delta' + 4e^{-2t(\varepsilon - \varepsilon')^2}.$$

Proof. Fix $i \in [k]$, multisets $S = S_1, \ldots, S_{i-1}$ and a transcript τ as in the statement of Lemma 3.7. We say that "S_i is good" if for all $r_i^{(j)} \in S_i$ it holds that $|Y_{\tau r_i^{(j)}} - X_{\tau r_i^{(j)}}| \leq \varepsilon$. By the law of total probability:

$$\Pr_{S_i,...,S_k}\left[|Y_\tau - X_\tau| > \varepsilon\right] \leq \Pr_{S_i,...,S_k}\left[|Y_\tau - X_\tau| > \varepsilon \;\middle|\; S_i \text{ is good}\right] + \Pr_{S_i,...,S_k}\left[S_i \text{ is not good}\right]. \tag{1}$$

We turn to bounding the two terms in the RHS of Eq. (1), from which Lemma 3.7 will follow.

Proposition 3.8. $\Pr_{S_i,...,S_k}\left[S_i \text{ is not good}\right] \leq t \cdot \delta'$.

Proof. Recall that by assumption, for all $r_i \in \{0,1\}^{rc_i}$, it holds that

$$\Pr_{S_{i+1},...,S_k}\left[|Y_{\tau r_i} - X_{\tau r_i}| > \varepsilon'\right] \leq \delta'.$$

Therefore,

$$\begin{aligned}
\Pr_{S_i,...,S_k}\left[S_i \text{ is not good}\right] &= \Pr_{S_i,...,S_k}\left[\exists r_i^{(j)} \in S_i : |Y_{\tau r_i^{(j)}} - X_{\tau r_i^{(j)}}| > \varepsilon'\right] \\
&= \mathop{\mathbb{E}}_{S_i}\left[\Pr_{S_{i+1},...,S_k}\left[\exists r_i^{(j)} \in S_i : |Y_{\tau r_i^{(j)}} - X_{\tau r_i^{(j)}}| > \varepsilon'\right]\right] \\
&\leq \mathop{\mathbb{E}}_{S_i}\left[\sum_{j=1}^{t} \Pr_{S_{i+1},...,S_k}\left[|Y_{\tau r_i^{(j)}} - X_{\tau r_i^{(j)}}| > \varepsilon'\right]\right] \\
&\leq t \cdot \delta'.
\end{aligned}$$

■

Proposition 3.9. $\Pr_{S_i,...,S_k}\left[|Y_\tau - X_\tau| > \varepsilon \;\middle|\; A\right] < 4 \cdot e^{-2t(\varepsilon - \varepsilon')^2}$.

Proof. We prove Proposition 3.9 using Hoeffding's Inequality on the sum of the variables $X_{\tau r_i^{(1)}}, ..., X_{\tau r_i^{(t)}}$. Denote $\bar{X} = \frac{1}{t}\sum_{r_i^{(j)} \in S_i} X_{\tau r_i^{(j)}}$ and recall that $S_i = \left\{r_i^{(j)}\right\}_{j \in [t]}$ was sampled as a uniformly random (multi-) subset of $\{0,1\}^{rc_i}$ and that the value of each $X_{\tau r_i^{(j)}}$ only depends on $r_i^{(j)}$ (which was sampled independently of all other $r_{i'}^{(j)}$), therefore there variables are independent and we can apply Hoeffding's Inequality (see Theorem 2.1) and get

$$\Pr_{S_i,...,S_k}\left[\left|\bar{X} - \mathop{\mathbb{E}}_{S_i,...,S_k}\left[\bar{X}\right]\right| > \varepsilon - \varepsilon'\right] < 2e^{-2t(\varepsilon - \varepsilon')^2}. \tag{2}$$

Thus, it remains to relate the LHS of Eq. (2) to $\Pr_{S_i,\ldots,S_k}\left[|Y_\tau - X_\tau| > \varepsilon \,\middle|\, A\right]$. First, observe that by definition of the value of a verifier node, $Y_\tau = \frac{1}{t}\sum_{r_i^{(j)}\in S_i} Y_{\tau r_i^{(j)}}$ and that when conditioned on A, it holds that $Y_{\tau r_i^{(j)}} \leq X_{\tau r_i^{(j)}} + \varepsilon'$ for each $r_i^{(j)} \in S_i$. Therefore, it holds that $Y_\tau = \frac{1}{t}\sum_{r_i^{(j)}\in S_i} Y_{\tau r_i^{(j)}} \leq \varepsilon' + \frac{1}{t}\sum_{r_i^{(j)}\in S_i} X_{\tau r_i^{(j)}} = \varepsilon' + \bar{X}$. Therefore, we get

$$\Pr_{S_i,\ldots,S_k}\left[|Y_\tau - X_\tau| > \varepsilon \,\middle|\, A\right] \leq \Pr_{S_i,\ldots,S_k}\left[\left|\bar{X} - X_\tau\right| > \varepsilon - \varepsilon' \,\middle|\, A\right]. \tag{3}$$

To get rid of the conditioning on A in the RHS of Eq. (3), we use Proposition 3.8 along with our assumption that $\delta' \leq \frac{1}{2t}$ to get that $\Pr[A] \geq \frac{1}{2}$ and therefore:

$$\begin{aligned}\Pr_{S_i,\ldots,S_k}\left[\left|\bar{X} - X_\tau\right| > \varepsilon - \varepsilon' \,\middle|\, A\right] &\leq \frac{\Pr_{S_i,\ldots,S_k}\left[\left|\bar{X} - X_\tau\right| > \varepsilon - \varepsilon'\right]}{\Pr_{S_i,\ldots,S_k}[A]} \\ &\leq 2\cdot \Pr_{S_i,\ldots,S_k}\left[\left|\bar{X} - X_\tau\right| > \varepsilon - \varepsilon'\right],\end{aligned}$$

where the first inequality follows from elementary probability theory and the second from the fact that $\Pr[A] \geq 1/2$.

Next, we show that the expectation of $\bar{X}$ is X_τ. Consider uniformly sampling t values $R_1, \ldots, R_t$ from $\{0,1\}^{rc_i}$. By the definition of $S_i = \left\{r_i^{(j)}\right\}_{j\in[t]}$, each R_j has an identical distribution to $r_i^{(j)}$. Therefore,

$$\begin{aligned}\mathop{\mathbb{E}}_{S_i,\ldots,S_k}\left[\bar{X}\right] &= \mathop{\mathbb{E}}_{S_i,\ldots,S_k}\left[\frac{1}{t}\sum_{r_i^{(j)}\in S_i} X_{\tau r_i^{(j)}}\right] \\ &=^{(*)} \mathop{\mathbb{E}}_{S_i}\left[\frac{1}{t}\sum_{r_i^{(j)}\in S_i} X_{\tau r_i^{(j)}}\right] \\ &= \mathop{\mathbb{E}}_{R_1,\ldots,R_t}\left[\frac{1}{t}\sum_{j\in[t]} X_{\tau R_j}\right] \\ &=^{(**)} X_\tau.\end{aligned}$$

The equality $(*)$ follows from the fact that the value of each $X_{\tau r_i^{(j)}}$ only depends on S_i and not on the subsequent $S_{i+1}, \ldots, S_k$ and the equality $(**)$ follows from the fact that $X_\tau = \mathbb{E}_{R_j}[X_{R_j}]$ by definition of X_τ when τ is a verifier node. Finally, this gives us

$$\Pr_{S_i,\ldots,S_k}\left[|Y_\tau - X_\tau| > \varepsilon \,\middle|\, A\right] \leq 2\cdot \Pr_{S_i,\ldots,S_k}\left[\left|\bar{X} - \mathop{\mathbb{E}}_{S_i,\ldots,S_k}\left[\bar{X}\right]\right| > \varepsilon - \varepsilon'\right] < 4e^{-2t\left(\varepsilon-\varepsilon'\right)^2},$$

and Proposition 3.9 follows. ∎

Plugging Propositions 3.8 and 3.9 into Eq. (1), we get

$$\begin{aligned}\Pr_{S_i,\ldots,S_k}\left[|Y_\tau - X_\tau| > \varepsilon\right] &\leq \Pr_{S_i,\ldots,S_k}\left[|Y_\tau - X_\tau| > \varepsilon \,\middle|\, S \text{ is good}\right] + \Pr\left[S \text{ is not good}\right] \\ &< 4e^{-2t\left(\varepsilon-\varepsilon'\right)^2} + t\cdot\delta',\end{aligned}$$

and Lemma 3.7 follows. ∎

The next lemma gives a bound on the distance of X_τ and Y_τ when τ is a prover node:

Lemma 3.10. *Fix a round $i \in [k]$ and multisets $S := S_1, \ldots, S_{i-1}$, and let τ be a prover node for round i in the tree $T'_{x,S}$. If for all $\pi_i \in \{0,1\}^{cc_i}$ it holds that $\Pr_{S_i,\ldots,S_k}\left[|Y_{\tau\pi_i} - X_{\tau\pi_i}| > \varepsilon\right] < \delta$ then it follows that $\Pr_{S_i,\ldots,S_k}\left[|Y_\tau - X_\tau| > \varepsilon\right] < 2^{cc_i}\cdot\delta$*

Proof. The lemma follows from the union bound, details follow. Recall that by the definition of the value of a prover node, it holds that $Y_\tau = \max_{\pi_i \in \{0,1\}^{cc_i}} Y_{\tau\pi_i}$ and $X_\tau = \max_{\pi_i \in \{0,1\}^{cc_i}} X_{\tau\pi_i}$. Therefore,

$$\begin{aligned}\Pr\left[|Y_\tau - X_\tau| > \varepsilon\right] &= \Pr\left[\exists \pi_i \in \{0,1\}^{cc_i} : |Y_{\tau\pi_i} - X_\tau| > \varepsilon\right] \\ &\leq \Pr\left[\exists \pi_i \in \{0,1\}^{cc_i} : |Y_{\tau\pi_i} - X_{\tau\pi_i}| > \varepsilon\right] \\ &\leq 2^{cc_i} \cdot \delta.\end{aligned}$$

■

Having bounded the growth in difference between X_τ and Y_τ for both prover and verifier nodes, we can now bound the actual difference between X_τ and Y_τ via induction, thus bounding the difference between $X_\perp$ and $Y_\perp$ (the values of the trees T_x and T'_x).

