SNARGs for Bounded Depth Computations from Sub-Exponential LWE

Yael Tauman Kalai* and Rachel Zhang†

July 11, 2020

Abstract

We construct a succinct non-interactive publicly-verifiable delegation scheme for any log-space uniform circuit under the sub-exponential LWE assumption, a standard assumption that is believed to be post-quantum secure. For a circuit of size S and depth D, the prover runs in time $\text{poly}(S)$, and the verifier runs in time $(D+n)\cdot S^{o(1)}$, where n is the input size. We obtain this result by slightly modifying the GKR protocol and proving that the Fiat-Shamir heuristic is sound when applied to this modified protocol. We build on the recent works of Canetti *et al.* (STOC 2019) and Peikert and Shiehian (Crypto 2020), which prove the soundness of the Fiat-Shamir heuristic when applied to a specific (non-succinct) zero-knowledge protocol.

As a corollary, by the work of Choudhuri *et al.* (STOC 2019), this implies that the complexity class PPAD is hard (on average) under the sub-exponential LWE assumption, assuming that #SAT with $o(\log n \cdot \log\log n)$ variables is hard (on average).

1 Introduction

In the past decade there have been tremendous efforts in constructing succinct and efficiently verifiable proofs. These efforts were motivated by the increasing popularity of cloud services and block-chain technologies. The question that is asked is the following: *Can one generate a short certificate for the correctness of a long computation?* Such certificates are currently used, for example, in various block-chains (such as ZCash and StarkWare) to prove the validity of transactions.

This task of constructing succinct proofs for long computations is believed to be impossible information theoretically. Indeed, all works on succinct proofs rely on some computational assumption and argue soundness of the scheme only against cheating provers who cannot break the underlying assumption. Such computationally sound proofs are referred to as *arguments* [16].

In this work we focus on constructing *succinct non-interactive arguments* (SNARGs), such as the ones needed for block-chain applications.[1] As almost all prior work, we consider the CRS model, which assumes the existence of a common reference string (CRS) known to all parties.[2] Indeed, without a CRS it is impossible to guarantee soundness against non-uniform cheating provers for schemes that are only computationally sound, since the latter implies that there exists a proof for an incorrect statement, and thus without a CRS a non-uniform adversary can simply hardwire such such a cheating proof. Our goal is to construct SNARGs under a standard cryptographic assumption which is believed to be resilient to quantum attacks. Despite extensive work in this area (which we elaborate on in Section 1.2), this goal was previously well out of reach.

*Microsoft Research and Massachusetts Institute of Technology. Email: yael@microsoft.com. This material is based upon work supported by DARPA under Agreement No. HR00112020023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Government or DARPA.

†Massachusetts Institute of Technology. Email: rachelyz44@gmail.com. Supported by the Paul E. Gray (1954) UROP Fund.

[1] In particular, we focus on SNARGs that are publicly verifiable.

[2] Our SNARG is in the common *random* string model. Namely, our CRS is a non-structured random string.

We consider the approach of constructing SNARGs by applying the Fiat-Shamir paradigm [31] to interactive delegation schemes, in particular, to the GKR protocol [37] for delegating bounded depth computations. The Fiat-Shamir paradigm is a general transformation for converting any public-coin interactive proof (or argument) to a non-interactive argument in the CRS model. It is used extensively in practice for constructing signature schemes [31], SNARGs [6, 53] and non-interactive zero knowledge proofs (NIZKs) [65].

Loosely speaking, the Fiat-Shamir transform converts an interactive proof (P, V) for a language L to a non-interactive argument (P', V') for L in the CRS model. The CRS consists of a hash function h chosen at random from some hash family $\mathcal{H}$. To compute a non-interactive proof for $x \in L$, the prover $P'(x)$ generates a transcript corresponding to $(P, V)(x)$, denoted by $(m_1, r_1, \ldots, m_{\ell-1}, r_{\ell-1}, m_\ell)$, by emulating $P(x)$ and replacing each random message r_i of the verifier by $r_i = h(m_1, r_1, \ldots, m_{i-1}, r_{i-1}, m_i)$. The verifier $V'(x)$ accepts if and only if $V(x)$ accepts this transcript and, for every $i \in [\ell - 1]$, indeed $r_i = h(m_1, r_1, \ldots, m_{i-1}, r_{i-1}, m_i)$.

