Electronic Colloquium on Computational Complexity, Report No. 22 (2018)

Efficient Batch Verification for UP

Omer Reingold[*] Stanford University

Guy N. Rothblum[†] Weizmann Institute

Ron D. Rothblum[‡] MIT and Northeastern University

February 1, 2018

Abstract

Consider a setting in which a prover wants to convince a verifier of the correctness of k NP statements. For example, the prover wants to convince the verifier that k given integers $N_1, \ldots, N_k$ are all RSA moduli (i.e., products of equal length primes). Clearly this problem can be solved by simply having the prover send the k NP witnesses, but this involves a lot of communication. Can interaction help? In particular, is it possible to construct *interactive* proofs for this task whose communication grows sub-linearly with k?

Our main result is such an interactive proof for verifying the correctness of any k UP statements (i.e., NP statements that have a unique witness). The proof-system uses only a constant number of rounds and the communication complexity is $k^{\delta} \cdot \mathsf{poly}(m)$, where $\delta > 0$ is an arbitrarily small constant, m is the length of a single witness, and the poly term refers to a fixed polynomial that only depends on the language and not on δ. The (honest) prover strategy can be implemented in polynomial-time given access to the k (unique) witnesses.

Our proof leverages "interactive witness verification" (IWV), a new type of proof-system that may be of independent interest. An IWV is a proof-system in which the verifier needs to verify the correctness of an NP statement using: (*i*) a sublinear number of queries to an alleged NP witness, and (*ii*) a short interaction with a powerful but untrusted prover. In contrast to the setting of PCPs and Interactive PCPs, here the verifier only has access to the *raw* NP witness, rather than some encoding thereof.

[*] Email: `reingold@stanford.edu`. Supported by NSF grant CCF-1749750.

[†] Email: `rothblum@alum.mit.edu`.

[‡] Email: `ronr@csail.mit.edu`. Research supported in part by NSF Grants CNS-1350619 and CNS-1414119, Alfred P. Sloan Research Fellowship, Microsoft Faculty Fellowship and in part by the Defense Advanced Research Projects Agency (DARPA), the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236 and by the Cybersecurity and Privacy Institute at Northeastern University.

ISSN 1433-8092

Contents

1 Introduction

The power of efficiently verifiable proof-systems is a central question in the study of computation. Interactive proofs, introduced in the seminal work of Goldwasser, Micali and Rackoff [GMR89], are interactive protocols between a randomized verifier and an untrusted prover. The prover convinces the verifier of the validity of a computational statement, usually framed as membership of an input x in a language $\mathcal{L}$. Soundness is unconditional. Namely, if the input is not in the language, then no matter what (unbounded and adaptive) strategy a cheating prover might employ, the verifier should reject with high probability over its own coin tosses. Interactive proofs have had a dramatic impact on complexity theory and on cryptography. Opening the door to randomized and interactive verification led to revolutionary notions of proof verification, such as zero knowledge interactive proofs [GMR89, GMW91] and probabilistically checkable proofs (PCPs) [BGKW88, FRS94, BFL91, BFLS91, FGL+96, AS92, ALM+98a]. Interactive proof-systems also allow for more efficient verification of larger classes of computations (compared with NP proof systems), as demonstrated in the celebrated IP = PSPACE Theorem [LFKN92, Sha92].

This work studies whether interactive proofs can allow for more efficient *batch verification* of NP statements. Namely:

Can an untrusted prover convince a verifier of the correctness of k NP *statements with communication complexity that is sublinear in k?*

Note that the naive solution is sending the k witnesses in their entirety. Looking ahead, our main result answers this question in the affirmative for a rich subclass of NP (the class UP of NP statements that have at most one witness). Along the way, we also introduce and study a new notion of proof-system: Interactive Witness Verification (IWV), which allow for the verification of NP statements using a sublinear number of queries to a "raw" (unencoded) NP witness and a short interaction with an untrusted prover. We construct IWVs for a rich subclass of NP, and these are a primary ingredient in our efficient batch verification protocol for UP.

Before proceeding to detail these contributions, we observe that the membership of k inputs in an NP language can be solved in space $O(\log k + m \cdot \mathsf{poly}(n))$, where n is the length of a single input and m is the length of a single NP witness. Thus, by the IP = PSPACE Theorem, there is an interactive proof for batch verification with communication complexity $\mathsf{poly}(\log k, n, m)$. A major caveat, however, is that the complexity of proving correctness (the running time of the *honest* prover) is *exponential in* $\mathsf{poly}(n, m)$. We, on the other hand, focus on batch verification where the honest prover runs in *polynomial time* given the k NP witnesses. We refer to such an interactive proof as having an *efficient prover*.[1]

1.1 Our Results

Our main result is an interactive proof-system, with an efficient prover, for verifying the correctness of k UP statements. Recall that UP refers to the subclass of NP statements for which correct

[1] Efficiency of the honest prover (given an NP witness) has been central in the study of zero-knowledge interactive proofs [GMR89, GMW91]. Our emphasis on an efficient honest prover is also inspired by the recent line of work on *doubly-efficient interactive proofs* [GKR08]. That line of work focuses on proofs for deterministic polynomial-time computations and the prover is required to run in polynomial-time without any auxiliary input. We remark that doubly efficient interactive proofs for deterministic computations do not seem to imply protocols for non-deterministic computations such as those we consider here.

statements have a *unique* witness. The canonical example (of a promise problem) in this class is unique-SAT in which one needs to distinguish unsatisfiable formulas from those having a unique satisfying assignment. Multiple other examples arise from cryptography, where problems related to factoring, discrete-log or lattices all have unique solutions.

Our protocol for UP batch verification uses a constant number of rounds and has communication complexity $(k^\delta \cdot \mathsf{poly}(m))$, for any arbitrarily small constant $\delta > 0$. For UP relations that are checkable in polynomial-time and bounded polynomial space, we can reduce the communication complexity to $(k^\delta \cdot m^{1+\delta})$. When the number of instances k is large, this is a significant improvement over the trivial solution in which the prover sends over all k witnesses.

Theorem 1 (Informally Stated, see Theorem 4.1)**.** *Let $\mathcal{L}$ be a language in* UP *with witnesses of length m. For every $\delta > 0$, there exists a constant-round interactive proof for verifying that k instances $x_1, \ldots, x_k$, each of length n, all belong to $\mathcal{L}$. The communication complexity is $k^\delta \cdot \mathsf{poly}(m)$. The verifier runs in time $\tilde{O}(k \cdot n) + k^\delta \cdot \mathsf{poly}(m)$, where n is the length of each of the instances. The honest prover runs in time $\mathsf{poly}(k, n, m)$ given the k unique witnesses.*

Comparison to prior work. Theorem 1 improves over the aforementioned protocol derived from IP = PSPACE theorem in two ways: (1) it has an efficient prover strategy (given the witnesses), and (2) it uses only a constant number of rounds of interaction (whereas the IP = PSPACE theorem uses $\mathsf{poly}(\log k, n, m)$ rounds).

Theorem 1 also improves over a prior result for batch verification of UP statements [RRR16]. The communication complexity of that protocol has an additional additive $k \cdot \mathsf{polylog}(m)$ term. In particular, even when k is larger than m, the [RRR16] protocol gives at most a quadratic saving over just naively sending all k witnesses. In contrast, our protocol yields an arbitrarily large polynomial saving in the parameter k.[2] A comparison of our techniques with those of [RRR16] is provided in Section 1.2.1.

A new type of proof-system. The protocol of Theorem 1 makes extensive use of a new type of proof-system that we introduce and construct. In a nutshell, these are proof-systems in which the verifier needs to check the correctness of an NP statement given oracle access to the NP witness. In contrast to PCPs, the verifier is only given access to the *raw* (i.e., original) NP witness, rather than to an encoding thereof. We allow the verifier to have a short interaction with an all powerful, but untrusted prover (this part of the interaction is similar to the interactive PCPs of Kalai and Raz [KR09]). We use the name "interactive witness verification" (IWV) for these proof-systems.

Jumping ahead, we remark that IWVs are closely related to *interactive proofs of proximity* (IPPs) [EKR04, RVW13]. Indeed, we show that IPPs for a class of deterministic polynomial-time computations directly imply IWVs for a related class of NP relations. The IWV protocol used in the proof of Theorem 1 is derived from known results on IPPs [RVW13]. We proceed to elaborate on the notion of IWVs.

[2] We remark that a linear dependence of the communication complexity on m is inherent. Indeed, by results of Goldreich *et al.* [GH98, GVW02], under complexity theoretic assumptions, even verification of a single instance (i.e., $k = 1$) requires $\Omega(m)$ communication.

1.1.1 Interactive Witness Verification

Motivation: sublinear witness verification. Suppose Alice wants to test whether a given graph G is 3-colorable. The validity of an alleged 3-coloring χ can be verified in polynomial time, but this requires reading every bit of the coloring. Can Alice verify χ's validity while reading a *sublinear* number of bits from χ?

At first glance, it may seem that PCPs give a direct solution to this problem. Recall that a PCP is an encoding of an NP witness that can be verified by reading only a very small number of bits. The celebrated PCP theorem [ALM$^+$98b] shows that every NP language has a PCP proof-system. However, the reason that PCPs do *not* solve Alice's problem is that she only has oracle access to the *original* 3-coloring χ of the graph. In contrast, in the PCP setting, the verifier is given oracle access to an *encoding* of the witness (e.g., via an error correcting code).

Indeed, sublinear witness verification for general NP relations is not possible (assuming that $\mathsf{P} \neq \mathsf{NP}$). Consider an NP relation for satisfiability where the witness is an encoding of the m-bit satisfying assignment w on a polynomial of high degree (say degree $10m$). The polynomial is given by a list of valuations on field elements and the satisfying assignment can be recovered by interpolating the polynomial. There are many possible polynomials that encode a given satisfying assignment. In particular, if Alice is given a *random* polynomial encoding the assignment, she must read $\Omega(m)$ valuations before she learns anything about the alleged satisfying assignment.

Adding interaction. While sublinear witness verification for general NP relations is not possible, this situation changes when we also allow *interaction*. Namely, we allow Alice to interact with an untrusted prover Bob who knows the entire 3-coloring of the graph. The communication should also be sublinear in the witness size (in particular, Bob cannot simply send χ to Alice). Given the 3-coloring, an honest Bob should run in polynomial time.