Lemma 3.11. *For any $\varepsilon > 0$ and $t = \tilde{\Omega}\left(cc \cdot k \cdot \varepsilon^{-2}\right)$, it holds that*

$$\Pr_\rho[|Y_\perp - X_\perp| > k \cdot \varepsilon] < 2^{cc+k} \cdot t^k \cdot e^{-2t\varepsilon^2}.$$

Proof. Fix ε, t as stated in the lemma. We prove the lemma by induction over the levels of the interaction tree. Namely, we prove the following claim:

Proposition 3.12. *For all $i \in [k]$ and any verifier node τ in round i and any prover node $\tau' := \tau, r_i$ which is a child of τ it holds that*

$$\Pr\left[|Y_\tau - X_\tau| > (k - i + 1) \cdot \varepsilon\right] < 2^{\sum_{j=i+1}^k cc_j} \cdot 2^{k-i+1} \cdot t^{k-i+1} \cdot e^{-2t\varepsilon^2},$$

$$\Pr\left[|Y_{\tau'} - X_{\tau'}| > (k - i) \cdot \varepsilon\right] < 2^{\sum_{j=i+1}^k cc_j} \cdot 2^{k-i} \cdot t^{k-1} \cdot e^{-2t\varepsilon^2}.$$

Proof. (of Proposition 3.12) By reverse induction on i. First note that the assumption $t = \tilde{\Omega}\left(cc \cdot k \cdot \varepsilon^{-2}\right)$ guarantees that $2^{cc+k} \cdot t^k \cdot e^{-2t\varepsilon^2} \leq \frac{1}{2t}$.

Base: For a full transcript τ'_k, the corresponding nodes in both trees T_x and T'_x are leaves, therefore $Y_{\tau'_k} = X_{\tau'_k}$ which trivially implies $\Pr[|Y_{\tau'_k} - X_{\tau'_k}| > 0] = 0 \Rightarrow \Pr[|Y_{\tau'_k} - X_{\tau'_k}| > 0] < e^{-2t\varepsilon^2}$. In addition, for all τ_k which is a parent of a leaf, we can directly apply Lemma 3.7 with $\varepsilon' = 0, \delta' = e^{-2t\varepsilon^2}$ (indeed $\delta' < 2^{cc+2k} \cdot t^k \cdot e^{-2t\varepsilon^2} \leq \frac{1}{2t}$) and get $\Pr_{S_k}\left[|Y_{\tau_k} - X_{\tau_k}| > \varepsilon - 0\right] < t \cdot e^{-2t\varepsilon^2} + 4 \cdot e^{-2t\varepsilon^2} \leq 2 \cdot t \cdot e^{-2t\varepsilon^2}$.

Hypotheses: Fix $i \in \{1, \ldots, k-1\}$. Assume that for all verifier nodes τ_{i+1} in round $i+1$ and all prover nodes $\tau'_{i+1} := \tau r_{i+1}$ it holds that:

$$\Pr\left[|Y_{\tau_{i+1}} - X_{\tau_{i+1}}| > (k-i) \cdot \varepsilon\right] < 2^{\sum_{j=i+2}^k cc_j} \cdot 2^{k-i} \cdot t^{k-i} \cdot e^{-2t\varepsilon^2},$$

$$\Pr\left[|Y_{\tau'_{i+1}} - X_{\tau'_{i+1}}| > (k-i-1) \cdot \varepsilon\right] < 2^{\sum_{j=i+2}^k cc_j} \cdot 2^{k-i-1} \cdot t^{k-i-1} \cdot e^{-2t\varepsilon^2}.$$

Step: Let τ_i in round i. For any r_i, define $\tau'_i := \tau r_i$. For any π_{i+1}, the node $\tau_{i+1} := \tau'_i \pi_{i+1}$ is a verifier node for round $i+1$, therefore by the induction hypotheses, it holds that

$$\Pr\left[|Y_{\tau_{i+1}} - X_{\tau_{i+1}}| > (k-i) \cdot \varepsilon\right] < 2^{\sum_{j=i+2}^k cc_j} \cdot 2^{k-i} \cdot t^{k-i} \cdot e^{-2t\varepsilon^2}.$$

Therefore, we can apply Lemma 3.10 on τ'_i and get:

$$\begin{aligned}\Pr\left[|Y_{\tau'_i} - X_{\tau'_i}| > (k-i) \cdot \varepsilon\right] &< 2^{cc_{i+1}} \cdot 2^{\sum_{j=i+2}^k cc_j} \cdot 2^{k-i} \cdot t^{k-i} \cdot e^{-2t\varepsilon^2} \\ &= 2^{\sum_{j=i+1}^k cc_j} \cdot 2^{k-i} \cdot t^{k-i} \cdot e^{-2t\varepsilon^2}.\end{aligned}$$

Denote $\delta' = 2^{\sum_{j=i+1}^{k} cc_j} \cdot 2^{k-i} \cdot t^{k-i} \cdot e^{-2t\varepsilon^2} \le 2^{cc+2k} \cdot t^k \cdot e^{-2t\varepsilon^2} \le \frac{1}{2t}$ and apply Lemma 3.7 on τ with $\varepsilon' = (k-i) \cdot \varepsilon$ and δ':

$$\begin{aligned}\Pr\left[|Y_\tau - X_\tau| > (k-i+1)\varepsilon\right] &< t \cdot 2^{\sum_{j=i+1}^{k} cc_j} \cdot 2^{k-i} \cdot t^{k-i} \cdot e^{-2t\varepsilon^2} + 4 \cdot e^{-2t\varepsilon^2} \\ &\le 2 \cdot t \cdot 2^{\sum_{j=i+1}^{k} cc_j} \cdot 2^{k-i} \cdot t^{k-i} \cdot e^{-2t\varepsilon^2} \\ &= 2^{\sum_{j=i+1}^{k} cc_j} \cdot 2^{k-i+1} \cdot t^{k-i+1} \cdot e^{-2t\varepsilon^2},\end{aligned}$$

and the claim follows. ■

We apply Proposition 3.12 on $i = 1$ and get that for any transcript τ consisting only of a first prover message, it holds that $\Pr\left[|Y_\tau - X_\tau| > k \cdot \varepsilon\right] < 2^{\sum_{j=2}^{k} cc_j} \cdot 2^k \cdot t^k \cdot e^{-2t\varepsilon^2}$. We can apply Lemma 3.10 on the root $\perp$ (which is a prover node) and get that

$$\Pr\left[|Y_\perp - X_\perp| > k \cdot \varepsilon\right] < 2^{cc_1} \cdot 2^{\sum_{j=2}^{k} cc_j} \cdot 2^k \cdot t^k \cdot e^{-2t\varepsilon^2} = 2^{cc+k} \cdot t^k \cdot e^{-2t\varepsilon^2}.$$

This concludes the proof of Lemma 3.11. ■

We are now ready to prove the main result of this section.

Proof. (of Lemma 3.3.) Let $(\mathcal{P}, \mathcal{V})$ be an IOP for $\mathcal{L}$ as stated in the lemma, denote the CRS IOP that we get by using the CRS to prune the game tree by $(\mathcal{P}', \mathcal{V}')$. By Proposition 3.6, we have the desired completeness and the same query complexity, communication complexity and number of rounds as in $(\mathcal{P}, \mathcal{V})$. Next, we use Lemma 3.11 with parameters $\varepsilon = \frac{\varepsilon_{gap}}{3k}, t = \frac{27}{\varepsilon_{gap}^2}(\lambda + cc) \cdot k^3$. Using this, we get the following:

$$\begin{aligned}\Pr\left[|Y_\perp - X_\perp| > \frac{\varepsilon_{gap}}{3}\right] &< 2^{cc+k} \cdot t^k \cdot e^{-2t\varepsilon^2} \\ &= 2^{cc+k+k\log t} \cdot e^{-6k(\lambda+cc)} \\ &< 2^{cc+k+k\log t} \cdot 2^{-6k(\lambda+cc)} \\ &= 2^{cc+k+k\log(\lambda+cc)+3k\log k} \cdot 2^{-6k(\lambda+cc)} \\ &<^{(*)} 2^{k\log(\lambda+cc)+3k\log k} \cdot 2^{-5k(\lambda+cc)} \\ &<^{(**)} 2^{-k(\lambda+cc)} \\ &< 2^{-\lambda},\end{aligned}$$

where inequality $(*)$ follows from the fact that $cc + k < k(cc + \lambda)$ and inequality $(**)$ follows from $k\log(\lambda + cc) + 3k\log k < 4k(cc + \lambda)$. This means that for any $x \notin \mathcal{L}$ and $\mathcal{P}^*$, with probability $1 - 2^{-\lambda}$ over the choice of ρ, the probability that $\langle \mathcal{P}^*, \mathcal{V}' \rangle(\rho, x)$ accepts is $\varepsilon_S + \varepsilon_{gap}$ over the coins of $\mathcal{V}'$.

