SNARGs for P from Sub-exponential DDH and QR

James Hulett[⋆], Ruta Jawale[⋆], Dakshita Khurana[⋆], and Akshayaram Srinivasan[⋆⋆]

[1] University of Illinois, Urbana-Champaign
[2] Tata Institute of Fundamental Research

Abstract. We obtain publicly verifiable Succinct Non-Interactive Arguments (SNARGs) for arbitrary deterministic computations and bounded space non-deterministic computation from standard group-based assumptions, without relying on pairings. In particular, assuming the sub-exponential hardness of both the Decisional Diffie-Hellman (DDH) and Quadratic Residuosity (QR) assumptions, we obtain the following results, where n denotes the length of the instance:

1. A SNARG for any language that can be decided in non-deterministic time T and space S with communication complexity and verifier runtime $(n+S)\cdot T^{o(1)}$.
2. A SNARG for any language that can be decided in deterministic time T with communication complexity and verifier runtime $n \cdot T^{o(1)}$.

1 Introduction

We consider the problem of constructing *succinct, publicly verifiable* arguments to certify the correctness of computation. By succinct, we refer to the setting where the running time of verifier is much smaller than the time required to perform the computation.

The problem of constructing such proof systems has received widespread attention over the last three decades. These are typically called succinct non-interactive arguments (SNARGs), where *argument* refers to any proof system whose soundness holds against polynomial-time provers (under cryptographic assumptions) and the non-interactive setting refers to a single message of communication sent by the prover to the verifier. As in prior work, our work focuses on constructions in the CRS model, where participants have access to a common reference string.

Until recently, a significant amount of prior work on SNARGs focused on constructions proven secure under non-falsifiable assumptions or shown secure

[⋆] Email: {`jhulett2,jawale2,dakshita`}`@illinois.edu`. Supported in part by DARPA SIEVE award under contract number HR001120C0024, a gift from Visa Research, and a C3AI DTI award. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

[⋆⋆] Email: `akshayaram.srinivasan@tifr.res.in` only in idealized models (such as the Random Oracle Model). Indeed, Gentry and Wichs [25] showed that if such an argument system satisfied a strong form of soundness called as adaptive soundness, then such non-falsifiable assumptions are necessary for SNARGs for NP. There has been recent exciting progress on constructing SNARGs for classes that are subsets of NP under falsifiable standard cryptographic assumptions, and in particular the LWE (Learning with Errors assumption), by instantiating the Fiat-Shamir paradigm, discussed next.

The Fiat-Shamir Paradigm. The Fiat-Shamir paradigm is a transformation that converts any public-coin interactive argument $(\mathcal{P}, \mathcal{V})$ for a language L to a non-interactive argument $(\mathcal{P}', \mathcal{V}')$ for L. The CRS consists of randomly chosen hash functions $h_1, \ldots, h_\ell$ from a hash family $\mathcal{H}$, where ℓ is the number of rounds in $(\mathcal{P}, \mathcal{V})$. To compute a non-interactive argument for $x \in L$, the prover $\mathcal{P}'(x)$ generates a transcript corresponding to $(\mathcal{P}, \mathcal{V})(x)$, by emulating $\mathcal{P}(x)$ and replacing each random verifier message by a hash of the transcript so far. The verifier $\mathcal{V}'(x)$ accepts if and only if $\mathcal{V}(x)$ accepts this transcript and all verifier challenges are computed correctly as the output of the hash function on the transcript so far. This paradigm is sound when applied to constant round protocols in the Random Oracle Model (ROM) [6,48]. At the same time there are counterexamples that demonstrate its insecurity in the plain model [4,26,16,5].

The recent work of Canetti *et al.* [15] and subsequent work of Peikert and Shiehian [47] proved the soundness of the Fiat-Shamir paradigm, assuming standard hardness of the Learning With Errors (LWE) problem, when applied to a *specific* zero-knowledge protocol. This gave the first NIZK argument from LWE. This work also obtained a SNARG for all bounded depth computations, assuming the existence of an FHE scheme with optimal circular security – which appears to be an extremely strong assumption. Subsequently, [33] gave an instantiation of the Fiat-Shamir paradigm applied to special classes of succinct proofs, which resulted in SNARGs for bounded depth computations from sub-exponential LWE [33]. Even more recently, Choudhuri et al. [22] gave a construction of SNARGs for the complexity class P from polynomial LWE, using which Kalai et al. [38] gave a construction of SNARGs for bounded-space nondeterministic computation under sub-exponential LWE. The LWE assumption is a structured cryptographic assumption that is known to imply among several other interesting cryptographic primitives, compact (leveled) homomorphic encryption. In fact, all aforementioned constructions of SNARGs implicitly make use of homomorphic encryption.

On the other hand, foundational group-based assumptions such as Decisional Diffie-Hellman and Quadratic Residuosity are not known to imply homomorphic encryption, and yet their (sub-exponential) variants have surprisingly, via the Fiat-Shamir paradigm, been shown to imply non-interactive zero-knowledge [14,32] as well as non-trivial SNARGs for batched NP statements [21]. This motivates the following question:

Do there exist SNARGs for P *(and beyond) from standard group-based assumptions like DDH and QR?*

1.1 Our Results

We address the above question and obtain the following positive results.

- We build a SNARG for the class of all non-deterministic computations requiring time $T(n)$ and space $S(n)$ (denoted by $\mathsf{NTISP}(T(n); S(n))$) where the prover runs in time $\mathsf{poly}(T(n))$ given a witness for the computation and the verifier runs in time $(n + S(n)) \cdot T(n)^{o(1)}$ where n is the instance length.
- Plugging the SNARG above into a compiler from [38], we obtain a SNARG for the class P where the prover runs in time $\mathsf{poly}(T(n))$ and the verifier runs in time $n \cdot T(n)^{o(1)}$.

Our construction for NTISP is obtained in three steps.

1. We develop a new *folding technique* for interactive succinct arguments, where we *recursively* break down a time-T computation into smaller subcomputations, each of time T/k (for an appropriate choice of k) and have the prover send batch proofs of the validity of each subcomputation. This can be viewed as a computational analogue of the RRR interactive proof [49].
2. We instantiate our protocol using batch interactive arguments for NP[3] that are "FS-compatible", which were in particular developed in [21] based on the hardness of QR. Here, FS-compatible refers to the fact that these interactive batch NP arguments can be soundly converted into SNARGs via the Fiat-Shamir paradigm. In addition, we show that our interactive argument for NTISP is FS-compatible as long as the underlying batch NP argument is FS-compatible.
3. We then soundly convert the above succinct interactive argument to a SNARG by making use of correlation-intractable hash functions for low-depth threshold circuits constructed in [32], based on sub-exponential hardness of DDH.

Finally, we note that the works of [2,29,42] observed that in addition to interactive proofs, the Fiat-Shamir paradigm can be soundly instantiated for special types of arguments. They observed that this is possible for arguments that have an *unconditionally sound mode*, and where the prover cannot detect whether the argument is unconditionally or computationally sound. These ideas were then extended to the setting of *succinct* arguments in [21,22]. As a contribution that may be of independent interest, we abstract out a notion of Fiat-Shamir compatibility of argument systems, which captures these broad requirements (including those used in [2,29,42,21,22]) that interactive *arguments* satisfy in order to soundly instantiate Fiat-Shamir from standard assumptions using known techniques.

1.2 Other Prior Work

The works of [43,30,41,23,24,9,8,7] obtain SNARGs for non-deterministic computations, with security either in the Random Oracle Model [6] or from non-falsifiable "knowledge assumptions." The schemes of [19,40,10,18,1,20,45] rely

[3] Batch arguments for NP allow a verifier to verify the correctness of k NP instances with circuit complexity smaller than k times the size of the NP verification circuit.

on assumptions related to obfuscation, which are both stronger in flavor and less widely studied than the ones used in this work. More recently, [35] constructed a SNARG (for deterministic computations) based on a (new) efficiently falsifiable decisional assumption on groups with bilinear maps. Later, a line of work [15,33,21,22,38] instantiated the Fiat-Shamir paradigm to finally result in SNARGs for P from the learning with errors (LWE) assumption. Very recently, the work of Gonzalez and Zacharias [28] constructed SNARGs from pairing-based assumptions. On the other hand, in this work, we obtain SNARGs from assumptions that hold in pairing-free groups.

Another line of work [36,37,34,12,3,13] built *privately verifiable* schemes for deterministic computations and a sub-class of non-deterministic computations, based on standard assumptions (specifically, the hardness of LWE or ϕ-hiding). These schemes, however, are not publicly verifiable. The CRS is generated together with a secret key which is needed in order to verify the proofs.

In the interactive setting, publicly verifiable schemes exist, even for non-deterministic computations, under standard cryptographic assumptions [39,11,46]. In fact some publicly verifiable interactive proof systems for restricted classes of computations exist even unconditionally, in particular for bounded depth [27] and bounded space computations [49].

2 Technical Overview

We start with a high-level overview of our recursively-built interactive argument. To begin with, we will only focus on languages that can be decided in deterministic time T and space S. The prover will run in time $\mathsf{poly}(T)$, and the size of our proofs will grow (linearly) in S.

2.1 Succinct Interactive Arguments for Bounded Space from Succinct Arguments for Batch NP

In what follows, we describe a form of interactive arguments for bounded space computations that can be soundly compressed via the Fiat-Shamir transform. We discuss why these ideas may seem to necessitate the use of LWE, and then describe how our folding technique helps get around the need for the LWE assumption while achieving $T^{o(1)}$ verification time.

Consider a deterministic computation that takes T steps: the prover and verifier agree on a (deterministic) Turing Machine $\mathcal{M}$, an input $y \in \{0,1\}^n$, and two configurations $u, v \in \{0,1\}^S$ (a configuration includes the machine's internal state, the contents of all memory tapes, and the position of the heads). The prover's claim is that after running the machine $\mathcal{M}$ on input y, starting at configuration u and proceeding for T steps, the resulting configuration is v. This is denoted by

$$(\mathcal{M}, y) : u \xrightarrow{T} v.$$

To prove correctness of this T-step computation, the prover will send $(k-1)$ alleged intermediate configurations

$$(s_1, s_2, \ldots, s_{k-1})$$

and will set $s_0 := u, s_k := v$, where for every $i \in [1,k]$, s_i is the alleged configuration of the machine $\mathcal{M}$ after T/k steps when starting at configuration s_{i-1}.

Now the prover will attempt to prove correctness of all these intermediate configurations: a naïve way to achieve this is to run k executions of the base protocol, one for every $i \in [k]$. But the trick to achieving succinctness will be to prove correctness of all configurations simultaneously in verification time that is significantly smaller than running the base protocol k times, while also not blowing up the prover's complexity by a factor of k. To enable this, the prover and verifier can rely on an appropriate succinct interactive argument for *batch* NP to establish that all responses would have been accepted by the verifier.

In a succinct argument for batch NP, a prover tries to convince a verifier that $(x_1, \ldots, x_k) \in \mathcal{L}^{\otimes k}$, in such a way that the proof size and communication complexity are smaller than the trivial solution where the prover simply sends all witnesses $(w_1, \ldots, w_k)$ to the verifier, and the verifier computes $\bigwedge_{i \in [k]} \mathcal{R}_{\mathcal{L}}(x_i, w_i)$. In particular, [21] recently obtained SNARGs for batching k NP instances (from QR and sub-exponential DDH) where the communication complexity is $\widetilde{O}(|C| + k \log |C|) \cdot \mathsf{poly}(\lambda)$, and verifier runtime is $\widetilde{O}(kn + |C|) \cdot \mathsf{poly}(\lambda)$, where λ is the security parameter, $|C|$ denotes the size of the verification circuit and n denotes the size of each instance. In our setting, $|C| \approx (T/k)$, which means that verification time for the SNARG will be $\widetilde{O}(k + T/k)$. Setting $k = O(\sqrt{T})$, we would obtain communication complexity (and verification runtime) that grows (approx.) with $O(\sqrt{T})$ and this is the best that one can hope for in this case [21]. However, in this work, we would like to achieve an overhead of $T^{o(1)}$.