More generally, an *interactive witness verification* for a given NP relation R for a language $\mathcal{L}$ is a protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$, who both get as input an instance x. In addition, the verifier has query access to an alleged witness w for x and the prover is given full explicit access to w. The prover wants to convince the verifier that indeed $(x, w) \in R$. Towards this end, the prover and verifier run an interactive protocol. The verifier $\mathcal{V}$, on examination of the communication transcript with $\mathcal{P}$ and the queried points in w, accepts or rejects the prover's claim. For completeness, if $(x, w) \in R$ then the verifier $\mathcal{V}$, when interacting with the honest prover $\mathcal{P}$, should accept. For soundness, we emphasize that *the witness w is fixed before the protocol begins.* I.e., the soundness property is that if $x \notin \mathcal{L}$, then for any *fixed* false witness w^*, and for any (unbounded) cheating prover strategy $\mathcal{P}^*$, if we run the interactive protocol between $\mathcal{V}$ and $\mathcal{P}^*$, where $\mathcal{V}$'s witness-queries are answered using the fixed (false) witness w^*, then $\mathcal{V}$ will reject w.h.p. over its coins tosses. In terms of complexity, our goal is to have both the number of witness-queries and the communication be significantly smaller than $m = |w|$.

Note that an IWV refers to a specific NP *relation*, and not only to the underlying NP language (which can have many possible NP relations). Indeed, every NP language has *some* witness relation with an extremely efficient IWV.[3] The point, however, is that we would like to construct IWVs for arbitrary NP relations, not just highly structured ones.

In particular, an IWV can be particularly useful in situations where the NP witness is outside the prover's control. For example, the witness could arise from nature in the form of a physical or

[3] Consider the relation in which witnesses are PCP proof strings. For such relations even the interaction with the prover is unnecessary.

biological observation, in which case it is not reasonable to assume that it is given in an encoded format. Thus, we find IWVs to be natural objects worthy of further study (beyond their importance as a technical tool in our protocol for UP batch verification).

Constructions. We construct IWVs for a large class of NP relations. Namely, any NP relation R that is checkable by bounded-space Turing machines or bounded-depth (logspace uniform) circuits. This includes, for example, the natural NP relations for 3-coloring, satisfiability, k-clique, etc. In the protocol the verifier only reads $\sqrt{m}$ bits of the witness and the communication complexity is also roughly $\sqrt{m}$ (recall that m denotes the witness length). Thus, the verifier only observes $\sqrt{m}$ bits of information about the witness. More precisely, we obtain the following result, which allows for a tradeoff between the query and communication complexities:

Theorem 2 (Informally Stated, see Theorem 3.3)**.** *Let q be a parameter. Let R be an* NP *relation with witness length m that is checkable either by a bounded space Turing Machine or by a bounded depth (logspace uniform) circuit. For every constant $\delta > 0$, and $q \in [m]$, there exists an* IWV *for R in which the verifier reads q bits of the witness and the communication complexity is $O(m^{1+\delta}/q)$. Furthermore, the prover, given the* NP *witness, can be implemented efficiently (i.e., in polynomial time). The verifier runs in time $\tilde{O}(n + q + (m^{1+\delta}/q))$. For bounded space relations the number of rounds is constant, and for bounded depth relations it is* $\mathsf{polylog}(n)$*.*

The proof of Theorem 2 relies on known constructions of *interactive proofs of proximity* (IPPs). Loosely speaking, IPPs are interactive proofs in which the verifier runs in time that is sub-linear in the input length and is convinced that the input is *close* to the language (see Section 1.3 for additional details and related works on IPPs). Specifically, Theorem 2 follows from an IPP construction of Rothblum, Vadhan and Wigderson [RVW13] (together with an extension due to [RRR16]).

In particular, IWVs are closely related to IPPs. To see this, observe that an IWV for a relation R may be thought of as an IPP, where the IPP input is the IWV witness w, and the goal is to check whether the witness is "close" to the language $\mathcal{L}_x = \{w' : (x, w') \in R\}$. Even though IWVs do not refer to the proximity of the witness to $\mathcal{L}_x$, observe that if x is not in the language, then $\mathcal{L}_x = \emptyset$, and so any fixed w will have "infinite" distance from $\mathcal{L}_x$. In general, IWVs seem to be a more relaxed object than IPPs (in particular, the above reduction gives instances with infinite distance).

Lower Bound for IWVs. We also give a lower bound for IWVs that shows that Theorem 2 is quantitatively almost tight. Moreover, the specific NP relation for which we demonstrate this lower bound is in the class of relations for which we assume an IWV in the proof of Theorem 1. For our lower bound to hold we assume the existence of an exponentially strong cryptographic pseudorandom generator. The proof of the lower bound follows from a similar lower bound of Kalai and Rothblum [KR15] for IPPs, but is somewhat simpler.

1.2 Technical Overview

Let $\mathcal{L}$ be a UP language and let R be the corresponding UP relation. By the proof of the Cook-Levin theorem we may assume without loss of generality that R is computable in NC^1 (more specifically, by a CNF formula).[4] Recall that our goal is to design a protocol, between a prover $\mathcal{P}_{\mathsf{batch}}$ and

[4] This step incurs a polynomial blowup in the witness size, which is the source of the $\mathsf{poly}(m)$ dependence in Theorem 1. This blowup can be avoided for many natural UP relations which are natively checkable in NC^1 and more

verifier $\mathcal{V}_{\mathsf{batch}}$ that both get as input k instances $x_1, \dots, x_k$. The prover, in addition, also gets witnesses $w_1, \dots, w_k$, each of length m, and needs to convince $\mathcal{V}$ that $x_1, \dots, x_k \in \mathcal{L}$.

For sake of simplicity, we will focus on constructing a protocol with small communication $O(k^\delta \cdot m^{1+\delta})$ for some small constant $\delta > 0$, and ignore the running time of the verifier. Obtaining a verifier that runs in time that is sub-linear in k amounts to maintaining concise descriptions of all the objects involved in the interaction. We ignore the verification time for this overview.[5]

A Warmup: Batch Verification with $\sqrt{k} \cdot m^{1+\delta}$ Communication. As a warmup, we first consider a quantitatively easier task: batch verification with communication complexity $\sqrt{k} \cdot m^{1+\delta}$ (rather then our eventual goal of $k^\delta \cdot m^{1+\delta}$). This task is already non-trivial and demonstrates most of our key ideas.

Consider an augmented UP relation $R^{\otimes k}$ defined as:

$$R^{\otimes k} = \Big\{ \big((x_1, \dots, x_k), (w_1, \dots, w_k) \big) \ : \ \forall j \in [k], \ (x_j, w_j) \in R \Big\}.$$

Note that $R^{\otimes k}$ is computable in NC^1. By Theorem 2, $R^{\otimes k}$ has an efficient IWV protocol. Setting the parameter q of Theorem 2 to $\sqrt{k}$, we obtain an IWV protocol for $R^{\otimes k}$ in which the verifier reads $\sqrt{k}$ bits of the witness $\mathbf{w} = (w_1, \dots, w_k)$ and with communication roughly $\sqrt{k} \cdot m^{1+\delta}$ (where recall that $m = |w_1| = ... = |w_k|$ is the length of a single witness).

We would like for $\mathcal{P}_{\mathsf{batch}}$ and $\mathcal{V}_{\mathsf{batch}}$ to run this IWV. An immediate objection that should arise is that in contrast to the IWV setting, we are now trying to construct a standard interactive proof and so the verifier does not have access to the witness string $\mathbf{w} = (w_1, \dots, w_k)$. To get around this, we will leverage a property of the IWV of Theorem 2. Specifically, that IWV operates in two phases: an online phase followed by an offline phase. First, in the online phase, the verifier does *not* have oracle access to the witness w, but is allowed to communicate with the prover. The result of this interaction is a set of coordinates Q of w that the verifier would like to read and a predicate ϕ to be applied to w_Q. In the second (offline) phase, the verifier is given oracle access to w but is no longer allowed to interact with the prover. Rather, the verifier just reads w_Q and accepts if and only if $\phi(w_Q) = 1$. The soundness condition remains unchanged. Namely, for any alleged witness w for a false statement, which is fixed prior to the interaction, with high probability over the coins tossed in the online phase, either the verifier rejects or it generates Q and ϕ such that $\phi(w_Q) = 0$.

The batch verification prover $\mathcal{P}_{\mathsf{batch}}$ and verifier $\mathcal{V}_{\mathsf{batch}}$ start by running the online phase of the IWV for $R^{\otimes k}$ on input $(x_1, \dots, x_k)$. The prover $\mathcal{P}_{\mathsf{batch}}$ also uses $\mathbf{w} = (w_1, \dots, w_k)$ as its witness string. Observe that for this online phase the verifier $\mathcal{V}_{\mathsf{batch}}$ does not need oracle access to $\mathbf{w}$ (indeed, as discussed above, $\mathcal{V}_{\mathsf{batch}}$ has no such oracle access).

At the end of the this phase, $\mathcal{V}_{\mathsf{batch}}$ obtains a set $Q \subseteq [k] \times [m]$, of size $|Q| \leq \sqrt{k}$, of points that it would like to read from $\mathbf{w} = (w_1, \dots, w_k)$ together with a predicate $\phi : \{0,1\}^{|Q|} \to \{0,1\}$ to be evaluated on $\mathbf{w}_Q$. However, since $\mathcal{V}_{\mathsf{batch}}$ does not have oracle access to $\mathbf{w}$, it is not immediately clear how we can leverage the soundness condition of IWVs in our current setting. Jumping ahead, we shall do so by using the uniqueness of the witnesses in a crucial way.

generally, for relations that are checkable by bounded space Turing machines or bounded depth (logspace uniform) circuits.

[5] One way getting an efficient verifier is to first construct a protocol with small communication but large verification time (as will be described in this overview), and then further delegate the verification task (which is a deterministic computation applied to the transcript of the interaction and the input) to the prover using a doubly-efficient interactive proof such as those of [GKR08] or [RRR16]. However, in our actual construction we show that the verifier can be implemented efficiently directly, without this additional trick.

Specifically, consider the fixed witness string $\hat{\mathbf{w}} = (\hat{w}_1, \ldots, \hat{w}_1)$, where for every $j \in [k]$: if $x_j \in \mathcal{L}$, then we set $\hat{w}_j$ to be the corresponding *unique* witness, and otherwise (in case $x_j \notin \mathcal{L}$) we set $\hat{w}_j$ as an arbitrary fixed string (e.g., 0^m). Since $\hat{\mathbf{w}}$ is an a priori fixed string, by the soundness of the IWV, if there exists some j^* such that $x_{j^*} \notin \mathcal{L}$, then, no matter what a cheating prover does, with high probability the IWV verifier generates ϕ and Q such that $\phi(\hat{\mathbf{w}}_Q) = 0$ (or it rejects, which case $\mathcal{V}_{\mathsf{batch}}$ also rejects).