Similarly, we can prove that and the randomness that $\mathcal{V}'$ uses in each round is $\lceil \log t \rceil = O\big(\log(\lambda \cdot cc)\big)$. This concludes the proof of Lemma 3.3. ■

Remark 3.13. *Goldreich and Håstad [GH98] prove a similar result for standard interactive proofs using the pruning technique, but their application is transforming interactive proofs into probabilistic algorithms. This section demonstrates that the technique can be used to reduce the per-round randomness of the interactive (oracle) proof, in the form of transforming any (public-coin)* IOP *to a (public-coin)* CRS IOP *with small randomness complexity. In addition, our proof presents a slight improvement in parameters over that in [GH98]. Namely, our sample size t only depends linearly on the prover-to-verifier communication complexity cc, whereas the proof in [GH98] has $t = \Omega(cc^4)$.*

3.2 Non-Uniform IOPs with Small Randomness

We demonstrate how we can use Lemma 3.3 to get an IOP with small randomness complexity which is sound in the traditional sense, i.e., for all $x \notin \mathcal{L}$ and any prover strategy $\mathcal{P}^*$, the verifier accepts with probability at most 0.5. The resulting IOP is *non-uniform*. Loosely speaking, a non-uniform IOPs is an IOP in which the prover and the verifier receive (polynomial) non-uniform advice.

Corollary 3.14. *Let $\mathcal{L}$ be any language. If $\mathcal{L}$ has a k-round* IOP *with communication complexity cc then $\mathcal{L}$ has a non-uniform k-round* IOP *with communication complexity $O(cc)$ and randomness complexity $O(k \cdot \log(n \cdot cc))$.*

Proof. By Lemma 3.3, there exists a CRS IOP for $\mathcal{L}$ with CRS-error $\left(2^{-n-1}\right)$ and soundness error 0.6, where the communication complexity is cc and the randomness complexity is $O(k \cdot \log(n \cdot cc))$. Denote this CRS IOP by $(\mathcal{P}, \mathcal{V})$.

By the non-adaptive soundness of the CRS IOP, for any $x \notin \mathcal{L}$ and any prover strategy $\mathcal{P}^*$, with probability $1 - 2^{-n-1}$ over ρ, the prover $\mathcal{P}^*$ succeeds in making $\mathcal{V}$ accept with probability at most 0.6. This is equivalent to saying that with probability $1 - 2^{-n-1}$ over ρ, every prover strategy can make $\mathcal{V}$ accept with probability at most 0.6. Formally:

$$\Pr_{\rho \xleftarrow{\$} \{0,1\}^{\mathrm{poly}(n)}} \left[\forall \mathcal{P}^* : \Pr\left[\langle \mathcal{P}^*, \mathcal{V} \rangle(\rho, x) = 1 \right] \leq 0.6 \right] = 1 - 2^{-n-1}.$$

By a union bound over all "no" instances of length n, it holds that

$$\begin{aligned} &\Pr_{\rho \xleftarrow{\$} \{0,1\}^{\mathrm{poly}(n)}} \left[\forall x \in \{0,1\}^n \setminus \mathcal{L}, \forall \mathcal{P}^* : \Pr\left[\langle \mathcal{P}^*, \mathcal{V} \rangle(\rho, x) = 1 \right] \leq 0.6 \right] \\ &\quad = \Pr_{\rho \xleftarrow{\$} \{0,1\}^{\mathrm{poly}(n)}} \left[\exists x \in \{0,1\}^n \setminus \mathcal{L}, \forall \mathcal{P}^* : \Pr\left[\langle \mathcal{P}^*, \mathcal{V} \rangle(\rho, x) = 1 \right] > 0.6 \right] \\ &\quad \leq 2^n \cdot 2^{-n-1} \\ &\quad = \frac{1}{2} \end{aligned}$$

for a sufficiently large n. Therefore, there exists a specific ρ for which $\Pr\left[\langle \mathcal{P}^*, \mathcal{V} \rangle(\rho, x) = 1 \right] \leq 0.6$ for all $x \in \{0,1\}^n \setminus \mathcal{L}$ and all prover strategies $\mathcal{P}^*$ - this ρ is the non-uniform advice of our non-uniform IOP. The length of the advice is indeed $\mathrm{poly}(n)$, and the soundness error can be reduced to 0.5 by parallel repetition. ■

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 2.6) with instance length n and witness length m we get a $\mathrm{poly}(m)$-space algorithm with $\mathrm{poly}(n)$-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 4.1. *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 2.1). 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 $\Pr_{y \leftarrow \mathcal{A}_1(x)}\left[\mathcal{A}_2(y) = 1\right] \geq 1 - \varepsilon_c$.*
- *If $x \notin \mathcal{L}$ then $\Pr_{y \leftarrow \mathcal{A}_1(x)}\left[\mathcal{A}_2(y) = 1\right] \leq \varepsilon_s$.*

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 4.1. *If a language $\mathcal{L}$ has a k-round* IOP *with communication complexity cc, query complexity qc and verifier run-time $T_{\mathcal{V}}$, then $\mathcal{L}$ can be decided in $O(cc + k \cdot \log cc)$-space with $\left(2^{qc+O(k\cdot\log cc)} \cdot T_{\mathcal{V}}\right)$ 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 4.2. *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 $\mathrm{poly}(m)$-space with $\mathrm{poly}(n)$ preprocessing. Similarly, if $R_{\mathcal{L}}$ has a succinct $o\left(\frac{m}{\log m}\right)$-round* IOP *with perfect completeness and $o(m) + O(\log n)$ query complexity then $\mathcal{L}$ can be decided in $\mathrm{poly}(m)$-space with $2^{o(m)} \cdot \mathrm{poly}(n)$ preprocessing. The soundness error in both algorithms is $2^{-\mathrm{poly}(m)}$.*

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 fact[10]:

Fact 4.3. *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 4.1, 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:

Lemma 4.4. *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 $f : \{0,1\}^n \to \{0,1\}^{O(2^{rc+qc} \cdot \log cc)}$ that can be computed in $O\left(T_{\mathcal{V}} \cdot 2^{rc+qc}\right)$ 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 4.4) 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 preform any actual computation. Instead, $\mathcal{V}_s$ looks at $f(x)$ whenever it makes a query

[10]See, e.g., [Gol08, Chapter 9]

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 $\pi_1, ..., \pi_k$, 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 4.4, 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 3.3. After getting the input x, we let the preprocessor sample a random CRS ρ, and then apply Lemma 4.4 on both of x and ρ and the IOP that takes them as an input. By setting $\lambda = cc$ in Lemma 3.3, we get the following properties:

1. The preprocessor that computes $f(x)$ would run in $2^{qc} \cdot T_{\mathcal{V}} \cdot \mathrm{poly}(cc)^k$ time.
2. The string $f(x)$ would have length $2^{qc} \cdot \mathrm{poly}(cc)^k$.
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 4.1 using Lemma 4.4 and Fact 4.3.

Proof. (of Theorem 4.1) 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 3.3 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 4.4, there exists a function $f : \{0,1\}^{n+|\rho|} \to \{0,1\}^{O(2^{qc} \cdot \mathrm{poly}(cc)^k)}$ 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 4.3, 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 Section 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 2.5). Furthermore, we do so under the minimal [OW93] cryptographic assumption of one-way functions.

In Section 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 5.1. *Assume the existence of one-way functions. Let $\mathcal{L} \in \mathsf{NP}$. If $(P_{\mathsf{IOP}}, V_{\mathsf{IOP}})$ is a T_V-uniform γ-compact* IOP *for $\mathcal{L}$, with soundness error $\varepsilon_{\mathsf{IOP}}$, 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 $\varepsilon_{\mathsf{IOP}} + 2^{-\lambda}$ and proof length $cc + \mathrm{poly}(\lambda, \gamma, \log cc)$, where $\lambda > 0$ is the security parameter. Furthermore, the running time of the* ZKP *verifier is $T_V + \mathrm{poly}(\lambda, \gamma, \log cc)$ and the running time of the prover is $T_P + \mathrm{poly}(\lambda, \gamma)$.*

5.1.1 The Transformation

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

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

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

Let x be the input. For $\langle P_{\mathsf{IOP}}(w), V_{\mathsf{IOP}}\rangle(x)$, 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 $\pi = (\pi_1, ..., \pi_k)$ and full access to its own random coins $r = (r_1, ..., r_k)$. In the local computation phase, V_{IOP} produces a predicate $\phi_{x,r} : \{0,1\}^{qc} \to \{0,1\}$ 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 $\phi : \{0,1\}^{qc} \to \{0,1\}$ is a predicate, $\rho \in \{0,1\}^{\mathrm{poly}(\lambda)}$ 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 $c \in \{0,1\}^{\mathrm{poly}(\lambda)}$ 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

Communication Preserving ZK Protocol

1. V generates a reference string $\rho \leftarrow Gen\left(1^{\lambda+1}\right)$ for the commitment scheme and sends it to P.
2. P generates a random PRF seed $S \in_R \{0,1\}^\lambda$, commits to it using $(com, dec) \leftarrow Com\left(1^{\lambda+1}, \rho, S\right)$ and sends the commitment com to V.
3. P initializes $\ell \leftarrow 0$, $p = F_S([cc])$ then performs with V the following "encrypted" version of the IOP: For $i = 1, \ldots, k$:
   - P sends an encryption of the i^{th} P_{IOP}'s message π_i by XORing it with fresh bits from p, i.e., sends $\alpha_i = p[\ell, .., \ell + |\pi_i| - 1] \oplus \pi_i$ and updates $\ell \leftarrow \ell + |\pi_i|$.
   - The verifier replies with the usual random coins r_i as in the IOP.