A Recursive Construction. The argument described above incurred an overhead of T/k because the verification circuit for each subcomputation had size T/k. However, what if we substituted this verification circuit with the (relatively efficient) verifier for a *succinct interactive argument for* T/k-time computations?

Specifically, assume there exists a *public-coin interactive* argument for verifying computations of size T/k. As before, suppose a prover wants to convince a verifier that

$$(\mathcal{M}, y) : u \xrightarrow{T} v.$$

The prover sends $(k-1)$ intermediate configurations, as before, and then prepares the first messages of all k interactive arguments, where the i^{th} interactive argument attests to the correctness of $(\mathcal{M}, y) : s_{i-1} \xrightarrow{T/k} s_i$. Instead of sending these messages in the clear, the prover sends to the verifier a *succinct commitment* to all k first messages. Here, following [21,22,38], one could use a keyed *computationally binding* succinct commitment whose key is placed in the CRS. In fact, looking ahead, we will require a commitment that that is binding to a (hidden) part of the input string [31], and in fact the bound parts of the input should be extractable given a trapdoor. We will call such commitments somewhere-extractable (SE) commitments. In more detail, these commitments have a key generation algorithm $\mathsf{Gen}(1^\lambda, i)$ that on input an index $i \in [k]$ outputs a commitment key ck together with an extraction trapdoor td, and an extraction algorithm that given td and any commitment string c outputs the unique i^{th} committed block (out of a total of k blocks). Moreover, the commitment key hides the index i in a CPA-sense.

Next, the verifier sends a single (public coin) message that serves as a challenge for all k arguments. Subsequently, the prover prepares a third message for all arguments, and commits to these messages, after which the verifier again generates a single (public coin) message that serves as its fourth message for all k arguments. The prover and verifier proceed until all rounds of all k arguments are committed, and then the prover (as before) must prove to the verifier that all committed transcripts would be accepted.

At this point, one solution is for the prover and verifier to engage in a batch NP argument (as before), where the prover must convince the verifier that for every $i \in [k]$, there is an opening to the commitment that would cause the verifier to accept. In what follows, we will rely on the fact that the batch NP SNARG can actually be obtained in two steps: first, build an interactive argument for batch NP, and next compress rounds of interaction via Fiat-Shamir. Indeed, the batch SNARG from [21] that we will use *is obtained* by first building an interactive argument and then compressing it by soundly instantiating the Fiat-Shamir paradigm. From this point on, unless otherwise specified, we will make use of the [21] *interactive* batch NP argument, and later separately use the fact that it can be soundly compressed via Fiat-Shamir based on sub-exponential DDH (a property referred to as FS-compatibility). This modified interactive argument $\langle \mathsf{P}, \mathsf{V} \rangle$ for T-time computations is described in Figure 1, and it relies on a protocol for T/k-time computations.

Batch NP and the Need for Local Openings. Unfortunately, the protocol described in Figure 1 is *not succinct*. In particular, each batch NP statement involves verifying an opening of the SE commitment, and therefore the verification complexity of batch NP grows with the complexity of verifying commitment openings. For this to be small, the SE commitment must satisfy an important property: namely, that it is possible to *succinctly decommit* to a part of the committed input in such a way that the size of the opening and complexity of verifying openings depend only on the part being opened, and do not grow with the size of input to the commitment. Unfortunately, such commitments are only known from the learning with errors (LWE) assumption[4]; and therefore we take a different route.

[4] In the full version of this paper, we show that one can in fact construct a commitment with somewhat succinct local openings from DDH or QR. However, these are significantly less succinct than their LWE-based counterparts, and using these commitments would lead to marginally worse parameters than one can get with the methods described next.

Emulation Phase.

1. P computes and sends $(k-1)$ intermediate configurations $(s_1, \ldots, s_{k-1})$ to V, where s_i is the configuration of machine $\mathcal{M}$ after T/k steps when starting at configuration s_{i-1}.
2. P prepares the first messages $\{m_1^{(i)}\}_{i\in[k]}$ for k interactive arguments, where the i^{th} interactive argument attests to the correctness of $(\mathcal{M}, x) : s_{i-1} \xrightarrow{T/k} s_i$. Next, P computes an SE commitment $c^{(1)}$ to these first messages, and sends the commitment string $c^{(1)}$ to V.
3. V generates a single (public coin) message for (a single copy of) the interactive argument for T/k-sized computation. All k arguments will share the same verifier message.
4. More generally, for every round $j \in [\rho]$ of the underlying interactive argument,
   - P computes the j^{th} round messages for all k interactive arguments where the i^{th} interactive argument attests to the correctness of $(\mathcal{M}, x) : s_{i-1} \xrightarrow{T/k} s_i$. Next, P computes an SE commitment $c^{(j)}$ to all these first messages, and sends the commitment string $c^{(j)}$ to V.
   - V generates a single (public coin) message for the underlying interactive argument for T/k-sized computation. All k arguments will share the same verifier message.

Batch NP Phase. P proves to V that there exists an opening of the commitment $c = (c^{(1)}, \ldots, c^{(\rho)})$ where for $i \in [k]$ the i^{th} opened value is an interactive argument such that:

1. The commitment verifier would accept the opening and
2. The verifier for the T/k interactive argument would accept the i^{th} argument.

Fig. 1. Recursively Defined Interactive Argument for Bounded Space Deterministic Computation

Coincidentally, in the *interactive arguments* for batch NP due to [21], the first step requires the prover to *commit* to witnesses $(w_1, \ldots, w_k)$ corresponding to each of the k instances $(x_1, \ldots, x_k)$. This is done via an SE commitment in such a way that when the commitment key is binding at index $i \in [k]$, the extraction algorithm outputs the i^{th} committed witness w_i. Moreover, this commitment does not need to have local openings; somewhere extractability suffices[5]. Finally,

[5] We remark that [21] also require some additional linear homomorphism properties from the commitment, but these are not necessary for our discussion.

the [21] protocol is actually an *argument of knowledge for one of the instances*: implicit in their proof is the fact that when the SE commitment keys (in the CRS) are binding on index i, no efficient prover can commit to w_i that is a non-witness for x_i and produce an accepting transcript (except with negligible probability).

This gives us a way out: in the Batch NP phase of our protocol, instead of proving that *there exists an opening to the commitment*, we omit sending commitments (since we already committed to all $\frac{T}{k}$ transcripts), and simply prove that for each of the transitions $s_{i-1} \to s_i$, there exists a prover strategy corresponding to verifier coins sent in the emulation phase, that would cause the verifier to accept. That is, the prover demonstrates membership of instances $(\widetilde{x}_1, \ldots, \widetilde{x}_k)$ in the language $\widetilde{\mathcal{L}}$, where for any $i \in [k]$,

$$\widetilde{x}_i = (s_{i-1}, s_i, y, \mathcal{M}, \beta)$$

and $\widetilde{\mathcal{L}}$ is the language of all such $\tilde{x}$ such that there exist prover messages that when combined with the verifier messages β create an accepting transcript. We note that an honest prover, by the end of the emulation phase in Figure 1, will already be committed to witnesses for this language.

Thus our final protocol has an emulation phase that is identical to Figure 1, but the batch NP phase is modified as described in Figure 2.

It may appear that the language $\widetilde{\mathcal{L}}$ will contain nearly all strings: since the protocol for T/k-sized computations is an *argument*, so there will exist prover messages even for instances not in the language. However, this would only be a problem if we relied on soundness of the batch NP protocol: on the other hand, we are able to use the fact that the [21] protocol is an *argument of knowledge* for the i^{th} statement when the SE commitment key is binding at index i. In particular, this means that if the SE commitment was binding at index i, then it is possible to *efficiently extract* a witness, i.e., an accepting transcript for the i^{th} subcomputation $s_{i-1} \to s_i$.

Now if the prover managed to break soundness of our protocol, this would imply that there exists an index $j \in [k]$ such that the machine $\mathcal{M}$ on input y does not transition from configuration s_{j-1} to s_j. But, if $j = i$, where i is the index where the SE commitment is binding, then one can in fact *extract* an accepting transcript for the j^{th} *incorrect* subcomputation $s_{j-1} \to s_j$. This can therefore be used to build a prover that contradicts soundness of the protocol for T/k-sized computations. Moreover, hiding of the index i ensures that $j = i$ occurs with non-negligible probability.

Finally, we point out that in the base case, i.e., for unit-time computations, the verifier simply checks the statement on its own (this takes one time-step).

The recursive protocol described so far satisfies succinctness for an appropriate choice of k (that we discuss later) but requires multiple rounds, since each round of recursion adds a few rounds of interaction. The goal of this work is to build a *non-interactive* argument, which we achieve by compressing this interactive argument to a SNARG based on correlation-intractable hash functions for low-depth threshold circuits. We discuss this in detail below.

Updated Batch NP Phase

– P and V define instances

$$(\widetilde{x}_1, \ldots \widetilde{x}_k) \text{ where } \widetilde{x}_i = (s_{i-1}, s_i, y, \mathcal{M}, \beta)$$

where β denote all verifier messages from the emulation phase.

– P additionally defines witnesses

$$(\widetilde{w}_1, \ldots \widetilde{w}_k)$$

where for every $i \in [k]$, $\widetilde{w}_i$ contains the prover messages for the i^{th} subcomputation for size $\frac{T}{k}$.

– Finally, define language

$$\widetilde{\mathcal{L}} = \{(s, s', y, \mathcal{M}, \beta) \ : \ \exists \text{ prover messages } \pi \text{ s.t.}$$
$$(\pi, \beta) \text{ is accepting transcript for } (\mathcal{M}, y) : s \xrightarrow{T/k} s'.\}$$

– P and V execute a batch NP argument to prove that for every $i \in [k]$, $\widetilde{x}_i \in \widetilde{\mathcal{L}}$, where they replace the first round of Batch NP (where prover SE-commits to witnesses) with the transcript of the emulation phase.

Fig. 2. Updated Batch NP Phase for Bounded Space Deterministic Computation

2.2 Obtaining a SNARG

We now discuss why this argument can be compressed by relying on the same CI hash functions as used in [21], leading to a sound SNARG.

Fiat-Shamir Compatible Batch NP. To soundly compress their *batch NP* interactive argument into a SNARG, the work of [21] (building on a line of recent works including [15,47,14,2,29,42,33,32]) relies on a special type of hash function, called a correlation intractable hash function. The prover generates verifier messages for the interactive protocol locally by applying this hash function to its partial transcripts, in effect eliminating the need to interact with a verifier. At a high level, a hash family $\mathcal{H}$ is correlation intractable (CI) for a relation $\mathcal{R}(x, y)$ if it is computationally hard, given a random hash key k, to find any input x such that $(x, \mathcal{H}(k, x)) \in \mathcal{R}$.

Given a CI hash function, the key observation is that if the BAD verifier challenge for the interactive argument, which allows a prover to cheat, can be computed by an *efficient* function, then replacing the verifier message by the output of a CI hash function results in a verifier message that does not allow a prover to cheat, except with negligible probability. But this paradigm is only applicable to protocols where the circuits computing BAD verifier challenges are supported by constructions of CI hash functions exist based on standard assumptions. In particular, CI hash functions from (sub-exponential) DDH are known for functions that are computable by constant (and in fact, $O(\log\log\lambda)$) depth threshold circuits. Recall that the [21] batch NP interactive argument has a first message that contains SE commitments to all witnesses; [21] show that the SE commitment they use (which they construct based on the QR assumption) allows for extraction in constant depth. Moreover, given the witness, all other computations can also be performed by constant depth threshold circuits. Therefore, their interactive arguments can be compressed based on the (sub-exponential) DDH assumption. [21] call this the *strong* FS-compatible property. We will now prove that our interactive arguments for bounded space, also inherit this property.

