Electronic Colloquium on Computational Complexity, Report No. 9 (2018)

Non-Interactive Delegation for Low-Space Non-Deterministic Computation

Saikrishna Badrinarayanan
UCLA
Email: saikrishna@cs.ucla.edu

Yael Tauman Kalai
Microsoft Research
Email: yael@microsoft.com

Dakshita Khurana
UCLA
Email: dakshita@cs.ucla.edu

Amit Sahai
UCLA
Email: sahai@cs.ucla.edu

Daniel Wichs
Northeastern University
Email: wichs@ccs.neu.edu

January 9, 2018

Abstract

We construct a delegation scheme for verifying non-deterministic computations, with complexity proportional only to the *non-deterministic space* of the computation. Specifically, letting n denote the input length, we construct a delegation scheme for any language verifiable in non-deterministic time and space $(\mathcal{T}(n), \mathcal{S}(n))$ with communication complexity $\mathsf{poly}(\mathcal{S}(n))$, verifier runtime $n \cdot \mathsf{polylog}(\mathcal{T}(n)) + \mathsf{poly}(\mathcal{S}(n))$, and prover runtime $\mathsf{poly}(\mathcal{T}(n))$.

Our scheme consists of only two messages and has adaptive soundness, assuming the existence of a sub-exponentially secure private information retrieval (PIR) scheme, which can be instantiated under standard (albeit, sub-exponential) cryptographic assumptions, such as the sub-exponential LWE assumption. Specifically, the verifier publishes a (short) public key ahead of time, and this key can be used by any prover to *non-interactively* prove the correctness of any *adaptively chosen* non-deterministic computation. Such a scheme is referred to as a non-interactive delegation scheme. Our scheme is *privately verifiable*, where the verifier needs the corresponding secret key in order to verify proofs.

Prior to our work, such results were known only in the Random Oracle Model, or under knowledge assumptions. Our results yield succinct non-interactive arguments based on sub-exponential LWE, for many natural languages believed to be outside of P.

ISSN 1433-8092

Contents

1 Introduction

Efficient verification of computation, also known as *delegation of computation*, is one of the most fundamental notions in computer science. Indeed, one of the most basic computational objects in computer science is the complexity class NP – that is defined as the class of problems whose computation can be verified in polynomial time.

The broader importance of the problem of delegating computation has become evident in recent years due to the increasing popularity of cloud computing. In this setting, one participant, the client or delegator (or verifier), would like to offload the computation of a function f to another participant, the server or worker (or prover). Such a client may not trust the server, and would therefore want the server to "prove" that the computation was done correctly. Clearly, the complexity of verifying such a proof should be significantly lower than the complexity of running f, while at the same time, not significantly blowing up the running time of the prover. The applicability of delegation schemes goes beyond cloud computing. For example, efficient verification of computation is used as a building block in some widely used crypto currencies [BCG+14].

In a non-interactive delegation scheme, which is the focus of this work, the verifier computes and publishes a common reference string (CRS), and any prover can use this CRS to generate succinct proofs. We emphasize that in non-interactive delegation schemes soundness is guaranteed even if a cheating prover chooses the computation *adaptively* depending on the CRS.

Two types of such delegation schemes were considered in the literature: *publicly verifiable* and *privately verifiable*. In a publicly verifiable scheme, in order to verify a proof, all that is needed is the CRS (and the proof). In a privately verifiable scheme the verifier generates the CRS along with a secret key, and in order to verify a proof one needs to use the secret key, and the scheme remains sound as long as the secret key is hidden from the prover.

In this work, we construct a non-interactive *privately verifiable* delegation scheme for *non-deterministic* computations, that is sound under standard sub-exponential hardness assumptions. Specifically, assuming sub-exponentially secure succinct PIR, we construct a non-interactive, privately verifiable, delegation scheme for $\mathsf{NTISP}(\mathcal{T}(n), \mathcal{S}(n))$, which is the class of all non-deterministic computations requiring time $\mathcal{T}(n)$ and space $\mathcal{S}(n)$. The prover runs in time $\mathsf{poly}(\mathcal{T}(n))$ given a witness for the computation, and the communication complexity is $\mathsf{poly}(\mathcal{S}(n))$, and the verifier running time is $n \cdot \mathsf{polylog}(T(n)) + \mathsf{poly}(\mathcal{S}(n))$, where n denotes the instance length.[1] Similar to previous privately verifiable delegation schemes (for deterministic computations), our scheme is vulnerable to the "verifier rejection problem": A cheating prover that observes whether or not the verifier rejects a large number of maliciously crafted proofs, can completely learn the verifier's secret key and violate soundness.

Perspective: the challenge of non-determinism. The problem of constructing succinct delegation from standard complexity assumptions has been studied in a long sequence of works (see Section 1.1 for a comprehensive exposition on prior work). However, until our work, all known work based on standard assumptions either only worked for *deterministic* computations or for batch verification of multiple NP statements. Thus, the following question has been a guiding inspiration for research on delegation over the last several years:

Is it possible to obtain succinct non-interactive arguments (SNARGs) for delegating non-deterministic *computations, based on standard and well-studied assumptions?*

Unfortunately, Gentry and Wichs [GW11] showed that it is impossible to achieve delegation for all of NP generically under standard assumptions, via black-box proofs of security. Despite

[1] For simplicity of exposition, we assume here that $\mathsf{poly}(\mathcal{S}(n))$ is larger than the security parameter.

the negative result of Gentry and Wichs, we tackle succinct non-interactive delegation for non-deterministic computations from well-studied assumptions.

Evading the barrier of [GW11] for non-deterministic computations. While the Gentry-Wichs [GW11] barrier may make it appear that the situation is hopeless, our first observation is that, looking more closely, it is plausible to circumvent this barrier for many interesting non-deterministic computations.

Consider, for example, the subset-sum problem. An instance $x := (N, y_1, \ldots y_N, t)$, where each $y_i \in \{0, \ldots, 2^\ell\}$, is in the language $\mathsf{Subset.Sum}_\ell$ if and only if there exists $S \subseteq [N]$ such that $\Sigma_{i \in S} y_i = t$. Note that the length of the witness for $\mathsf{Subset.Sum}_\ell$ is N. However, this problem can be decided with non-deterministic space $O(\ell)$, because such a machine can non-deterministically decide whether to include each y_i in the partial sum, and updating the partial sum only takes space $O(\ell)$.[2] Since any language requiring non-deterministic space S can be decided in deterministic time 2^S, $\mathsf{Subset.Sum}_\ell$ is in $\mathsf{DTIME}(2^{O(\ell)})$. Now, looking more closely at the negative result of [GW11], we see that their result only implies that a delegation scheme from standard assumptions for $\mathsf{Subset.Sum}_\ell$ would require communication complexity roughly $\Omega(\ell)$. And yet, $\mathsf{Subset.Sum}_\ell$ is very interesting even when $\ell \ll N$. Indeed, even for values of $\ell = \mathsf{polylog}(N)$, subset-sum is believed to be outside of P. In these cases, matching the [GW11] lower bound would be a significant improvement over the trivial scheme where the prover sends the witness of size N. Therefore we ask whether it is possible to achieve delegation with complexity growing with $\mathsf{poly}(\ell)$ as opposed to growing with N?

We give a positive answer to this question. In fact, as already described above, we address the more general setting of languages in $\mathsf{NTISP}(\mathcal{T}(n), \mathcal{S}(n))$, and construct a non-interactive delegation scheme for such languages, with communication complexity $\mathsf{poly}(\mathcal{S}(n))$ and verifier runtime equal to $n \cdot \mathsf{polylog}(T(n)) + \mathsf{poly}(S(n))$. Note that before our work, there were no known constructions of such a delegation scheme, even under strong but "standard-type" assumptions such as indistinguishability obfuscation[3].

Beyond subset-sum, there are many examples of computations where the non-deterministic space complexity $\mathcal{S}$ is small. For example, verifying that a Merkle (or more generally any) tree hash value was correctly computed on some message requires space just $O(\kappa)$, where κ denotes the security parameter. We expand further on other interesting instantiations when discussing our results below.

1.1 Prior Work

There is a long body of work on delegating computation. The question of delegation of computation was first studied by Kilian [Kil92] and Micali [Mic94]. Kilian gave a four-message protocol for delegating NP computations, and Micali showed how to obtain a non-interactive delegation for NP in the random oracle model. Indeed, especially from the point of view of applications, *non-interactive* delegation protocols are significantly more interesting.

The works of [GKR08, RRR16] construct delegation schemes with *statistical soundness* (i.e., security against computationally unbounded cheating provers). These schemes are inherently interactive, and also inherently only handle bounded classes of computations. Specifically, the delegation scheme from [GKR08] is sound for bounded depth (deterministic) computation, and the

[2] As is standard when defining space complexity, we do not count the length of the input in the space usage of the machine.

[3] The work of [SW14] built such protocols assuming indistinguishability obfuscation, but they require a long CRS (as long as the witness). In this work, we focus on the truly succinct setting where both the CRS and proof are significantly smaller than the size of the input and witness.

delegation scheme from [RRR16] is sound for bounded space (deterministic) computation.

Non-interactive delegation schemes are inherently only *computationally sound*, i.e., soundness is only guaranteed against cheating provers that cannot break some hardness assumption. Indeed most previous work in the literature (as well as this work) focus on achieving computational soundness. These prior works can be roughly clustered into two sets.

Delegating non-deterministic computations based on non-standard assumptions. There is an extensive body of work (including [Gro10, Lip12, DFH12, GGPR13, BCI$^+$13, BCCT13, BCC$^+$14, BISW17]), achieving non-interactive, publicly verifiable, delegation for all non-deterministic computations under so called *knowledge assumptions*. These works get the strongest possible results at the price of relying on non-standard assumptions. Knowledge assumptions are of a different nature than standard complexity assumptions, as (similarly to the random oracle model) they restrict the class of adversaries considered to those that perform computations in a certain way.[4] We emphasize that currently we do not know how to construct such delegation schemes even under strong assumptions such as indistinguishability obfuscation (iO).

Delegating deterministic computations. The literature on delegation schemes for deterministic computations can be partitioned into two classes: Those that rely on *standard cryptographic assumptions* but are privately verifiable [KR09, KRR13, KRR14, KP15, DNR16, BHK17], and those that are publicly verifiable but rely on indistinguishability obfuscation [BGL$^+$15, CHJV15, CH16, CCHR15, KLW15, CCC$^+$16, ACC$^+$15], or on other non-standard assumptions (seemingly related to obfuscation) [PR17]. Our work follows the former line of works.

We emphasize that most of these works construct a *two-message* delegation scheme, where the first message does not depend on the instance chosen by the prover. The difference between this and a non-interactive delegation scheme (as also described in [BHK17]) is that the former only guarantees *non-adaptive* soundness, namely, soundness is guaranteed only if the prover first chooses the computation, and then succeeds in proving correctness for a random message (CRS) sent by the verifier. Non-interactive delegation guarantees *adaptive* soundness, even if the prover chooses the computation depending on the CRS. The only known works from this list that construct non-interactive delegation are [PR17, BHK17], where [PR17] relies on non-standard assumptions and [BHK17] relies on the existence of a succinct PIR scheme. Recently, [JKKR17] also constructed (non-succinct) arguments achieving variants of zero-knowledge in this setting. Our work follows the footsteps of [BHK17] in this regard and achieves adaptive soundness assuming a (sub-exponentially secure) succinct PIR scheme.

1.2 Our Results

Our main result is the following:

Informal Theorem 1. *Assuming sub-exponentially secure succinct PIR, we construct non-interactive, privately verifiable, delegation scheme for* $\mathsf{NTISP}(\mathcal{T}(n), \mathcal{S}(n))$*, where the prover runs in time* $\mathsf{poly}(\mathcal{T}(n))$*, the communication complexity is* $\mathsf{poly}(\mathcal{S}(n))$*, and the verifier running time is* $n \cdot \mathsf{polylog}(T(n)) + \mathsf{poly}(\mathcal{S}(n))$*, where* n *denotes the instance length.*[5]

We refer the reader to Theorem 1 in Section 5 for our formal theorem. As mentioned above, before our work, there were no known constructions of such a delegation scheme, even under

[4] For example, the Knowledge-of-Exponent assumption [Dam92] assumes that any adversary that given (g, h) computes (g^z, h^z), must do so by "first" computing z and then computing (g^z, h^z).