   Denote $\alpha = (\alpha_1, \ldots, \alpha_k)$ and $r = (r_1, ..., r_k)$.
4. P and V emulate $(P_{\mathcal{L}'}, V_{\mathcal{L}'})$ with $\left(\phi_{x,r}, \rho, Q_{x,r}, \alpha\left[Q_{x,r}\right], com\right)$ as common input and where $P_{\mathcal{L}'}$ further uses (S, dec) as its witness, with security parameter $\lambda + 1$ and V answers accordingly.

Figure 1: The ZK protocol from Theorem 5.1

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:

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

The length of an instance (ϕ, ρ, Q, y, c) is $|\phi| + \mathrm{poly}(\lambda) + qc \cdot O(\log cc)$. The language $\mathcal{L}'$ is clearly in NP since given S and d (which have $\mathrm{poly}(\lambda)$ length), a verifier can verify that $Ver(1^\lambda, \rho, c, S, d) = 1$ in $\mathrm{poly}(\lambda)$ time, compute $y \oplus F_S[Q]$ in $\mathrm{poly}(\lambda, qc, \log cc)$ time and verify $\phi\left(y \oplus F_S[Q]\right) = 1$ 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 5.1. *The protocol in Fig. 1 is a public-coin interactive proof for $\mathcal{L}$. The proof length is $cc + \mathrm{poly}(\lambda, \gamma, \log cc)$, the soundness error is $\varepsilon_{\mathsf{IOP}} + 2^{-\lambda}$, the verifier runs in time $T + \mathrm{poly}(\lambda, \gamma, \log cc)$ and the prover runs in time $T_P + \mathrm{poly}(\lambda, \gamma)$.*

Lemma 5.2. *The protocol in Fig. 1 is computational zero-knowledge w.r.t. auxiliary input.*

5.1.2 Proof of Lemma 5.1

We start by proving the completeness and soundness of the protocol, then move on to establishing the complexity bounds.

Completeness. Let $x \in \mathcal{L}$. By the correctness of (Gen, Com, Ver), the decommitment string dec and the seed S that the honest prover have satisfy the first condition of $\mathcal{L}'$, i.e., $Ver\left(1^{\lambda+1}, \rho, com, dec, S\right) = 1$. In addition, $\alpha_1, \ldots, \alpha_k$ are the encryptions of the honest IOP verifier messages $\pi_1, ..., \pi_k$, so for $y = \alpha[Q_{x,r}]$ we get that $y \oplus F_S[Q_{x,r}] = \pi[Q_{x,r}]$. Since $x \in \mathcal{L}$ and $(P_{\mathsf{IOP}}, V_{\mathsf{IOP}})$ has perfect completeness, we get that the second condition of $\mathcal{L}'$ is satisfied, i.e., $\phi_{x,r}(y \oplus F_S[Q_{x,r}]) = 1$. Therefore, $(\phi_{x,r}, \rho, Q_{x,r}, \alpha[Q_{x,r}], com) \in \mathcal{L}'$. By the perfect completeness of $(P_{\mathcal{L}'}, V_{\mathcal{L}'})$, the verifier accepts in step 4.

Soundness. Let $x \notin \mathcal{L}$ and P^* be a fixed cheating prover strategy. In step 2, P^* sends an alleged commitment com^* of a seed S, then in the step 3, P^* sends k messages $\alpha^* = \alpha_1^*, .., \alpha_k^*$. In step 4, P^*

tries to convince V that $(\phi_{x,r}, \rho, Q_{x,r}, \alpha^*[Q_{x,r}], com^*) \in \mathcal{L}'$, i.e., that there exists a dec^*, S such that $Ver\left(1^{\lambda+1}, \rho, com^*, dec^*, S\right) = 1$ and $\phi_{x,r}(\alpha^*[Q_{x,r}] \oplus F_S[Q_{x,r}]) = 1$. The idea is that the committed values do not satisfy the predicate (with high enough probability) by the soundness of the IOP, and the probability that the prover can reveal different values is negligible by the binding property of the commitment scheme. This is formalized in the following claim:

Claim 5.2.1. $\Pr\left[\left(\phi_{x,r}, \rho, Q_{x,r}, \alpha^*[Q_{x,r}], com^*\right) \in \mathcal{L}'\right] \leq \varepsilon_{\mathsf{IOP}} + 2^{-\lambda-1}$.

Proof. We say that a string S is consistent with com^* if there exists some decommitment string dec^* s.t. $Ver\left(1^{\lambda}, \rho, S, com^*, dec^*\right) = 1$. Denote by A_1 the event that no S is consistent with com^*, by A_2 the event that exactly one S is consistent with com^* and by A_3 that more than one value is consistent with com^*. These three events partition the probability space. Denote by B the event that $\left(\phi_{x,r}, \rho, Q_{x,r}, \alpha^*[Q_{x,r}], com^*\right) \in \mathcal{L}'$. By the binding property of the commitment scheme, the probability over $\rho \leftarrow Gen\left(1^{\lambda}\right)$ of A_3 happening is at most $2^{-\lambda-1}$. Assuming that there is exactly one valid value S that can be revealed, i.e., that event A_2 happens, we get that the decryption of the messages $\alpha_1, ..., \alpha_k$ w.r.t. that S constitutes a prover strategy for the IOP. By the soundness of $(P_{\mathsf{IOP}}, V_{\mathsf{IOP}})$, the probability over the random coins of V_{IOP} that this strategy succeeds is at most $\varepsilon_{\mathsf{IOP}}$. Finally, assuming A_1, the probability of B happening is 0. So in total,

$$\begin{aligned}
&\Pr\left[\left(\phi_{x,r}, \rho, Q_{x,r}, \alpha^*[Q_{x,r}], com^*\right) \in \mathcal{L}'\right] \\
&\quad = \Pr\left[A_1 \wedge B\right] + \Pr\left[A_2 \wedge B\right] + \Pr\left[A_3 \wedge B\right] \\
&\quad \leq \Pr\left[B \mid A_2\right] + \Pr\left[A_3\right] \\
&\quad \leq \varepsilon_{\mathsf{IOP}} + 2^{-\lambda-1},
\end{aligned}$$

where the probability is over ρ and r. This completes the proof of the claim. ■

Assuming $(\phi_{x,r}, \rho, Q_{x,r}, \alpha^*[Q_{x,r}], com^*) \notin \mathcal{L}'$, the verifier V rejects with probability at least $2^{-\lambda-1}$ by the soundness of $(P_{\mathcal{L}'}, V_{\mathcal{L}'})$. Thus, overall, the verifier accepts with probability at most $2^{-\lambda-1} + \varepsilon_{\mathsf{IOP}} + 2^{-\lambda-1} = \varepsilon_{\mathsf{IOP}} + 2^{-\lambda}$
and soundness follows.

Proof length. We analyze the length of the prover messages for each step:

1. In step 1, the prover does not send any messages.
2. In step 2, the prover commits to a seed $S \in \{0,1\}^{\lambda}$ with security parameter $\lambda + 1$ which requires $\mathrm{poly}(\lambda)$ bits of communication.
3. In step 3, the number of bits that the prover sends is exactly cc.
4. In step 4, the communication depends on $(P_{\mathcal{L}'}, V_{\mathcal{L}'})$. The communication complexity is polynomial in the size of the tuple $(\phi_{x,r}, \rho, Q_{x,r}, \alpha[Q_{x,r}], com)$ as well as S and dec. We recall that $(P_{\mathsf{IOP}}, V_{\mathsf{IOP}})$ is γ-compact, therefore $|\phi_{x,r}| = O(\gamma(n))$. In addition, the size of the predicate bounds the number of queries, i.e., $qc = O(\gamma(n))$. The query set $Q_{x,r}$ can be represented using $qc \cdot O(\log cc)$ bits and the commitment com, decommitment dec, seed S and CRS ρ have $\mathrm{poly}(\lambda)$ size. In total, the communication complexity used in $(P_{\mathcal{L}'}, V_{\mathcal{L}'})$ is $\mathrm{poly}(\lambda, \gamma, \log cc)$.

Therefore, the total number of bits sent by the prover is $cc + \mathrm{poly}(\lambda, \log cc, \gamma)$.

Public-coin. The protocol is public-coin since the underlying IOP and ZKP are both public-coin and the verifier can just send the random coins it used in Gen to generate ρ.