Fiat-Shamir Compatible Bounded Space Arguments. To begin, we assume that all cryptographic primitives (SE commitments, CI hash functions) satisfy T-security, meaning that no $\mathsf{poly}(T)$-size adversary can break the primitive with advantage better than $\mathsf{negl}(T)$.

Our interactive argument begins with P sending $(k-1)$ intermediate configurations to V. Observe that it is possible to verify (in time $\leq T$) whether or not a given intermediate configuration is correct[6]. Of course, the verifier should not be verifying intermediate configurations directly (as this will make verification inefficient).

As discussed above, a cheating prover must output at least one pair of consecutive intermediate configurations s_i, s_{i+1} such that $\mathcal{M}$ does not transition from s_i to s_{i+1} in T/k steps. Moreover, by T-index hiding of the SE commitment, if the SE commitment is set to be binding at a random index i', the probability (over the randomness of i') that the prover cheats on the ${i'}^{th}$ underlying T/k interactive argument must be (negligibly) close to $1/k$. Finally, because the SE commitment is extractable, in this mode, it becomes possible for a reduction to *extract* an accepting transcript of the underlying T/k argument corresponding to a false statement.

Peeling off the recursion just a little, we observe that the (T/k) interactive argument itself begins with the prover sending $(k-1)$ intermediate configurations, each corresponding to (T/k^2) steps of the Turing Machine $\mathcal{M}$. Again, one pair of consecutive configurations s'_j, s'_{j+1} must be such that $\mathcal{M}$ does not transition from s'_j to s'_{j+1} in (T/k^2) steps. Moreover, by index hiding of the SE commitment used in the (T/k) argument, if the (T/k) commitment is set to be binding at a uniformly random index j', the probability that the prover cheats on the ${j'}^{th}$ underlying (T/k^2) argument in addition to cheating on the ${i'}^{th}$ (T/k) argument must be (negligibly) close to $(1/k^2)$. We can recurse $\log_k T$ times all the way to the base case, where the base argument is simply a unit-time computation where the verifier checks the statement on its own. Moreover, letting π denote the unit-time protocol obtained by peeling all layers of the recursion, we

[6] This becomes somewhat non-trivial in the non-deterministic setting, which we discuss in an upcoming subsection.

can establish that with probability (close to) $(1/k^{\log_k T}) = 1/T$, π corresponds to a false statement. The rest of our analysis will be conditioned on this event.

Assuming that the base statement π (that the prover is statistically bound to) at the end of the first message is false, we must now understand the distribution of BAD verifier challenges in subsequent messages of the argument system. Note that the very next message will consist of the batch NP phase of the interactive argument for k-size computations, encrypted under $(\log_k T - 1)$ layers of SE commitments. This phase starts with commitments to k witnesses (in this case, the witnesses are empty transcripts), each one proving the correctness of one of the unit-size subcomputations. The false statement from the emulation phase immediately determines which one of the batch statements is incorrect. As long as the SE commitment is binding at this index, the BAD function at this lowest layer of recursion will correspond to the set of verifier challenges in the corresponding batch NP argument that allow the prover to cheat within that argument. This means that the BAD function can be computed by *peeling off* layers of the commitment (i.e. performing $\log_k T$ sequential extractions), and then computing the BAD function for the batch NP argument (which we know is efficiently computable by a constant-depth circuit).

Next, going back up one step, we have the protocol corresponding to k^2-sized computations. It will again be the case that assuming the SE commitment binds at the right index, the BAD function at this layer of recursion will correspond to the set of verifier challenges in the corresponding batch NP argument that allow the prover to cheat within that argument. This means that the BAD function can be computed by peeling off $\log_k T - 1$ layers of the commitment, and then computing the BAD function for the batch NP argument (which we know is efficiently computable by a constant-depth threshold circuit).

More generally, the BAD function of our protocol corresponds to extracting from upto $\log_k T$ layers of commitments, and feeding the result as input to the BAD function circuit of the interactive argument for batch NP.

Communication Complexity and Verifier Runtime. Considering now the efficiency of the verifier, we note that in the emulation phase, the verifier simply has to read prover messages and generate random strings. Thus, for our overview, it suffices to focus on the batch NP phase, as the time taken there will dominate that of the emulation phase. If we were to use a trivial batch NP protocol that simply provided all k witnesses and asked the verifier to check them all, this would mean that the run time of the T verifier would increase by a factor of k over the run time of the T/k verifier. Unrolling the recursion, unfortunately, we would obtain a T-time verifier. Luckily, we are not constrained to use only a trivial batch NP protocol; by being more efficient, we can improve upon the above analysis. Indeed, applying the batch NP described above, we can improve the k multiplicative overhead to a polynomial in λ overhead, where λ is the security parameter of our batch NP scheme.[7]

[7] For simplicity of exposition, we are here ignoring some additional additive overhead as well as polylogarithmic multiplicative factors.

By choosing k and λ such that $\lambda << k$, we can ensure that the difference in verifier efficiency over the $\log_k T$ levels between the unit protocol and the T protocol is $\lambda^{c \cdot \log_k T}$ for some constant c, which can be set to $T^{o(1)}$ by a careful choice of parameters. Since the verifier run time is an upper bound on the communication complexity of the protocol (as the verifier needs to at a minimum read all the messages), this gives us the same bound on the size of the proof.

This completes an overview of our SNARGs for deterministic bounded space computation. In what follows, we will discuss how to extend these ideas to the non-deterministic setting.

2.3 SNARGs for Bounded Space Non-Deterministic Computation

When the machine $\mathcal{M}$ is non-deterministic it is a-priori no longer clear how to argue or even define "correctness" of intermediate configurations. It may be tempting to consider defining correctness of intermediate configurations with respect to both the instance and the witness. However, the witness used can potentially change every time the prover is queried, and is therefore not well defined. It may also in general be too large to be sent as part of the SNARG.

However, inspired by [3], we observe that if the non-deterministic Turing Machine reads each bit of the witness only once, then it becomes possible to get around this barrier. Similar to [3], we consider the class $\mathsf{NTISP}(T(n), S(n))$ of all languages recognizable by nondeterministic Turing Machines in time $O(T(n))$ and space $O(S(n))$. Recall that a non-deterministic Turing Machine allows each step of the computation to non-deterministically transition to a new state. This, in a sense, corresponds to the setting where each bit of the witness is read at most once (and if the machine wishes to remember previous non-deterministic choices it must explicitly write them down on its worktape). Thus an alternative way to describe this class is as the class of languages $\mathcal{L}$ with a corresponding witness relation $R_{\mathcal{L}}$, recognizable by a layered circuit $C_{n,m}$ parameterized by $n = |x|$ and $m = m(n) = |w|$, that on input a pair (x, w) outputs 1 if and only if $R_{\mathcal{L}}(x, w) = 1$. 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 $D = O(T(n))$ and width $W = O(S(n))$, where W may be smaller than n and m.

The SNARG Construction. The construction remains largely similar to the one in the deterministic setting. The only (syntactical) difference is that Step 1 in the recursively defined interactive argument from Figure 1 is modified to send wire assignments $(W_1, \ldots W_{k-1})$ to $(k-1)$ intermediate layers of the circuit, each at a depth interval of D/k from the base layer. Next, for every $i \in [k]$, the prover runs (parallel) interactive arguments proving that there is an assignment to witness wires such that configuration W_i transitions to W_{i+1} in depth D/k.

Analysis. As discussed above, unlike the deterministic setting, it appears difficult define a notion of "correctness" of these intermediate wire assignments. Instead, inspired by [3], we define the notion of an *accepting layer.*

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 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 $\mathsf{Acc}^D(x) = 1$. The set $\mathsf{Acc}^{D-1}(x)$ contains 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 $i < (D-1)$, the set $\mathsf{Acc}^i(x)$ is defined recursively in a similar manner. That is, for $i < (D-1)$, $\mathsf{Acc}^i(x)$ is the set of all possible assignments to intermediate wires connecting gates in layer i to gates in layer $i+1$, such that when the instance wires for layer i are set consistently with x, there exists an assignment to the witness wires for layer $i+1$, such that the transition function applied to these wires outputs intermediate wires connecting layer $i+1$ to layer $(i+2)$ that lie in the set $\mathsf{Acc}^{i+1}(x)$. We note that the lowest i for which this definition is meaningful is $i = 1$, since there are no intermediate wires before the first layer.

By this definition, for $x \notin R_{\mathcal{L}}$, the set $\mathsf{Acc}^1(x)$ is empty. This implies that for any set of claimed intermediate configurations $(W_1, \ldots, W_{k-1})$ sent by P (and for $W_k = 1$), there must exist an $i \in [k-1]$ such that $W_{i+1} \in \mathsf{Acc}^{i+1}(x)$ but $W_i \notin \mathsf{Acc}^i(x)$. This means that there is *no* set of assignments to witness wires that would lead to a correct transition from W_i to W_{i+1}. This means that the prover must be cheating in the i^{th} interactive argument for T/k-time (non-deterministic) computation.

Moreover, as observed in [3], for any width W and depth D non-deterministic computation, it is possible to decide whether a set of wire assignments are in $\mathsf{Acc}^i(x)$, for any $i \in [D]$ in time $\mathsf{poly}(D, 2^W)$. This is done via a straightforward dynamic programming approach. We will set parameters so that the SE commitment is index-hiding against $\mathsf{poly}(T, 2^S)$-size adversaries. This, together with the previous claim implies that if the SE commitment is set to be binding at a random index i', the probability that the prover cheats on the i'^{th} underlying T/k interactive argument must be (negligibly) close to $1/k$. Moreover, because the SE commitment is extractable, in this mode, it becomes possible for a reduction to *extract* an accepting transcript of the underlying T/k argument for a false statement.

At this point, it becomes possible to apply the same recursive argument as in the deterministic setting to argue that with probability (negligibly) close to $1/k^{\log_k T} = 1/T$, the base argument corresponds to a false statement. Conditioned on this event, it becomes possible to analyze the batch NP phase in a manner similar to the analysis in the deterministic setting.

SNARG*s for* P*.* We rely on the recent work of [38] to compile our SNARGs for non-deterministic bounded-space computations to SNARGs for P.

To this end, we observe that for any language $L \in \mathsf{NTISP}(T, S)$, it holds that $L^{\otimes k} \in \mathsf{NTISP}(kT, S+T)$, where $L^{\otimes k}$ is the language of k instances from L. This implies SNARGs for batch NP with improved parameters than the [21] SNARGs, from sub-exponential DDH and QR. In particular, this implies SNARGs for batching k instances that have a description of size n, and proving that the batched instances are in $L^{\otimes k}$ where $L \in \mathsf{NTISP}(T, S)$, with communication complexity and verifier runtime $k^{o(1)}(n + \mathsf{poly}(T+S))$. By plugging this into a compiler of [38] from Batch SNARGs to SNARGs for P, we obtain SNARGs for T-time deterministic computations with overhead $T^{o(1)}$ from sub-exponential DDH and QR. We point out that the [38] compiler as stated also requires SE commitments that allows for committing to T values with local openings of size $\mathsf{polylog}(T)$. However, we show that for our setting of parameters, it suffices to have a weaker local opening property, where openings are of size $T^{o(1)}$. We build such commitments from any (sub-exponentially index-hiding) SE commitment without local openings, therefore obtaining our final results also from sub-exponential DDH and QR.