This paradigm is extremely elegant and simple, and despite its abundant use in practice, its soundness is still poorly understood. It is known to be sound in the Random Oracle Model (ROM) [5, 62], yet at the same time there are counterexamples that demonstrate its insecurity when applied to interactive arguments [3, 4, 36]. Recently, there have been several works that prove its soundness when applied to interactive proofs [17–19, 44], albeit under extremely strong assumptions.

Very recently, the beautiful work of Canetti *et al.* [17] and the followup work of Peikert and Shienhian [60] prove the soundness of the Fiat-Shamir paradigm assuming standard LWE when applied to a *specific* protocol: namely, (a variant of) the three round zero-knowledge proof of graph Hamiltonicity. This result gives the first NIZK construction from LWE.

The focus of this work is on constructing SNARGs: our goal is to apply the Fiat-Shamir paradigm to a *succinct* interactive proof. The work of Canetti *et al.* mentioned above, also proves the soundness of the Fiat-Shamir paradigm when applied to the (succinct) GKR protocol, albeit under a very strong assumption: the existence of a fully homomorphic encryption (FHE) scheme that has *perfect circular security*, that is, every poly-size adversary, given $\mathsf{Enc}(\mathsf{sk})$, outputs sk with probability at most the probability of guessing (i.e., probability at most $\mathrm{poly}(\kappa) \cdot 2^{-\kappa}$, where κ is the security parameter). By contrast, in our work we prove the soundness of the Fiat-Shamir paradigm applied to (a variant of) the GKR protocol, assuming a much weaker assumption: the sub-exponential hardness of LWE.

1.1 Our Results

In this work we build upon [17, 60] and prove soundness of the Fiat-Shamir paradigm applied to (a variant of) GKR, assuming the sub-exponential hardness of LWE, an assumption that is widely believed to be true and is not known to be broken under quantum attacks.

Theorem 1.1 (Informal)**.** *The Fiat-Shamir transform is sound when applied to (a variant of) the* GKR *protocol, with respect to a specific hash family, under the sub-exponential hardness of* LWE*.*

We obtain this theorem by first proving the soundness of the Fiat-Shamir transformation when applied to the sum-check protocol. The efficiency of the resulting non-interactive sum-check degrades double-exponentially with the number of rounds in the underlying interactive sum-check protocol.

Theorem 1.2 (Informal)**.** *The Fiat-Shamir transform is sound when applied to the sum-check protocol, with respect to a specific hash family, under the sub-exponential hardness of* LWE*. The resulting non-interactive protocol has communication complexity that is $\kappa^{2^{O(m)}}$, where m is the number of rounds in the sum-check.*

The reason for this significant degradation in efficiency stems from the complexity leveraging technique that we use to prove soundness, which we elaborate on in Section 2.2. This leveraging requires us to use a hash function h_i, to replace the i'th message of the verifier, with the property that for every i, the number of bits that h_i outputs is much larger than the number of bits that h_{i-1} outputs. As a result, the communication complexity of the non-interactive sum-check protocol blows up double-exponentially with the number of rounds (see Theorem 4.4 in Section 4 for the precise statement). In particular, the SNARG we obtain is not as succinct as the interactive GKR protocol (which has communication complexity $D \cdot \mathrm{polylog}(S)$). Specifically, we obtain the following corollary.

Corollary 1.3 (Informal)**.** *Assume the sub-exponential hardness of* LWE*. Then for* $\epsilon = \omega(1/\log\log S)$*, there exists a* SNARG *for any log-space uniform circuit* C *of size* S*, depth* D*, and input size* n*, where the prover runs in time* $\mathrm{poly}(S)$ *and the verifier runs in time* $(D+n)\cdot S^{\epsilon}$*.*

We also show how to use Theorem 1.2 to obtain PPAD hardness, following the blueprint of Choudhuri *et al.* [25]. The complexity class PPAD was defined by Papadimitriou [58], and its importance, as well as the motivation for studying its hardness, stems from the fact that the problem of finding a Nash equilibrium is known to be PPAD-complete [23, 29]. Indeed, in recent years, several works established the hardness of PPAD [12, 25, 26, 30, 32, 39], albeit under strong cryptographic assumptions. We elaborate on these works in Section 1.2. We use Theorem 1.2 to obtain the following corollary.