Verifier time. The verifier runs $Gen(1^{\lambda+1})$ in step 1 and then simulates V_{IOP} until it computes $\phi_{x,r}, Q_{x,r}$. This takes $T + \mathrm{poly}(\lambda)$ times. In step 4, the verifier simulates $V_{\mathcal{L}'}$ which takes $\mathrm{poly}(\lambda, \gamma, \log cc)$. In total, the running time of the verifier V is $T + \mathrm{poly}(\lambda, \gamma, \log cc)$.

Prover time. The prover P runs $Com\left(1^{\lambda+1}, \rho, S\right)$ and then simulates P_{IOP} (while encrypting each message using F_S) and $P_{\mathcal{L}'}$. Running Com requires $\mathrm{poly}(\lambda)$ time. Simulating P_{IOP} can be done in T_P time, and simulating $P_{\mathcal{L}'}$ is done in polynomial time since P has the witness S, d, that is $\mathrm{poly}(\lambda, \gamma, \log cc)$. Note that $\log cc = o(T_P)$ since T_P includes generating and sending all of the IOP prover messages which have length cc. In total, the prover P runs in $T_P + \mathrm{poly}(\lambda, \gamma)$.
This completes the proof of Lemma 5.1.

5.1.3 Proof of Lemma 5.2

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 2.15, and denote by $V^*_{\mathcal{L}'}$ the residual verifier strategy[11] that V^* uses in step 4. Let $x \in \mathcal{L}$ be an instance, $z \in \{0,1\}^{\mathrm{poly}(|x|)}$ 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 $View_{V^*(z)}(x) := View^{P(w)}_{V^*(z)}(x)$, consists of x,z, the commitment com, the encrypted messages α and the view of $V^*_{\mathcal{L}'}$ in step 4, denoted by $View'\left(\phi_{x,r}, \rho, Q_{x,r}, \alpha[Q_{x,r}], com, z_{\mathcal{L}'}, (S, dec)\right) := View^{P_{\mathcal{L}'}(S,dec)}_{V^*_{\mathcal{L}'}(z_{\mathcal{L}'})}\left(\phi_{x,r}, \rho, Q_{x,r}, \alpha[Q_{x,r}], com\right)$ 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

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

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 $\alpha\left[Q_{x,r}\right]$ twice. Recall that we assume $(P_{\mathsf{IOP}}, V_{\mathsf{IOP}})$ has perfect completeness, therefore for any verifier messages ρ, r and honestly generated prover messages com and α, it holds that $\left(\phi_{x,r}, \rho, Q_{x,r}, \alpha\left[Q_{x,r}\right], com\right) \in \mathcal{L}'$. 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 $View^{P(w)}_{V^*(z)}(x)$. We start with a high-level description. Given as input x, z, the simulator emulates step 1 of 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.

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 2.10) 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:

[11]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.

ZK Simulator for Theorem 5.1

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 $(c_0, d_0) \leftarrow Com\left(1^\lambda, \rho, 0^\lambda\right)$. "Send" c_0 to V^*.
3. For $i = 1,..,k$: Generate a random β_i and "send" it to V^* and get in response r_i.
4. Output $Sim'_{V^*}\left(x, r, z, \beta, \rho, c_0, 1^\lambda\right)$.

Figure 2: The simulator Sim_{V^*} for Theorem 5.1

Proposition 5.3. *For all $x \in \mathcal{L}$ and auxiliary input $z \in \{0,1\}^{\text{poly}(|x|)}$,*

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

Recall that Sim_{V^*} applies the simulator $Sim'_{V^*_{\mathcal{L}'}}$ on a "mock" instance instead of a "yes" instance. The following proposition states that the parts in which the "mock" instance differs from a "yes" instance are computationally indistinguishable:

Proposition 5.4. *Let $x \in \mathcal{L}$ and $\pi \in \{0,1\}^{cc}$ be any string. For any* CRS *$\rho \in \{0,1\}^{\text{poly}(\lambda)}$, the following distribution ensembles are computationally indistinguishable:*

- $\left\{\left(\beta, c_0, 1^\lambda\right)\right\}_{\lambda\in\mathbb{N}}$ *where $\beta \overset{\$}{\leftarrow} \{0,1\}^{cc}$, and $(c_0, d_0) \leftarrow Com(1^{\lambda+1}, \rho, 0^\lambda)$.*
- $\left\{\left(\alpha, c_S, 1^\lambda\right)\right\}_{\lambda\in\mathbb{N}}$ *where $S \overset{\$}{\leftarrow} \{0,1\}^\lambda$, $\alpha \leftarrow F_S([|\pi|]) \oplus \pi$ and $(c_S, d_S) \leftarrow Com(1^{\lambda+1}, \rho, S)$.*

Proof. (of Proposition 5.4) We prove the claim using a hybrid argument. Denote the first and second ensemble by H and H', respectively. We define the intermediate hybrid $I = \left\{\left(\alpha, c_0, 1^\lambda\right)\right\}_{\lambda\in\mathbb{N}}$, where $(c_0, d_0) \leftarrow Com(1^{\lambda+1}, \rho, 0^\lambda)$ and $\alpha = F_S([|\pi|]) \oplus \pi$. We note that I is defined similarly to H' except that instead of committing to the PRF seed $S \in \{0,1\}^\lambda$, we commit to 0^λ. Observe that by the computational hiding property of (Gen, Com, Ver), the commitments are computationally indistinguishable and therefore $I \overset{c}{\approx} H'$. By Fact 2.13, the string $\alpha = F_S([|\pi|]) \oplus \pi$ is pseudorandom, i.e., it is computationally indistinguishable from the random string β and therefore $H \overset{c}{\approx} I$. We get that $H \overset{c}{\approx} H'$, and the proposition follows. ■

We are now ready to prove Proposition 5.3.

Proof. (of Proposition 5.3) Fix $x \in \mathcal{L}$ and some auxiliary input $z \in \{0,1\}^{\text{poly}(x)}$. Recall that V^* is deterministic, therefore we can assume a fixed CRS ρ that V^* sends as its first message when interacting with P over x and auxiliary input z.

Let $S \overset{\$}{\leftarrow} \{0,1\}^\lambda$ be a random variable that denotes a seed. We look at the following distributions that depend in S:

- The commitment c_S of S relative to the CRS ρ.
- The transcript (π, r) of the interaction between P_{IOP} and a verifier that responds according to the messages that V^* sends, when given as input also the commitment c_S as well as explicit access to the encryption of each prover message in each round.

We also let β be a uniform distribution over $\{0,1\}^{cc}$ and c_0 be the distribution of the commitment $Com(1^{\lambda+1}, \rho, 0^\lambda)$ and denote $\alpha = F_S([|\pi|]) \oplus \pi$. Now, we define the following distribution ensembles:

- $H = \left\{H_\lambda = \left(x, r, z, \beta, \rho, c_0, 1^\lambda\right)\right\}_{\lambda\in\mathbb{N}}$.

- $H' = \left\{ H'_\lambda = \left(x, r, z, \alpha, \rho, c_S, 1^\lambda\right) \right\}_{\lambda \in \mathbb{N}}$.

By Proposition 5.4, the distribution ensembles $\left\{(\beta, c_0, 1^\lambda)\right\}_{\lambda \in \mathbb{N}}$ and $\left\{(\alpha, c_S, 1^\lambda)\right\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable. Since H,H' are mere extensions of those ensemble (obtained by concatenating the identical distributions x, r, z, ρ), we get that $H \overset{c}{\approx} H'$. By Fact 2.10, $Sim'_{V^*_{\mathcal{L}'}}(H) \overset{c}{\approx} Sim'_{V^*_{\mathcal{L}'}}(H')$. Note that H is a distribution of "mock" inputs for $Sim'_{V^*_{\mathcal{L}'}}$ (that Sim_{V^*} passes to $Sim'_{V^*_{\mathcal{L}'}}$). By the perfect completeness of $(P_{\mathsf{IOP}}, V_{\mathsf{IOP}})$, for any verifier messages r and honestly generated prover messages π, it holds that $(\phi_{x,r}, \rho, Q_{x,r}, \alpha[Q_{x,r}], c_S) \in \mathcal{L}'$, therefore H' is a distribution of "yes" inputs for $Sim'_{V^*_{\mathcal{L}'}}$.

Leveraging the fact that H' corresponds to "yes" instances for $\mathcal{L}'$ with auxiliary input $z_{\mathcal{L}'} = (x, r, \alpha, \rho)$, we can use the zero-knowledge property of $(P_{\mathcal{L}'}, V_{\mathcal{L}'})$ and get that:

$$\left\{ View'\left(\phi_{x,r}, \rho, Q_{x,r}, \alpha[Q_{x,r}], c_S, z_{\mathcal{L}'}, \lambda\right) \right\}_{\lambda \in \mathbb{N}} \overset{c}{\approx} Sim'_{V^*_{\mathcal{L}'}}(H') \overset{c}{\approx} Sim'_{V^*_{\mathcal{L}'}}(H). \tag{4}$$

On the other hand, by the construction of Sim_{V^*}, its output is exactly:

$$\left\{ Sim_{V^*}\left(x, z, 1^\lambda\right) \right\}_{\lambda \in \mathbb{N}} = Sim'_{V^*_{\mathcal{L}'}}(H). \tag{5}$$

Putting Eq. (5) and Eq. (4) together, we get that

$$\left\{ Sim_{V^*}\left(x, z, 1^\lambda\right) \right\}_{\lambda \in \mathbb{N}} \overset{c}{\approx} \left\{ View'\left(\phi_{x,r}, \rho, Q_{x,r}, \alpha[Q_{x,r}], com, z_{\mathcal{L}'}, \lambda\right) \right\}_{\lambda \in \mathbb{N}}.$$

Finally, recall that the view of V^* is:

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

So in total,

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

and the proposition follows. ■

Proof. (of Lemma 5.2) Let $x \in \mathcal{L}$ and w be the corresponding witness. Fix some polynomial time verifier V^* and auxiliary input z. By Proposition 5.3, the output of the simulator Sim_{V^*} on input $(x, z, 1^\lambda)$ is computationally from the view $View^{P(w)}_{V^*(z)}(x, \lambda)$. 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 5.1 and Lemma 5.2 complete the proof of Theorem 5.1.