Suppose that indeed $\phi(\hat{\mathbf{w}}_Q) = 0$. While our verifier $\mathcal{V}_{\mathsf{batch}}$ does not have oracle access to $\hat{\mathbf{w}}$, since $|Q| \leq \sqrt{k}$, we can ask the prover $\mathcal{P}_{\mathsf{batch}}$ to send all of the witnesses belonging to the subset $S \subseteq [k]$ of witnesses that are "touched" by Q (i.e., S is the projection of the set $Q \subseteq [k] \times [m]$ to its first coordinate). Denote the set of alleged witnesses that the prover sends by $\tilde{\mathbf{w}} = (\tilde{w}_j)_{j \in S}$. Sending $\tilde{\mathbf{w}}$ only costs us an additional $|S| \cdot m = O(\sqrt{k} \cdot m)$ communication. The verifier $\mathcal{V}_{\mathsf{batch}}$ checks that $(x_j, \tilde{w}_j) \in R$, for all $j \in S$, and that $\phi(\tilde{\mathbf{w}}) = 1$. To argue that soundness holds, observe that if for some $j \in S$ it holds that $x_j \notin \mathcal{L}$, then for any $\tilde{w}_j$ that a potential cheating prover $\mathcal{P}^*_{\mathsf{batch}}$ might send, it holds that $(x_j, \tilde{w}_j) \notin \mathcal{R}$ and so $\mathcal{V}_{\mathsf{batch}}$ rejects. On the other hand, since $\mathcal{L} \in \mathsf{UP}$, if $x_j \in \mathcal{L}$ for all $j \in S$ (which can certainly happen), the prover $\mathcal{P}^*_{\mathsf{batch}}$ has to send the *unique* witnesses $\tilde{\mathbf{w}} = \hat{\mathbf{w}}_S$ (in order for these witnesses to satisfy the relation R) in which case $\mathcal{V}_{\mathsf{batch}}$ rejects when checking that $\phi(\tilde{\mathbf{w}}) = \phi(\hat{\mathbf{w}}_S) = 1$. Thus, in any case $\mathcal{V}_{\mathsf{batch}}$ rejects and soundness follows.

Taking a step back, our proof of soundness strongly leverages the uniqueness of the witnesses to emulate query access to an a priori fixed witness by having the prover send the relevant parts of the witness a posteriori (i.e., after the interactive phase of the IWV).

Observe that by our setting of parameters, the IWV part of the protocol uses $\sqrt{k} \cdot m^{1+\delta}$ communication, and actually sending the witnesses $(\hat{w}_j)_{j \in S}$ adds only an additional $\sqrt{k} \cdot m$ communication. Overall we obtain communication complexity $\sqrt{k} \cdot m^{1+\delta}$.

Batch Verification with $k^\delta \cdot m$ Communication. The main idea for obtaining smaller communication $k^\delta \cdot m^{1+\delta}$, where $\delta > 0$ is an arbitrary small constant, is to recursively apply the solution for the warmup case (with slightly different parameters). We describe our approach in detail next.

First, in contrast to the warmup case, we use the IWV of Theorem 2 with respect to parameter $q = k^{1-\delta}$ (rather than $q = \sqrt{k}$). Thus, we have an IWV for the relation R_k with communication $k^\delta \cdot m^{1+\delta}$ and query complexity $k^{1-\delta}$.

The prover $\mathcal{P}_{\mathsf{batch}}$ and $\mathcal{V}_{\mathsf{batch}}$ run the IWV as in the warmup. The main difference is that now the set S of instances that are queried in the IWV is of size $k^{1-\delta}$ and we cannot afford for $\mathcal{P}_{\mathsf{batch}}$ to send $(w_j)_{j \in S}$ explicitly to $\mathcal{V}_{\mathsf{batch}}$. Rather, we observe that after running the (online part of the) IWV what remains to be checked is that for every $j \in S$ it holds that $x_j \in \mathcal{L}$ *and* that the corresponding (unique) witnesses $(w_j)_{j \in S}$ satisfy $\phi((w_j)_{j \in S})$.

Our approach is to check that these conditions hold by a recursive application of our batch verification protocol on $(x_j)_{j \in S}$. Observe that the number of instances has shrunk from k to $k^{1-\delta}$ so we have made significant progress. Actually, batch verification per se does not suffice since it only guarantees that $x_j \in \mathcal{L}$ for every $j \in S$ but does not guarantee that $\phi((w_j)_{j \in S})$. Still, we can obtain also the latter condition by using the fact that ϕ is computable in NC_1 and therefore can be incorporated into the augmented relation which is defined for the next round.

Thus, in each iteration of the recursion we shrink the number of instances by a k^δ factor. After $\ell = O(1/\delta)$ iterations we will be left with a constant number of instances and the prover $\mathcal{P}^{\mathsf{batch}}$ can send the corresponding witnesses explicitly. As in the warmup case, the verifier checks that all these witnesses satisfy the base relation R individually and that they jointly satisfy the predicate

ϕ_ℓ generated by the last iteration of the recursion.

By our setting of parameter for the IWV, the communication complexity in each one of the iterations is $k^\delta \cdot m^{1+\delta}$. At the final step the verifier sends $O(1)$ witnesses in the clear so that also adds at most $O(m)$ communication. The prover can be implemented efficiently (given the UP witnesses) since the underlying IWV has an efficient prover strategy. As described above, the verifier's running time might be as high as $\Omega(k \cdot m)$ (naively, that will be complexity of the first iteration) but in the technical sections we provide a more refined analysis that shows how to implement the verifier more efficiently.

1.2.1 Technical Comparison with [RRR16]

As noted above, a less efficient UP batch verification protocol was presented in [RRR16]. We briefly review that approach and the differences from the current work. We refer the reader to [RRR16] and to Goldreich's recent survey [Gol17] for more details on the [RRR16] protocol.

The high level idea in the [RRR16] batch verification protocol is for the prover to generate PCP proof strings for all the k instances and send a short "checksum" of these PCPs. The verifier in turn, generates PCP queries and asks the prover to reveal the answer to these queries for all k PCPs. The checksum is designed to guarantee that a cheating prover must either answer these queries in a way that is consistent with predetermined PCPs, or alternatively, it can send answers that are inconsistent with the correct PCPs of *many* of the statements. For UP the correct PCPs are unique, and so the latter situation can be checked directly by having the verifier specify a random subset of the PCP proofs for the prover to fully reveal (also here, similarly to our protocol, recursion can be used to obtain improved parameters).

The current work completely avoids the use of checksums and PCPs. Instead, we use IWVs to force a cheating prover to make false claim about a subset of the witnesses. IWVs, together with the uniqueness of the witnesses, let us accomplish this without requiring the verifier to make explicit queries to the witnesses. In terms of complexity, the key difference is that in the [RRR16] protocol, the prover must reveal the PCP answers for all the k instances. This step adds $\Omega(k)$ to the communication complexity, which we avoid.

1.3 Additional Related Works

Doubly Efficient Interactive Proofs. Goldwasser, Kalai and Rothblum [GKR08] introduced a variant of interactive proofs, called *doubly-efficient* interactive proofs, in which both the prover and the verifier are highly efficient. Namely, the honest prover must run in time that is proportional to the complexity of the statement being proved, whereas the verifier should be much faster than the complexity of the computation. Soundness is required to hold against computationally unbounded provers.

Goldwasser *et al.* [GKR08] construct such doubly efficient interactive proofs for every language in (logspace-uniform) NC. The aforementioned recent work of Reingold *et al.* [RRR16] gives such proof-systems for languages computable in polynomial-time and some bounded polynomial space. This includes in particular the complexity class SC. Moreover the latter protocol only requires a constant number of rounds of interaction. We note that batch verification of certain types of interactive proofs (that generalize UP batch verification) was a key ingredient in the [RRR16] result.

We remark that [GKR08] and [RRR16] doubly-efficient interactive proof-systems are for *deterministic* computations and do not seem to immediately imply a protocol for batch NP, or even UP, verification.

Batch Verification with Computational Soundness. A recent work of Brakerski, Holmgren and Kalai [BHK17] shows an *argument-system* for batch verification of NP statements. We emphasize that they only obtain soundness against polynomial-time cheating provers whereas we achieve statistical soundness against computationally unbounded provers. In addition, the result of [BHK17] relies on an unproven cryptographic assumption (specifically a computational private information retrieval scheme) whereas our result is unconditional. On the other hand, the protocol of [BHK17] also offers significant advantages (some of which are likely to be infeasible in the context of interactive proofs with statistical soundness). First, their result holds for any NP language (rather than just UP). Second, their protocol requires only 2 messages whereas our protocol requires a larger (but still constant) number of rounds. Lastly, under strong enough assumptions, [BHK17] achieve communication that depends only poly-logarithmically on k, whereas our dependence is any arbitrarily small polynomial.

Interactive PCPs and PCIPs. Interactive PCPs, introduced by Kalai and Raz [KR09] are a generalization of PCPs in which the PCP verifier can, in addition to reading bits of the PCP, also interact with a prover. Our notion of IWV can be thought of as a restriction of Interactive PCPs in which the PCP proof string is exactly the NP witness and cannot be further encoded.

A generalization of interactive PCPs, called probabilistically checkable interactive proofs (PCIPs) was recently introduced by Reingold *et al.* [RRR16] and Ben Sasson *et al.* [BCS16].[6] Loosely speaking, these are interactive proofs in which the verifier only reads few bits of each message. Again, and in contrast to IWVs, the prover is allowed to send long messages of its choice (e.g., a PCP encoding of the NP witness).

Interactive Proofs of Proximity (IPPs). As mentioned above, interactive proofs of proximity (IPPs) form an important component in our UP batch verification protocol. IPPs were introduced in [EKR04, RVW13] and have seen a considerable amount of progress in recent years. The latter work gives a general purpose sub-linear IPP protocol for general bounded depth computations (which was extended to bounded space computations in [RRR16]). A non-interactive variant of IPPs was studied in [GR15, FGL14]. Highly efficient protocols for restricted classes of computations (i.e., context free languages and small read-once branching programs) were given in [GGR15]. A study of a computational variant of IPPs was initiated in [KR15]. The latter work also shows a lower bound for IPPs. As mentioned above, we use their proof technique to derive a similar lower bound for IWVs. Gur and Rothblum [GR17] show a round hierarchy for IPPs. Most recently, IPPs were considered in the context of zero-knowledge [BRV18] and distribution testing [CG18].