Corollary 1.4 (Informal)**.** *Assume the sub-exponential hardness of* LWE*, and assume that* #SAT *with* $o(\log n \cdot \log\log n)$ *variables that is hard (on average). Then* CLS *is hard (on average).*[3]

We note that [25] relied on the soundness of the Fiat-Shamir transform when applied to the sum-check protocol, and on (distributional) hardness of #SAT with $\mathrm{polylog}(n)$ variables, to obtain PPAD-hardness. The reason we need to strengthen the #SAT assumption and require hardness with $o(\log n \cdot \log\log n)$ variables stems from the fact that our non-interactive sum-check protocol is not as succinct as one would hope for (due to the double-exponential growth in the round complexity). We refer the reader to Section 7 for details.

1.2 Related Work

In this section we elaborate on the related work, starting with the related work on delegating computation in Section 1.2.1, and continuing with the related work on PPAD-hardness in Section 1.2.2.

1.2.1 Related Work on Delegating Computation

Many delegation schemes have been proposed in the literature. These schemes can be roughly divided into three categories.

SNARGs. Extensive work, starting from the seminal work of Micali [54] and continuing with [7–9,28,33,38,49], construct SNARGs for non-deterministic computations. However, the soundness of these schemes is proved either in the Random Oracle Model [5] or based on non-standard hardness assumptions known as "knowledge assumptions."[4] Such assumptions have been criticized for being non-falsifiable (as in [56]) and for yielding non-explicit security reductions. We mention that some of these works form the basis of several efficient implementations which are used in practice. Other schemes (for deterministic computations) are known based on non-standard assumptions related to obfuscation [1, 10, 20, 21, 24, 48] or multilinear maps [57].

Very recently, [42] constructed a SNARG (for deterministic computations) based on an efficiently falsifiable decisional assumption on groups with bilinear maps. While this assumption seems reasonable, it is known to be broken with quantum attacks. Independently, [17] constructed a SNARG for all bounded depth computations, assuming the existence of a FHE scheme with perfect circular security. While this assumption is not known to be broken by quantum attacks, it appears to be an extremely strong assumption.

Designated verifier schemes. A line of works starting from [45, 46] and continuing with [2, 14, 15, 41] designed delegation schemes for deterministic computations and a sub-class of non-deterministic computations, based on standard assumptions (such as computational private-information retrieval). These schemes, however, are not publicly verifiable. Rather, the CRS is generated together with a secret key, which is needed in order to verify the proof.

[3] By hardness on average, we mean that there is an efficiently sampleable distribution over instances, such that the underlying problem (#SAT or a problem in CLS) is hard w.r.t. this distribution.

[4] For example, the Knowledge-of-Exponent assumption [27] asserts that any efficient adversary that is given two random generators (g, h) and outputs (g^z, h^z) must also "know" the exponent z.

Interactive schemes. In the interactive setting, we can achieve publicly verifiable schemes, even for non-deterministic computations, under standard assumptions. Kilian [47] constructed a four message protocol for any NP language with polylog communication, assuming the existence of a hash family that is collision-resistant w.r.t. sub-exponential time adversaries. It has been shown that this scheme can be converted into a three message protocol assuming a multi-collision resistant hash function [11]. The work of [59] constructs a two-message delegation scheme in addition to a (hard to compute) CRS for low-depth circuits, assuming an attribute-based encryption scheme.

Finally, we mention that in the interactive setting, we can achieve publicly verifiable schemes even unconditionally. For example, [37] and [63] give interactive delegation schemes for bounded depth and bounded space computations with unconditional soundness. As mentioned above, in this work we build on the GKR protocol from [37] to obtain our SNARG.