5.2 Constructing Succinct ZKPs

Our next step is to use Theorem 5.1 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 5.2. *(Extension of [RR20]) Let $\mathcal{L} \in \mathsf{NP}$ 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 $\mathrm{poly}(m+n)$ and space $s \geq \log m$. For any constants $\beta, \gamma \in (0,1)$, there exists a $\beta^{-O\left(\frac{1}{\beta}\right)}$-round IOP for $\mathcal{L}$ with soundness error $\frac{1}{2}$ and $(\gamma\beta)^{-O\left(\frac{1}{\beta}\right)}$ 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 $\mathrm{poly}\left(n^\beta, (\gamma\beta)^{-\frac{1}{\beta}}, s\right)$ additional communication. In addition, the IOP is $\left(\tilde{O}(n) + \mathrm{poly}\left(n^\beta, (\gamma\beta)^{-\frac{1}{\beta}}, s\right)\right)$-uniform $\mathrm{poly}\left(n^\beta, (\gamma\beta)^{-\frac{1}{\beta}}, s\right)$-compact and the prover runs in $\mathrm{poly}(n)$ time.*

Remark 5.5. *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 fist message only depends on the witness length, whereas we do not make that assumption and therefore Theorem 5.2 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 5.2 is proved in the full version [NR22].*

The soundness error in Theorem 5.2 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 5.6. *There exists a fixed constant $\xi > 0$ such that the following holds. Let $\mathcal{L} \in \mathsf{NP}$ with a corresponding relation $\mathcal{R}_\mathcal{L}$ in which the instances have length n and witnesses have length m such that $m \le n$ and $\mathcal{R}_\mathcal{L}$ can be decided in $\mathrm{poly}(n)$ 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 $(1+\gamma)\cdot m + O\big(\log \frac{1}{\varepsilon}\big)\cdot\gamma\cdot n^\beta$, 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 $\mathrm{poly}(n)$ time.*

Corollary 5.6 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 5.1 on the IOP from Corollary 5.6 and get a succinct ZKP for all languages in that class. This yields our main theorem:

Theorem 5.3. *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 $m \le n$, that can be decided in $\mathrm{poly}(n)$ 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 $(1+\gamma)\cdot m + \gamma \cdot n^\beta \cdot \mathrm{poly}(\lambda)$ proof length, perfect completeness and soundness error $2^{-\lambda}$. Furthermore, the verifier runs in time $\tilde{O}(n) + n^\beta \cdot \mathrm{poly}(\lambda)$ and the prover runs in $\mathrm{poly}(n)$ time.*

Proof. The relation $\mathcal{R}_\mathcal{L}$ satisfies the premise of Corollary 5.6. Therefore for any constant $\gamma_0 \in (0,1)$ and any function $\varepsilon = \varepsilon(n) \in (0,1)$ there exists a constant $\beta'_0 \in (0,1)$ such that for any $\beta_0 \in (0, \beta'_0)$ there exists a $\tilde{O}(n)$-uniform n^β-compact IOP for the corresponding language $\mathcal{L}$ with soundness error ε, constant query complexity, communication complexity $(1+\gamma)\cdot m + O\left(\log\frac{1}{\varepsilon}\right)\cdot\gamma\cdot n^\beta$ and prover time $\mathrm{poly}(n)$, if given the witness. Assuming one-way functions exist and choosing $\varepsilon = 2^{-\lambda-1}$, we can apply Theorem 5.1 with security parameter $\lambda+1$ on the previous IOP to get a public-coin ZKP for $\mathcal{L}$ with the following properties:

- Soundness error $2^{-\lambda-1} + 2^{-\lambda-1} = 2^{-\lambda}$.
- Proof length $(1+\gamma_0)\cdot m + O(\lambda)\cdot\gamma_0\cdot n^{\beta_0} + \mathrm{poly}\left(\lambda, n^{\beta_0}, \log n\right)$.
- The verifier runs in $\tilde{O}(n) + \mathrm{poly}\left(\lambda, n^{\beta_0}, \log n\right)$.
- The prover runs in $\mathrm{poly}(n)$ time.

By choosing a sufficiently small β'_0 and $\gamma_0 = \gamma$, we get $(1+\gamma)\cdot m + \gamma n^\beta \cdot \mathrm{poly}(\lambda, \log n)$ proof length and $\tilde{O}(n) + n^\beta \cdot \mathrm{poly}(\lambda, \log n)$ verifier time. Since $\mathrm{polylog}(n) = O(n^\beta)$, we can drop the $\log n$ and get the desired proof length and verifier time. ∎

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] Benny Applebaum and Eyal Golombek. On the randomness complexity of interactive proofs and statistical zero-knowledge proofs. In Stefano Tessaro, editor, *2nd Conference on Information-Theoretic Cryptography, ITC 2021, July 23-26, 2021, Virtual Conference*, volume 199 of *LIPIcs*, pages 4:1–4:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[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.

[BDRV18] Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. From laconic zero-knowledge to public-key cryptography - extended abstract. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 674–697. Springer, 2018.

[BGG$^+$88] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56. Springer, 1988.

[Bra19] Zvika Brakerski. Fundamentals of fully homomorphic encryption. In Oded Goldreich, editor, *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 543–563. ACM, 2019.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015*, volume 9015 of *Lecture Notes in Computer Science*, pages 468–497. Springer, 2015.

[CY20] Alessandro Chiesa and Eylon Yogev. Barriers for succinct arguments in the Random Oracle Model. 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 47–76. Springer, 2020.

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

[GGI$^+$15] Craig Gentry, Jens Groth, Yuval Ishai, Chris Peikert, Amit Sahai, and Adam D. Smith. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. *J. Cryptol.*, 28(4):820–843, 2015.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

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

[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.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. 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 171–185. Springer, 1986.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications.* Cambridge University Press, 2004.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective.* Cambridge University Press, 2008.

[GR18] Tom Gur and Ron D. Rothblum. Non-interactive proofs of proximity. *Comput. Complex.*, 27(1):99–207, 2018.

[GS10] Oded Goldreich and Or Sheffet. On the randomness complexity of property testing. *Comput. Complex.*, 19(1):99–133, 2010.

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

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[HPV77] John E. Hopcroft, Wolfgang J. Paul, and Leslie G. Valiant. On time versus space. *J. ACM*, 24(2):332–337, 1977.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[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.

[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, *ICALP 2008*, volume 5126 of *Lecture Notes in Computer Science*, pages 536–547. Springer, 2008.

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

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptol.*, 4(2):151–158, 1991.

[New91] Ilan Newman. Private vs. common random bits in communication complexity. *Inf. Process. Lett.*, 39(2):67–71, 1991.

[NR22] Shafik Nassar and Ron D. Rothblum. Succinct interactive oracle proofs: Applications and limitations. Cryptology ePrint Archive, Paper 2022/281, 2022. https://eprint.iacr.org/2022/281.

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way fuctions are essential for non-trivial zero-knowledge. In *Second Israel Symposium on Theory of Computing Systems, ISTCS 1993, Natanya, Israel, June 7-9, 1993, Proceedings*, pages 3–17. IEEE Computer Society, 1993.

[PTC77] Wolfgang J. Paul, Robert Endre Tarjan, and James R. Celoni. Space bounds for a game on graphs. *Math. Syst. Theory*, 10:239–251, 1977.

[RR20] 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.

[RRR21] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. *SIAM J. Comput.*, 50(3), 2021.

[Sma14] How do we know that P != LINSPACE without knowing if one is a subset of the other? https://mathoverflow.net/questions/40770/how-do-we-know-that-p-linspace-without-knowing-if-one-is-a-subset-of-the-othe, 2014. [Online; accessed 2-2-2022].

A More on Conjecture 1.2

In this section, we elaborate further on Conjecture 1.2. We start by recalling it:

Conjecture 1.2. *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* $\mathrm{poly}(m)$ *space and $t(n,m)$-time probabilistic preprocessing, for any $t \in \mathcal{T}$.*

Circuit Value Problem. The *circuit value problem* (CVAL) is the computational problem in which you are given the description of a Boolean circuit C and an input x and need to output $C(x)$. CVAL is known to be P-complete under log-space reductions. Under the widely believed conjecture that $\mathsf{P} \not\subseteq \mathsf{Space}(\log)$ it holds that CVAL cannot be solved in logarithmic space. But little has been achieved in solving CVAL even in sub-linear space; in particular, the most space-efficient algorithm known to date, which follows the so-called "pebbling" approach yields a $O(\frac{n}{\log n})$-space algorithm [HPV77] where n is the size of the circuit. It was later proved that for general circuits, the pebbling approach cannot do much better [PTC77], i.e., there is a space lower bound of $\Omega(\frac{n}{\log n})$ for solving CVAL using pebbling. Since the number of variables in a circuit can be much smaller than its size, and in particular, the circuit size can be super-polynomial in the number of variables, we believe it is reasonable to assume that a general circuit cannot be evaluated in space that is polynomial in the number of variables.