FS-compatible Arguments. In the body of our paper, we abstract out some general properties of our interactive arguments, and define a class of FS-compatible interactive arguments that can be soundly compressed using the Fiat-Shamir paradigm based on our technique. We show that any interactive batch NP argument that is an "FS-compatible argument" can also be converted into a proof, (intuitively) as long as its first message essentially contains a succinct commitment to witnesses for all the NP statements. We define FS-compatible interactive arguments to be those that satisfy a variant of round-by-round soundness [15] *w.r.t. a predicate.* This predicate is computed as a function of the first message of the interactive argument[8] and a trapdoor associated with the CRS. Intuitively, we will say that an interactive argument is FS-compatible w.r.t. a predicate ϕ if transcripts that satisfy the predicate, also satisfy round-by-round soundness with sparse and efficiently computable BAD verifier challenges. Moreover, in order to ensure that these arguments can be soundly converted into SNARGs based on CI hash functions, we will require that the predicate be "non-trivial". That is, any adversary that produces accepting transcripts for false statements with non-negligible probability should also produce accepting transcripts *that satisfy the predicate* and correspond to false statements, with non-trivial probability. We show the non-triviality of our predicate using the index hiding property of the underlying SE commitments.

Roadmap. A formalization of the FS-compatible property and a proof that such arguments can be converted to SNARGs can be found in Section 4. Next, in Sections 5 and 6 we formalize our constructions of SNARGs for deterministic and non-deterministic bounded-space computations, respectively. We also combine

[8] More generally, this can be computed as a function of the entire transcript.

the latter with recent work [38] to obtain SNARGs for P in Section 6.5. Due to shortage of space, we only provide constructions and theorem statements, and defer proofs to the full version of the paper.

3 Preliminaries

In what follows, when we say we assume (T_1, T_2)-hardness of an efficiently falsifiable assumption, we mean that there exists a negligible function $\mu(\cdot)$ such that no $\mathsf{poly}(T_1)$-size adversary can falsify the assumption with probability better than $\mu(T_2)$.

3.1 Correlation Intractable Hash Functions

In this section, we recall the notion of a CI hash family. We start by recalling the notion of a hash function family.

Definition 1. *A hash family $\mathcal{H}$ is associated with algorithms $(\mathcal{H}.\mathsf{Gen}, \mathcal{H}.\mathsf{Hash})$, and a parameter $n = n(\lambda)$, such that:*

– *$\mathcal{H}.\mathsf{Gen}$ is a PPT algorithm that takes as input a security parameter 1^λ and outputs a key k.*
– *$\mathcal{H}.\mathsf{Hash}$ is a polynomial time computable (deterministic) algorithm that takes as input a key $k \in \mathcal{H}.\mathsf{Gen}(1^\lambda)$ and an element $x \in \{0,1\}^{n(\lambda)}$ and outputs an element y.*

We consider hash families $\mathcal{H}$ such that for every $\lambda \in \mathbb{N}$, every key $k \in \mathcal{H}.\mathsf{Gen}(1^\lambda)$ and every $x \in \{0,1\}^{n(\lambda)}$, the output $y = \mathcal{H}.\mathsf{Hash}(k, x)$ is in $\{0,1\}^\lambda$.

Definition 2 (Correlation Intractable). *[17,15] Fix any $T_1 = T_1(\lambda) \geq \mathsf{poly}(\lambda)$ and $T_2 = T_2(\lambda) \geq \mathsf{poly}(\lambda)$. A hash family $\mathcal{H} = (\mathcal{H}.\mathsf{Gen}, \mathcal{H}.\mathsf{Hash})$ is said to be (T_1, T_2) correlation intractable (CI) for a family $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$ of efficiently enumerable relations if the following two properties hold:*

– *For every $\lambda \in \mathbb{N}$, every $R \in \mathcal{R}_\lambda$, and every $k \in \mathcal{H}.\mathsf{Gen}(1^\lambda)$, the functions R and $\mathcal{H}.\mathsf{Hash}(k, \cdot)$ have the same domain and the same co-domain.*
– *For every $\mathsf{poly}(T_1)$-size $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$ and every $R \in \mathcal{R}_\lambda$,*

$$\Pr_{\substack{k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \\ x \leftarrow \mathcal{A}(k)}} [(x, \mathcal{H}.\mathsf{Hash}(k, x)) \in R] = \mu(T_2(\lambda)).$$

We will use the following theorems from prior work.

Theorem 1. *[32] Fix any $T = T(\lambda) \geq 2^{\lambda^\epsilon}$ for some $0 < \epsilon < 1$. Assuming the (T, T)-hardness of DDH, there exists a constant $c > 0$ such that for any $B = B(\lambda) = \mathsf{poly}(\lambda)$, depth $L \leq O(\log \log \lambda)$ and any family $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$ of relations that are enumerable by threshold circuits of size $B(\lambda)$ and depth L, there exists a (T, T) correlation intractable (CI) hash family $\mathcal{H} = (\mathcal{H}.\mathsf{Gen}, \mathcal{H}.\mathsf{Hash})$ computable in time $(B(\lambda) \cdot \lambda \cdot L)^c$, for $\mathcal{R}$ (Definition 2).*

3.2 Somewhere Extractable (SE) Commitments

Definition 3 (SE Commitments). *A somewhere extractable (SE) commitment consists of PPT algorithms* $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Extract})$ *along with an alphabet* $\Sigma = \{0,1\}^{\ell_{\mathsf{blk}}}$ *and a fixed polynomial* $p = p(\cdot)$ *satisfying the following:*

- $(\mathsf{ck}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda, L, \ell_{\mathsf{blk}}, i)$*: Takes as input an integer* $L \leq 2^\lambda$*, block length* ℓ_{blk} *and integer* $i \in \{0, \dots, L-1\}$ *and outputs a public commitment key* ck *along with an extraction trapdoor* ek*.*
- $h \leftarrow \mathsf{Com}(\mathsf{ck}, x)$*: is a deterministic polynomial time algorithm that takes as input* $x = (x[0], \dots, x[L-1]) \in \Sigma^L$ *and outputs* $h \in \{0,1\}^{\ell_{\mathsf{com}}}$*.*
- $\pi \leftarrow \mathsf{Open}(\mathsf{ck}, x, i)$*: Given the commitment key* ck*,* $x \in \Sigma^L$ *and an index* $i \in \{0, \dots, L-1\}$*, outputs proof* $\pi \in \{0,1\}^{\ell_{\mathsf{open}}}$*.*
- $b \leftarrow \mathsf{Verify}(\mathsf{ck}, y, i, u, \pi)$*: Given a commitment key* ck *and* $y \in \{0,1\}^{\ell_{\mathsf{com}}}$*, an index* $i \in \{0, \dots, L-1\}$*, opened value* $u \in \Sigma$ *and a proof* $\pi \in \{0,1\}^{\ell_{\mathsf{open}}}$*, outputs a decision* $b \in \{0,1\}$*.*
- $u \leftarrow \mathsf{Extract}(\mathsf{ek}, y)$*: Given the extraction trapdoor* ek *and a commitment* $y \in \{0,1\}^{\ell_{\mathsf{com}}}$*, outputs an extracted value* $u \in \Sigma$*.*

We require the following properties:

- ***Correctness****: For any integers* $L \leq 2^\lambda$ *and* $i \in \{0, \dots, L-1\}$*, any* $\mathsf{ck} \leftarrow \mathsf{Gen}(1^\lambda, L, i)$*,* $x \in \Sigma^L$*,* $\pi \leftarrow \mathsf{Open}(\mathsf{ck}, x, j)$*: we have that* $\mathsf{Verify}(\mathsf{ck}, \mathsf{Com}(\mathsf{ck}, x), j, x[j], \pi) = 1$*.*
- ***Index Hiding****: We consider the following game between an attacker* $\mathcal{A}$ *and a challenger:*
  - *The attacker* $\mathcal{A}(1^{T_1})$ *outputs an integer* L *and two indices* $i_0, i_1 \in \{0, \dots, L-1\}$*.*
  - *The challenger chooses a bit* $b \leftarrow \{0,1\}$ *and sets* $\mathsf{ck} \leftarrow \mathsf{Gen}(1^\lambda, L, i_b)$*.*
  - *The attacker* $\mathcal{A}$ *gets* ck *and outputs a bit* b'*.*

  We say that an SE commitment satisfies (T_1, T_2) *index-hiding if for every* $\mathsf{poly}(T_1)$*-size attacker* $\mathcal{A}$ *there exists a negligible function* $\mu(\cdot)$ *such that:*

  $$\left| \Pr[\mathcal{A} = 1 | b = 0] - \Pr[\mathcal{A} = 1 | b = 1] \right| = \mu(T_2)$$

  in the above game.
- ***Somewhere Extractable****: We say that a commitment is somewhere extractable if there is a negligible function* μ *such that for every* $L(\lambda) \leq 2^\lambda$ *and* $i \in \{0, \dots L-1\}$*,*

  $$\Pr_{(\mathsf{ck},\mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda, L, i)} \left[\begin{smallmatrix} \exists y \in \{0,1\}^{\ell_{\mathsf{com}}}, \quad u \in \Sigma, \quad \pi \in \{0,1\}^{\ell_{\mathsf{open}}} \\ s.t.\ \mathsf{Verify}(\mathsf{ck}, y, i, u, \pi) = 1 \ \wedge \ \mathsf{Extract}(\mathsf{ek}, y) \neq u \end{smallmatrix} \right] = \mu(T_2)$$

Theorem 2 (SE Commitments from QR [21]). *Fix any* $T_1 = T_1(\lambda) \geq \mathsf{poly}(\lambda)$ *and* $T_2 = T_2(\lambda) \geq \mathsf{poly}(\lambda)$*. Assuming* (T_1, T_2) *hardness of QR, there exists an SE commitment satisfying Definition 3 where the extraction algorithm can be implemented by a threshold circuit of constant depth, and which satisfies* (T_1, T_2)*-index hiding. Furthermore, this satisfies the following properties:* $\ell_{\mathsf{com}} = \ell_{\mathsf{blk}}\lambda$*,* $\ell_{\mathsf{open}} = \ell_{\mathsf{blk}}L$*,* $|\mathsf{ck}| = \ell_{\mathsf{blk}}L\lambda$*,* $|\mathsf{ek}| = \ell_{\mathsf{blk}}\lambda$*, the running time of* Gen *and* Verify *is* $\ell_{\mathsf{blk}}L\lambda$ *and the running time of* $\mathsf{Extract}$ *is* $\ell_{\mathsf{blk}}\mathsf{poly}(\lambda)$*.*

4 Fiat-Shamir for Arguments

In this section, we define a class of (multi-round) interactive arguments to which the Fiat-Shamir paradigm can be soundly applied, based on an (appropriate) correlation-intractable hash function. In particular, we will define a few properties that a multi-mode interactive argument should satisfy, in order to be converted to a non-interactive one by applying our technique. We begin with a natural definition of multi-mode interactive arguments:

Definition 4 (N-Mode Protocols). *Let $N(\lambda) \geq \lambda$ be a function. We say that $\mathit{\Pi} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ is an N-mode protocol for a language $\mathcal{L}$ if the following property holds:*

- ***Syntax:*** Setup *is a randomized algorithm that obtains input a security parameter λ and some $i \in [N(\lambda)]$.* Setup *outputs common reference string* CRS *and auxiliary information* aux *such that* aux *contains i.*

Next, we define a notion of a predicate, that applies to the first prover message, the instance and a trapdoor in the CRS.