1.4 Open Questions

We conclude the introduction by mentioning two open questions on batch verification:

1. Do there exist interactive proofs for batch verification of arbitrary NP statements (rather than just UP statements)? Ideally such a proof-system would be both constant-round and

[6] Ben Sasson *et al.* refer to these as Interactive Oracle Proofs.

have an efficient prover strategy, but even getting a result satisfying only one of these two requirements would be very interesting. We mention that we do not know a way to extend our UP batch verification to NP via the Valiant-Vazirani randomized reduction from NP to UP [VV86].

2. The communication complexity in our protocol is proportional to k^{δ}. Is it possible to obtain a similar result with communication that grows only poly logarithmically with k? (Such a result would likely require a super constant number of rounds.) In particular, a quantitative improvement to the interactive proof of proximity of [RVW13] could yield such a result.[7]

1.5 Organization

Section 2 contains definitions and notations. In Section 3 we formally define our notion of interactive witness verification IWV, show that they exist for bounded NP relations and demonstrate a lower bound on their complexity. In Section 4 we present our protocol for batch verification of UP statements.

2 Preliminaries

Throughout this work we use NC^1 to refer to the class of logspace uniform Boolean circuits of logarithmic depth and constant fan-in. Namely, $\mathcal{L} \in \mathsf{NC}^1$ if there exists a logspace Turing machine M that on input 1^n outputs a full description of a logarithmic depth circuit $C : \{0,1\}^n \to \{0,1\}$ such that for every $x \in \{0,1\}^n$ it holds that $C(x) = 1$ if and only if $x \in \mathcal{L}$. We recall that the class SC refers to languages decidable by Turing machines that run in polynomial-time and poly-logarithmic space.

We next define a notion of succinct representation of circuits. Loosely speaking, a function $f : \{0,1\}^n \to \{0,1\}$ has a succinct representation if there is a short string $\langle f \rangle$, of poly-logarithmic length, that describes f. That is, $\langle f \rangle$ can be expanded to a full description of f. The actual technical definition is slightly more involved and in particular requires that the full description of f be an NC_1 (i.e., logarithmic depth) circuit:

Definition 2.1 (Succinct Description of Functions)**.** *We say that a function $f : \{0,1\}^n \to \{0,1\}$ of size s has a* succinct description *if there exists a string $\langle f \rangle$ of length $\mathsf{polylog}(n)$ and a logspace Turing machine M (of constant size, independent of n) such that on input 1^n, the machine M outputs a full description of an NC^1 circuit C such that for every $x \in \{0,1\}^n$ it holds that $C(\langle f \rangle, x) = f(x)$. We refer to $\langle f \rangle$ as the succinct description of f.*

We also define succinct representation for sets $S \subseteq [k]$. Roughly speaking this means that the set can be described by a string of length $\mathsf{polylog}(k)$. The formal definition is somewhat more involved:

Definition 2.2 (Succinct Description of Sets)**.** *We say that a set $S \subseteq [k]$ of size s has a succinct description if there exists a string $\langle S \rangle$ of length $\mathsf{polylog}(k)$ and a logspace Turing machine M such that on input 1^k, the machine M outputs a full description of a depth $\mathsf{polylog}(k)$ and size*

[7] By slightly adjusting our parameters we could obtain a batch verification protocol with communication complexity $k^{o(1)} \cdot m^{1+o(1)}$ (and a super constant number of rounds of interaction). Still, achieving communication $\mathsf{polylog}(k) \cdot m$ seems beyond our current techniques.

$\mathsf{poly}(s, \log k)$ *circuit (of constant fan-in) that on input* $\langle S \rangle$ *outputs all the elements of* S *as a list (of length* $s \cdot \log(k)$*).*

We emphasize that the size of the circuit that M outputs is proportional to the actual size of the set S, rather than the universe size k.

2.1 Interactive Proofs

An interactive proof-system, as defined by Goldwasser, Micali and Rackoff [GMR89], is a protocol between a polynomial-time verifier $\mathcal{V}$ and a computationally unbounded prover $\mathcal{P}$. The two parties interact and at the end of the interaction the verifier accepts if and only if the given computational statement is correct (with high probability).

We denote by $(\mathcal{P}, \mathcal{V})(x)$ the output of $\mathcal{V}$ after interacting with $\mathcal{P}$ on common input x. If either $\mathcal{V}$ or $\mathcal{P}$ are given additional explicit inputs than we shall denote this by $(\mathcal{P}(y), \mathcal{V}(z))(x)$ which refers to the output of $\mathcal{V}$ after interacting with $\mathcal{P}$ where $\mathcal{V}$ gets as input (x, z) and $\mathcal{P}$ gets as input (x, y). We extend the foregoing notation to implicit access to the input by placing the implicit input as a superscript. Thus, by $(\mathcal{P}, \mathcal{V}^z)(x)$ we refer to the output of $\mathcal{V}$ after interacting with $\mathcal{P}$, where $\mathcal{V}$ has oracle access to z and both parties have explicit access to x.

Definition 2.3. *An* interactive proof *for a language* $\mathcal{L}$ *is an interactive protocol between a polynomial-time verifier* $\mathcal{V}$ *and a computationally unbounded prover* $\mathcal{P}$*. Both parties are given as input a string* x *and must satisfy the following two properties:*

- **Completeness:** *If* $x \in \mathcal{L}$*, then*
$$\Pr[(\mathcal{P}, \mathcal{V})(x) = 1] = 1.$$
- **Soundness:** *If* $x \notin \mathcal{L}$*, then for every possible cheating strategy* $\mathcal{P}^*$*,*
$$\Pr[(\mathcal{P}^*, \mathcal{V})(x) = 1] \leq 1/2.$$

If $\mathcal{L} \in \mathsf{NP}$, we say that an interactive proof for $\mathcal{L}$ has an *efficient prover* if the honest prover strategy $\mathcal{P}$ can be implemented an polynomial-time given access to an NP witness.

Remark 2.4 (On Completeness and Soundness Errors)**.** *We note that Definition 2.3 can be generalized to allow for an error in the completeness condition. For simplicity however, and since our protocols achieve perfect completeness, we avoid doing so.*[8] *We also remark that the soundness error (and completeness error, if ones allows for such) can be reduced at an exponential rate by either sequential or parallel repetition (see, e.g., [Gol99, Lemma C.1]).*

2.1.1 Doubly Efficient Interactive Proofs

Doubly-efficient interactive proofs, introduced by Goldwasser *et al.* [GKR08] are interactive proofs in which the prover is relatively efficient (i.e., runs in time proportional to the complexity of the computation), whereas the verifier is extremely efficient (i.e., running in almost linear time). We shall use a recent result of Reingold *et al.* [RRR16] giving such doubly efficient interactive proofs for bounded space computations. The reason that we use the more recent protocol of [RRR16] rather than that of [GKR08], is mainly because the former only requires a constant number of rounds.

[8] Fürer *et al.* [FGM$^+$89] show how to transform any interactive proof to one having perfect completeness. Their transformation however does not preserve the efficiency of the prover.

Theorem 2.5 (Interactive Proofs for Bounded Space ([RRR16]))**.** *Let $T = T(n)$ and $S = S(n)$ such that $n \leq T \leq \mathsf{exp}(n)$ and $\log(T) \leq S \leq \mathsf{poly}(n)$.*

Let $\mathcal{L} \in \mathsf{DTISP}(T, S)$ and let $\tau = \tau(n) \in (0, 1/2)$ such that $\mathsf{poly}(1/\tau) \leq \log(T)$. Then, $\mathcal{L}$ has a public-coin interactive proof with perfect completeness and soundness error $\frac{1}{2}$. The number of rounds is $(1/\tau)^{O(1/\tau)}$. The communication complexity is $T^{O(\tau)} \cdot \mathsf{poly}(S)$. The (prescribed) prover runs in time $T^{1+O(\tau)} \cdot \mathsf{poly}(S)$ time, and the verifier runs in time $\big(n \cdot \mathsf{polylog}(T) + T^{O(\tau)} \cdot \mathsf{poly}(S)\big)$.

3 Interactive Witness Verification

For an NP relation R, we denote by $R(x)$ the set of witnesses for x, namely $R(x) = \{w : R(x, w) = 1\}$.

Definition 3.1 (Interactive Witness Verification IWV)**.** *An* Interactive Witness Verification Protocol *(*IWV*) for an* NP *relation R, is an interactive protocol between a computationally unbounded prover $\mathcal{P}$ and a verifier $\mathcal{V}$ on a given input x. The verifier also has* oracle *access to an alleged witness w. The protocol must satisfy the following two requirements:*

- **Completeness:** *If $(x, w) \in R$ then*
$$\Pr\Big[\big(\mathcal{P}, \mathcal{V}^{w}\big)(x, |w|) = 1\Big] = 1.$$

- **Soundness:** *If $R(x) = \emptyset$ (i.e., x is not in the underlying* NP *language), then for every a priori fixed w^* and every prover strategy $\mathcal{P}^*$:*
$$\Pr\Big[\big(\mathcal{P}^*, \mathcal{V}^{w^*}\big)(x, |w^*|) = 1\Big] \leq 1/2.$$

The query complexity *$q = q(|x|, |w|)$ is the number of bits that $\mathcal{V}$ reads from w and the* communication complexity *$\mathsf{cc} = \mathsf{cc}(|x|, |w|)$ is the number of bits exchanged between $\mathcal{V}$ and $\mathcal{P}$ in the protocol.*

We say that the IWV *has an* efficient prover*, if the honest prover strategy $\mathcal{P}$ can be implemented in polynomial time, if the prover is given explicit access to w (i.e., the same witness to which the verifier has oracle access to).*

Loosely speaking, we say that an IWV is *oblivious* if the verifier makes all its queries non-adaptively at the end of the interaction. Put differently, at the end of the interaction the verifier specifies some query set Q of bits from the witness, and a predicate ϕ and accepts if and only if $\phi(w_Q) = 1$. For technical considerations in our proof, we actually require that the verifier generate succinct descriptions of Q and ϕ (see Definitions 2.1 and 2.2 for the precise technical definition of succinct descriptions of functions and sets). This allows the verifier to run in time that is sublinear in the sizes of Q and ϕ. We proceed to the formal definition:

Definition 3.2 (Oblivious IWV)**.** *An* Oblivious IWV *for an* NP *relation R with witness length m, is an interactive protocol between a computationally unbounded prover $\mathcal{P}$ and a verifier $\mathcal{V}$ on a given input x. At the end of the interaction either the verifier rejects or it outputs a succinct description $\langle Q \rangle$ of a set $Q \subseteq [m]$ of size q and succinct description $\langle \phi \rangle$ of a predicate $\phi : \{0, 1\}^q \to \{0, 1\}$ such that*

- **Completeness:** *If* $(x, w) \in R$ *then*

$$\Pr\left[\mathcal{V} \textit{ does not reject and } \phi(w_Q) = 1\right] = 1.$$

- **Soundness:** *If* $R(x) = \emptyset$ *(i.e., x is not in the underlying* NP *language), then for every a priori fixed w^* and every prover strategy $\mathcal{P}^*$:*

$$\Pr\left[\mathcal{V} \textit{ does not reject and } \phi(w^*_Q) = 1\right] \leq 1/2.$$

The query complexity *of an oblivious* IWV *is q, the size of the set Q, and the* communication complexity cc *is the number of bits exchanged between $\mathcal{V}$ and $\mathcal{P}$ in the protocol.*

We say that the IWV *has an efficient prover strategy, if the honest prover strategy $\mathcal{P}$ can be implemented in polynomial time, if the prover is given explicit access to w (i.e., the same witness to which the verifier has oracle access to).*

We will rely on the following theorem, which is established in Section 3.1. Loosely speaking, this result shows that any NP relation verifiable by small depth circuits has an IWV in which we can trade off the query and communication complexities, such that their product is roughly equal to the witness length. In particular, it yields IWV protocols for all such NP relations in which the complexity scales with the square root of the witness length.

Theorem 3.3. *[*IWV*s for Bounded Space Relations] Let R be an* NP *relation with witness length $m = m(n)$, which can be verified in $\mathsf{poly}(n)$ time and space $S = S(n)$. Then, for every parameter $q = q(n, m)$ and constant $\delta > 0$, there exists a constant-round oblivious* IWV *for R. The query complexity is q and the communication complexity is $\mathsf{cc} = \mathsf{cc}(n, m) = \big((m/q) \cdot m^\delta \cdot \mathsf{poly}(S)\big)$. The verifier runs in time $\Big(n \cdot \mathsf{polylog}(n, m) + \tilde{O}(\mathsf{cc})\Big)$. The prover runs in time $\mathsf{poly}(n, m)$, given as input the* NP *witness.*

We remark that a similar result holds for any language computable in (log-space uniform) NC, where in the case of NC the number of rounds is $\mathsf{polylog}(n, m)$ rather than constant, and the m^δ terms in the communication complexity and the verifier runtime are replaced by $m^{o(1)}$.

3.1 Constructing IWVs for NP

The main technical tool that we use to prove Theorem 3.3 is the *interactive proofs of proximity* (IPPs) protocol of Rothblum, Vadhan and Wigderson [RVW13]. First, in Section 3.1.1 we introduce the model of IPPs and state the [RVW13] result. Then, in Section 3.1.2, we prove Theorem 3.3.

3.1.1 Background on IPPs

Loosely speaking, IPPs are interactive proofs in which the verifier runs in sub-linear time in the input length and is assured that the input is *close* to the language. Actually, we will think of the input of the verifier as being composed of two parts: a short input $x \in \{0, 1\}^n$ to which the verifier has direct access and a long input $y \in \{0, 1\}^m$ to which the verifier has oracle access. The goal is for the verifier to run in time that is sub-linear in m and to verify that y is far from any y' such that the pair (x, y') are in the language. Since such languages are composed of input pairs, we refer to them as *pair languages*.

Definition 3.4 (Interactive Proof of Proximity (IPP) [EKR04, RVW13])**.** *An interactive proof of proximity (IPP) for the pair language $\mathcal{L}$ is an interactive protocol with two parties: a (computationally unbounded) prover $\mathcal{P}$ and a computationally bounded verifier $\mathcal{V}$. Both parties get as input $x \in \{0,1\}^n$ and a proximity parameter $\varepsilon > 0$. The verifier also gets oracle access to $y \in \{0,1\}^m$ whereas the prover has full access to y. At the end of the interaction, the following two conditions are satisfied:*

1. **Completeness:** *For every pair $(x, y) \in \mathcal{L}$, and proximity parameter $\varepsilon > 0$ it holds that*
$$\Pr\left[\left(\mathcal{P}(y), \mathcal{V}^y\right)(x, |y|, \varepsilon) = 1\right] = 1.$$
2. **Soundness:** *For every $\varepsilon > 0$, $x \in \{0,1\}^n$ and y that is ε-far from the set $\{y' : (x, y') \in \mathcal{L}\}$, and for every computationally unbounded (cheating) prover $\mathcal{P}^*$ it holds that*
$$\Pr\left[\left(\mathcal{P}^*(y), \mathcal{V}^y\right)(x, |y|, \varepsilon) = 1\right] \leq 1/2.$$

An IPP for $\mathcal{L}$ is said to have query complexity $q = q(n, m, \varepsilon)$ if, for every $\varepsilon > 0$ and $(x, y) \in \mathcal{L}$, the verifier $\mathcal{V}$ makes at most $q(|x|, |y|, \varepsilon)$ queries to y when interacting with $\mathcal{P}$. The IPP is said to have communication complexity $\mathsf{cc} = \mathsf{cc}(n, m, \varepsilon)$ if, for every $\varepsilon > 0$ and pair $(x, y) \in \mathcal{L}$, the communication between $\mathcal{V}$ and $\mathcal{P}$ consists of at most $\mathsf{cc}(|x|, |y|, \varepsilon)$ bits.

We are now ready to state the main result of [RVW13]. Actually we will use an extension, due to [RRR16], of the [RVW13] IPP.[9]

Theorem 3.5 (IPPs for Bounded Space Computations, [RVW13, RRR16])**.** *Let $\mathcal{L}$ be a pair language that is computable in $\mathsf{poly}(n, m)$ time and space $S = S(n, m)$. For every constant $\delta > 0$, there is an IPP for $\mathcal{L}$ with the following parameters. For every input pair $(x, y) \in \{0,1\}^n \times \{0,1\}^m$, the query complexity is $q = q(n, m, \varepsilon) = (1/\varepsilon) \cdot m^{O(\delta)}$, the communication complexity is $\mathsf{cc} = \mathsf{cc}(n, m, \varepsilon) = (\varepsilon \cdot m^{1+O(\delta)} \cdot \mathsf{poly}(S))$, and the number of rounds is constant. The honest prover runs in time $\mathsf{poly}(n, m, 1/\varepsilon)$ and the verifier runs in time $n \cdot \mathsf{polylog}(n, m) + \tilde{O}(q + \mathsf{cc})$.*

Furthermore, the verification can be implemented in two phases. In the communication *phase the verifier interacts with the prover without querying y. The verifier's running time in this phase is $n \cdot \mathsf{polylog}(n, m) + \tilde{O}(\mathsf{cc})$. At the end of the communication phase, the verifier either rejects or it outputs a succinct description $\langle Q \rangle$ of a set $Q \subseteq [m]$ of size q and succinct description $\langle \phi \rangle$ of a predicate $\phi : \{0,1\}^q \to \{0,1\}$ which can be computed by a (logspace uniform) NC^1 circuit of size $\tilde{O}(q)$. In the* query *phase, the verifier only queries y_Q and accepts if and only if $\phi(y_Q) = 1$.*

3.1.2 Proof of Theorem 3.3

Let R be an NP relation with witness length $m = m(n)$, which can be verified in $\mathsf{poly}(n)$ time and space $S = S(n)$. Let $\delta > 0$ be a constant and let $q = q(n, m) \in [m]$ be a parameter.

View R as a pair language. Namely each input-witness pair (x, w) is viewed as a an input pair (x, w) for the pair language. Let $(\mathcal{P}, \mathcal{V})$ by the IPP for R guaranteed by Theorem 3.5 with respect

[9] The [RRR16] IPP extends the [RVW13] result from bounded depth computations to also hold for bounded space computations. Also, and more importantly for our purposes, the [RRR16] IPP only requires a constant number of rounds (for languages computable in bounded space).

to proximity parameter $\varepsilon = \frac{m^{O(\delta)}}{q}$. We claim that $(\mathcal{P}, \mathcal{V})$ is an IWV for R, where queries to the IPP input oracle y are emulated by querying the IWV witness oracle.

Completeness follows immediately from the completeness of the IPP. For soundness, let $x \in \{0,1\}^n$ such that $R(x) = \emptyset$. Fix an alleged witness string $\tilde{y}$ and an IWV prover strategy $\mathcal{P}^*$. We view $\mathcal{P}^*$ as a cheating prover strategy for the IPP $(\mathcal{P}, \mathcal{V})$ with respect to the input pair $(x, \tilde{y})$. Since $R(x) = \emptyset$, it holds that $(x, \tilde{y})$ is at infinite distance from the set $\{y' : (x, y') \in R\}$ (in particular the distance is more than ε). Thus, by the IPP soundness, the verifier rejects with probability at least $1/2$.

The fact that the foregoing IWV is *oblivious*, as well as its complexity, follows from the furthermore part of Theorem 3.5.

3.2 Lower Bound for IWVs

In this section we show a lower bound on the efficiency of IWVs that, loosely speaking, shows that Theorem 3.3 is tight. This lower bound relies on the existence of an *exponentially strong cryptographic pseudorandom generator (PRG)*. By exponentially strong, we mean that the output of the generators, when evaluated of a random string of length m, computationally indistinguishable from a uniformly random string even for adversaries running in time $2^{m/100} \cdot \mathsf{poly}(m)$. We remark that by assuming only sub-exponential hardness (i.e., hardness against $2^{m^{\varepsilon}}$ time adversaries) we can still obtain a meaningful (albeit weaker) lower bound.

For an NP relation R, we denote by $R^{\otimes k}$ the relation

$$R^{\otimes k} \stackrel{\text{def}}{=} \Big\{ \big((x_1, \ldots, x_k), (w_1, \ldots, w_k)\big) \;:\; \forall j \in [k],\ (x_j, w_j) \in R \text{ and } |x_1| = \ldots = |x_k| \Big\}.$$

Theorem 3.6. *Assume the existence of an exponentially hard cryptographic PRG. Then, there exists an* NP *relation* R *with witnesses of size* m*, such that for every* $k \le \mathsf{poly}(m)$*, every* IWV *for* $R^{\otimes k}$ *must have either query complexity* $q = \Omega(k)$ *or communication complexity* $\mathsf{cc} = \Omega(m)$*. Furthermore, if the PRG is injective, then* R *is a* UP *relation.*

The proof of Theorem 3.6 follows in a straightforward way from the IPP lower bound of Kalai and Rothblum [KR15]. We provide a proof sketch below.