Connection to CSAT. Observe that CVAL is a much easier problem than CSAT. Beyond the fact that CSAT is NP complete whereas CVAL is in P, there is a log-space reduction from CVAL to CSAT. Therefore if CVAL cannot be solved in space $s > \log n$ then CSAT cannot be solved in space s. But the connection between those two problems seems even stronger: the natural way to decide if a Boolean circuit of size n over m variables is satisfiable is by enumerating all possible assignments and running the best CVAL algorithm wrt each assignment.

Connection to Conjecture 1.2 If a circuit of size n over m variables can be solved in $\mathrm{poly}(m)$ space with some non-trivial time probabilistic preprocessing (say $\mathrm{poly}(n)$) then this would imply that either it can be solved in a single phase; in $\mathrm{poly}(m)$ space or non-trivial time, or it would imply a surprising interplay between space and time complexities and in particular, yield an interesting space-time trade-off for CSAT.

B IOP for small space relations

In this section, we prove Theorem 5.2, which states that there exists a succinct IOP for NP relations that can be verified in small space. This result is a small extension of the main result of [RR20] and uses the same IOP construction from that paper.

Theorem 5.2. *(Extension of [RR20]) Let $\mathcal{L} \in \mathsf{NP}$ 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 $\mathrm{poly}(m+n)$ and space $s \geq \log m$. For any constants $\beta, \gamma \in (0,1)$, there exists a $\beta^{-O\left(\frac{1}{\beta}\right)}$-round* IOP *for $\mathcal{L}$ with soundness error $\frac{1}{2}$ and $(\gamma\beta)^{-O\left(\frac{1}{\beta}\right)}$ 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 $\mathrm{poly}\left(n^{\beta}, (\gamma\beta)^{-\frac{1}{\beta}}, s\right)$ additional communication. In addition, the* IOP *is $\left(\tilde{O}(n) + \mathrm{poly}\big(n^{\beta}, (\gamma\beta)^{-\frac{1}{\beta}}, s\big)\right)$-uniform $\mathrm{poly}\left(n^{\beta}, (\gamma\beta)^{-\frac{1}{\beta}}, s\right)$-compact and the prover runs in $\mathrm{poly}(n)$ time.*

The difference between Theorem 5.2 and the main result of [RR20] is the IOP is compact, as well as the fact that we do not require the instance and witness lengths to be polynomially related. Since Theorem 5.2 is a minor extension of a previous work, we opt to give an overview of that work. We start by defining the key ingredients that are used in the proof, formally state the results that we rely on and discuss the main ideas behind the proofs of these results. Finally, in Appendix B.2, we prove Theorem 5.2.

B.1 Summary of Previous Work

B.1.1 Preliminaries

The protocol we discuss relies on a variant of interactive oracle proofs called *interactive oracle proofs of proximity* (IOPP). Loosely speaking, this is an IOP in which the verifier only has oracle access to the input and accepts if the input is in the language and rejects if the input is "far" from the language. The notion of distance that we use is relative Hamming distance. Formally, for any $x \in \{0,1\}^*$, language $\mathcal{L} \neq \emptyset$ and parameter $\delta \in (0,1)$, we say that x is *δ-far* from $\mathcal{L}$ if the relative Hamming distance between x and any $x' \in \mathcal{L}$ is at least δ. IOPPs are formally defined as follows.

Definition B.1. *A public-coin* IOPP *w.r.t. proximity parameter $\delta > 0$ and soundness error $\varepsilon > 0$ for a language $\mathcal{L}$ is a pair $(\mathcal{P}, \mathcal{V})$ of probabilistic algorithms that satisfy the following requirements:*

- ***Input:*** *$\mathcal{P}$ receives $x \in \{0,1\}^n$ as input and $\mathcal{V}$ has oracle access to x.*
- ***Interaction phase:*** *$\mathcal{P}$ and $\mathcal{V}$ interact for $k(n)$ rounds in the following manner: in round each i, $\mathcal{P}$ sends a message π_i and $\mathcal{V}$ replies with a random string r_i. Denote $r = r_1 ... r_k$ and $\pi = \pi_1 ... \pi_k$.*
- ***Local computation phase:*** *$\mathcal{V}$ deterministically (based on r) produces a query vector $\vec{ql}_r \in [|x| + |\pi|]^q$ of q queries and a predicate $\mathcal{V}_r : \{0,1\}^q \to \{0,1\}$.*
- ***Evaluation phase:*** *$\mathcal{V}$ queries π and x, plugs the values into the predicate and outputs $\mathcal{V}_r\left((x \circ \pi)\left[\vec{ql}_r\right]\right)$.*
- ***Completeness:*** *If $x \in \mathcal{L}$ then $\mathcal{V}$ outputs 1 with probability 1.*
- ***Soundness:*** *If x is δ-far from $\mathcal{L}$ then for any prover strategy $\mathcal{P}^*$, when $\mathcal{V}$ interacts with $\mathcal{P}^*$, it accepts with probability at most ε.*

A key ingredient in optimizing the communication complexity is a property of error-correcting codes called *local decomposability.* Roughly speaking, a code that has this property allows us to compute any coordinate in the encoding of a message m by making a constant number of queries to encodings of sub-strings of m. This is particularly useful when a verifier has a large portion of the string, say an NP instance x, the prover has the remaining part, say the corresponding NP witness w, and the protocol dictates that the prover sends an encoding of $x \circ w$. In this case, the verifier can compute the encodings of the sub-strings corresponding to x and the prover can send the encoding of w. The property is formally defined below.

Definition B.2 (Locally decomposable code)**.** *Let $C : \{0,1\}^n \to \{0,1\}^{n'}$ be an error-correcting code, and let ℓ be an integer that divides n. We say that C is* locally ℓ-decomposable *if there exists a* base code *$C_0 : \{0,1\}^{\frac{n}{\ell}} \to \{0,1\}^*$ and a deterministic oracle machine A such that the following holds. For any $m^{(1)}, \dots, m^{(\ell)} \in \{0,1\}^{\frac{n}{\ell}}$, given oracle access to the codewords $C_0\left(m^{(1)}\right), \dots, C_0\left(m^{(\ell)}\right)$ and explicit access to $i \in [n']$, the machine A makes at most $O(1)$ queries to each oracle and outputs the i-th coordinate of $C\left(m^{(1)} \circ \dots \circ m^{(\ell)}\right)$.*

The proof of Theorem 5.2 closely follows that of [RR20, Theorem 3.1]. Roughly speaking, [RR20, Theorem 3.1] states that NP relations that can be verified in polynomial time and small space have an IOP with communication complexity that is very close to the witness length. We fully state their man result next.

Theorem B.1. *([RR20, Theorem 3.1]) Let $\mathcal{L} \in$ NP with corresponding relation $\mathcal{R}_{\mathcal{L}}$ in which the instances have length m and witnesses have length n, where n and m are polynomially related and $m \geq n$, and such that $\mathcal{R}_{\mathcal{L}}$ can be decided in time $\mathrm{poly}(m)$ and space $s \geq \log m$. Let $\gamma = \gamma(m) \in (0,1)$ and $\beta = \beta(m) \in (0,1)$ be nice functions such that $\mathrm{poly}\left(\frac{1}{\beta}\right) \leq \log m$ and $\gamma \geq n^{-O(\beta)}$. Then there exists a $\beta^{-O\left(\frac{1}{\beta}\right)}$-round* IOP *for $\mathcal{L}$ with soundness error $\frac{1}{2}$. The query complexity is $(\gamma\beta)^{-O\left(\frac{1}{\beta}\right)}$ and the communication consists of a first (deterministic) message sent by the prover of length $(1+\gamma) \cdot m$ bits followed by $\mathrm{poly}\left(n^{\beta}, (\gamma\beta)^{-\frac{1}{\beta}}, s\right)$ additional communication. In addition, the* IOP *is prover-efficient and the verifier runs in $\tilde{O}(n) + \mathrm{poly}\left(n^{\beta}, (\gamma\beta)^{-\frac{1}{\beta}}, s\right)$.*

Remark B.3. *[RR20] actually handles some sub-constant values of β, γ. In our context, it is enough to handle constant values.*

Overview of proof of [RR20, Theorem 3.1]. The proof uses an IOPP to construct an IOP and then uses locally decomposable codes to optimize the communication complexity.

From IOPPs to IOPs. Let $\mathcal{L} \in \mathsf{NP}$ and let $\mathcal{R}_{\mathcal{L}}$ be the corresponding NP relation. Fixing an error-correcting code E, we define a language of pairs that consist of an encoded instance and a corresponding witness (that remains unchanged), that is:

$$\mathcal{L}' = \{(E(x), w) : (x, w) \in \mathcal{R}_{\mathcal{L}}\}.$$

Assuming there exists an IOPP for $\mathcal{L}'$ (note that the pair $(E(x), w)$ is an implicit input to the verifier and there is no explicit input), we can construct an IOP for $\mathcal{L}$ in the following manner: given $x \in \mathcal{L}$, the prover sends the witness w to the verifier who computes $E(x)$ on its own, then both parties run the IOPP to decide whether $(E(x), w)$ is in $\mathcal{L}'$. The key observation here is that if indeed $x \in \mathcal{L}$ then there exists a w that would make the IOPP verifier accept. However, if $x \notin \mathcal{L}$ and the code has a large enough distance, then for any w, the pair $(E(x), w)$ would be far from $\mathcal{L}'$ and the IOPP verifier would reject by the soundness of the IOPP. The communication of this IOP consists of a first prover message that contains w followed by the communication of the IOPP.