Definition 5 (Predicate). *ϕ is a predicate for an N-mode (Definition 4) protocol $\mathit{\Pi} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ if ϕ has the following property:*

- ***Syntax**: For any $i \in [N(\lambda)]$, ϕ takes as input instance x, the first prover message α_1, and some auxiliary information* aux *computed by* $\mathsf{Setup}(1^\lambda, i)$*. ϕ outputs a binary value in $\{0, 1\}$.*

Definition 6 ((T', N)-Non-Trivial Predicate). *Let $\mathit{\Pi} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be an N-mode (Definition 4) public-coin interactive proof system for a language $\mathcal{L}$. We say that a predicate ϕ for $\mathit{\Pi}$ (Definition 5) is time-T' non-trivial for $\mathit{\Pi}$ if the following properties hold:*

- ***Syntax**: For any $\lambda \in \mathbb{Z}^+$, any instance x, any $i \in [N(\lambda)]$, any $(\mathsf{CRS}, \mathsf{aux}) \in \mathsf{Support}(\mathsf{Setup}(1^\lambda, i))$, and any (partial) transcript $\tau = (\alpha_1, \beta_1, \ldots, \alpha_j)$ for some $j \in [\rho(\lambda)]$, we define $\phi(x, \tau, \mathsf{aux}) = \phi(x, \alpha_1, \mathsf{aux})$.*
- ***Non-Triviality**: There exists a polynomial $p(\cdot)$ such that for $\lambda \in \mathbb{Z}^+$, and any $\mathsf{poly}(T')$-time adversary $\mathcal{A}$, if there exists a polynomial $q(\cdot)$ such that:*

$$\Pr_{\substack{i \leftarrow [N], \\ (\mathsf{CRS}, \mathsf{aux}) \leftarrow \mathsf{Setup}(1^\lambda, i), \\ (x, \alpha_1) \leftarrow \mathcal{A}(\mathsf{CRS})}} [x \notin \mathcal{L} \wedge x \neq \bot] \geq \frac{1}{q(\lambda)}$$

then

$$\Pr_{\substack{i \leftarrow [N], \\ (\mathsf{CRS}, \mathsf{aux}) \leftarrow \mathsf{Setup}(1^\lambda, i), \\ (x, \alpha_1) \leftarrow \mathcal{A}(\mathsf{CRS})}} [\phi(x, \alpha_1, \mathsf{aux}) = 1 | x \notin \mathcal{L} \wedge x \neq \bot] \geq \frac{1}{p(N(\lambda))}.$$

- ***Efficiency**: ϕ can be evaluated in* $\mathsf{poly}(T')$ *time.*

4.1 Round-by-Round Soundness

We now define a notion of round-by-round soundness for interactive arguments w.r.t. a predicate ϕ. The definition below is a generalization of the definition in [15] to the setting of interactive arguments.

Unlike [15], we don't define State on the empty transcript, instead only starting to define it once the first prover message has been sent. The key difference from [15] is that we define the State function on the first prover message to reject when the predicate $\phi(x, \alpha_1, \mathsf{aux}) = 1$, instead of defining it to reject when $x \notin \mathcal{L}$. In particular, if we apply the definition below with the predicate $\phi(x, \alpha_1, \mathsf{aux}) = x \notin \mathcal{L}$ (and modify the syntax of Setup appropriately), we will recover the definition in [15,33].

Definition 7 (b-Round-by-Round Soundness w.r.t. ϕ). *[15] Let $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be a public-coin N-mode (Definition 4) interactive proof system for a language $\mathcal{L}$. We say that Π is b-round-by-round sound with respect to predicate ϕ (Definition 5), if there exists* State *such that, denoting the size of every verifier message by λ, for any $i \in [N(\lambda)]$, any $(\mathsf{CRS}, \mathsf{aux}) \in \mathsf{Support}(\mathsf{Setup}(1^\lambda, i))$, the following properties hold:*

1. ***Syntax:*** State *is a deterministic function that takes as input the* CRS*, an instance x, a transcript prefix τ, and auxiliary information* aux *computed by* Setup. State *outputs either* accept *or* reject.
   For every x, every non-empty transcript $\tau = (\alpha_1, \beta_1, \ldots, \alpha_j, \beta_j)$, and any next prover message α_{j+1}, we have
$$\mathsf{State}(\mathsf{CRS}, x, \tau, \mathsf{aux}) = \mathsf{State}(\mathsf{CRS}, x, \tau \| \alpha_{j+1}, \mathsf{aux}).$$
2. ***End Functionality:*** *For every x and every first prover message α_1,* $\mathsf{State}(\mathsf{CRS}, x, \alpha_1, \mathsf{aux}) = \mathsf{reject}$ *iff $\phi(x, \alpha_1, \mathsf{aux}) = 1$. For every complete transcript τ, if $\mathcal{V}(\mathsf{CRS}, x, \tau) = 1$,* $\mathsf{State}(\mathsf{CRS}, x, \tau, \mathsf{aux}) = \mathsf{accept}$.
3. ***Sparsity:*** *For every x and transcript prefix $\tau = (\alpha_1, \beta_1, \ldots, \alpha_{j-1}, \beta_{j-1}, \alpha_j)$, if $\phi(x, \alpha_1, \mathsf{aux}) = 1$ and* $\mathsf{State}(\mathsf{CRS}, x, \tau, \mathsf{aux}) = \mathsf{reject}$*, it holds that*
$$\Pr_{\beta \leftarrow \{0,1\}^\lambda}[\mathsf{State}(\mathsf{CRS}, x, \tau \| \beta, \mathsf{aux}) = \mathsf{accept}] \leq b(\lambda) \cdot 2^{-\lambda}. \tag{1}$$

4.2 FS-Compatible Arguments

In the following definition, we formalize the requirements from round-by-round sound arguments w.r.t. ϕ that allow them to be compressed by the Fiat-Shamir paradigm via our approach.

Definition 8 (FS-Compatible Multi-mode Argument with Respect to ϕ). *For some $\rho, N : \mathbb{Z}^+ \to \mathbb{Z}^+$, let $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be a ρ-round N-mode (Definition 4) public-coin interactive argument system where* Setup *is a randomized algorithm that obtains input a security parameter λ and some $i \in [N(\lambda)]$. For any $B, b, d : \mathbb{Z}^+ \to \mathbb{Z}^+$, we say that Π is (B, b, d) FS-compatible with respect to predicate ϕ (Definition 5) if the following properties hold:*

1. ***Completeness:*** *For any* $\lambda \in \mathbb{Z}^+$*,* $i \in \{1, 2, \ldots, N(\lambda)\}$*, and* $x \in \mathcal{L}$*, we have*

$$\Pr_{\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda, i)}[\langle \mathsf{P}, \mathsf{V}\rangle(\mathsf{CRS}, x) = \mathsf{accept}] = 1.$$

2. b***-Round-by-round soundness w.r.t.*** ϕ***:*** *Π is b-round-by-round sound with respect to* ϕ *(Definition 7); let* State *be the corresponding state function.*
3. d***-depth*** B***-efficient*** BAD ***w.r.t.*** ϕ***:*** *For any* $\lambda \in \mathbb{Z}^+$*, any* $i \in [N(\lambda)]$*, any* $(\mathsf{CRS}, \mathsf{aux}) \in \mathsf{Support}(\mathsf{Setup}(1^\lambda, i))$*, there exists a (non-uniform) randomized function* $\mathsf{BAD}_{\mathsf{aux}}$ *that satisfies the following guarantees:*
   - ***Syntax:*** $\mathsf{BAD}_{\mathsf{aux}}$ *is hardwired with* aux *and takes as input the* CRS*, instance* x*, a partial transcript* $\tau = (\alpha_1, \beta_1, \ldots, \alpha_i)$*; and potentially additional uniform randomness* r*.*
   - BAD ***w.r.t.*** ϕ***:*** *For every* x *and every* $\tau \triangleq (\alpha_1, \beta_1, \ldots, \alpha_{j-1}, \beta_{j-1}, \alpha_j)$ *s.t.* $\mathsf{State}(\mathsf{CRS}, x, \tau,$ $\mathsf{aux}) = \mathsf{reject}$ *and* $\phi(x, \alpha_1, \mathsf{aux}) = 1$*,* $\mathsf{BAD}_{\mathsf{aux}}(\mathsf{CRS}, x, \tau)$ *enumerates the set* $\mathcal{B}_{\mathsf{CRS}, \phi, \mathsf{aux}, \tau}$*, where*

     $$\mathcal{B}_{\mathsf{CRS}, \phi, \mathsf{aux}, \tau} := \{\beta \ : \ \mathsf{State}(\mathsf{CRS}, x, \tau \| \beta, \mathsf{aux}) = \mathsf{accept}\}.$$

     If $\mathcal{B}_{\mathsf{CRS}, \mathsf{aux}} = \emptyset$*,* $\mathsf{BAD}_{\mathsf{aux}}(\mathsf{CRS}, x, \tau)$ *outputs* $\perp$*. By Equation* (1)*,* $|\mathcal{B}_{\mathsf{CRS}, \mathsf{aux}}| \leq b(\lambda)$*.*
   - d***-Depth,*** B***-Efficient computation:*** $\mathsf{BAD}_{\mathsf{aux}}$ *can be evaluated by a* $d(\lambda)$*-depth (non-uniform) threshold circuit of size* $B = B(\lambda)$*.*

4.3 From FS-Compatible Arguments to SNARGs

In what follows, we formally state our theorem that arguments satisfying Definition 8 with respect to a non-trivial predicate (Definition 6) can be soundly compressed to obtain a SNARG; the proof of this is deferred to the full version of the paper.

Definition 9 **((T', N)-Sound Non-interactive Arguments).** *For any* $T' = T'(\lambda)$ *and* $N = N(\lambda)$*, we say that a* N*-mode protocol (Definition 4)* $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ *is a non-interactive argument for a language* $\mathcal{L}$ *if the following properties hold:*

- **Completeness:** *For any* $\lambda \in \mathbb{Z}^+$*, any* $i \in [N(\lambda)]$*, and* $x \in \mathcal{L}$*, we have that*

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda, i) \\ \tau \leftarrow \mathsf{P}(1^\lambda, \mathsf{CRS})}}[\mathsf{V}(\mathsf{CRS}, x, \tau) = \mathsf{accept}] = 1.$$

- N**-Mode Indistinguishability of CRS:** *There exists a negligible function* $\mu(\cdot)$ *such that for any* $i_1, i_2 \in [N(\lambda)]$*, and any* $\mathsf{poly}(T')$*-time adversary* $\mathcal{A}$ *we have that*

$$\left| \Pr_{\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda, i_1)}[\mathcal{A}(\mathsf{CRS}) = 1] - \Pr_{\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda, i_2)}[\mathcal{A}(\mathsf{CRS}) = 1] \right| = \mu(N(\lambda)).$$

– **Adaptive Soundness:** *There exists a negligible function $\mu(\cdot)$ such that for any $\lambda \in \mathbb{Z}^+$, any $i \in [N(\lambda)]$, and any non-uniform* $\mathsf{poly}(T')$*-time adversary $\mathcal{A}$ we have that*

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda, i) \\ (x,\tau) \leftarrow \mathcal{A}(1^\lambda, \mathsf{CRS})}} [x \notin \mathcal{L} \ \wedge \ \mathsf{V}(\mathsf{CRS}, x, \tau) = 1] \leq \mu(N(\lambda)).$$

Theorem 3 (FS-Compatible). *Suppose that there exist N, T', B, b, d (all functions of λ) where $N, T' \geq \lambda$. Let $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be a $\rho(\lambda)$-round $N(\lambda)$-mode (Definition 4) protocol for a language $\mathcal{L}$ decidable in (deterministic) time* $\mathsf{poly}(T')$. *Let Π have prover runtime T_P and verifier runtime T_V. Let $\mathcal{H}$ be a hash function. If Π and $\mathcal{H}$ are such that:*

– *Π is (B, b, d)-FS-compatible according to Definition 8 with respect to a (T', N) nontrivial predicate ϕ (Definition 6).*
– *$\mathcal{H}$ is (T', N) CI (Definition 2) for all relations sampleable by d-depth threshold circuits of size B, and is computable in time $p(B)$ for some fixed polynomial $p(\cdot)$.*

Then $\Pi_{\mathsf{FS}}^{\mathcal{H}}$, which is the protocol where every verifier message is computed by the prover by hashing the latest prover message, is a (T', N)-sound non-interactive argument system for $\mathcal{L}$ (Definition 9). $\Pi_{\mathsf{FS}}^{\mathcal{H}}$ has $\rho(\lambda) \cdot p(B(\lambda)) + T_\mathsf{P}$ prover runtime and $\rho(\lambda) \cdot p(B(\lambda)) + T_\mathsf{V}$ verifier runtime.