Proof Sketch. Let $G : \{0,1\}^m \to \{0,1\}^n$ be an exponentially strong PRG. Let $R_G = \{(y, s) : y = G(s)\}$. Clearly $R_G \in \mathsf{NP}$ and if G is injective, then $R_G \in \mathsf{UP}$. Suppose toward a contradiction that there exists an IWV $(\mathcal{P}, \mathcal{V})$ for $\mathcal{R}_G^{\otimes k} = \Big\{ \big((y_1, \ldots, y_k), (s_1, \ldots, s_k)\big) : \forall j \in [k],\ y_j = G(s_j) \Big\}$ with query complexity $k/100$ and communication complexity $m/100$.

The proof is composed of two steps. First, we use $(\mathcal{P}, \mathcal{V})$ to construct a relatively efficient interactive proof for R_G (i.e., with communication $m/100$). The second step is to show that such an interactive proof violates the exponential hardness of G.

We start with the first step: constructing an interactive proof $(\mathcal{P}', \mathcal{V}')$ for R_G - i.e., deciding whether a given string is in the image of G. Actually, we only achieve a relaxed notion of interactive proof. Specifically we have the following two relaxations:

- (Average-case Completeness:) Completeness holds for *most* inputs in the language but not necessarily for all inputs. Namely, for most s, the verifier $\mathcal{V}'$ accepts with high probability after interacting with the prover $\mathcal{P}'$ on common input $G(s)$.

- (Common Random String:) Both the prover and verifier have access to a (relatively long) common random string. We do not count this random string as part of the communication complexity of the protocol.

We proceed to describe the interactive proof $(\mathcal{P}', \mathcal{V}')$ for $R_G^{\otimes k}$. The common random string consists of $\big((s_1, \dots, s_k), i\big) \in_R (\{0,1\}^m)^k \times [k]$. We define $y_j = G(s_j)$, for all $j \in [k]$. In addition to the common random string, the verifier $\mathcal{V}'$ and prover $\mathcal{P}'$ are given as input $y \in \{0,1\}^n$, and $\mathcal{V}'$ needs to decide whether there exists $s \in \{0,1\}^m$ such that $G(s) = y$ (i.e., whether $R_G(y) =\neq \emptyset$).

The interactive proof proceeds as follows. The prover $\mathcal{P}'$ and verifier $\mathcal{V}'$ run the IWV $(\mathcal{P}, \mathcal{V})$ where the input is $(y_1, \dots, y_{i-1}, y, y_{i+1}, \dots, y_k)$ and the witness, to which $\mathcal{V}$ only gets oracle access, is $(s_1, \dots, s_{i-1}, 0^m, s_{i+1}, \dots, s_k)$.

To show that average-case completeness holds, let $s \in_R \{0,1\}^m$ and consider the execution of $(\mathcal{P}', \mathcal{V}')$ on input $y = G(s)$. Consider a mental experiment in which we run the IWV with the witness $(s_1, \dots, s_{i-1}, s, s_{i+1}, \dots, s_k)$ (rather than $(s_1, \dots, s_{i-1}, 0^m, s_{i+1}, \dots, s_k)$ as in the real execution). By the completeness of the IWV, in this mental experiment, $\mathcal{V}$ accepts.

Observe that, conditioned on not querying s, the view of $\mathcal{V}$ is identical in the real execution and in the mental experiment, and so it will accept also in the real execution. Moreover, since i and y are random, and $\mathcal{V}$ makes at most $k/100$ queries, with constant probability, $\mathcal{V}$ does not query s and average-case completeness follows.

Soundness of $(\mathcal{P}', \mathcal{V})$ is easier to show. Specifically, if $R_G(y) = \emptyset$ then $R_G^{\otimes k}\big(y_1, \dots, y_{i-1}, y, y_{i+1}, \dots, y_\ell\big) = \emptyset$ and so by the soundness of the IWV, the verifier $\mathcal{V}$ rejects with high probability given oracle access to any fixed witness (in particular, $(s_1, \dots, s_{i-1}, 0^m, s_{i+1}, \dots, s_k)$) and no matter what the cheating prover does.

Thus, $(\mathcal{P}', \mathcal{V}')$ is an interactive proof for R_G. Observe that $(\mathcal{P}', \mathcal{V}')$ has the same communication as $(\mathcal{P}, \mathcal{V})$ - namely, $m/100$.

The second step of the proof is to observe that the foregoing interactive proof can be emulated by an *algorithm* A running in time $2^{m/100} \cdot \mathsf{poly}(m,k)$. This is similar to the proof that $\mathsf{IP} \subseteq \mathsf{PSPACE}$ (i.e., interactive proofs can be emulated by bounded space machines).[10] Thus, using the fact that $k = \mathsf{poly}(m)$, the PRG can be broken in time $2^{m/100} \cdot \mathsf{poly}(m)$ time, in contradiction to our assumption.

□

4 Batch Verification for UP

For an NP relation R, we denote by $R^{\otimes k}$ the relation

$$R^{\otimes k} \stackrel{\text{def}}{=} \Big\{\big((x_1, \dots, x_k), (w_1, \dots, w_k)\big) \;:\; \forall j \in [k],\ (x_j, w_j) \in R \text{ and } |x_1| = \ldots = |x_k|\Big\}.$$

Theorem 4.1 (Batch Verification for UP)**.** *Let R be a UP relation that is verifiable in NC^1, with witnesses of length $m = m(n)$ such that m and n are polynomially related. Let $k = k(n) \geq 1$ and let $\delta > 0$ be a constant. There exists a constant-round interactive proof system for $R^{\otimes k}$ such that the verifier runs in time $\Big(\tilde{O}(n \cdot k) + k^\delta \cdot m^{1+\delta}\Big)$, the (honest) prover runs in time $\mathsf{poly}(n,m,k)$ and the communication complexity is $\big(k^\delta \cdot m^{1+\delta}\big)$.*

[10] We cannot afford to count the CRS as part of the communication of the interactive proof, since it is of length $m \cdot k + \log(k) \gg m/100$. Rather, observe that A can simply sample the CRS directly, in time $\mathsf{poly}(m,k)$ (rather than enumerating over all possible CRS strings).

By the proof of the Cook-Levin theorem every UP language has a UP relation that is verifiable in NC^1 (albeit with a polynomial blowup in the witness size). Thus, Theorem 4.1 is applicable to *any* UP language.

As described in the technical overview (see Section 1.2), our batch verification protocol works in iterations, where the goal of each iteration is to significantly reduce the number of instances that are still "alive". In Section 4.1 we describe the iterative step and then in Section 4.2 we describe the UP batch verification protocol.

4.1 The Iterative Step

We first describe the main step in our proof, corresponding to a single iteration of the protocol that was described in Section 1.2. Loosely speaking, this step shows an interactive protocol, where if we start with a false claim about a subset of the k UP statements, then at the end of the protocol, with high probability, we will have a false claim about a smaller subset of the statements.

This step, which appears next in Lemma 4.1.1, is where we rely on the existence of IWVs for general NC^1 relations. Lemma 4.1.1 can be instantiated with any such IWV. Later, in Section 4.2, we will use Lemma 4.1.1 instantiated with the IWVs that were shown to exist (unconditionally) in Theorem 3.3.

Lemma 4.1.1. *Suppose that for every parameter q, every NP relation computable in NC^1 has an oblivious IWV with an efficient prover such that with respect to inputs of size n and witness of size m, the proof-system has soundness error $\varepsilon = \varepsilon(n,m,q)$, verifier complexity $\mathcal{V}\mathsf{time} = \mathcal{V}\mathsf{time}(n,m,q)$, prover complexity $\mathcal{P}\mathsf{time}(n,m,q)$ (assuming the prover is given access to the NP witness), round complexity $r(n,m,q)$, query complexity q and communication complexity $\mathsf{cc} = \mathsf{cc}(n,m,q)$.*

Let R be a UP relation computable in NC^1, with witnesses of length $m = m(n)$, and let $\delta > 0$ be a constant. There exists an interactive protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$ such that the following holds. Both parties get as input $\mathbf{x} = (x_1, \dots, x_k) \in (\{0,1\}^n)^k$ and succinct descriptions $\langle S \rangle$ and $\langle \phi \rangle$, of a set $S \subseteq [k]$ of size s and circuit ϕ, respectively. The prover $\mathcal{P}$ also gets witnesses $\mathbf{w} = (w_1, \dots, w_k) \in (\{0,1\}^m)^k$ as an additional input. The two parties interact and at the end of the interaction $\mathcal{V}$ either rejects or outputs succinct descriptions $\langle S' \rangle$ and $\langle \phi' \rangle$ of a subset $S' \subseteq S$, of size $s^{1-\delta}$, and circuit ϕ', respectively, such that:

- **(Completeness:)** *If $(x_j, w_j) \in R$ for all $j \in S$, and $\phi(\mathbf{w}|_S) = 1$, then, with probability 1, after interacting with $\mathcal{P}$, the verifier $\mathcal{V}$ outputs $\langle S' \rangle$ and $\langle \phi' \rangle$ such that $\phi'(\mathbf{w}|_{S'}) = 1$.*

- **(Soundness:)** *If either (1) there exists $j \in S$ such that $R(x_j) = \emptyset$, or (2) $(x_j, w_j) \in R$ for all $j \in S$ but $\phi(\mathbf{w}|_S) = 0$, then, for every prover strategy $\mathcal{P}^*$, with probability $1 - \varepsilon$, after interacting with $\mathcal{P}^*$, the verifier $\mathcal{V}$ either rejects or outputs $\langle S' \rangle$ and $\langle \phi' \rangle$ such that one of the following holds:*

  1. *$\exists j \in S'$ such that $R(x_j) = \emptyset$; or*
  2. *$\phi'(\mathbf{w}|_{S'}) = 0$.*

- **(Complexity:)** *The protocol $(\mathcal{P}, \mathcal{V})$ has verifier complexity $\mathcal{V}\mathsf{time}\big(n \cdot k + \mathsf{poly}(\log n, \log k), s \cdot m, q\big)$, prover complexity $\mathcal{P}\mathsf{time}\big(n \cdot k + \mathsf{poly}(\log n, \log k), s \cdot m, q\big)$ (assuming the prover is given access to the k UP witnesses), round complexity $r = r\big(n \cdot k + \mathsf{poly}(\log n, \log k), s \cdot m, q\big)$ and communication complexity $\mathsf{cc} = \mathsf{cc}\big(n \cdot k + \mathsf{poly}(\log n, \log k), s \cdot m, q\big)$, where $q = s^{1-\delta}$.*

Proof. Let R be a UP relation computable in NC^1. We consider a related NP relation R_k defined as follows. The input to R_k is $(x_1,\ldots,x_k,\langle S\rangle,\langle\phi\rangle)$ and the witness is a sequence $\mathbf{w}|_S=(w_j)_{j\in S}$. The relation checks that (1) for every $j\in S$ it holds that $(x_j,w_j)\in R$, and (2) that $\phi(\mathbf{w}|_S)=1$. Observe that membership in R_k can be decided in (logspace uniform) NC^1, and therefore, by the lemma's hypothesis there exists an oblivious IWV $(\mathcal{P},\mathcal{V})$ for R_k where we use parameter $q=s^{1-\delta}$, where s is the size of the set S.