[5] We assume that $\mathsf{poly}(\mathcal{S}(n))$ is larger than the security parameter.

strong but "standard-type" assumptions such as indistinguishability obfuscation. Moreover, even 2-message delegation schemes for such languages were only known under knowledge assumptions.

More details on the barrier of [GW11]. Recall that the Gentry-Wichs [GW11] barrier shows that it is impossible to construct a succinct non-interactive delegation scheme for NP, and prove that it is sound via a black-box reduction to a standard assumptions. This result has left many researchers hopeless, and willing to settle for relying on (non-standard) knowledge assumptions. Surprisingly, we observe that it is possible to circumvent this barrier for many interesting non-deterministic computations.

In particular, the high-level idea behind the [GW11] impossibility result, is that a reduction that breaks the assumption, cannot distinguish between pairs (x, π) generated by a (possibly inefficient) cheating prover, where $x \notin L$ and π is a proof of length ℓ, and a pair $(\tilde{x}, \tilde{\pi})$ where $\tilde{x} \in L$ and $\tilde{\pi}$ is an efficiently generated proof. This is true, assuming the underlying NP language is 2^{ℓ}-hard.

We note that any computation in $\mathsf{NTISP}(\mathsf{poly}(n), \mathcal{S}(n))$ is also in $\mathsf{DTIME}\left(\mathsf{poly}(n) \cdot 2^{\mathcal{S}(n)}\right)$, and hence is not $\mathsf{poly}(n) \cdot 2^{O(\mathcal{S}(n))}$-hard. Therefore, the [GW11] does not rule out the possibility of a non-interactive delegation scheme where the proofs are of length $\mathsf{poly}(\mathcal{S}(n))$, which is exactly what we show is possible. The [GW11] result does show that our results are somewhat tight for this complexity class.

On the other hand, the result of [GW11] still leaves open the possibility that for any computation in $\mathsf{DTIME}(2^{\ell}) \cap \mathsf{NP}$, it is plausible to achieve an argument system with efficiency $\mathsf{poly}(\ell)$. Our result achieves this for an important subclass: $\mathsf{NTISP}(\mathit{poly}(n), \ell)$. An interesting question for future work is whether it is possible to go beyond this class or close the gap entirely.

Remark. As noted above, since $\mathsf{NTISP}(\mathcal{T}(n), \mathcal{S}(n))$ is contained in $\mathsf{DTIME}\left(\mathcal{T}(n) \cdot 2^{\mathcal{S}(n)}\right)$, if we were not concerned with the running time of the prover, it would have been possible to use [KRR14, BHK17] to achieve succinctness and verifier runtime according to Informal Theorem 1, but with prover running time as high as $\mathsf{poly}(\mathcal{T}(n) \cdot 2^{\mathcal{S}(n)})$. A key challenge, that is addressed by our work, is to maintain the honest prover's running time to be $\mathsf{poly}(\mathcal{T}(n))$, while obtaining succinctness. We stress that retaining the prover's running time as close as possible to $\mathcal{T}(n)$ is critical for applications.

Natural languages in $\mathsf{NTISP}(\mathcal{T}(n), \mathcal{S}(n))$. We observe that the class $\mathsf{NTISP}(\mathcal{T}(n), \mathcal{S}(n))$, where the non-deterministic space complexity $\mathcal{S}(n)$ is small compared to $\mathcal{T}(n)$, contains many natural NP languages. These include problems in NP such as subset-sum for interesting parameter settings, as well as other languages that admit dynamic programming solutions such as the knapsack problem and the partition problem. Another natural NP relation in this class is verifying tree-based computations, such as checking if a Merkle hash value was correctly computed on some message. This requires space proportional only to the depth of the tree. More generally, the class NSC^k, which is equal to $\mathsf{NTISP}(\mathsf{poly}(n), \log^k(n))$, for $k > 2$ is not known to be in P and contains interesting problems including special variants of SAT. We also note that batch verification of multiple NP statements requires space proportional to that required for verification of one statement. Thus, in particular, we get an *adaptive non-interactive* delegation scheme for batch NP (assuming sub-exponentially secure PIR scheme). This is in contrast to the work of [BHK17], which constructed a *2-message* (non-adaptive) delegation scheme for batch NP assuming a polynomially secure PIR scheme. We refer the reader to Appendix A for details.

1.3 Brief Overview of Our Techniques

We begin with a very high level overview of the main new ideas in our work. In the next section, we explain our techniques in more detail.

Our work builds upon, but fundamentally departs from, the ideas of [KRR13, PR17, BHK17]. Loosely speaking, these works show that if there exists a (possibly cheating) prover that convinces the verifier to accept a computation, say $M(x) = 1$, then the prover can be converted into a "local assignment generator" on the intermediate values of the computation. The local assignment generator, with locality ℓ (which depends on the communication complexity of the delegation scheme), takes as input a set of ℓ intermediate wires of the computation, and outputs an assignment to the results of these intermediate wires, such that the assignment is locally consistent.

All prior work, implicitly or explicitly, first convert any cheating prover that generates accepting proofs in the underlying delegation protocol, into a local assignment generator, and then argue that the existence of a local assignment generator implies that the computation must be correct.

Prior works on delegating deterministic computation argued soundness, while only relying on low locality of the assignment generator. Loosely speaking, this was done by inductively arguing that when such a local assignment generator is invoked repeatedly on different subsets of wires, all the wires must be assigned their **correct** value with overwhelming probability.

We emphasize that for deterministic computations, the notion of "correctness" of a wire assignment with respect to a fixed input x, is well defined. However, we deal with **non-deterministic** computations, and hence *there may be many non-deterministic choices that make many assignments correct, but these assignments are inconsistent with each other.* This is a problem because these conflicting non-deterministic choices cannot be pieced together; one set of correct non-deterministic choices may be incompatible with another set of correct non-deterministic choices.

Due to this discrepancy between deterministic and non-deterministic computations, it was believed that this blueprint is only applicable to deterministic computations. This is supported by the known result that only *deterministic* languages, computable in time T, have MIPs with no-signaling soundness and polylog(T) provers. Nevertheless in this work, we do use this blueprint for non-deterministic computations. For languages computable in non-deterministic time T and space S, we achieve no-signaling MIPs with $\mathsf{poly}(S, \log T)$ provers, where the provers given the witness, run in time $\mathsf{poly}(T)$.

To this end, we give up entirely on trying to piece together different computations to argue about the correctness of one global computation, as was done in all previous work. Instead, we use a special property of space-$\mathcal{S}$ non-deterministic computations: that in fact, in time $2^{O(\mathcal{S})}$, it is possible to determine whether a particular intermediate configuration of the computation, of size $O(\mathcal{S})$, can ever lead to an accepting state, regardless of what non-deterministic choices were made before or after reaching this intermediate configuration. We call such an intermediate configuration an *accepting configuration.* We then use cryptographic tools to argue that our protocol ensures that every accepting intermediate configuration must be preceded by another accepting intermediate configuration.

Notably, we use the fact that it is possible to check whether a configuration is accepting in time $2^{O(\mathcal{S})}$ only in a "mental experiment" within the proof of soundness; the actual running time of the honest prover is not impacted. We are able to argue via induction that a cheating prover can only cause the verifier to accept if in fact there exists a sequence of non-deterministic choices that would have caused the computation to accept.

We describe this in much more detail, but still informally, in the following section, where we also point out other bottlenecks that we must overcome to achieve our result.

2 Detailed Technical Overview

We now provide a more detailed overview of our main techniques. We begin by discussing known approaches to delegating computation, and the key bottlenecks in previous work.

2.1 Prior Work and Key Bottlenecks

Our work is based on the heuristic suggested by [BMW98], that uses any multi-prover interactive proof (MIP) scheme and any succinct PIR scheme to construct a 2-message delegation scheme. For simplicity, in this overview we use a special kind of succinct PIR – a *fully homomorphic encryption* (FHE) scheme. Such a scheme allows to perform homomorphic computations on ciphertexts, allowing the transition $\mathsf{FHE}_{\mathsf{pk}}(x) \to \mathsf{FHE}_{\mathsf{pk}}(f(x))$ with computational complexity proportional to that of f.

The model of multi-prover interactive proofs was introduced by Ben-Or et. al. [BGKW88]. In this model, the verifier interacts with two (or more) provers, it sends each prover a query and each prover replies with an answer. Importantly, it is assumed that the provers do not communicate during the protocol, so that each answer depends only on the corresponding query. Intuitively, this can be enforced by placing the provers in different rooms (without any connection to the outside world). This proof model was proven to be extremely powerful. Babai, Fortnow and Lund [BFL91] proved that any proof of length T can be converted into a 2-prover interactive proof where the two queries and two answers are of length $\mathsf{polylog}(T)$. Namely, [BFL91] proved that $\mathsf{MIP} = \mathsf{NEXP}$.

Thus, if we were willing to assume the existence of two non-communicating provers, then we could use these results from the early 90's to construct a delegation scheme, where the client interacts with two servers, and soundness is ensured as long as these two servers do not interact during the proof process. However, we do not want to make such an assumption, since in many applications (such as for crypto-currencies) this is not a realistic assumption, and for other applications (such as cloud computing) the non-communicating assumption may be too strong, or at the very least simply expensive.

The [BMW98] heuristic uses cryptography to emulate two (or more) non-communicating provers using a single prover. This heuristic is simple and elegant: First the verifier generates the queries for the MIP provers, and encrypts each query independently under a different (independently generated) public key, and sends all the encrypted queries to the (single) prover. Then, the prover computes the answer to each query homomorphically, and sends all the encprypted answers to the verifier. The verifier decrypts these answers, and accepts if the MIP verifier would have accepted these answers.

The intuition for why this heuristic was believed to be sound is the following: When a cheating prover answers each of the queries, the other queries are encrypted using different (independently generated) keys, and hence are indistinguishable from encryptions of 0 (by the security of the FHE scheme). Therefore, each answer should be indistinguishable from the answer the cheating prover would have provided in the case where the other queries were all 0, and clearly having encryptions of 0 cannot help a prover cheat, since he can generate these encryptions on his own.

Surprisingly, despite this intuition, Dwork et. al. [DLN$^+$01] showed that this heuristic, in general, can be insecure. The reason is that the soundness of the MIP is ensured only against cheating provers that answer each query *locally*, only as a function of the corresponding query. In this delegation scheme a cheating prover is not restricted to use local strategies. Rather the security of the FHE scheme ensures that each answer (provided by a cheating prover) does not "signal" information about the other queries, since if it did then we could use this prover to break the security of the FHE scheme.

However, there are strategies that are neither signaling nor local. Dwork et. al. [DLN+01] refer to such strategies as "spooky interactions". Such strategies are known in the quantum literature as *no-signaling* strategies. The intuition above suggests that these no-signaling strategies are useless. However, in the quantum literature it is well known that this is not the case.

Very recently, [DHRW16] showed that indeed the [BMW98] heuristic is insecure! Specifically, they construct an MIP scheme and a FHE scheme, for which when applying the [BMW98] heuristic to these MIP and FHE schemes, the resulting delegation scheme is not sound. To this end, they construct an MIP scheme whose soundness can be broken via a no-signaling strategy, and this no-signaling strategy can be implemented under the layer of the FHE.