5 FS-compatible Arguments for Bounded Space Computations

In this section, we describe and prove FS-compatibility of our interactive arguments for bounded space computation. Before providing a formal theorem, in the following subsection, we define an FS-Compatible Batch NP argument with respect to a batch predicate and SE commitment. We will bootstrap interactive arguments for batch NP satisfying this definition to obtain interactive arguments for bounded space computation.

5.1 FS-Compatible Batch NP Arguments

Let $\Pi_\mathsf{BNP} = (\mathsf{Setup}_\mathsf{BNP}, \mathsf{P}_\mathsf{BNP}, \mathsf{V}_\mathsf{BNP})$ be a public-coin argument system for $\mathcal{R}^k$ for some circuit satisfiability relation $\mathcal{R}$ such that $\mathsf{Setup}_\mathsf{BNP}(1^\lambda, i)$ runs $(\mathsf{ck}, \mathsf{ek}) \leftarrow \mathcal{C}.\mathsf{Gen}(1^\lambda, k, i)$ for some SE commitment scheme $\mathcal{C}$ and puts ck in CRS and (i, ek) in aux. Then we define a predicate ϕ_BNP such that

$$\phi_\mathsf{BNP}((x_1, \ldots, x_k), \alpha_1, \mathsf{aux}) = \Big((x_i, \mathcal{C}.\mathsf{Extract}(\mathsf{ek}, \alpha_1)) \notin \mathcal{R}\Big).$$

Theorem 4 (FS-compatible Batch NP w.r.t. $\mathcal{C}$ [21]). *Assuming the hardness of* QR, *for any $n = n(\lambda), m = m(\lambda), s = s(\lambda), k = k(\lambda)$, and field $\mathbb{F}$ where $|\mathbb{F}| \leq 2^\lambda$ there exists an FS-compatible Batch NP w.r.t. $\mathcal{C}$ and ϕ_BNP, where $\mathcal{C}$ satisfies Definition 3, for $\mathcal{R}^k_{n,m,s,\mathbb{F}}$ where $\mathcal{R}_{n,m,s,\mathbb{F}}$ is any* $\mathsf{C\text{-}SAT}$ *relation.*

5.2 Bounded-Space Protocol Construction

For any $T \in \mathbb{N}$, consider a language $\mathcal{L}_T$ that contains the set of all strings $(\mathcal{M}, s_0, s_T, y)$ where $\mathcal{M}$ is the description of a Turing machine, s_0 is the initial state, s_T is the final state and y is an input such that running $\mathcal{M}$ on y with the start state to be s_0 for T time steps results in the final state s_T. We construct an interactive FS-compatible argument for the language $\mathcal{L}_T$.

For any $k, \gamma \geq 1$ where $k^\gamma = T$, for every $\ell \in [\gamma]$, we construct an argument for k^ℓ-time, S-space computations in Figure 4 in terms of an interactive argument for $k^{\ell-1}$-time, S-space computations.

- Let $\Pi_0 = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ denote a trivial protocol (Figure 3) for unit-time computations where the verifier given a machine $\mathcal{M}$, instance x and states s_0, s_1, outputs 1 if $\mathcal{M}(x, s_0)$ transitions to state s_1 in one time step. $\mathsf{Setup}(1^\lambda)$ outputs $(\perp, \perp)$.
- Let $\Pi_{k^{\ell-1}} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be a ρ-round public-coin protocol for $(k^{\ell-1})$-time computations with ν-length prover messages whose verifier $\mathsf{V} = (\mathsf{V}_1, \ldots, \mathsf{V}_\rho)$ where $r^{(i)} \leftarrow \mathsf{V}_i(1^\lambda, |x|)$ for $i \in [\rho - 1]$ and $\{0,1\} \leftarrow \mathsf{V}_\rho(x, \tau)$ for transcript τ.
- Let $\mathcal{C} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Extract})$ be an SE commitment satisfying Definition 3.
- Let Π_{BNP} be a batch NP protocol for circuit satisfiability that is FS-compatible with respect to the commitment $\mathcal{C}$ (see section 5.1 in the full version of the paper for a formal definition of FS-compatibility for batch NP protocols).

Unit Time Interactive Protocol

P and V obtain an instance $x^{(0)}$, which P wishes to prove is in the language

$$\mathcal{L}^{(0)} \triangleq \Big\{(\mathcal{M}, y, s_0, s_1) \ : \ s_1 \leftarrow \mathcal{M}(y, s_0, 1)\Big\}.$$

That is, $\mathcal{M}$ with initial state s_0 reaches state s_1 in one time step on input y.

1. P sends dummy message α to V.
2. V sends dummy message β to P.
3. V computes $s_1' \leftarrow \mathcal{M}(y, s_0)$. V accepts iff $s_1' = s_1$.

Fig. 3. Unit Time Interactive Protocol $(\mathsf{Setup}, \mathsf{P}, \mathsf{V})$

Interactive Argument for k^ℓ-Time S-Space Computation

Common Input: The common input for P and V is an instance $x = (\mathcal{M}, s_0, s_T, y)$ of the language $\mathcal{L}_{k^\ell}$. $\mathsf{Setup}(1^\lambda, i, k)$:

- Parse i as a tuple $(i_1, i_2, \ldots, i_\ell) \in [k]^\ell$.
- Obtain $(\mathsf{ck}, \mathsf{ek}) \leftarrow \mathcal{C}.\mathsf{Gen}(1^\lambda, i_\ell, k)$ and $(\mathsf{CRS}', \mathsf{aux}') = \mathit{\Pi}_{k^{\ell-1}}.\mathsf{Setup}(1^\lambda, (i_1, \ldots, i_{\ell-1}), k)$.
- Output $\mathsf{CRS} = (\mathsf{CRS}', \mathsf{ck}), \mathsf{aux} = (\mathsf{aux}', \mathsf{ek}, (i_1, \ldots, i_\ell))$.

- **Initial Processing.** P computes $s = (s_0, \ldots, s_k)$ for initial state s_0 and $\{s_j \triangleq \mathcal{M}\left(y, s_0, 1^{T \cdot j/k}\right)\}_{j \in [k]}$. P sends s to V.
- P and V define k instances $(x'_1, \ldots, x'_k)$ for language $\mathcal{L}_{k^{\ell-1}}$ where $x'_j = (\mathcal{M}, s_{j-1}, s_j, y)$ for $j \in [k]$.

- **Emulation Phase.** For every $r \in [1, \rho]$, let ν denote the maximum message size of $\mathit{\Pi}_{k^{\ell-1}}.\mathsf{P}$. P computes k parallel executions of $\mathit{\Pi}_{k^{\ell-1}}.\mathsf{P}$'s r^{th} round message, $\mathit{\Pi}_{k^{\ell-1}}.\mathsf{P}_r$:

$$\pi^{(r)} = \begin{bmatrix} | & & | \\ \pi^{(r)}[1] & \cdots & \pi^{(r)}[\nu] \\ | & & | \end{bmatrix}$$
$$\triangleq \begin{bmatrix} — \pi_1^{(r)} \triangleq \mathit{\Pi}_{k^{\ell-1}}.\mathsf{P}_r(\mathsf{CRS}', x'_1, \mathcal{L}_{k^{\ell-1}}, \{\beta^{(1)}, \ldots, \beta^{(r-1)}\}) — \\ \cdots \\ — \pi_k^{(r)} \triangleq \mathit{\Pi}_{k^{\ell-1}}.\mathsf{P}_r(\mathsf{CRS}', x'_k, \mathcal{L}_{k^{\ell-1}}, \{\beta^{(1)}, \ldots, \beta^{(r-1)}\}) — \end{bmatrix}.$$

P sends $C^{(r)} = (C^{(r)}[1], \ldots, C^{(r)}[\nu])$ where $C^{(r)}[j] \triangleq \mathcal{C}.\mathsf{Com}(\mathsf{ck}, \pi^{(r)}[j])$ for $j \in [\nu]$.
- V sends $\mathit{\Pi}_{k^{\ell-1}}.\mathsf{V}$'s r^{th} round message computed as $\beta^{(r)} \leftarrow \mathit{\Pi}_{k^{\ell-1}}.\mathsf{V}_r(1^\lambda)$.

- **Batch NP Phase.** P and V define the instances $(x''_1, \ldots, x''_k)$ and P defines the witnesses $(\omega''_1, \ldots, \omega''_k)$ as follows:
For $j \in [k]$, $x''_j = (x'_j, \{\beta^{(r)}\}_{r \in [\rho]})$, $\omega''_j = \{\pi_j^{(r)}\}_{r \in [\rho]}$.
- Define language $\mathcal{L}'' \triangleq$
$$\Big\{(x, \{\beta_r\}_{r \in [\rho]}) : \exists \{\pi_r\}_{r \in [\rho]} \text{ s.t. } \mathit{\Pi}_{k^{\ell-1}}.\mathsf{V}(\mathsf{CRS}', x, \{\pi_r, \beta_r\}_{r \in [\rho]}) = 1\Big\}.$$
- P and V execute $\mathit{\Pi}_{\mathsf{BNP}}$ on input ck, instances $(x''_1, \ldots, x''_k)$ and witnesses $(\omega''_1, \ldots, \omega''_k)$ where the first round message of P in $\mathit{\Pi}_{\mathsf{BNP}}$ is ignored and replaced by $\{C^{(r)}\}_{r \in [\rho]}$ as sent in the emulation phase.
- If $\mathit{\Pi}_{\mathsf{BNP}}.\mathsf{V}$ accepts, then V accepts.

Fig. 4. Bounded Space Computation Protocol $\mathit{\Pi}_{k^\ell}$ w.r.t. $\mathit{\Pi}_{\mathsf{BNP}}$ and $\mathcal{C}$

5.3 Non-trivial predicate for Bounded-Space Protocol

We start with the description of the predicate ϕ for the protocol Π_{k^γ}. Let $\Pi_T = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be the protocol defined by Figure 4, where $T = k^\gamma$. The predicate ϕ equals ϕ_γ, where ϕ_ℓ is defined recursively for every x, α, aux and $\ell \in [\gamma]$.

- $\phi_0(x, \alpha, \mathsf{aux}) = 1 \iff x \notin \mathcal{L}^{(0)}$.
- $\phi_\ell(x, \alpha, \mathsf{aux})$ for $\ell \in [1, \gamma]$: Parse $\mathsf{aux} = (\mathsf{aux}', ek, (i_1, \ldots, i_\ell))$ and parse $\alpha = ((s_0, \ldots, s_k), C^{(1)})$. Define instances $(x'_1, \ldots, x'_k)$ as in Figure 4, where $x'_j = (\mathcal{M}, s_{j-1}, s_j, y)$ for $j \in [k]$.
  Set $\phi_\ell(x, \alpha, \mathsf{aux}) = (x'_{i_\ell} \notin \mathcal{L}_{k^{\ell-1}}) \wedge \phi_{\ell-1}(x'_{i_\ell}, \mathcal{C}.\mathsf{Extract}(\mathsf{ek}, C^{(1)}), \mathsf{aux}')$.