We use $(\mathcal{P},\mathcal{V})$ to construct a protocol $(\mathcal{P}_k,\mathcal{V}_k)$ as required in the theorem's statement. Given as common input $(x_1,\ldots,x_k,\langle S\rangle,\langle\phi\rangle)$, the verifier $\mathcal{V}_k$ and prover $\mathcal{P}_k$ run $(\mathcal{P},\mathcal{V})$ with respect to the common input $(x_1,\ldots,x_k,\langle S\rangle,\langle\phi\rangle)$, where $\mathcal{P}_k$ gets as an auxiliary input also $\mathbf{w}|_S=(w_j)|_{j\in S}$.

If $\mathcal{V}$ rejects then $\mathcal{V}_k$ immediately rejects. Otherwise, $\mathcal{V}$ outputs succinct NC^1 descriptions $\langle Q'\rangle$ and $\langle\phi'\rangle$ where $Q'\subseteq[k]\times[m]$, of size $k^{1-\delta}$ specifies which locations to read from $\mathbf{w}|_S$ and ϕ' is a predicate specifying whether $\mathcal{V}$ would have accepted had it read those bits. For simplicity, and without loss of generality, we assume that Q specifies $k^{1-\delta}$ of the witnesses $w_1,\ldots,w_k$ entirely and ignores the rest. Let $S'\subseteq[k]$ denote the witnesses that the Q refers to. The verifier $\mathcal{V}_k$ outputs $\langle S'\rangle$ and $\langle\phi'\rangle$.

Completeness. Let $x_1,\ldots,x_k$ be a sequence of inputs, $S\subseteq[k]$ a set and ϕ a circuit such that there exist unique $\mathbf{w}|_S=(w_j)|_{j\in S}$ such that $(x_j,w_j)\in R$ for all $j\in[k]$ and $\phi(\mathbf{w}|_S)=1$. The IWV protocol is run with respect to an input $\big((x_1,\ldots,x_k,\langle S\rangle,\langle\phi\rangle),\mathbf{w}_S\big)\in R_k$. Thus, by the completeness of the IWV, with probability 1, it holds that $\phi'(\mathbf{w}_{S'})=1$.

Soundness. Suppose that either there exists $j\in S$ such that $R(x_j)=\emptyset$, or $\mathbf{w}|_S=(w_j)_{j\in S}$ consists of the corresponding unique witnesses and $\langle\phi\rangle$ is such that $\phi(\mathbf{w}|_S)=0$. Let $\mathcal{P}^*$ be a cheating prover strategy. To show that the soundness condition holds, it suffices to prove the following claim:

Claim 4.1.1.

$$\Pr\Big[\big(\forall j\in S',\ R(x_j)\neq\emptyset\big)\ \textit{and}\ \big(\phi'(\mathbf{w}|_{S'})=0\big)\Big]\leq\varepsilon$$

Proof. For every $j\in S$, if $R(x_j)\neq\emptyset$ then define $\hat{w}_j=w_j$ (i.e., the unique witness for x_j), whereas if $R(x_j)=\emptyset$, then define $\hat{w}_j$ as some arbitrary fixed string (e.g., 0^m).

We view $\mathcal{P}^*$ as an adversary for the oblivious IWV protocol, with respect to the a priori fixed witness string $\hat{\mathbf{w}}|_S=(\hat{w}_j)|_{j\in S}$. By the soundness condition of the oblivious IWV (see Definition 3.2), it holds that:

$$\Pr[\phi'(\hat{\mathbf{w}}_{S'})=1]\leq\varepsilon.$$

For all $j\in S$ we have that, if $R(x_j)\neq\emptyset$ then $w_j=\hat{w}_j$. Thus,

$$\Pr\Big[\big(\forall j\in S',\ R(x_j)\neq\emptyset\big)\ \text{and}\ \big(\phi'(\mathbf{w}|_{S'})=0\big)\Big]\leq\Pr\big[\phi'(\hat{\mathbf{w}}|_{S'})=0\big)\big]\leq\varepsilon,$$

and the claim follows.

□

The UP Batching Protocol $(\mathcal{P}_{\mathsf{batch}}, \mathcal{V}_{\mathsf{batch}})$

Common Input: $\mathbf{x} = (x_1, \ldots, x_k) \in (\{0,1\}^n)^k$.
Prover's Auxiliary Input: witnesses $\mathbf{w} = (w_1, \ldots, w_k) \in (\{0,1\}^m)^k$.

1. Set $\langle S_1 \rangle$ to be a concise description of the set $S_1 = [k]$, and $\langle \phi_1 \rangle$ to be a concise description of a circuit $\phi_1 : (\{0,1\}^m)^k \to \{0,1\}$ that always outputs 1.
2. For $i = 1, \ldots, \ell - 1$, where $\ell = O(1/\delta)$:
   (a) Run $(\mathcal{P}_{\mathsf{reduction}}, \mathcal{V}_{\mathsf{reduction}})$ on common input $(\mathbf{x}, \langle S_i \rangle, \langle \phi_i \rangle)$, and with respect to parameter $q_i = s_i^{1-\delta}$, where s_i is the size of the set S_i, and with soundness error $\varepsilon = 1/(10\ell)$.[a] More specifically, $\mathcal{V}_{\mathsf{batch}}$ emulates $\mathcal{V}_{\mathsf{reduction}}$ and $\mathcal{P}_{\mathsf{batch}}$ emulates $\mathcal{P}_{\mathsf{reduction}}$, using $\mathbf{w}|_{S_i}$ as the auxiliary input.
   (b) If $\mathcal{V}_{\mathsf{reduction}}$ rejects then V_{batch} immediately rejects. Otherwise $\mathcal{V}_{\mathsf{reduction}}$ outputs $\langle S_{i+1} \rangle$ and $\langle \phi_{i+1} \rangle$.
3. $\mathcal{P}_{\mathsf{batch}}$ sends to $\mathcal{V}_{\mathsf{batch}}$ the witnesses $\mathbf{w}|_{S_\ell} = (w_j)_{j \in S_\ell}$.
4. The verifier $\mathcal{V}_{\mathsf{batch}}$ expands $\langle S_\ell \rangle$ to a full description of the set S_ℓ. The prover and verifier then run the doubly efficient interactive proof of Theorem 2.5 on input $\big((x_j)_{j \in S_\ell}, (w_j)_{j \in S_\ell}, \langle \phi_\ell \rangle\big)$ checking that for every $j \in S_\ell$ it holds that $(x_j, w_j) \in R$ and that $\phi_\ell(\mathbf{w}|_{S_\ell}) = 1$ (the protocol of Theorem 2.5 is used with a sufficiently small parameter $\tau > 0$ to be determined in the analysis). If all checks pass then $\mathcal{V}_{\mathsf{batch}}$ accepts and otherwise it rejects.[b]

[a] Such soundness amplification can be achieved by repeating the base protocol $O(\log(\ell))$ times in parallel.

[b] This step could be replaced by having the verifier directly check by itself that $(x_j, w_j) \in R$ for every $j \in S_\ell$. However, doing so introduces an additive overhead of $\mathsf{poly}(n, m)$ to the verifier's running time (arising from the complexity of the relation R) which we can reduce to $\tilde{O}(n+m)$ by using the interactive proof of Theorem 2.5.

Figure 1: UP Batching

Complexity. The stated complexity follows from the complexity of the IWV, which is run on an input of size $n \cdot k + \mathsf{poly}(\log n, \log k)$ (the concatenation of the k inputs and succinct representations $\langle S \rangle$ and $\langle \phi \rangle$), witness size $s \cdot m$ (i.e., the length of $(w_i)|_{i \in S}$ - the concatenation of the relevant witnesses) and with respect to the parameter $q = s^{1-\delta}$.

□

4.2 The Batch Verification Protocol: Proof of Theorem 4.1

Let $(P_{\mathsf{reduction}}, V_{\mathsf{reduction}})$ be the protocol guaranteed by Lemma 4.1.1, with respect to UP relation R and the IWV protocol of Theorem 3.3. We construct a protocol $(\mathcal{P}_{\mathsf{batch}}, \mathcal{V}_{\mathsf{batch}})$ satisfying the requirement of Theorem 4.1. The protocol is described in Fig. 1.

To complete the proof of Theorem 4.1 we need to show that completeness and soundness hold, as well as analyze the complexity of the protocol.

Completeness. Let $x_1, \ldots, x_k$ such that there exist (unique) witnesses $\mathbf{w} = (w_1, \ldots, w_k)$ such that $(x_j, w_j) \in R$, for every $j \in [k]$. Let $S_1, \ldots, S_\ell$ and $\phi_1, \ldots, \phi_\ell$ be the sets and formulas, respectively, generated in the interaction between $\mathcal{P}_{\mathsf{batch}}$ and $\mathcal{V}_{\mathsf{batch}}$.

Claim 4.1.2. *For every $i \in [\ell]$, with probability 1, it holds that $\mathcal{V}_{\mathsf{batch}}$ does not reject prior to the i^{th} iteration and $\phi_i(\mathbf{w}_{S_i}) = 1$.*

Proof. We prove by induction on i. In the case base ϕ_1 always outputs 1 and so the claim holds trivially. Suppose that the claim holds for some value i. Thus, $\phi_i(\mathbf{w}_{S_i}) = 1$.