Optimizing the communication complexity To achieve an even shorter communication, we utilize the fact that the verifier has explicit access to a large portion of the input, namely $E(x)$ (recall that this is not the case for IOPP verifiers). Suppose that there exists a code C such that the first message that the IOPP prover sends is the non-systematic part of $C(E(x), w)$. If the code C is locally ℓ-decomposable w.r.t. some base code C_0 for $\ell = \left\lfloor \frac{n'+m}{m} \right\rfloor$. Then we can chop $(E(x), w)$ to ℓ chunks of length m: $y_1, \ldots, y_\ell$ where $y_\ell = w$ and simulate every query to $C(E(x), w)$ by making a constant number of queries to each of $C_0(y_i)$. Since the verifier has computed $E(x)$, then it can simulate the queries to $C_0(y_i)$ for all $i < \ell$ independently of the prover. This means that the prover only has to send $C_0(y_\ell) = C_0(w)$ as the first message, instead of $C(E(x), w)$ in its entirety. This is formally stated in the following lemma:

Lemma B.4. *([RR20, Lemma 9.1]) Let $\mathcal{L} \in \mathsf{NP}$ with corresponding relation $\mathcal{R}_{\mathcal{L}}$, in which the instances have length n and witness have length m where m and n are polynomially related and $m \leq n$. Let $E = \left\{E_n : \{0,1\}^n \to \{0,1\}^{n'}\right\}_{n \in \mathbb{N}}$ be a code ensemble with relative distance $\delta > 0$ and quasi-linear time encoding. Let $C_0 = \left\{C_0^{(n)} : \{0,1\}^n \to \{0,1\}^{n'}\right\}_{n \in \mathbb{N}}$ be a code ensemble with rate $1 - \eta$ and quasi-linear time encoding. Let $\ell = \left\lfloor \frac{n'+m}{m} \right\rfloor$ and suppose that $C = \left\{C^{(m \cdot \ell)} : \{0,1\}^{(m \cdot \ell)} \to \{0,1\}^*\right\}_{n \in \mathbb{N}}$ is a systematic locally ℓ-decomposable code w.r.t. base code C_0.*

Let $\mathcal{L}' = \{(E(x), w) : (x, w) \in \mathcal{R}_{\mathcal{L}}\}$ and suppose that $\mathcal{L}'$ has a q-query r-round IOPP w.r.t. proximity parameters $\frac{\delta}{2}$ and soundness error ε with the following properties:

- *The first message in the IOPP is the non-systematic part of $C(E(x), w)$.*
- *The rest of the communication in the IOPP is of length cc.*
- *The IOPP verifier runs n time T_V and the IOP prover runs in time T_P.*

Then $\mathcal{L}$ has an $O(q(n' + m))$-query $r(n' + m)$-round IOP with soundness error $\varepsilon(n' + m)$. The communication consists of a first message sent by the prover of length $m' = \frac{1}{1-\eta} \cdot m$ bits followed by $cc(n' + m)$ additional communication. The IOP verifier runs in time $\tilde{O}(n) + T_V(n' + m)$ and the IOP prover runs in time $\tilde{O}(n) + T_P(n' + m)$.

We now look at the particular IOPP used in the proof of [RR20, Theorem 3.1], which is presented in the following lemma:

Lemma B.5. *([RR20, Lemma 8.1]) Let $\mathcal{L}$ be a language computable in time $\mathrm{poly}(n)$ and space $s \geq \log n$. Then for any constants $\beta, \gamma \in (0, 1)$ the following holds. There exists an IOPP for $\mathcal{L}$ with respect to proximity parameter $\delta > 0$ with communication complexity $\gamma \cdot n + \mathrm{poly}\left(n^\beta, (\gamma\beta)^{-\frac{1}{\beta}}, \frac{1}{\delta}, s\right)$, query complexity $\mathrm{poly}\left((\gamma\beta)^{-\frac{1}{\beta}}, \frac{1}{\delta}\right)$, round complexity $\beta^{O\left(-\frac{1}{\beta}\right)}$ and soundness error $\frac{1}{2}$.*

Moreover,

- *The verifier's running time is* $\mathrm{poly}\left(n^\beta, (\gamma\beta)^{-\frac{1}{\beta}}, \frac{1}{\delta}, s\right)$.
- *The prover's running time is* $\mathrm{poly}(n)$.
- *There exists a systematic code* $C = \left\{C_n : \{0,1\}^n \to \{0,1\}^{n'}\right\}_{n \in \mathbb{N}}$ *of rate at least* $\frac{1}{1+\gamma}$ *such that the communication phases consists of a single prover's message of length* $\gamma \cdot n$ *which is the non-systematic part of* C_n *followed by a two-way communication of length* $\mathrm{poly}\left(n^\beta, (\gamma\beta)^{-\frac{1}{\beta}}, \frac{1}{\delta}, s\right)$. *Furthermore,* C_n *is locally* $O\left(n^{1-\beta}\right)$*-decomposable w.r.t. a base code* C_0 *of rate at least* $1-\gamma$ *that is encodable in quasi-linear time.*

B.2 Proof of Theorem 5.2

We now prove our two extensions over [RR20, Theorem 3.1].

B.2.1 Achieving compactness

Showing compactness is straightforward, we simply keep track of the size of the predicate produced by the IOP verifier. In particular, we note that the predicate produced by the IOP verifier in [RR20, Lemma 9.1] depends on the running time of the IOPP verifier which it simulates.

Proposition B.6. *The* IOP *from [RR20, Lemma 9.1] is* $\tilde{O}(n)+T_V(n'+m)$*-uniform* $T_V(n'+m)$*-compact.*

Proof. Recall that the IOP verifier computes $E(x)$ on its own and then runs the IOPP verifier on input $(E(x), w)$ and simply answers accordingly. This means that in the local computation phase, the IOP verifier produces the same predicate that is produced by the IOPP verifier. Therefore, the size of the circuit that evaluates the predicate is bounded from above by the running time of the IOPP verifier which is $T_V(n'+m)$. In addition, the running time of the IOP verifier is $\tilde{O}(n) + T_V(n'+m)$. The claim follows by definition Definition 2.5. ■

By choosing a sufficiently small β in [RR20, Lemma 8.1] and plugging it into [RR20, Lemma 9.1] with Proposition B.6, we get [RR20, Theorem 3.1] with the desired compactness property as stated in Theorem 5.2.

B.2.2 Handling smaller witness size

As for removing the polynomial relation assumption between the instance and witness lengths, we take a closer look at the communication of the IOP from [RR20, Lemma 9.1] and show how we can utilize the property of local decomposability even if the instance and witness lengths are not polynomially related. This costs us an additive factor of n^β in the communication complexity for some small constant β, but this is a price we are willing to pay in our context.

Proposition B.7. *For any instance size* n*, witness size* m *and constant* $\beta \in (0,1)$*, if the code* C *is locally* ℓ*-decomposable code w.r.t. base code* C_0 *where* $\ell = O\left(n^{1-\beta}\right)$*, then the first prover message in the* IOP *from [RR20, Lemma 9.1] can be modified to have length* $\frac{1}{1-\eta} \cdot m + \frac{\eta}{1-\eta} \cdot O\left(n^\beta\right)$*, where* $1-\eta$ *is the rate of* C_0*.*

Recall that optimizing the first prover message is done in [RR20, Lemma 9.1] by utilizing that the code C is locally $\left\lfloor \frac{n'+m}{m} \right\rfloor$-decomposability and taking advantage of the fact that there exists a constant $c > 1$ such that $n' = O(m^c)$, therefore by setting $\beta = \frac{1}{c}$ in [RR20, Lemma 8.1], the IOP prover can simply send $C_0(w)$ instead of $C(E(x), w)$. What we have here is slightly different, but the same technique can still be applied. Assuming that $n = \omega(m^c)$ for every constant $c > 1$ (and therefore $n' = \omega(m^c)$), it follows that $n'^\beta = \omega(m)$. Since C is locally ℓ-decomposable, then we can still split the string $(E(x), w)$ into $O\left(n^{1-\beta}\right)$ sub-strings of size $O\left(n^\beta\right)$ each and w would only be contained in the last sub-string $y \circ w$, where y is a suffix of x of length $O\left(n^\beta\right)$. The prover sends the encoding of that last sub-string, allowing the verifier to simulate each query to $C(E(x), w)$ by making $O(1)$ queries to the encoding $C_0(y \circ w)$. In fact, the prover only has to send the non-systematic part of $C_0(y \circ w)$ as well as w. The correctness of [RR20, Lemma 9.1] is preserved, and the only change is that the length of the first message is now $\frac{1}{1-\eta} \cdot m + \frac{\eta}{1-\eta} \cdot O\left(n^{\beta}\right)$. Proposition B.7 follows.

Combining Proposition B.6 with Proposition B.7 yields Theorem 5.2 which is, as mentioned above, a mild extension of [RR20, Theorem 3.1].