Theorem 5 (Non-trivial predicate). *For every $T = T(\lambda), T' = T'(\lambda)$, assuming the (T', T)-index hiding property of SE commitments, ϕ is a (T', T)-non-trivial predicate for the protocol Π_T.*

5.4 FS-Compatibility for Bounded-Space Protocol

Theorem 6 (FS-Compatibility w.r.t. Predicate ϕ). *Let $\mathcal{C}$ be a somewhere extractable commitment (Definition 3) with security parameter λ whose extraction algorithm* Extract *has depth d_{Extract} and size B_{Extract}. Suppose that Π_{BNP} is a k-mode $(B_{\mathsf{BNP}}, b_{\mathsf{BNP}}, d_{\mathsf{BNP}})$-FS-compatible batch NP argument with respect to $\mathcal{C}$ and ϕ_{BNP} for some $B_{\mathsf{BNP}}, b_{\mathsf{BNP}}, d_{\mathsf{BNP}}, k$ that are all functions of λ.*

Then for any $T = T(\lambda) \geq \lambda$ and $k = k(\lambda)$, Π (Fig. 4) is a T-mode (B, b, d)-FS-compatible argument (Definition 8) with respect to the predicate ϕ, where

$$B = \log_k T \cdot B_{\mathsf{Extract}} + B_{\mathsf{BNP}}, \quad b = b_{\mathsf{BNP}}, \quad d = \log_k T \cdot d_{\mathsf{Extract}} + d_{\mathsf{BNP}}.$$

Furthermore, Π has both communication complexity and verifier complexity $|\Pi_{k^\gamma}.\mathsf{V}| = (kS + |y|) \cdot (\lambda \cdot \log(kS + |y|))^{O(\gamma)}$ and prover complexity $\mathsf{poly}(k^\gamma)$ for a fixed polynomial $\mathsf{poly}(\cdot)$.

Corollary 1. *Assuming the subexponential hardness of QR and subexponential hardness of DDH, there exists a* SNARG *for any time-T space-S deterministic computation with $\left(T^{\frac{c}{\sqrt{\log\log\log T}}} \cdot (S + n)\right)$ verifier runtime and communication complexity, as well as $\mathsf{poly}(T, S)$ prover runtime, where n denotes the size of the input, and c is a constant > 0.*

6 FS-compatible Arguments for Non-Deterministic Bounded Space

We now describe our interactive arguments for $\mathsf{NTISP}(T(n), S(n))$, which is the class of all languages recognizable by non-deterministic Turing Machines in time $T(n)$ and space $S(n)$. Such a 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[9]. 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 $O(S(n))$ space is used, and that runs in $O(T(n))$ time.

First, we introduce some notation and provide some background on NTISP computations. The following subsection closely mirrors [3].

6.1 Background.

Fix any $L \in \mathsf{NTISP}(T(n), S(n))$. Denote by $\mathcal{R}_L$ its corresponding NP relation, and denote by $M = M_L$ a $T(n)$-time $S(n)$-space (non-deterministic) Turing machine for deciding L. M can alternately be defined as a 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 $C^M_{n,m}$. Any such Turing machine M can be converted into a layered circuit, denoted by $C^M_{n,m}$, 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, $C^M_{n,m}$ is a layered circuit, with $W = O(S(n))$ denoting the maximum of the number of gates and number of wires in each layer, and depth $D = O(T(n))$, such that an incoming wire to a gate in layer $i + 1$ is either an input wire (i.e. a wire that reads the input), a witness wire (i.e. a wire that is attached to a trivial witness gate with fan-in and fan-out 1 whose output equals its input, and whose input wire reads the witness), or the output wire of a gate in layer i. Moreover, any witness gate has fan-out 1 (this corresponds to read-once access to the witness tape), and any layer of the circuit reads at most one (unique) bit from the witness tape. In addition, there is a deterministic Turing machine M' of space $O(\log T)$ that on input n outputs the (description of the) circuit $C^M_{n,m}$.

6.2 Interactive Arguments for Bounded Space Non-Deterministic Computation.

For any $\ell \geq 1$, we construct an interactive argument that proves correctness of wire assignments to layered circuits $\mathsf{Ckt}_{n,m}$ where $n = |x|$ and $m = |w|$ that are of the form described above (i.e. corresponding to computations of a Turing Machine M). We will assume that $\mathsf{Ckt}_{n,m}$ has depth $D = k^\ell$ and width W, and describe an interactive argument in terms of an interactive argument for $k^{\ell-1}$-depth, W-width circuits. We will prove in subsequent sections that this protocol is FS-compatible.

[9] 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.

– Let $\Pi_0 = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ denote a trivial protocol for unit-depth circuits where the prover sends a dummy message followed by a dummy verifier message, and the verifier given a circuit $\mathsf{Ckt}_{n,1}$ with a single layer, instance x s.t. $|x| = n$ and states s_0, s_1, outputs 1 iff $s_1 = \mathsf{Ckt}_{n,1}(x, 0, s_0, 1)$ or $s_1 = \mathsf{Ckt}_{n,1}(x, 1, s_0, 1)$. $\mathsf{Setup}(1^\lambda)$ outputs $(\perp, \perp)$.
– Let $\Pi_{k^{\ell-1}} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be a ρ-round public-coin protocol for $(k^{\ell-1})$-time computations with ℓ-length prover messages whose verifier $\mathsf{V} = (\mathsf{V}_1, \ldots, \mathsf{V}_\rho)$ where $r^{(i)} \leftarrow \mathsf{V}_i(1^\lambda, |x|)$ for $i \in [\rho - 1]$ and $\{0, 1\} \leftarrow \mathsf{V}_\rho(x, \tau)$.
– Let $\mathcal{C} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Extract})$ be an SE commitment satisfying Definition 3.
– Let Π_{BNP} be a batch NP protocol for circuit satisfiability.

For any $D \in \mathbb{N}$, consider language $\mathcal{L}_D$ defined by the NP relation $\mathcal{R}_{\mathcal{L}_D}$ where $\mathcal{R}_{\mathcal{L}_D}(x, w) = 1$ iff $x = (M', n, s^n, s^D, y)$ where M' outputs a description of a circuit Ckt such that s^n is a set of wire assignments to the intermediate and input wires in the n^{th} layer of the circuit, y and w respectively define assignments to all input and witness wires in the circuit, and s^D is a set of consistent wire assignments to intermediate and input wires of the circuit at layer $D + n$.

Our argument is identical to the one in Figure 4, except for the following (syntactic) changes to the inputs of both players, and to the initial processing.

Inputs. The common input for P and V is an instance $x = (M', s_0, s_T, y)$ of the language $\mathcal{L}_{k^\ell}$. P also obtains a witness ω, such that $(x, \omega) \in \mathcal{R}_{\mathcal{L}_D}$.

Initial Processing.

– P sends $s = (s_0, \ldots, s_k)$ for $\{s_i \triangleq \mathsf{Ckt}\,(y, \omega, s_0, iD/k)\}_{i \in [k]}$ to V.
– P and V define instances $(x_1, \ldots, x_k)$ where $\{x_i \triangleq (M', \frac{(i-1)D}{k}, s_{i-1}, s_i, y)\}_{i \in [k]}$ of the language $\mathcal{L}_{k^{\ell-1}}$.
– P partitions ω into witnesses $\omega_1, \ldots, \omega_k$ where for all $i \in [k]$, ω_i is used to generate assignments to witness wires in layers $\frac{(i-1)D}{k}$ through $\frac{iD}{k}$.

6.3 Non-Trivial Predicate

Let $\Pi_T = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be the protocol defined by Figure 4, where $D = k^\gamma$, and with modifications from the previous section. The predicate ϕ equals ϕ_γ, where ϕ_ℓ is defined recursively for every x, α, aux and $\ell \in [\gamma]$.

– $\phi_0(x, \alpha, \mathsf{aux}) = 1 \iff x \notin \mathcal{L}^{(0)}$.
– $\phi_\ell(x, \alpha, \mathsf{aux})$ for $\ell \in [1, \gamma]$: Parse $\mathsf{aux} = (\mathsf{aux}', ek, (i_1, \ldots, i_\ell))$ and parse $\alpha = ((s_0, \ldots, s_k), C^{(1)})$. Define instances $(x'_1, \ldots, x'_k)$ as in Figure 4, where $x'_j = (\mathcal{M}, s_{j-1}, s_j, y)$ for $j \in [k]$. Finally, set $\phi_\ell(x, \alpha, \mathsf{aux}) = (x'_{i_\ell} \notin \mathcal{L}_{k^{\ell-1}}) \wedge \phi_{\ell-1}(x'_{i_\ell}, \mathcal{C}.\mathsf{Extract}(\mathsf{ek}, C^{(1)}), \mathsf{aux}')$.

Theorem 7 (Non-trivial predicate). *Assuming the $(T \cdot 2^S, T)$-index hiding property of SE commitments, ϕ is a $(T \cdot 2^S, T)$-non-trivial predicate for the protocol Π_T where $T = k^\gamma$.*

The proof of this theorem builds on [3] and appears in the full version.

6.4 FS-Compatibility w.r.t. Predicate Φ

Theorem 8 (FS-**Compatibility w.r.t. Predicate** ϕ). *Let $\mathcal{C}$ be a somewhere extractable commitment (Definition 3) whose extraction circuit has depth d_{Extract} and size B_{Extract}. Let Π_{BNP} be a k-mode ρ-round $(B_{\mathsf{BNP}}, b_{\mathsf{BNP}}, d_{\mathsf{BNP}})$-FS-compatible batch NP with respect to $\mathcal{C}$, for some $B_{\mathsf{BNP}}, b_{\mathsf{BNP}}, d_{\mathsf{BNP}}, k$ (all functions of λ).*

Then for any $T = T(\lambda) \geq \lambda$ and $k = k(\lambda)$, Π defined above is a T-mode (B, b, d)-FS-compatible argument (Definition 8) with respect to the predicate ϕ defined above, where $B = \log_k T \cdot B_{\mathsf{Extract}} + B_{\mathsf{BNP}}$, $\quad b = b_{\mathsf{BNP}}$, $\quad d = \log_k T \cdot d_{\mathsf{Extract}} + d_{\mathsf{BNP}}$.

Furthermore, Π has communication complexity and verifier runtime $|\Pi_{k^\gamma}.\mathsf{V}| = (kS + |y|) \cdot (\lambda \cdot \log(kS + |y|))^{O(\gamma)}$ and prover runtime $\mathsf{poly}(T)$ given the witness, for a fixed polynomial $\mathsf{poly}(\cdot)$.

The proof of this theorem follows identically to that of Theorem 6. In particular, the proofs of completeness, round-by-round soundness, FS-compatibility and efficiency follow identically to that of Theorem 6. We obtain the following corollaries of this theorem.

Corollary 2. *Assuming the $(T \cdot 2^S, T)$-hardness of QR, for any time-T space-S non-deterministic computation, there is a T-mode (B, b, d)-FS-compatible argument (Definition 8) w.r.t. a $(T \cdot 2^S, T)$ non-trivial predicate ϕ, where each verifier message is of size λ (which also denotes the security parameter), and where verifier runtime and communication complexity are bounded by $T^{\frac{c}{\sqrt{\log\log\log T}}} \cdot (S + |y|)$, c is a constant > 0, $|y|$ denotes the size of the input and where $\lambda = T^{\frac{1}{\log\log\log T}}$, where $B = T^{\frac{c}{\sqrt{\log\log\log T}}}, b = \mathsf{poly}(\lambda), d = O(\sqrt{\log\log\log T})$.*

Proof. We set λ such that $\lambda^\gamma = k$, this implies that $T = \lambda^{\gamma^2}$, and $\log T = \gamma^2 \log \lambda$. We also set $\gamma^2 = \log\log\log T$, This implies that $\lambda = T^{\frac{1}{\gamma^2}} = T^{\frac{1}{\log\log\log T}}$, and $\log T < \log^2 \lambda$. Substituting, this implies that there is a constant $c > 0$ such that

$$|\Pi_T.\mathsf{V}| \leq (kS + |y|) \cdot (k^c)$$

which implies that there is a constant $c > 0$ such that

$$|\Pi_T.\mathsf{V}| \leq T^{\frac{c}{\sqrt{\log\log\log T}}} \cdot (S + |y|)$$

Finally, we note that $\lambda = T^{\frac{1}{\log\log\log T}}$, which completes our proof.

The following Corollary follows from Corollary 2, Theorem 3 and Theorem 1, where we set the security parameter $\lambda = \max(S^{1/\epsilon}, T^{\frac{1}{\log\log\log T}})$, where ϵ is the smaller of the subexponential parameters for QR/DDH hardness. Then, $(2^{\lambda^\epsilon}, 2^{\lambda^\epsilon})$-hardness of QR implies the conditions of the corollary above. In addition, $(2^{\lambda^\epsilon}, 2^{\lambda^\epsilon})$-hardness of DDH implies the conditions of Theorem 1.