Since $(x_j, w_j) \in R$ for all $j \in S_i$, and $\phi_i(\mathbf{w}_{S_i}) = 1$, by the completeness of $(P_{\mathsf{reduction}}, \mathcal{V}_{\mathsf{reduction}}))$ it holds that $V_{\mathsf{reduction}}$ does not reject and outputs $\langle S_{i+1} \rangle$ and $\langle \phi_{i+1} \rangle$ such that $\phi_{i+1}(\mathbf{w}_{S_{i+1}}) = 1$. The claim follows. □

Thus, at the end of the loop $\phi_\ell(\mathbf{w}_{S_\ell}) = 1$. Since $\mathcal{P}_{\mathsf{batch}}$ sends the correct (unique) witnesses $\mathbf{w}_{S_\ell}$ in Step 3, by the completeness of the interactive proof-system of Theorem 2.5, the verifier $\mathcal{V}_{\mathsf{batch}}$ accepts.

Soundness. Let $x_1, \dots, x_k \in \{0,1\}^n$ such that $\exists j^* \in [k]$ with $R(x_{j^*}) = \emptyset$. Let $\mathcal{P}^*$ be a cheating prover strategy. For every $j \in [k]$, if $R(x_j) \neq \emptyset$, let w_j be the corresponding unique witness. Purely for notational convenience, we also define w_j to be an arbitrary string, for every $j \in [k]$ such that $R(x_j) = \emptyset$. Let $\mathbf{w} = (w_1, \dots, w_k)$.

For every $i \in [\ell]$, let E_i denote the conjunction of the following three events:

- The verifier $\mathcal{V}_{\mathsf{batch}}$ has not rejected prior to the start of the i^{th} iteration; and
- For every $j \in S_i$ it holds that $R(x_j) \neq \emptyset$; and
- $\phi_i(\mathbf{w}|_{S_i}) = 1$

Setting $\varepsilon = 1/(10\ell)$, we have the following central claim:

Claim 4.1.3. *For every $i \in [\ell]$:*

$$\Pr[E_i] \leq (i-1) \cdot \varepsilon$$

Proof. We prove the claim by induction on i. For the base case $i = 1$, since $R(x_{j^*}) = \emptyset$ and $j^* \in [k] = S_1$, it holds that

$$\Pr[E_1] \leq \Pr\left[\forall j \in S_1,\ R(x_j) \neq \emptyset\right] \leq \Pr[R(x_{j^*}) \neq \emptyset] = 0.$$

Assume that the claim holds for some $i \in [\ell - 1]$. By elementary probability theory,

$$\Pr[E_{i+1}] \leq \Pr[E_i] + \Pr\left[E_{i+1} | \neg E_i\right].$$

By the induction hypothesis the first term is bounded by $(i-1)\cdot\varepsilon$ and so to complete the proof we need to bound the second term by ε. This holds since:

$$\begin{aligned}
\Pr[E_{i+1} \mid \neg E_i] &\leq \Pr\left[E_{i+1} \mid \mathcal{V}_{\mathsf{batch}} \text{ rejects prior to iteration } i\right] + \Pr\left[E_{i+1} \;\middle|\; \begin{array}{c} \left(\exists j \in S_i \text{ s.t. } R(x_j) = \emptyset\right) \\ \text{or} \\ \left(\phi_i(\mathbf{w}|_{S_i}) = 0\right) \end{array}\right] \\
&= \Pr\left[E_{i+1} \;\middle|\; \begin{array}{c} \left(\exists j \in S_i \text{ s.t. } R(x_j) = \emptyset\right) \\ \text{or} \\ \left(\phi_i(\mathbf{w}|_{S_i}) = 0\right) \end{array}\right] \\
&\leq \varepsilon,
\end{aligned}$$

where the equality is since if $\mathcal{V}_{\mathsf{batch}}$ rejects prior to iteration i then clearly it also rejects prior to iteration $i+1$ (and so E_{i+1} does not occur), and the last inequality follows directly from the soundness condition of Lemma 4.1.1 (where recall that we used that protocol with soundness error $\varepsilon = 1/(10\ell)$). □

Thus, with probability at least $1 - \ell \cdot \varepsilon \geq 0.9$, one of the following events occurs at the end of the loop:

1. $\mathcal{V}_{\mathsf{batch}}$ has already rejected; or
2. There exists $j \in [S_\ell]$ such that $R(x_j) = \emptyset$; or
3. $\phi_\ell(\mathbf{w}|_{S_\ell}) = 0$.

We show that in each of these cases, the verifier rejects with high probability. For the first case this is immediate. In the second case, the cheating prover must send some incorrect witness w_j^*. Thus, the verifier and prover run the doubly efficient interactive proof-system of Theorem 2.5 on a false input, and by the soundness condition of that protocol, the verifier rejects with probability 0.9.

Lastly, if $\phi_\ell(\mathbf{w}_{S_\ell}) = 0$, then either $\mathcal{P}^*$ sends witnesses that are not the unique witnesses, in which case again the protocol of Theorem 2.5 is run on a false statement or $\mathcal{P}^*$ sends the unique witnesses but in this case the statement is still false since $\phi_\ell(\mathbf{w}|_{S_\ell}) = 1$. Thus, in both cases, by the soundness of Theorem 2.5, the verifier rejects with probability 0.9.

Thus, in all cases the verifier rejects with probability at least $0.9^2 \geq 1/2$.

Complexity. For every $i \in [\ell]$, let s_i denote the size of the set S_i generated in the interaction. By Lemma 4.1.1 it holds that $s_i \leq k^{1-(i-1)\cdot\delta}$.

Consider the i^{th} iteration of the loop, for some $i \in [\ell - 1]$. Let $n_i = n \cdot k + \mathsf{polylog}(n,k)$ and $m_i = s_i \cdot m \leq k^{1-(i-1)\cdot\delta} \cdot m$ and recall that we set $q_i = s_i^{1-\delta} \leq k^{1-i\cdot\delta}$. By Theorem 3.3, together with Lemma 4.1.1, the i^{th} iteration takes a constant number of rounds and has:

- Communication complexity:

$$\begin{aligned} \mathsf{cc}_i &= (m_i/q_i) \cdot m_i^\delta \cdot \mathsf{polylog}(n,m) \\ &\leq \left((k^{1-(i-1)\cdot\delta} \cdot m)/k^{1-i\cdot\delta}\right) \cdot (k \cdot m)^\delta \cdot \mathsf{polylog}(n,m) \\ &= k^{2\delta} \cdot m^{1+\delta} \cdot \mathsf{polylog}(n,m) \end{aligned}$$

- Verifier running time:

$$\begin{aligned} \mathcal{V}\mathsf{time}_i &= n_i \cdot \mathsf{polylog}(n_i, m_i) + \tilde{O}(\mathsf{cc}_i) \\ &\leq n \cdot k \cdot \mathsf{polylog}(n,k,m) + k^{2\delta} \cdot m^{1+\delta} \cdot \mathsf{polylog}(n,m,k) \end{aligned}$$

- And prover running time (given the UP witnesses):

$$\begin{aligned} \mathcal{P}\mathsf{time}_i &= \mathsf{poly}(n_i, m_i) \\ &= \mathsf{poly}(n,m,k) \end{aligned}$$

To analyze the last two steps of the protocol, first observe that s_ℓ (i.e., the size of the final set S_ℓ) has size $s_\ell \leq k^{1-(\ell-1)\dot{\delta}} = O(1)$. Thus, Step 3 only adds an additional $s_\ell \cdot m = O(m)$ communication.

As for the verification time, generating the set S_ℓ takes time $\mathsf{poly}(s_\ell, \log k) = \mathsf{polylog}(k)$ (by the definition of concise description of sets, see Definition 2.2). The protocol of Theorem 2.5, is run on a logspace computation and with its parameter τ set to be a sufficiently small constant so that the communication is $O(m)$. This protocol takes an additional $O(1)$ rounds, the verifier runs in time $\tilde{O}(n+m)$, the prover runs in time $\mathsf{poly}(n, m)$ and the communication complexity is $O(m)$.

The parameters stated in the theorem's statement now follows by taking resetting δ to be sufficiently small (e.g., take $\delta' = \delta/4$) and the fact that m and n are polynomially related.

Acknowledgments

We thank Oded Goldreich for illuminating conversations and particularly for his insights that helped us crystallize the notion of Interactive Witness Verification.

References

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

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

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; A new characterization of NP. In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 2–13, 1992.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. Cryptology ePrint Archive, Report 2016/116, 2016. http://eprint.iacr.org/.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31, 1991.

[BGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 113–131, 1988.

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

[BRV18] Itay Berman, Ron D. Rothblum, and Vinod Vaikuntanathan. Zero-knowledge proofs of proximity. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 19:1–19:20, 2018.

[CG18] Alessandro Chiesa and Tom Gur. Proofs of proximity for distribution testing. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 53:1–53:14, 2018.

[EKR04] Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Fast approximate probabilistically checkable proofs. *Inf. Comput.*, 189(2):135–159, 2004.

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

[FGL14] Eldar Fischer, Yonatan Goldhirsh, and Oded Lachish. Partial tests, universal tests and decomposability. In *ITCS*, pages 483–500, 2014.

[FGM$^+$89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. *Advances in Computing Research*, 5:429–442, 1989.

[FRS94] Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. *Theor. Comput. Sci.*, 134(2):545–557, 1994.

[GGR15] Oded Goldreich, Tom Gur, and Ron D. Rothblum. Proofs of proximity for context-free languages and read-once branching programs - (extended abstract). In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 666–677, 2015.

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

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[Gol99] Oded Goldreich. *Modern cryptography, probabilistic proofs and pseudorandomness*, volume 17 of *Algorithms and Combinatorics.* Springer-Verlag, 1999.

[Gol17] Oded Goldreich. Overview of the doubly-efficient interactive proof systems of RRR. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:102, 2017.

[GR15] Tom Gur and Ron D. Rothblum. Non-interactive proofs of proximity. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 133–142, 2015.

[GR17] Tom Gur and Ron D. Rothblum. A hierarchy theorem for interactive proofs of proximity. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 39:1–39:43, 2017.

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

[KR09] Yael Tauman Kalai and Ran Raz. Probabilistically checkable arguments. In *CRYPTO*, pages 143–159, 2009.

[KR15] Yael Tauman Kalai and Ron D. Rothblum. Arguments of proximity - [extended abstract]. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 422–442, 2015.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

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

[RVW13] Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *STOC*, pages 793–802, 2013.

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.