Yet, [KRR13] and other follow-up works, showed that the approach of [BMW98] can be proven sound if the underlying MIP is sound against (statistically) *no-signaling provers*. Such provers are not restricted to answering each query locally (as required by classical MIPs), rather each answer can be a function of all the queries, as long as the answer does not *signal* information about the other queries. More formally, the requirement is that for any subset of queries, the marginal distribution of the answers to this subset of queries, are (statistically) independent of the other queries. The work of [KRR13] consists of two parts:

1. First, they show that any prover that cheats in the delegation scheme (constructed via the [BMW98] heuristic) with *noticeable* probability, can be converted into a (statistically) no-signaling prover that cheats in the underlying MIP with *noticeable* probability. This step involves relying on the underlying FHE scheme to guarantee the no-signaling property.

2. Second, they construct an MIP scheme with soundness against (statistically) no-signaling cheating provers.

This approach was later explored further in context of public verifiability by [PR17], and extended (in the privately verifiable setting) to RAM computations and to non-interactive (as opposed to 2-message) delegation by [KP15, BHK17].

An immediate barrier for non-deterministic computations. It appears that this approach is doomed to fail for general non-deterministic computations, since it is known that MIP's with no-signaling soundness (with polynomial time verifiers and polynomial communication complexity) exist only for languages in EXP. We show a new approach that can nevertheless be made to work for low-space non-deterministic computation. To unravel how this is achieved, we give some more details about the overall approach used by prior work, and how we (fundamentally) deviate from it. Many of these works relied on an abstraction developed by [PR17], which abstracts techniques from [KRR13], known as *a local assignment generator*. We give an informal description of this primitive, since it is relevant for our work.

Local Assignment Generator. For any (fixed) Turing machine M that decides a language, it will be useful to think of a circuit C^M (parameterized appropriately by input length, which we skip here for simplicity), as a tableau or layered circuit that describes the evolution of the computation of M. For any input $\{0,1\}^n$, an ℓ-local assignment generator for the computation $M(x) = y$, is an oracle Turing machine that on input a subset of wires for the circuit C^M, outputs assignments of values to these wires. This generator is required to output assignments to wires that satisfy the following properties:

- **Everywhere ℓ-Local Consistency.** The wire assignments output by C^M to any subset of at most ℓ wires, must be locally consistent with each other, and consistent with the input x and the output y, with overwhelming probability.

- **No-Signaling.** For any two subsets of wires Q_1, Q_2, let $Q := Q_1 \cap Q_2$. Then, the *marginal* distribution of assignments to wires in Q when the assignment generator is queried on Q_1 should be indistinguishable from the *marginal* distribution of assignments to wires in Q when the assignment generator is queried on Q_2.

  If these two distributions are computationally indistinguishable then the local assignment generator is said to satisfy the *computational* no-signaling criterion. If these two distributions are T-indistinguishable (i.e., indistinguishable by circuits of size $\mathsf{poly}(T)$) then we say that the local assignment generator satisfies the T-no-signaling criterion.

In all prior work in this line or work, the analysis (implicitly or explicitly) consists of converting any (possibly cheating) prover that generates an accepting proof in the underlying delegation scheme (with non-negligible probability), into a local assignment generator (for that computation), where the locality ℓ of the local assignment generator is proportional to the communication complexity of the delegation scheme. It is useful to think of the communication complexity as being polynomial in the security parameter κ (we will elaborate more on this later): note that the size of the instance and the tableau are allowed to be much larger than κ, and in particular can be as large as 2^κ.

Deriving Soundness via the Local Assignment Generator. The existence of a local assignment generator with locality equal to the size of the entire tableau naturally implies soundness. However, such high locality would blow up the communication complexity, and the corresponding delegation protocol would no longer be succinct.

On the other hand, ℓ-local consistency (for small ℓ) only guarantees that assignments to ℓ-sized subsets of wires are consistent with respect to each other; however, it may be possible that an entire subset of wires could be set incorrectly and yet consistently with each other (note that intuitively the verifier can only check consistency, not correctness). Nevertheless, the works of [KRR13, KRR14, KP15] argue inductively, that local consistency, together with no-signaling, actually implies correctness of *all wire assignments*[6]. We note that to this end, the works of [KRR14, KP15] had to change the tableau of the computation C^M; indeed in [KRR14] a low degree extension was added to each layer in C^M (this was referred to as the "augmented circuit"), and in [KP15] Merkle hashes were added to each layer of the computation.

The work of [BHK17] extends this to the adaptive setting, where the instance x is output by the prover and therefore by the local assignment generator based on the verifier message. In this setting, the instance x may change each time the local assignment generator is queried. These works show that it is possible to convert an (adaptive) prover for the delegation scheme that outputs accepting proofs for $x \notin L$ with noticeable probability, to an (adaptive) local assignment generator that outputs $x \notin L$ together with assignments, that are no-signaling and locally consistent with overwhelming probability.

The Key Barrier to Non-Determinism. This approach breaks down completely when trying to extend it to non-deterministic computations. The first reason is that it is no longer clear how to argue or even define "correctness" of wire assignments any more. Indeed, it is known that local consistency does not imply global consistency in this setting.

For non-deterministic computations, it may be tempting to consider defining correctness of wire assignments with respect to both the instance x and the witness w. However, even when the instance x is fixed in advance, the purported witness w being used by the prover is unknown and can change every time the prover is queried! This witness is too large to send in its entirety, and

[6] This is also referred to as global consistency.

hence is not well defined, and in particular, one cannot define correctness of an assignment with respect to this (undefined) witness.[7]

At a more intuitive level, the key issue is the following: When the circuit C^M requires reading the i^{th} bit of the witness w at two different steps in the computation, the no-signaling ℓ-local assigner that invokes a cheating prover, may output different assignments to w_i every time it is queried. It is possible that these assignments are always locally consistent with neighboring wires, and yet when assigning values to the global circuit, both steps in the computation are assigned different values for w_i. The local assignment generator therefore does not output a globally consistent assignment to C^M, and it is no longer clear that soundness holds. This presents an avenue for attack by the adversary that existing techniques did not know how to overcome.

2.2 New Techniques

Our first observation is that if the non-deterministic circuit C^M reads each bit of the witness only once, then the intuitive attack described in the previous paragraph is no longer valid. Armed with this observation, we consider the class $\mathsf{NTISP}(\mathcal{T}(n), \mathcal{S}(n))$ of all languages recognizable by non-deterministic Turing Machines in time $O(\mathcal{T}(n))$ and space $O(\mathcal{S}(n))$. Recall that a non-deterministic Turing Machine allows each step of the computation to non-deterministically transition to a new state. Thus, in a sense, this corresponds to the setting where each bit of the witness is read at most once.[8] An alternative way to describe this class is as the class of languages L with a corresponding witness relation R_L, recognizable by a deterministic Turing Machines with access to an input tape and a read-only, *read-once* witness tape, in addition to a work tape where only $O(\mathcal{S}(n))$ space is used, and that runs in $O(\mathcal{T}(n))$ time. Any such Turing Machine M can be converted into a layered circuit $C^M_{n,m}$, parameterized by $n = |x|$ and $m = m(n) = |w|$, that on input a pair (x, w) outputs 1 if and only if $R_L(x, w) = 1$. Each layer of gates in this circuit has input wires that directly read the instance, or directly read the witness, or are the output wires of gates in the previous layer. Moreover, each bit of the witness is read by at most one layer. This circuit has depth $D = O(\mathcal{T}(n))$ and width $W = O(\mathcal{S}(n))$, where W may be smaller than n.

This corresponds to *exactly* the type of circuit we described above – where no two different layers read the same bit of the witness. We next describe why for such computations, local consistency implies global consistency, assuming the locality is as large as the space, and describe the technical hurdles that we encounter along the way.

From Local Consistency to Soundness. As already noted, any Turing Machine M that recognizes languages in $\mathsf{NTISP}(\mathcal{T}(n), \mathcal{S}(n))$ can be represented by a layered circuit $C^M_{n,m}$ that takes input (x, w) where $|x| = n, |w| = m$. Moreover, $C^M_{n,m}$ has depth $D = O(n \cdot \mathcal{T}(n))$ and width $W = O(\mathcal{S}(n))$, and is such that every layer of gates obtains input wires that either directly read the instance, or directly read the witness, or are output wires from a previous layer. Finally, each bit of the witness is read (directly) by at most one layer.

As mentioned above, it is not clear how to define "correctness" of a wire assignment in this (non-deterministic) setting. We therefore give up on trying to formalize any notion of "correctness" of wire assignments. Our key insight is that instead we can define the notion of an "accepting layer".

[7] This can be overcome if the witness is hashed (using a Merkle Hash) *in advance*; see [KP15].

[8] If a non-deterministic Turing Machine wishes to remember what non-deterministic choices it made, it has to write them down to its work tape.

Defining Accepting Layers. We define the notion of an accepting layer inductively, starting from the output layer.[9]

One can define a layer in terms of the gates in that layer, or in terms of the wires that are input to the gates in that layer. We choose the latter. In particular, the output layer consists only of the output wire, and thus the only valid assignment for this layer is the symbol 1. For each layer i, we partition the wires that are input to gates in layer i into three sets: intermediate wires, instance wires, and witness wires. Intermediate wires for layer i are all the wires connecting gates in layer $(i-1)$ to gates in layer i; instance wires for layer i are all wires that directly read the instance x and are input to gates in layer i; and witness wires for layer i are all wires that directly read the witness and are input to gates in layer i.

We define Acc_x^D as the set of all possible assignments to *intermediate* wires connecting a gate in layer $(D-1)$ to a gate in layer D, such that when the instance wires for layer D are set consistently with x, *there exists some assignment to the witness wires for layer* D, such that the transition function applied to these wires results in output 1. For each layer $i < D$, the set Acc_x^i is defined recursively in a similar manner. That is, for $i < D$, Acc_x^i is the set of all possible assignments to intermediate wires connecting gates in layer $(i-1)$ to gates in layer i, such that when the instance wires for layer i are set consistently with x, *there exists an assignment to the witness wires for layer* i, such that the transition function applied to these wires outputs intermediate wires for layer $i+1$ that lie in the set Acc_x^{i+1}.

We note that the lowest i for which this definition is meaningful is $i = 2$, since there are no intermediate wires before the first layer. Moreover, by definition, for every $i \geq 2$, and for any assignment to the intermediate wires for layer i, such that the assignment is in Acc_x^i, it holds that there exists a (partial) assignment to the witness such that this layer i configuration, on the input x and on the (partial) witness, proceeds to an accepting output configuration. This follows from the fact that each bit of the witness is read by at most one layer, and hence the bits of the witness read in each layer can be pasted together in a consistent manner, to lead a configuration in Acc_x^i to an accepting output configuration, as desired.

This fact leads to the following important observation: Suppose the local assignment generator, when queried on all input wires to layers 1 and 2 (simultaneously), outputs an assignment that satisfies the following properties (with overwhelming probability):

- The assignment is locally consistent.
- The assignment to intermediate wires in layer 2 is in Acc_x^2.

Then there must exist an accepting witness for x. This follows from the following simple analysis: The local consistency implies that there exists a partial witness that leads from the input layer to layer 2 in Acc_x^2. As noted above, by the definition of Acc_x^2, there is a partial witness that leads from the layer 2 assignment in Acc_x^2 to the output 1 in the final layer. The fact that each bit of the witness is read by at most one layer, implies that these two partial witnesses can be pasted together in a consistent manner, to lead from the input layer to the output layer being 1, as desired.

It thus remains to prove that the local assignment generator indeed outputs a locally consistent assignment in Acc_x^2 (with overwhelming probability). The local consistency property is ensured by the definition of a local assignment generator. We thus focus on proving that the assignment to the intermediate wires for layer 2 is in Acc_x^2 (with overwhelming probability).

[9] Here again, our approach departs from all prior works in that we must perform induction top down (as opposed to previous works which perform the induction bottom up).

This follows from the crucial observation that for any width W and depth D computation, it is possible to decide whether a set of wire assignments are in Acc^i_x, for any $i \in [D]$ in time $\mathsf{poly}(D, 2^W)$. This is done via a straightforward dynamic programming approach.