Corollary 3. *Assuming the subexponential hardness of QR and subexponential hardness of DDH, there exists a SNARG for any time-T space-S non-deterministic computation with verifier runtime and communication complexity $T^{\frac{c}{\sqrt{\log\log\log T}}} \cdot (S + n)$ and prover runtime $\mathsf{poly}(T, S)$ given the witness, where n denotes the size of the input, and c is a constant > 0.*

We also obtain the following corollary about improved SNARGs for Batch NTISP, which follows from the observation that if $L \in \mathsf{NTISP}(T, S)$, then $L^{\otimes k} \in \mathsf{NTISP}(kT, S)$, where $L^{\otimes k}$ is the language containing k instances of L. This is because we can verify the k different instances by verifying each one individually in time T, and reusing the same workspace for every instance.

Corollary 4. *For every $L \in \mathsf{NTISP}(T, S)$ and every $k \geq S$, assuming the sub-exponential hardness of QR and DDH, there exists a SNARG for $L^{\otimes k}$ where verifier runtime and communication complexity are bounded by $(kT)^{\frac{c}{\sqrt{\log\log\log kT}}} \cdot (S + n)$ and prover runtime is $\mathsf{poly}(k, T, S)$ given the NP witnesses where n denotes the size of the claimed (potentially succinctly described) instance of $L^{\otimes k}$, and $c > 0$ is a constant.*

6.5 SNARGs for P and beyond

Given the SNARG for batch-NTISP above, we can use methods from [38] to build a SNARG for any language decidable in deterministic time T (and in fact, any language that has a no-signaling PCP, just as in [38]). We instantiate what is essentially their approach with different parameters, specifically, while they obtain $\mathsf{polylog}(T)$ overhead from sub-exponential LWE, we obtain $T^{o(1)}$ overhead from sub-exponential DDH and QR. Thus, we have:

Corollary 5. *Let $\mathcal{L}$ be a language and $T = T(n)$ be a function such that $\mathsf{poly}(n) \leq T(n) \leq \exp(n)$ and $\mathcal{L} \in \mathsf{DTIME}(T)$. Then assuming the subexponential hardness of QR and DDH, there exists a SNARG for $\mathcal{L}$ with prover time $\mathsf{poly}(T)$, verifier time $n \cdot \mathsf{poly}\left(T^{o(1)}\right)$, and communication complexity $n \cdot \mathsf{poly}\left(T^{o(1)}\right)$.*

References

1. Ananth, P., Chen, Y., Chung, K., Lin, H., Lin, W.: Delegating RAM computations with adaptive soundness and privacy. In: TCC. pp. 3–30 (2016)
2. Badrinarayanan, S., Fernando, R., Jain, A., Khurana, D., Sahai, A.: Statistical ZAP arguments. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT. Lecture Notes in Computer Science, vol. 12107, pp. 642–667. Springer (2020)
3. Badrinarayanan, S., Kalai, Y.T., Khurana, D., Sahai, A., Wichs, D.: Succinct delegation for low-space non-deterministic computation. In: STOC. pp. 709–721 (2018)
4. Barak, B.: How to go beyond the black-box simulation barrier. In: FOCS. pp. 106–115 (2001)
5. Bartusek, J., Bronfman, L., Holmgren, J., Ma, F., Rothblum, R.D.: On the (in)security of kilian-based snargs. In: Hofheinz, D., Rosen, A. (eds.) TCC. Lecture Notes in Computer Science, vol. 11892, pp. 522–551. Springer (2019)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM Conference on Computer and Communications Security. pp. 62–73. ACM (1993)
7. Bitansky, N., Canetti, R., Chiesa, A., Goldwasser, S., Lin, H., Rubinstein, A., Tromer, E.: The hunting of the SNARK. IACR Cryptology ePrint Archive **2014**, 580 (2014), `http://eprint.iacr.org/2014/580`

8. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKS and proof-carrying data. In: STOC. pp. 111–120 (2013)
9. Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct non-interactive arguments via linear interactive proofs. In: TCC. pp. 315–333 (2013). https://doi.org/10.1007/978-3-642-36594-2_18, `http://dx.doi.org/10.1007/978-3-642-36594-2_18`
10. Bitansky, N., Garg, S., Lin, H., Pass, R., Telang, S.: Succinct randomized encodings and their applications. IACR Cryptology ePrint Archive **2015**, 356 (2015)
11. Bitansky, N., Kalai, Y.T., Paneth, O.: Multi-collision resistance: a paradigm for keyless hash functions. In: Diakonikolas, I., Kempe, D., Henzinger, M. (eds.) STOC. pp. 671–684. ACM (2018)
12. Brakerski, Z., Holmgren, J., Kalai, Y.T.: Non-interactive delegation and batch NP verification from standard computational assumptions. In: STOC. pp. 474–482 (2017)
13. Brakerski, Z., Kalai, Y.: Witness indistinguishability for any single-round argument with applications to access control. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC. vol. 12111, pp. 97–123. Springer (2020)
14. Brakerski, Z., Koppula, V., Mour, T.: NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO. Lecture Notes in Computer Science, vol. 12172, pp. 738–767. Springer (2020)
15. Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D., Wichs, D.: Fiat-shamir: from practice to theory. In: Charikar, M., Cohen, E. (eds.) STOC. pp. 1082–1090. ACM (2019)
16. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM **51**(4), 557–594 (2004)
17. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM **51**(4), 557–594 (2004)
18. Canetti, R., Holmgren, J.: Fully succinct garbled RAM. In: ITCS. pp. 169–178. ACM (2016)
19. Canetti, R., Holmgren, J., Jain, A., Vaikuntanathan, V.: Succinct garbling and indistinguishability obfuscation for RAM programs. In: STOC. pp. 429–437. ACM (2015)
20. Chen, Y., Chow, S.S.M., Chung, K., Lai, R.W.F., Lin, W., Zhou, H.: Cryptography for parallel RAM from indistinguishability obfuscation. In: ITCS. pp. 179–190. ACM (2016)
21. Choudhuri, A.R., Jain, A., Jin, Z.: Non-interactive batch arguments for NP from standard assumptions. IACR Cryptol. ePrint Arch. **2021**, 807 (2021), `https://eprint.iacr.org/2021/807`
22. Choudhuri, A.R., Jain, A., Jin, Z.: Snargs for P from LWE. IACR Cryptol. ePrint Arch. p. 808 (2021), `https://eprint.iacr.org/2021/808`
23. Damgård, I., Faust, S., Hazay, C.: 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. pp. 54–74 (2012). https://doi.org/10.1007/978-3-642-28914-9_4, `http://dx.doi.org/10.1007/978-3-642-28914-9_4`
24. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: 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. pp. 626–

645 (2013). https://doi.org/10.1007/978-3-642-38348-9_37, http://dx.doi.org/10.1007/978-3-642-38348-9_37

25. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: STOC. pp. 99–108 (2011)
26. Goldwasser, S., Kalai, Y.T.: On the (in)security of the fiat-shamir paradigm. In: FOCS. pp. 102– (2003)
27. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: Interactive proofs for muggles. J. ACM **62**(4), 27 (2015)
28. González, A., Zacharakis, A.: Fully-succinct publicly verifiable delegation from constant-size assumptions. In: Nissim, K., Waters, B. (eds.) Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13042, pp. 529–557. Springer (2021). https://doi.org/10.1007/978-3-030-90459-3_18, https://doi.org/10.1007/978-3-030-90459-3_18
29. Goyal, V., Jain, A., Jin, Z., Malavolta, G.: Statistical zaps and new oblivious transfer protocols. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, May 10-14, 2020, Proceedings, Part III. vol. 12107, pp. 668–699. Springer (2020)
30. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 6477, pp. 321–340. Springer (2010)
31. Hubácek, P., Wichs, D.: On the communication complexity of secure function evaluation with long output. In: Roughgarden, T. (ed.) Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015. pp. 163–172. ACM (2015). https://doi.org/10.1145/2688073.2688105, https://doi.org/10.1145/2688073.2688105
32. Jain, A., Jin, Z.: Non-interactive zero knowledge from sub-exponential DDH. In: Canteaut, A., Standaert, F. (eds.) Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12696, pp. 3–32. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5_1, https://doi.org/10.1007/978-3-030-77870-5_1
33. Jawale, R., Kalai, Y.T., Khurana, D., Zhang, R.: Snargs for bounded depth computations and PPAD hardness from sub-exponential LWE. In: Khuller, S., Williams, V.V. (eds.) STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021. pp. 708–721. ACM (2021). https://doi.org/10.1145/3406325.3451055, https://doi.org/10.1145/3406325.3451055
34. Kalai, Y.T., Paneth, O.: Delegating RAM computations. In: Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II. pp. 91–118 (2016). https://doi.org/10.1007/978-3-662-53644-5_4, https://doi.org/10.1007/978-3-662-53644-5_4
35. Kalai, Y.T., Paneth, O., Yang, L.: How to delegate computations publicly. In: Charikar, M., Cohen, E. (eds.) Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019. pp. 1115–1124. ACM (2019). https://doi.org/10.1145/3313276.3316411, https://doi.org/10.1145/3313276.3316411

36. Kalai, Y.T., Raz, R., Rothblum, R.D.: Delegation for bounded space. In: Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013. pp. 565–574 (2013). https://doi.org/10.1145/2488608.2488679, http://doi.acm.org/10.1145/2488608.2488679
37. Kalai, Y.T., Raz, R., Rothblum, R.D.: How to delegate computations: the power of no-signaling proofs. In: STOC. pp. 485–494. ACM (2014)
38. Kalai, Y.T., Vaikuntanathan, V., Zhang, R.Y.: Somewhere statistical soundness, post-quantum security, and snargs. Cryptology ePrint Archive, Report 2021/788 (2021), https://ia.cr/2021/788
39. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: Proceedings of the 24th Annual ACM Symposium on Theory of Computing. pp. 723–732. ACM (1992)
40. Koppula, V., Lewko, A.B., Waters, B.: Indistinguishability obfuscation for turing machines with unbounded memory. In: STOC. pp. 419–428. ACM (2015)
41. Lipmaa, H.: Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In: TCC. pp. 169–189 (2012)
42. Lombardi, A., Vaikuntanathan, V., Wichs, D.: Statistical ZAPR arguments from bilinear maps. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12107, pp. 620–641. Springer (2020). https://doi.org/10.1007/978-3-030-45727-3_21, https://doi.org/10.1007/978-3-030-45727-3_21
43. Micali, S.: CS proofs (extended abstracts). In: 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994. pp. 436–453 (1994). https://doi.org/10.1109/SFCS.1994.365746, http://dx.doi.org/10.1109/SFCS.1994.365746, full version in [44]
44. Micali, S.: Computationally sound proofs. SIAM J. Comput. **30**(4), 1253–1298 (2000)
45. Paneth, O., Rothblum, G.N.: On zero-testable homomorphic encryption and publicly verifiable non-interactive arguments. In: TCC. pp. 283–315 (2017)
46. Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: Verifiable computation from attribute-based encryption. In: Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings. pp. 422–439 (2012). https://doi.org/10.1007/978-3-642-28914-9_24, https://doi.org/10.1007/978-3-642-28914-9_24
47. Peikert, C., Shiehian, S.: Noninteractive zero knowledge for NP from (plain) learning with errors. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11692, pp. 89–114. Springer (2019). https://doi.org/10.1007/978-3-030-26948-7_4, https://doi.org/10.1007/978-3-030-26948-7_4
48. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U.M. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 1070, pp. 387–398. Springer (1996)
49. Reingold, O., Rothblum, G.N., Rothblum, R.D.: 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. pp. 49–62 (2016). https://doi.org/10.1145/2897518.2897652, http://doi.acm.org/10.1145/2897518.2897652