Moreover, crucially, we ensure that the local assignment generator we obtain from the cheating prover is $\mathsf{poly}(D, 2^W)$-no-signaling, so that when querying the local assignment generator with the intermediate wires for layer i, possibly along with other queries, whether the assignment to these intermediate wires is in Acc^i_x or not is *independent* of the other queries. This is ensured by assuming that the underlying PIR (or FHE) scheme is $\mathsf{poly}(D, 2^W)$-secure, which in turn is done by setting the security parameter to be $\kappa = \mathsf{poly}(W, \log D)$, and assuming sub-exponential security of the PIR (or FHE) scheme.

Given this, we can prove the more general claim that for every $i \in [D]$ when querying the local assignment generator on all intermediate wires for layers i (possibly along with other queries) then it outputs an assignment in Acc^i_x (with overwhelming probability). We prove this by backward induction.

- In the base case, this holds for layer D just by local consistency between the input wires of layer D and the output of the circuit.
- Next, assume by induction that the local assignment generator, when queried on intermediate wires for layer $i+1$ (possibly along with other queries), outputs an assignment that is in Acc^{i+1}_x (with overwhelming probability).
- Query the local assignment generator on all the wires for i and all the wires for layer $i+1$ (simultaneously). By our induction hypothesis, the assignment generator outputs an assignment to the intermediate wires for layer $i+1$ that is in Acc^{i+1}_x (with overwhelming probability).
- Next, by local consistency of the assignment generator, the assignments to the wires for layer i must be consistent with the assignment for the wires for layer $(i+1)$, with respect to some witness assignment for this layer and with respect to the instance x. Thus, if the assignment in layer $(i+1)$ is accepting, the assignment in layer i must also be in Acc^i_x (with overwhelming probability).
- It remains to note that by the $\mathsf{poly}(D, 2^W)$-no-signaling property, the assignment for the intermediate wires for layer i are in Acc^{i+1}_x when queried with *any* wires.

This completes a rough overview of our induction strategy. Our actual proof requires additional care, including handling adaptive choices of instance x, and ensuring that negligible factors in the induction do not grow exponentially. A detailed exposition is provided in upcoming sections. In Section 3, we describe basic notation and primitives that we use, and in Section 4 we state our new definitions (and notations). In Section 5, we state and prove our formal theorem.

3 Preliminaries

We begin by introducing some notation. Throughout this paper, we denote the security parameter by κ. For any function $\mathbf{T}(\kappa)$ we denote by $\mathsf{negl}(\mathbf{T}(\kappa))$ any function that vanishes faster that $\frac{1}{\mathsf{poly}(\mathbf{T}(\kappa))}$ for every large enough κ.

Definition 1. *Two distribution ensembles $\{X_\kappa\}_{\kappa\in\mathbb{N}}$ and $\{Y_\kappa\}_{\kappa\in\mathbb{N}}$ are said to be* $\mathbf{T}$-indistinguishable *if for every distinguisher $\mathcal{D}$ of size $\mathsf{poly}(\mathbf{T}(\kappa))$,*

$$|\Pr_{x\leftarrow X_\kappa}[\mathcal{D}(x) = 1] - \Pr_{y\leftarrow Y_\kappa}[\mathcal{D}(y) = 1]| = \mathsf{negl}(\mathbf{T}(\kappa)).$$

The following definition of succinct PIR is taken from [BHK17].

Definition 2 (Succinct PIR)**.** *A succinct 2-message PIR scheme is a tuple of PPT algorithms* $(\mathsf{PIR.Send}, \mathsf{PIR.Respond}, \mathsf{PIR.Decode})$ *where:*

- $(q, s) \leftarrow \mathsf{PIR.Send}(1^\kappa, i)$*: Given* 1^κ *and* $i \in [2^\kappa]$*, outputs a query string* q *and secret state* s*.*
- $a \leftarrow \mathsf{PIR.Respond}(q, D)$*: Given a query string* q *and database* $D \in \{0,1\}^{\leq 2^\kappa}$*, outputs a response string* a*.*
- $x \leftarrow \mathsf{PIR.Decode}(1^\kappa, s, a)$*: Given an answer* a *and state* s*, outputs an element* $x \in \{0,1\}^\kappa$*.*

These algorithms form a succinct PIR if they satisfy the succinctness, correctness and security properties described below:

- *The scheme is succinct if* $|a| = \mathsf{poly}(\kappa)$*.*
- *The scheme is (perfectly) correct if for every* $i \leq 2^\kappa$ *and every* $D \in \{0,1\}^{\leq 2^\kappa}$ *with* $|D| \geq i$*, when setting* $(q, s) \leftarrow \mathsf{PIR.Send}(1^\kappa, i), a \leftarrow \mathsf{PIR.Respond}(q, D)$ *and* $x \leftarrow \mathsf{PIR.Decode}(1^\kappa, s, a)$*, then* $x = D[i]$ *with probability* 1*.*
- *The scheme is* $\mathbf{T}(\kappa)$*-secure if for any* $i, i' \in [2^\kappa]$*, for* $(q, s) \leftarrow \mathsf{PIR.Send}(1^\kappa, i)$ *and* $(q', s') \leftarrow \mathsf{PIR.Send}(1^\kappa, i')$*, it holds that* q *and* q' *are* $\mathbf{T}$*-indistinguishable.*

Delegating Non-Deterministic Computation We now describe our model for private-key non-interactive delegation for non-deterministic computations.

Our model is similar to prior work, and in particular is similar to [BHK17], except that we consider the Turing machine model of computation instead of the RAM model from [BHK17]. In our protocols, we will fix a (universal) non-deterministic Turing machine M and its maximum running time $\mathcal{T}$ ahead of time for simplicity. As we describe later, this can be removed without loss of generality. Below, we describe the syntax of a private-key delegation scheme.

A non-interactive delegation scheme corresponding to a fixed non-deterministic Turing machine M consists of PPT algorithms, $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Setup}(1^\kappa) \to (\mathsf{pp}, \mathsf{sk})$: A PPT algorithm that takes as input 1^κ, and outputs public parameters pp and secret key sk.
- $\mathsf{Prove}(1^\kappa, \mathsf{pp}, x, w) \to \pi$: A deterministic algorithm that takes input public parameters pp, runs in time $\mathsf{poly}(\mathcal{T}(|x|), \kappa)$ and outputs a proof π that Turing machine M on input x outputs 1 within $\mathcal{T}$-time steps.
- $\mathsf{Verify}(1^\kappa, \mathsf{pp}, \mathsf{sk}, x, \pi) \to b$: A deterministic algorithm that outputs an acceptance bit b.

4 Definitions

The Class NTISP. $\mathsf{NTISP}(\mathcal{T}(n), \mathcal{S}(n))$ is defined as the class of languages accepted by non-deterministic Turing machines taking time $\mathcal{T}(n)$ and using space $\mathcal{S}(n)$. Alternately, we can think of it as the class of languages that can be verified in deterministic time $\mathcal{T}(n)$ and space $\mathcal{S}(n)$ with read-once access to the witness tape, on any input of length n.

Fix any $L \in \mathsf{NTISP}(\mathcal{T}(n), \mathcal{S}(n))$. Denote by $\mathcal{R}_L$ its corresponding NP relation, and denote by $M = M_L$ a $\mathcal{T}(n)$-time $\mathcal{S}(n)$-space (non-deterministic) Turing machine for deciding L. We can think of M as a deterministic two-input Turing machine, that takes as input a pair (x, w) and outputs 1 if and only if $(x, w) \in \mathcal{R}_L$.

Corresponding Layered Circuit $\mathcal{C}^M_{n,m}$. Any such Turing machine M can be converted into a layered circuit, denoted by $\mathcal{C}^M_{n,m}$, which takes as input a pair (x, w), where $n = |x|$ and $|w| = m = m(n)$ (where $m(n)$ is an upper bound on the length of a witness corresponding to a length n instance), and outputs 1 if and only if $M(x, w) = 1$.

Moreover, $\mathcal{C}^M_{n,m}$ is a layered circuit, with $W = O(\mathcal{S}(n))$ denoting the maximum of the number of gates and number of wires in each layer, and depth $D = O(\mathcal{T}(n))$, such that a child of a gate in layer $i+1$ is either an input gate (or a negation of an input gate), or a witness gate (i.e., a gate corresponding to the witness part of the input), or a gate in layer i. Moreover, any witness gate has fan-out 1 (this corresponds to read-once access to the witness tape). In addition, there is a deterministic Turing machine of space $O(\log \mathcal{T})$ that on input n outputs the (description of the) circuit $\mathcal{C}^M_{n,m}$.

Notation for Wires of $\mathcal{C}^M_{n,m}$. We introduce some detailed notation for the wires of $\mathcal{C}^M_{n,m}$.

We call all wires that are inputs to gates in layer i, the wires for layer i. The set of wires for layer i is denoted by $\mathbf{q}^i$, and a set of assignments to these wires is denoted by $\mathbf{a}^i$. The j^{th} wire in layer i is denoted by q^i_j, and a (boolean) assignment to this wire is denoted by a^i_j.

We partition the wires for layer i into three sets, denoted by $\mathsf{Instance}^i, \mathsf{Witness}^i, \mathsf{Intermediate}^i$, where $\mathsf{Instance}^i$ is the set of all wires for layer i that read the instance x, $\mathsf{Witness}^i$ is the set of all wires for layer i that read the witness w, and $\mathsf{Intermediate}^i$ is the set of remaining wires for layer i which are output wires of gates in layer $(i-1)$.

Notation. In what follows, for simplicity we abuse notation: for two vectors $\mathbf{q} = (q_1, \ldots, q_\ell)$ and $\mathbf{q}' = (q'_1, \ldots, q'_{\ell'})$, we denote by $\mathbf{q}' \subseteq \mathbf{q}$ the fact that for all $i \in [\ell']$, there exists j such that $q'_i = q_j$.

Definition 3 (Adaptive Local Assignment Generator)**.** *We let $\{\mathcal{C}_{n,m}\}$ denote a family of circuits that take two inputs (x, w), where $|x| = n, |w| = m = m(n)$, and $Q[n]$ denote the set of all wires of $\mathcal{C}_{n,m}$.*

For any $\mathbf{T} = \mathbf{T}(\kappa)$, $n = n(\kappa)$, and $\ell_{\max} = \ell_{\max}(\kappa)$, a $(\mathbf{T}, n, \ell_{\max})$-adaptive local assignment generator Assign *for the family of circuits $\{\mathcal{C}_{n,m}\}$ is a probabilistic Turing machine that takes as input a security parameter 1^κ, and a tuple of wire-identifiers $\mathbf{q} \in [Q(n)]^\ell$ of $\mathcal{C}_{n,m}$'s wires, where $n = n(\kappa)$ and $\ell \leq \ell_{\max}(\kappa)$, and outputs an input $x \in \{0,1\}^n$ and assignments $\mathbf{a} \in \{0,1\}^\ell$, that satisfy the following properties:*

- **Everywhere $(\ell_{\max}, \mathbf{T})$-Local Consistency.** *For every security parameter 1^κ, and any vector of wires $\mathbf{q} = (q_1, \ldots q_\ell) \in [Q(n)]^\ell$, where $n = n(\kappa)$ and $\ell \leq \ell_{\max}(\kappa)$, with probability $1 - \mathsf{negl}(\mathbf{T}(\kappa))$ over a draw:*

$$(x, \mathbf{a} = (a_1, \ldots a_\ell)) \leftarrow \mathsf{Assign}(1^\kappa, \mathbf{q}),$$

the assignment $\mathbf{a}$ is locally consistent with the computation of $\mathcal{C}_{n,m}$ on input $(x, \cdot)$. That is, for all $i, j, k \in [\ell]$:

1. *If $q_i = q_j$, then $a_i = a_j$.*
2. *If q_i, q_j are the input wires of an AND gate and q_k is its output wire, then $a_k = a_i \cdot a_j$.*
3. *If q_i is an input wire of a NOT gate and q_j is its output wire, then $a_j = 1 - a_i$.*
4. *If q_i is an input wire in $\mathcal{C}_{n,m}$ then the value a_i is consistent with x.*
5. *If q_i is the output wire of $\mathcal{C}_{|x|,m(|x|)}$ then $a_i = 1$.*

- **T-Computational No-signaling.** *For every security parameter 1^κ, every $\ell \leq \ell_{\max}$, every wire-vector $\mathbf{q} = (q_1, \ldots q_\ell) \in [Q(n)]^\ell$, every $\ell' \leq \ell$ and $\mathbf{q}' \triangleq (q'_1, \ldots q'_{\ell'})$ such that $\mathbf{q}' \subseteq \mathbf{q}$, the distributions:*

$$(x', \mathbf{a}') \text{ and } (x, \mathbf{a}_{|\mathbf{q}'})$$

are $\mathbf{T}$-indistinguishable (according to Definition 1), over the randomness of sampling $(x, \mathbf{a}) \leftarrow \mathsf{Assign}(1^\kappa, \mathbf{q})$ and $(x', \mathbf{a}') \leftarrow \mathsf{Assign}(1^\kappa, \mathbf{q}')$.

From Succinct Delegation to Adaptive Local Assignment Generators. As in [BHK17], we fix a Turing machine M and a time bound $\mathcal{T}$, and consider a succinct non-interactive delegation scheme with the following syntax: the verifier V on input 1^κ computes a pair $(\mathsf{pp}, \mathsf{sk})$ and outputs pp to the prover P. Next, the prover P on input $(1^\kappa, \mathsf{pp})$ outputs an instance x together with a proof π that Turing machine M on input x outputs 1 within $\mathcal{T}$-time steps. The verifier, upon receiving a pair (x, π) from the prover, uses his secret key sk, to compute the output 0 or 1, denoting whether or not the proof was accepted.

Imported Theorem 1. *[KRR14, BHK17] Fix any (possibly non-deterministic) Turing machine M, and fix functions $n = n(\kappa) \leq 2^\kappa$ and $\mathcal{T} = \mathcal{T}(n)$ such that $\mathcal{T}(n(\kappa)) \in [\max\{n, \kappa\}, 2^\kappa]$, and any locality parameter $\ell = \ell(\kappa) \in [\kappa, \mathcal{T}(n)]$. Fix any function $\mathbf{T} = \mathbf{T}(\kappa) \in [\kappa, 2^\kappa]$ and assume the existence of a succinct $\mathbf{T}$-secure succinct PIR scheme. Then there exists a (succinct) non-interactive argument (P, V) for proving that M, on input x of length n, outputs 1 within $\mathcal{T}(n)$ steps, such that for any security parameter κ the following holds:*

- *The communication complexity[10] of (P, V) on input 1^κ is $\ell \cdot \mathsf{poly}(\kappa)$.*
- *The runtime of V, on input 1^κ and upon receiving an instance x of length n from the prover, is $n \cdot \mathsf{polylog}(\mathcal{T}(n)) + \ell \cdot \mathsf{poly}(\kappa)$. Moreover, if the verifier has oracle access to the low-degree extension of the instance x generated by the prover,[11] then by making a single oracle call on a random point, the verifier's runtime can be reduced to $\ell \cdot \mathsf{poly}(\kappa)$.*
- *The runtime of P given (x, w), where $|x| = n$ and $M(x, w)$ outputs 1 within $\mathcal{T} = \mathcal{T}(n)$ steps, is $\mathsf{poly}(\mathcal{T}(n))$.*
- **Completeness.** *For every $\kappa \in \mathbb{N}$ and every (x, w) such that $|x| = n$ and $M(x, w) = 1$ within $\mathcal{T}(n)$ steps,*

$$\Pr[(P(x, w), V)(1^\kappa) = 1] = 1 - \mathsf{negl}(\kappa),$$

where the probability is over the randomness of V.

- **Local Soundness.** *For any constant $c \in \mathbb{N}$ there exists a probabilistic polynomial time oracle machine Assign_c such that the following holds: If there exists a $\mathsf{poly}(\mathbf{T}(\kappa))$-size cheating prover P^* such that for infinitely many $\kappa \in \mathbb{N}$,*

$$\Pr[P^*(1^\kappa, \mathsf{pp}) = (x, \pi) : (\mathsf{pp}, \mathsf{sk}) \leftarrow V(1^\kappa) \ \wedge \ (1^\kappa, \mathsf{sk}, x, \pi) \in \mathsf{CHEAT}] \geq \frac{1}{\kappa^c},$$

where $(1^\kappa, \mathsf{sk}, x, \pi) \in \mathsf{CHEAT}$ if and only if $V(1^\kappa, \mathsf{sk}, x, \pi) = 1$, $|x| = n(\kappa)$, and M on input x does not output 1 within $\mathcal{T}(n)$ time steps, then for these κ's, $\mathsf{Assign}_c^{P^*}$ is an adaptive $(\mathbf{T}, n, \ell)$-local assignment generator for the layered circuit $C^M_{n,m}$ corresponding to the

[10] We do not include the statement x as part of the communication.

[11] The low degree extension is w.r.t. H, F, m described in [KRR14, BHK17], where $|H| \approx \log \mathcal{T}, m \approx \log\log \mathcal{T}$, and F is a field containing H of size $\approx \mathsf{polylog}(\mathcal{T})$.

Turing machine M. Furthermore, the marginal distribution of x output by $\mathsf{Assign}_c^{P^*}$ is $\mathbf{T}$-indistinguishable from the marginal distribution of x output by P^* conditioned on succeeding in the proof.

5 Succinct Delegation for Computations in $\mathsf{NTISP}(\mathcal{T}, \mathcal{S})$

In this section, we state and prove our main theorem, which roughly says that there exists a succinct non-interactive delegation scheme for languages in $\mathsf{NTISP}(\mathcal{T}(n), \mathcal{S}(n))$, where the communication complexity is $\mathsf{poly}(S(n), \kappa)$, the verifier's runtime is $n \cdot \mathsf{polylog}(\mathcal{T}(n)) + \mathsf{poly}(S(n), \kappa)$, and the prover's runtime is $\mathsf{poly}(\mathcal{T}(n))$. The actual theorem is stated assuming $S(n) \geq \kappa$, so that we get a delegation scheme with communication complexity $\mathsf{poly}(\kappa)$, verifier runtime $n \cdot \mathsf{polylog}(\mathcal{T}(n)) + \mathsf{poly}(\kappa)$, and prover runtime $\mathsf{poly}(\mathcal{T}(n))$.

Theorem 1. *Fix any (possibly non-deterministic) Turing machine M. Fix functions $n = n(\kappa)$ and $\mathcal{T} = \mathcal{T}(n)$ such that $\mathbf{T}(\kappa) \triangleq \mathcal{T}(n) \cdot 2^{W(n)} \leq 2^\kappa$, where $W = W(n)$ denotes the width of the layered circuit $C^M_{m,n}$ (described in Section 4), and such that $\mathcal{T}(n) \geq \max\{n, \kappa\}$. Assume the existence of a succinct $\mathbf{T}$-secure PIR scheme. Then there exists a (succinct) non-interactive argument (P, V) for proving that M, on an input x of length at most n, outputs 1 within $\mathcal{T}(|x|)$ steps, such that for any security parameter κ the following holds:*

- *The communication complexity of (P, V) on input 1^κ is $\mathsf{poly}(\kappa)$.*
- *The runtime of V, on input 1^κ and upon receiving an instance x such that $|x| \leq n$ from the prover, is $|x| \cdot \mathsf{polylog}(\mathcal{T}(n)) + \mathsf{poly}(\kappa)$. Moreover, as in Imported Theorem 1, if the verifier has oracle access to the low-degree extension of x, then the verifier's runtime can be reduced to $\mathsf{poly}(\kappa)$.*
- *The runtime of P given (x, w), where $|x| \leq n$ and $M(x, w)$ outputs 1 within $\mathcal{T} = \mathcal{T}(|x|)$ steps, is $\mathsf{poly}(\mathcal{T}(|x|), \kappa)$.*[12]
- **Completeness.** *For every $\kappa \in \mathbb{N}$ and every (x, w), such that $|x| \leq n$ and $M(x, w) = 1$ within $\mathcal{T}(|x|)$ steps,*
$$\Pr[(P(x, w), V)(1^\kappa) = 1] = 1 - \mathsf{negl}(\kappa),$$
where the probability is over the randomness of V.
- **Soundness.** *For any $\mathsf{poly}(\mathbf{T})$-size cheating prover P^*,*
$$\Pr[P^*(1^\kappa, \mathsf{pp}) = (x, \pi) : (\mathsf{pp}, \mathsf{sk}) \to V(1^\kappa) \ \wedge \ (1^\kappa, \mathsf{sk}, x, \pi) \in \mathsf{CHEAT}] = \mathsf{negl}(\kappa),$$
where $(1^\kappa, \mathsf{sk}, x, \pi) \in \mathsf{CHEAT}$ if and only if $V(1^\kappa, \mathsf{sk}, x, \pi) = 1$, and $|x| \leq n$, and M on input x does not output 1 within $\mathcal{T}(|x|)$ time steps.

Remark 1. *We emphasize that even though in Theorem 1, the Turing machine M and time bound $\mathcal{T}$ are fixed in advance, this theorem gives* full adaptivity*. This is the case, since one can set the (fixed in advance) Turing machine to be a universal Turing machine $\mathcal{U}$ and set a time bound $\mathcal{T}$ as in the theorem statement. The prover can now choose an input (M, x, T) adaptively, and $\mathcal{U}$ on input (M, x, T) outputs 1 if and only if M on input x outputs 1 within T steps, and $T \leq \mathcal{T}$.*

The running time of the honest prover can be further reduced to $\mathsf{poly}(T(|x|), \kappa)$ instead of $\mathsf{poly}(\mathcal{T}(|x|), \kappa)$, by having the verifier send (in parallel) a message corresponding to $\mathcal{T}_i = 2^i$ for every $i \in [\log |\max\{n, \kappa\}|, \kappa - 2W]$, and having the prover choose the specific i adaptively.

[12] Note that $|x|$ can be arbitrarily small, and thus $\mathsf{poly}(\mathcal{T}(|x|))$ may be significantly smaller than κ.

Proof. In order to prove Theorem 1, we show that it suffices to prove the following lemma, which is identical to Theorem 1, except that we restrict $|x| = n$ (as opposed to $|x| \leq n$). We state the lemma formally for the sake of completeness.

Lemma 1. *Fix any (possibly non-deterministic) Turing machine M. Fix functions $n = n(\kappa)$ and $\mathcal{T} = \mathcal{T}(n)$ such that $\mathbf{T}(\kappa) \triangleq \mathcal{T}(n) \cdot 2^{W(n)} \leq 2^{\kappa}$, where $W = W(n)$ denotes the width of the layered circuit $C^M_{m,n}$ (described in Section 4), and such that $\mathcal{T}(n) \geq \max\{n, \kappa\}$. Assume the existence of a succinct $\mathbf{T}$-secure PIR scheme. Then there exists a (succinct) non-interactive argument (P, V) for proving that M, on an input x of length n, outputs 1 within $\mathcal{T}(n)$ steps, such that for any security parameter κ the following holds:*

- *The communication complexity of (P, V) on input 1^{κ} is $\mathsf{poly}(\kappa)$.*
- *The runtime of V, on input 1^{κ} and upon receiving an instance x such that $|x| = n$ from the prover, is $n \cdot \mathsf{polylog}(\mathcal{T}(n)) + \mathsf{poly}(\kappa)$. Moreover, as in Imported Theorem 1, if the verifier has oracle access to the low-degree extension of x, then by making a single oracle call on a random point, the verifier's runtime can be reduced to $\mathsf{poly}(\kappa)$.*
- *The runtime of P given (x, w), where $|x| = n$ and $M(x, w)$ outputs 1 within $\mathcal{T} = \mathcal{T}(n)$ steps, is $\mathsf{poly}(\mathcal{T}(n))$.*
- **Completeness.** *For every $\kappa \in \mathbb{N}$ and every (x, w), such that $|x| = n$ and $M(x, w) = 1$ within $\mathcal{T}(n)$ steps,*
$$\Pr[(P(x, w), V)(1^{\kappa}) = 1] = 1 - \mathsf{negl}(\kappa),$$
where the probability is over the randomness of V.
- **Soundness.** *For any $\mathsf{poly}(\mathbf{T})$-size cheating prover P^*,*
$$\Pr[P^*(1^{\kappa}, \mathsf{pp}) = (x, \pi) : (\mathsf{pp}, \mathsf{sk}) \to V(1^{\kappa}) \ \wedge \ (1^{\kappa}, \mathsf{sk}, x, \pi) \in \mathsf{CHEAT}] = \mathsf{negl}(\kappa),$$
where $(1^{\kappa}, \mathsf{sk}, x, \pi) \in \mathsf{CHEAT}$ if and only if $V(1^{\kappa}, \mathsf{sk}, x, \pi) = 1$, and $|x| = n$, and M on input x does not output 1 within $\mathcal{T}(n)$ time steps.

We first prove Lemma 1, and later prove that Lemma 1 implies Theorem 1.

Proof of Lemma 1. The succinct delegation scheme that we use is exactly the one from Imported Theorem 1 with locality $\ell = \ell(\kappa) = W(n) \cdot \mathsf{poly}(\kappa)$. The completeness and efficiency guarantees follow immediately from the guarantees of Imported Theorem 1, where the latter follows from the fact that $W(n) \leq \mathsf{poly}(\kappa)$. Thus, it suffices to prove soundness.

Assume for contradiction that for the non-interactive argument (P, V), given by Imported Theorem 1 with locality parameter ℓ, there exists a (non-uniform) $\mathsf{poly}(\mathbf{T})$-size cheating prover P^* and there exists a constant $c > 0$, such that for infinitely many $\kappa \in \mathbb{N}$,

$$\Pr[P^*(1^{\kappa}, \mathsf{pp}) = (x, \pi) : (\mathsf{pp}, \mathsf{sk}) \leftarrow V(1^{\kappa}) \ \wedge \ (1^{\kappa}, \mathsf{sk}, x, \pi) \in \mathsf{CHEAT}] \geq \frac{1}{\kappa^c},$$

where $(1^{\kappa}, \mathsf{sk}, x, \pi) \in \mathsf{CHEAT}$ if and only if $V(1^{\kappa}, \mathsf{sk}, x, \pi) = 1$, and $|x| = n$, and M on input x does not output 1 within $\mathcal{T}(n)$ time steps.

By the local soundness property of (P, V), there exists a PPT oracle machine Assign_c such that for these κ's, $\mathsf{Assign}_c^{P^*}$ is a $(\mathbf{T}, n, \ell)$-adaptive local assignment generator for the corresponding layered circuit $C^M_{n,m}$ of depth $D = O(\mathcal{T})$ and width W. We will use this machine to derive a contradiction.

Notation. In what follows, we introduce additional notation about the circuit $C^M_{n,m}$ that is used in the remaining proof. We assume that there are exactly W wires in any layer of $C^M_{n,m}$. This assumption is without loss of generality since the number of wires is at most W, and we can ensure exactly W wires by padding with dummy wires. We let

$$\delta_i : \{0,1\}^W \to \{0,1\}^{|\mathsf{Intermediate}^{i+1}|}$$

denote the corresponding transition function that takes as input an assignment to all wires for layer i and outputs an assignment to all intermediate wires for layer $i+1$.

We next define the sets Acc^i_x for all layers i of $C^M_{n,m}$.

Defining the set Acc^i_x. For any layer $i \in [D]$, we define the set Acc^i_x recursively, as follows:

- Acc^D_x is the set of all possible assignments $\{a^D_j\}_{j:(q^D_j \in \mathsf{Intermediate}^D)}$ to intermediate wires, such that when the assignment $\{a^D_j\}_{j:(q^D_j \in \mathsf{Instance}^D)}$ to the wires in $\mathsf{Instance}^D$ are set consistently with x, there exists an assignment to the witness wires $\{a^D_j\}_{j:(q^D_j \in \mathsf{Witness}^D)}$ such that for this assignment,
$$\delta_D(\mathbf{a}^D = \{a^D_1, a^D_2, \ldots a^D_W\}) = \{1\}.$$
In other words, we require that the output of the D^{th} layer (and hence the output of the circuit) is 1.

- For $i \in [D-1]$, Acc^i_x is defined as the set of all possible assignments $\{a^i_j\}_{j:(q^i_j \in \mathsf{Intermediate}^i)}$ to intermediate wires such that when the assignment $\{a^i_j\}_{j:(q^i_j \in \mathsf{Instance}^i)}$ to wires in $\mathsf{Instance}^i$ are set consistently with x, there exists an assignment to the witness wires $\{a^i_j\}_{j:(q^i_j \in \mathsf{Witness}^i)}$ such that for this assignment,
$$\delta_i(\mathbf{a}^i = \{a^i_1, a^i_2, \ldots a^i_W\}) \in \mathsf{Acc}^{i+1}_x$$

We begin by proving the following claim.

Claim 1. *There exists a Turing machine $\mathcal{Y}$, that takes as input a triplet $(x, \mathbf{a}, k)$ where $|x| = n$, runs in time $D(n) \cdot 2^{2W(n)}$ and decides whether $\mathbf{a} \in \mathsf{Acc}^k_x$ for $k \in [D(n)]$.*

Proof. The Turing machine $\mathcal{Y}$ on input $(x, \mathbf{a}, k)$ does the following: By backward induction, starting from $i = D(n)$ until $i = k$, it computes a table consisting of all the elements in Acc^i_x. We note that given the table of all the elements in Acc^{i+1}_x it takes time $2^{2W(n)}$ to compute the table of all elements in Acc^i_x. Thus, going one by one from $i = D(n)$ to $i = k$, it takes time $D(n) \cdot 2^{2W(n)}$ to compute all such tables. Once $\mathcal{Y}$ computes the table of all the elements in Acc^k_x, all that remains is to check whether $\mathbf{a}$ is in this table. The Turing machine $\mathcal{Y}$ is formally described in Figure 1.

It is easy to verify by inspection that $\mathcal{Y}$ outputs 1 if and only if $\mathbf{a} \in \mathsf{Acc}^k_x$. Furthermore, for every layer, the Turing machine $\mathcal{Y}$ checks at most $2^{2W(n)}$ sets of assignments, each in time at most $\mathsf{poly}(W(n), n)$. Since there are at most $D(n)$ layers, $\mathcal{Y}$ runs in time $D(n) \cdot 2^{2W(n)}$, as desired. □

We next prove the following claim.

Claim 2. *For every $i \in [D(n) - 1]$,*

$$\Pr[\mathbf{a}^i_{\mathsf{Intermediate}} \in \mathsf{Acc}^i_x] = 1 - \mathsf{negl}(\kappa), \tag{1}$$

where the probability is over the randomness of sampling $(x, (\mathbf{a}^i, \mathbf{a}^{i+1})) \leftarrow \mathsf{Assign}^{P^*}_c(1^\kappa, (\mathbf{q}^i, \mathbf{q}^{i+1}))$.

This is the key technical claim that helps prove soundness.

Description of Turing machine $\mathcal{Y}$

- Obtain input $(x, \mathbf{a}, k)$.

1. Set $\mathsf{Acc}_x^{D+1} = \{1\}$.
2. Set $i = D$.
3. While $i \geq k$, compute Acc_x^i as follows:
   - List all possible assignments to intermediate wires $\{a_j^i\}_{j:q_j^i \in \mathsf{Intermediate}^i}$ for gates in layer i, such that:
     When instance wires $\{a_j^i\}_{j:(q_j^i \in \mathsf{Instance}^i)}$ are set consistently with x, there exists an assignment to the witness wires $\{a_j^i\}_{j:(q_j^i \in \mathsf{Witness}^i)}$ such that for this assignment, $\{a_j^{i+1}\}_{j:(q_j^{i+1} \in \mathsf{Intermediate}^{i+1})} \in \mathsf{Acc}_x^{i+1}$, where $\delta_i(\mathbf{a}^i = \{a_1^i, a_2^i, \ldots a_W^i\}) = \{a_j^{i+1}\}_{j:(q_j^{i+1} \in \mathsf{Intermediate}^{i+1})}$.
   - Set $i = i - 1$ and repeat Step 3.
4. Output 1 if $\mathbf{a} \in \mathsf{Acc}_x^k$, else output 0.

Figure 1: Algorithm to decide if $\mathbf{a} \in \mathsf{Acc}_x^k$ for any $\mathbf{a}, x$ and any $k \in [2, D]$.

Proof of Claim 2. First, we show that Equation (1) holds for $i = D(n)$. This follows directly by everywhere $(\ell, \mathbf{T})$-local consistency of $\mathsf{Assign}_c^{P^*}$ (see Definition 3), applied to $i = D(n)$.

Suppose for contradiction that Equation (1) does not hold for some $j \in [D(n)-1]$. This implies that there exists some index $i \in [D(n)-1]$ such that for infinitely many $\kappa \in \mathbb{N}$,

$$\Pr[\mathbf{a}_{\mathsf{Intermediate}}^{i+1} \in \mathsf{Acc}_x^{i+1}] - \Pr[\mathbf{a}_{\mathsf{Intermediate}}^{i} \in \mathsf{Acc}_x^{i}] \geq \frac{1}{p(\kappa) \cdot D(n)}, \tag{2}$$

where the first probability is over sampling $(x, (\mathbf{a}^{i+1}, \mathbf{a}^{i+2})) \leftarrow \mathsf{Assign}_c^{P^*}(1^\kappa, (\mathbf{q}^{i+1}, \mathbf{q}^{i+2}))$, and the second is over the randomness of sampling $(x, (\mathbf{a}^i, \mathbf{a}^{i+1})) \leftarrow \mathsf{Assign}_c^{P^*}(1^\kappa, (\mathbf{q}^i, \mathbf{q}^{i+1}))$.

First, by the $\mathbf{T}$-no-signaling property of the adaptive local assignment generator (see Definition 3), the distributions $(\widetilde{x}, \widetilde{\mathbf{a}}^{i+1})$ and $(x, \mathbf{a}^{i+1})$ are $\mathbf{T}$-indistinguishable (as per Definition 1), where the first is sampled as

$$\left(\widetilde{x}, (\widetilde{\mathbf{a}}^{i+1}, \widetilde{\mathbf{a}}^{i+2})\right) \leftarrow \mathsf{Assign}_c^{P^*}\left(1^\kappa, (\mathbf{q}^{i+1}, \mathbf{q}^{i+2})\right)$$

and the second is sampled as

$$\left(x, (\mathbf{a}^i, \mathbf{a}^{i+1})\right) \leftarrow \mathsf{Assign}_c^{P^*}\left(1^\kappa, (\mathbf{q}^i, \mathbf{q}^{i+1})\right).$$

Therefore,

$$\left|\Pr[\widetilde{\mathbf{a}}_{\mathsf{Intermediate}}^{i+1} \in \mathsf{Acc}_{\widetilde{x}}^{i+1}] - \Pr[\mathbf{a}_{\mathsf{Intermediate}}^{i+1} \in \mathsf{Acc}_x^{i+1}]\right| = \mathsf{negl}(\mathbf{T}(\kappa)) \tag{3}$$

where the first probability is over sampling $\left(\widetilde{x}, (\widetilde{\mathbf{a}}^{i+1}, \widetilde{\mathbf{a}}^{i+2})\right) \leftarrow \mathsf{Assign}_c^{P^*}\left(1^\kappa, (\mathbf{q}^{i+1}, \mathbf{q}^{i+2})\right)$ and the second is over sampling $(x, (\mathbf{a}^i, \mathbf{a}^{i+1})) \leftarrow \mathsf{Assign}_c^{P^*}(1^\kappa, (\mathbf{q}^i, \mathbf{q}^{i+1}))$.

This is because if Equation (3) did not hold, then by Claim 1, it is possible to distinguish between the distributions $(\widetilde{x}, \widetilde{\mathbf{a}}^{i+1})$ and $(x, \mathbf{a}^{i+1})$ in time $\mathcal{T}(n) \cdot 2^{2W(n)} = \mathsf{poly}(\mathbf{T}(\kappa))$ by checking

whether the assignment is in Acc_x^{i+1}, and this would contradict the $\mathbf{T}$-indistinguishability.

Combining Equation (3) with Equation (2), we have that for the index $i \in [j, D-2]$ that satisfies Equation (2):

$$\Pr[\mathbf{a}^{i+1}_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^{i+1}] - \Pr[\mathbf{a}^{i}_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^{i}] \geq \frac{1}{2p(\kappa)D(n)} \tag{4}$$

where both probabilities are over the randomness of sampling $(x, (\mathbf{a}^i, \mathbf{a}^{i+1})) \leftarrow \mathsf{Assign}_c^{P^*}(1^\kappa, (\mathbf{q}^i, \mathbf{q}^{i+1}))$.

We next show that Equation (4) and everywhere $(\ell, \mathbf{T})$-local consistency of $\mathsf{Assign}_c^{P^*}$ according to Definition 3 contradict each other.

Recall that for $(\mathbf{a}^i, \mathbf{a}^{i+1}, x) \leftarrow \mathsf{Assign}_c^{P^*}\left(1^\kappa, (\mathbf{q}^i, \mathbf{q}^{i+1})\right)$, if $\Pr[\mathbf{a}^{i+1}_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^{i+1}] \geq \frac{1}{\mathsf{poly}(\mathbf{T}(\kappa))}$, local consistency implies that:

$$\Pr[\mathbf{a}^{i}_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^{i} | \mathbf{a}^{i+1}_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^{i+1}] = 1 - \mathsf{negl}(\mathbf{T}(\kappa))$$

where the probability is over $(\mathbf{a}^i, \mathbf{a}^{i+1}, x) \leftarrow \mathsf{Assign}_c^{P^*}\left(1^\kappa, (\mathbf{q}^i, \mathbf{q}^{i+1})\right)$. This implies that

$$\begin{aligned}\Pr[(\mathbf{a}^{i}_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^{i}) \ \wedge \ (\mathbf{a}^{i+1}_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^{i+1})] &= \Pr[\mathbf{a}^{i+1}_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^{i+1}] \cdot (1 - \mathsf{negl}(\mathbf{T}(\kappa))) \\ &= \Pr[\mathbf{a}^{i+1}_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^{i+1}] - \mathsf{negl}(\mathbf{T}(\kappa))\end{aligned}$$

where the probability is over $(\mathbf{a}^i, \mathbf{a}^{i+1}, x) \leftarrow \mathsf{Assign}_c^{P^*}\left(1^\kappa, (\mathbf{q}^i, \mathbf{q}^{i+1})\right)$. Therefore,

$$\Pr[\mathbf{a}^{i}_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^{i}] \geq \Pr[\mathbf{a}^{i+1}_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^{i+1}] - \mathsf{negl}(\mathbf{T}(\kappa))$$

where the probability is over $(\mathbf{a}^i, \mathbf{a}^{i+1}, x) \leftarrow \mathsf{Assign}_c^{P^*}\left(1^\kappa, (\mathbf{q}^i, \mathbf{q}^{i+1})\right)$. This contradicts Equation (4) and completes the proof of Claim 2. □

In what follows we use Claim 2 to complete the proof of Lemma 1. Intuitively, we would like to use Claim 2 with $i = 1$, and argue that if $(\mathbf{a}^1_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^1)$ then it must be the case that indeed $M(x) = 1$ within $\mathcal{T}(n)$ steps. However, note that $\mathbf{a}^1_{\mathsf{Intermediate}}$ is the empty set, since the first layer of gates only reads the instance (we assume without loss of generality that the first layer of gates does not read any witness wires). Thus, $\mathbf{a}^2_{\mathsf{Intermediate}}$ is the very first non-trivial layer of intermediate wires in the circuit.

By the definition of Acc_x^2 and because each bit of the witness is read by at most one layer starting at layer 2, $(\mathbf{a}^2_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^2)$ if and only if $\exists w : C^M_{n,m}(x, w) = 1$.

By Claim 2 (for $i = 2$),

$$\Pr[\mathbf{a}^{2}_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^{2}] = 1 - \mathsf{negl}(\kappa)$$

where the probability is over sampling $(\mathbf{a}^2, \mathbf{a}^3, x) \leftarrow \mathsf{Assign}_c^{P^*}(1^\kappa, (\mathbf{q}^2, \mathbf{q}^3))$.

Thus,

$$\Pr[\mathbf{a}^{2}_{\mathsf{Intermediate}} \in \mathsf{Acc}_x^{2}] = 1 - \mathsf{negl}(\kappa) \tag{5}$$

must also hold over the randomness of sampling $(\mathbf{a}^1, \mathbf{a}^2, x) \leftarrow \mathsf{Assign}_c^{P^*}(1^\kappa, (\mathbf{q}^1, \mathbf{q}^2))$. This follows by the $\mathbf{T}$ no-signaling property of the adaptive local assignment generator (see Definition 3) applied similarly as in the proof of Equation (3) above.

This implies that

$$\Pr[\exists w : C^M_{n,m}(x, w) = 1] = 1 - \mathsf{negl}(\kappa),$$

where the probability is over sampling $((\mathbf{a}^1, \mathbf{a}^2), x) \leftarrow \mathsf{Assign}_c^{P^*}(1^\kappa, (\mathbf{q}^1, \mathbf{q}^2))$.

However, recall that P^* generates x such that $M(x)$ does not output 1 within $\mathcal{T}(n)$ steps, and the instance x generated by $\mathsf{Assign}_c^{P^*}$ is $\mathbf{T}$-indistinguishable from the instance x generated by P^*. Therefore, the instance x generated by $\mathsf{Assign}_c^{P^*}$ is such that with probability $1-\mathsf{negl}(\kappa)$, $M(x)$ does not output 1 within $\mathcal{T}(n)$ steps. This gives a contradiction and completes the proof of Lemma 1. $\square$

We now complete the proof of Theorem 1. Note that the main difference between Theorem 1 and Lemma 1 is that Lemma 1 restricts $|x|$ to be an a-priori fixed value n. The prover's runtime for the delegation scheme in Lemma 1 grows as a function of this fixed n. On the other hand, in Theorem 1, we allow the prover to pick any x where $|x| \leq n$, such that the prover's runtime only grows as a function of $|x|$.

Given a delegation scheme satisfying Lemma 1 with inputs of fixed length n, we describe how to generically build a delegation scheme satisfying Theorem 1.

- First, we sparsify the inputs of the Turing machine M as follows: assume that M takes only inputs of length 2^i for $i \in \mathbb{N}$. This assumption is without loss of generality since we encode each input $x \in \{0,1\}^n$ as an input of length exactly $2^{\lceil \log(n+1) \rceil}$. This is done by padding each such x with a single 1 followed by zero's. Moreover, we think of the Turing machine M as first removing the padding from the padded input, and only then running M.

- Let us denote by Π_i the succinct delegation scheme for inputs of length 2^i, satisfying Lemma 1.

  Given a bound $n = 2^j$ on the size of x according to Theorem 1, we construct protocol $\widetilde{\Pi}$ for Theorem 1 where the verifier sends the first message for all the j protocols $\Pi_1, \Pi_2, \ldots, \Pi_j$, in parallel. Recall that for each $i \leq j$, the protocol Π_i is for a fixed input size 2^i.

  The prover on input instance x of length 2^i for some $i \leq j$, sends the requisite prover message according to Π_i, and does not provide a message for any of the other protocols.

The protocol $\widetilde{\Pi}$ is described formally in Figure 2.

Description of Succinct Delegation Protocol $\widetilde{\Pi} = (P_{\widetilde{\Pi}}, V_{\widetilde{\Pi}})$.

1. Let Π_i denote the succinct delegation protocol from Lemma 1 for inputs of length 2^i.
2. $V_{\widetilde{\Pi}}(1^\kappa)$ computes $(\mathsf{pp}_i, \mathsf{sk}_i) \leftarrow V_{\Pi_i}(1^\kappa)$, saves $\{\mathsf{sk}_i\}_{i\in[j]}$ and outputs $\{\mathsf{pp}_i\}_{i\in[j]}$.
3. $P_{\widetilde{\Pi}}\left(1^\kappa, \{\mathsf{pp}_i\}_{i\in[j]}, x, w\right)$ computes $i = \log(|x|)$ and runs $P_{\Pi_i}(1^\kappa, \mathsf{pp}_i, x, w)$ to output (x, π).
4. $V_{\widetilde{\Pi}}(1^\kappa, \{\mathsf{pp}_i\}_{i\in[j]}, \{\mathsf{sk}_i\}_{i\in[j]}, x, \pi)$ executes $V_{\Pi_i}(1^\kappa, \mathsf{pp}_i, \mathsf{sk}_i, x, \pi)$ for $i = \log(|x|)$.

Figure 2: Succinct Delegation with Prover Runtime $\mathsf{poly}(\mathcal{T}(|x|))$.

Completeness and soundness of $\widetilde{\Pi}$ follow directly from that of Π. The fact that $j \leq \kappa$ implies that the communication complexity of $\widetilde{\Pi}$ remains $\mathsf{poly}(\kappa)$, the runtime of P and V remain the same (up to a multiplicative factor in the security parameter κ). This gives the required protocol for Theorem 1 with running time of the prover only growing with $|x|$, thus completing the proof. $\square$

Remark 2. *We note that Theorem 1 can be optimized so that the communication complexity and verifier running time only grow with the space required by the configurations of the Turing machine that make a non-deterministic step (as opposed to the space required by all configurations of the Turing machine). This is done by augmenting the deterministic layers of the circuit using [KRR14, BHK17] and relying on [BHK17] to achieve adaptive soundness with locality only $O(\kappa)$ for the deterministic layers. We defer additional details to a future version of the paper.*

References

[ACC+15] Prabhanjan Ananth, Yu-Chi Chen, Kai-Min Chung, Huijia Lin, and Wei-Kai Lin. Delegating RAM computations with adaptive soundness and privacy. *IACR Cryptology ePrint Archive*, 2015:1082, 2015.

[ACL+14] Eric Allender, Shiteng Chen, Tiancheng Lou, Periklis A. Papakonstantinou, and Bangsheng Tang. Width-parametrized SAT: time–space tradeoffs. *Theory of Computing*, 10:297–339, 2014.

[BCC+14] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *IACR Cryptology ePrint Archive*, 2014:580, 2014.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *STOC*, pages 111–120. ACM, 2013.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474, 2014.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, pages 315–333, 2013.

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

[BGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. pages 113–131, 1988.

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. *IACR Cryptology ePrint Archive*, 2015:356, 2015.

[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.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based snargs and their application to more efficient obfuscation. In *Advances in Cryptology - EUROCRYPT*

2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III, pages 247–277, 2017.

[BMW98] Ingrid Biehl, Bernd Meyer, and Susanne Wetzel. Ensuring the integrity of agent-based computations by short proofs. In *Mobile Agents, Second International Workshop, MA'98, Stuttgart, Germany, September 1998, Proceedings*, pages 183–194, 1998.

[CCC+16] Yu-Chi Chen, Sherman S. M. Chow, Kai-Min Chung, Russell W. F. Lai, Wei-Kai Lin, and Hong-Sheng Zhou. Cryptography for parallel RAM from indistinguishability obfuscation. In *ITCS*, pages 179–190. ACM, 2016.

[CCHR15] Ran Canetti, Yilei Chen, Justin Holmgren, and Mariana Raykova. Succinct adaptive garbled RAM. *IACR Cryptology ePrint Archive*, 2015:1074, 2015.

[CH16] Ran Canetti and Justin Holmgren. Fully succinct garbled RAM. In *ITCS*, pages 169–178. ACM, 2016.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In *STOC*, pages 429–437. ACM, 2015.

[Dam92] Ivan Damgård. *Towards Practical Public Key Systems Secure Against Chosen Ciphertext attacks*, pages 445–456. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 54–74, 2012.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. Cryptology ePrint Archive, Report 2016/272, 2016. `http://eprint.iacr.org/`.

[DLN+01] Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim, and Omer Reingold. Succinct proofs for np and spooky interactions. Unpublished manuscript, 2001.

[DNR16] Cynthia Dwork, Moni Naor, and Guy N. Rothblum. Spooky interaction and its discontents: Compilers for succinct two-message argument systems. Cryptology ePrint Archive, Report 2016/291, 2016. `http://eprint.iacr.org/`.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 113–122. ACM, 2008. Full version in [GKR15].

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

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340. Springer, 2010.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 99–108, New York, NY, USA, 2011. ACM.

[HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS '15, pages 163–172, New York, NY, USA, 2015. ACM.

[JKKR17] Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 158–189. Springer, 2017.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732. ACM, 1992.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *STOC*, pages 419–428. ACM, 2015.

[KP15] Yael Tauman Kalai and Omer Paneth. Delegating RAM computations. *IACR Cryptology ePrint Archive*, 2015:957, 2015.

[KR09] Yael Tauman Kalai and Ran Raz. Probabilistically checkable arguments. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 143–159, 2009.

[KRR13] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 565–574. ACM, 2013.

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *STOC*, pages 485–494. ACM, 2014.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC*, pages 169–189, 2012.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 436–453. IEEE Computer Society, 1994. Full version in [Mic00].

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

[PR17] Omer Paneth and Guy N. Rothblum. On zero-testable homomorphic encryption and publicly verifiable non-interactive arguments. Cryptology ePrint Archive, Report 2017/903, 2017. `http://eprint.iacr.org/2017/903`.

[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.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484, 2014.

A Natural Languages Computable in Low Non-Deterministic Space

Computation in $\mathsf{NSC}^{\mathsf{k}}$. Our results are especially relevant for the class $\mathsf{NSC}^{\mathsf{k}}$, which is defined to be the class of languages in $\mathsf{NTISP}(\mathsf{poly}, \mathsf{log}^k n)$. Note that NSC^k for $k > 2$ is not known to be in P. As an example, SAT of tree-width $\mathsf{log}^k n$ is contained in NSC^{k+1} [ACL+14].

We now consider more general NP-complete languages that require low non-deterministic space (and for specific parameter settings, lie in NSC^k).

NP-Complete Languages Decidable in Low Non-Deterministic Space. We observe that variants of the subset sum, knapsack and partition problems lie in $\mathsf{NTISP}(\mathsf{poly}(n), \mathcal{S}(n))$ for small $\mathcal{S}(n)$ (eg., $\mathcal{S}(n) = O(\mathsf{polylog}(n))$. These problems are known to admit (super-polynomial time) dynamic programming solutions that are significantly more efficient than brute-force solutions.

Subset Sum. On input a set of N numbers, each of size ℓ bits, and a target value V, the problem is to find some subset $\mathcal{S}$ of these N numbers such that the sum of all elements in $\mathcal{S}$ equals V. Formally, the NP language is defined as follows:

$$\mathsf{SUBSET.SUM}_{\ell,N} = \left\{ T = (V, x_1, \ldots, x_N) \text{ where } \forall i \in [N],\ |x_i| = \ell \;\middle|\; \exists\ \mathcal{S} \subseteq 2^N \text{ s.t. } \Sigma_{i \in \mathcal{S}}\ x_i = V \right\}$$

We note that $|V| = (\ell \log N)$ bits, and $|(x_1, \ldots x_N)| = \ell N$ bits. A canonical representation of the witness w has a single bit indicating whether or not x_i is part of the solution set for $i \in [N]$, therefore $|w| = N$ bits. The certificate checking algorithm is as follows. Given an input T and a witness w represented as an N bit vector, do:

- Initialize sum to 0.
- For each $i \in [N]$, if $w[i] = 1$, do $\mathsf{sum} = \mathsf{sum} + x_i$.
- If $\mathsf{sum} = V$, output 1 indicating that $T \in \mathsf{SUBSET.SUM}_{\ell,N}$. Else, output 0.

The space required by the above algorithm reading each bit of the witness once is at most $(\mathsf{log}N + \ell)$, which is just the space needed to maintain a running sum. Therefore, setting the size of input $n = \ell N$ and setting $\ell = \mathsf{polylog}(N)$, $\mathsf{SUBSET.SUM}_{\ell,N} \in \mathsf{NTISP}(\mathsf{poly}(n), \mathsf{polylog}(n))$.

We remark that another NP-Complete problem – partition – is in fact, just a special case of the Subset Sum problem and hence also fits the analysis above.

Knapsack. On input a set of N items and an associated (weight, value) pair for each item, a target weight W and a target value V, the goal is to determine a subset $\mathcal{S}$ of the N items such that the sum of the weights of all the items in the set is at most W and the sum of the values of all the items in the set is at least V. Formally, the NP language is defined as:

$$\mathsf{KNAP}_{\ell,N} = \Bigg\{ T = (V, W, v_1, \ldots, v_N, \mathsf{wt}_1, \ldots, \mathsf{wt}_N) \text{ where } \forall i \in [N],\ |v_i| = |\mathsf{wt}_i| = \ell$$

$$\Big|\ \exists\ \mathcal{S} \subseteq 2^N \text{ s.t. } \Sigma_{i \in \mathcal{S}}\ v_i \geq V \text{ and } \Sigma_{i \in \mathcal{S}}\ \mathsf{wt}_i \leq W \Bigg\}$$

The size of input $n = |W| + |V| + N \cdot (2\ell)$ bits, where $|W|, |V| \leq \ell \log N$. A canonical representation of the witness w has a single bit indicating whether or not the i^{th} item is a part of the solution, therefore $|w| = N$ bits. The certificate checking algorithm is as follows, on input T and witness w:

- Initialize weight and value to 0.
- For each $i \in [N]$, if $w[i] = 1$, do $\mathsf{value} = \mathsf{value} + v_i$ and $\mathsf{weight} = \mathsf{weight} + \mathsf{wt}_i$.
- If $\mathsf{value} \geq V$ and $\mathsf{weight} \leq W$, output 1 indicating that $T \in \mathsf{KNAP}_{\ell,N}$. Else, output 0.

The space required by this algorithm reading each bit of the witness once is at most $4\ell \log N$. Therefore, setting the size of input $n = \ell N$ and setting $\ell = \mathsf{polylog}(n)$, $\mathsf{KNAP}_{\ell,N} \in \mathsf{NTISP}(\mathsf{poly}(n), \mathsf{polylog}(n))$.

Tree-Based Computation. Tree based computations are commonly used in cryptography. We formally define a class of languages whose verification requires computing over a tree structure, and while on inputs of length n the witness size may be exponential in n, these languages lie in $\mathsf{NTISP}(\mathsf{poly}(n), \mathsf{poly}(n))$.

Definition 4 (Tree-Computable Languages)**.** *Consider an NP language L with an associated witness-checking Turing Machine $\mathcal{A}$ that computes R_L, namely given as input a statement x of length n and a witness $w = (w_1, \ldots, w_N)$ where $N = N(n)$ and $|w_i| = n$ for all $i \in N$, outputs 1 if and only if $x \in L$. Then we say that L is defined to be* Tree-Computable *if the Turing machine $\mathcal{A}$ has the following structure:*

- *First, create a binary tree with the leaves of the tree corresponding to the witness w. That is, $\forall i \in [N]$, w_i is the i^{th} leaf of the tree.*
- *Consider all pairs of leaves w_i, w_{i+1} for all odd $i \in [N]$. Their parent node is computed by applying a deterministic function that takes as input (w_i, w_{i+1}, x) and outputs a string of length n using space $O(n)$.*
- *Repeat recursively on each layer until there is only one node at the root. This requires saving $O(\log N)$ nodes in memory, which is the depth of the tree. Let* value *denote the root node.*

- *Finally, $\mathcal{A}$ outputs 0 or 1 by evaluating a deterministic function that takes as input* (value, x).

Setting parameters such that $n \geq \mathsf{polylog}(\mathsf{N})$, we can see that any tree-computable NP language L is in $\mathsf{NTISP}(\mathsf{poly}(n), \mathsf{poly}(n))$. Note that the witness may be as large as $N = \exp^{(n^{1/c})}$ in size, where $c \geq 0$ is a constant. As a result, such proofs will be significantly shorter than the witness.

An example of a tree-computable language is the following. Consider the FHE-based construction of somewhere statistically binding hash in [HW15]. In this construction, the hash key contains an FHE public key, and the range of this hash function consists only of **valid** ciphertexts under this key. This gives rise to a hash tree function $H : \{0,1\}^{nN} \to \{0,1\}^n$. Given a target output V, we say that (H, V) is in the language L if and only if there exists a message $m = (m_1, m_2, \ldots m_N)$ such that $H(m_1, m_2, \ldots m_N) = V$. It is immediate that L is a tree-computable language. Here, it can be useful to have a prover computing a hash tree provide a (succinct) proof that the hash computation was correctly done, and this will ensure that the output of the prover corresponds to a valid ciphertext. We anticipate that our proof systems will also find use in other tree-based computations.

Non-Interactive Batch Delegation of Non-Deterministic Computation. Our protocol yields succinct non-interactive (adaptive) delegation for non-deterministic computation, which allows a prover to prove, given $\{(x_i, w_i)\}_{i\in[N]}$ where $|x_i| \leq n$ for all i, that $x_1, \ldots x_N$ are all in L which is a $\mathsf{NTISP}(\mathcal{T}(n), \mathcal{S}(n))$ language. This is done with verification and communication complexity proportional to the non-deterministic space complexity of verifying *one* computation. Using the optimization from Remark 2, this will be at most the size of a single witness.

Corollary 2. *Assume the existence of sub-exponentially secure succinct PIR. Then for any language L in $\mathsf{NTISP}(\mathcal{T}(n), \mathcal{S}(n))$ with corresponding relation R_L, there exists a (succinct) non-interactive argument (P, V) for proving that $\bigwedge_{i\in[N]} x_i \in L$ where $|x_i| = n$ for $i \in [N]$, satisfying completeness and soundness properties against adaptive PPT provers as defined in Theorem 1.*

The communication complexity of the protocol and the run time of the verifier V on input 1^κ is $\mathsf{poly}(\mathcal{S}(n), \kappa, \log N, \log(\mathcal{T}(n)))$ and $nN \cdot \mathsf{polylog}(\mathcal{T}(n)) + \mathsf{poly}(\mathcal{S}(n), \kappa, \log N, \log(\mathcal{T}(n)))$ respectively[13]*. The run time of the prover P given $\{(x_i, w_i)\}_{i\in[N]}$ such that $R_L(x_i, w_i) = 1$ for all $i \in [N]$, is $\mathsf{poly}(N \cdot \mathcal{T}(n), \kappa)$.*

Prior to our work, [BHK17] achieved non-adaptive security, communication complexity and verifier run-time growing with the length of one witness, while relying on polynomially secure succinct PIR.

[13] As in Imported Theorem 1, if the verifier has oracle access to the low-degree extension of $\{x_i\}_{i\in[N]}$, then the verifier's runtime can be reduced to $\mathsf{poly}(\mathcal{S}(n), \kappa, \log N)$.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il