Despite all the above works, constructing SNARGs under standard post-quantum assumptions has remained a major open problem.

1.2.2 Related Work on PPAD Hardness

Recently, there have been a proliferation of results proving the hardness of the class PPAD, defined by Papadimitriou [58]. In his original paper, Papadimitriou suggested proving the hardness of PPAD under cryptographic assumptions. After two decades of little progress on the question, a recent sequence of works [12, 25, 26, 30, 32, 39] established the hardness of PPAD (and even that of a subclass known as CLS $\subseteq$ PPAD) under strong cryptographic assumptions. The first line of works, starting with that of Bitansky, Paneth and Rosen [12], assumes sub-exponentially secure indistinguishability obfuscation, or functional encryption [32, 39].

The second line of works, which is more relevant to us, started with the work of Choudhuri *et al.* [25] and relies on unambiguous incrementally updateable proofs. In particular, [25, 61] each construct a hard-on-average instance in the class CLS under cryptographic assumptions. The work of [25] does so assuming the adaptive soundness of the Fiat-Shamir transformation when applied to the sum-check protocol (and assuming that #SAT with polylog(n) variables is hard on average). They then use the work of Canetti *et al.* [17], which proves adaptive soundness of the Fiat-Shamir transformation applied to sum-check, to obtain PPAD hardness assuming the existence of a perfectly secure FHE. The work of [61] relies on the soundness of the Fiat-Shamir transformation when applied to Pietrzak's interactive proof for repeated squaring [61] (and assuming hardness of repeated squaring modulo a composite). Very recently, Lombardi and Vaikuntantanathan [50] proved soundness of the Fiat-Shamir transform applied to Pietrzak's interactive proof for repeated squaring, assuming LWE is $2^{\lambda^{1-\epsilon}}$-hard (w.r.t. a hash function that takes time $T(\lambda) = 2^{\lambda^\epsilon}$ time to compute), thus obtaining average-case CLS hardness under this assumption (and assuming that repeated squaring is 2^{λ^ϵ}-hard). Independently, Kalai, Paneth, and Yang [43] obtained average-case CLS hardness under a quasi-polynomial-time assumption on groups with bilinear maps (and assuming SAT with $\log(n)^{1+\epsilon}$ variables is hard on average).

In this work, we obtain average-case CLS hardness assuming the sub-exponential hardness of LWE (and assuming #SAT with $o(\log n \cdot \log\log n)$ variables is hard on average). If we weaken the #SAT assumption to a worst-case variant, namely, we assume that #SAT with $o(\log n \cdot \log\log n)$ variables is hard in the worst case, an assumption that is implied by the (non-uniform) exponential-time hypothesis (ETH) when the number of variables is $\omega(\log n)$, then we obtain worst-case CLS hardness.

2 Our Techniques

We build on ideas from [17, 60] to construct a SNARG for bounded depth computations by applying the Fiat-Shamir transformation to a modified version of GKR. In what follows, we first discuss the hash functions constructed by [17,60] and explain the difficulties with directly applying these hash functions to the sum-check and GKR protocols. We then explain our basic idea of using complexity leveraging to prove security of the

Fiat-Shamir paradigm applied to the sum-check protocol. Finally, we explain the technical hurdles that come into play when applying this leveraging technique to the original GKR protocol. Specifically, the technical problem we face is related to adaptivity. We show how to circumvent this adaptivity problem by slightly modifying the original GKR protocol.

2.1 The Hash Functions from [17, 60]

We follow the "bad-challenge function" methodology of Canetti *et al.* [17] for proving the soundness of Fiat-Shamir paradigm. In this work, we use their method to avoid *several* bad challenges.

For simplicity, consider an arbitrary 3-round proof system for some language L, and denote the transcript by (α, β, γ) (that is, the prover first sends α, the verifier replies with β, and the prover returns γ). The soundness of this 3-round proof system implies that for every $x \notin L$ and α, there is only a negligible fraction of the β's for which there exists γ such that (α, β, γ) is an accepting transcript for $x \in L$. Denote by BAD_α the set of such β's. Suppose that for every α, the set BAD_α is small, say smaller than some polynomial Q.

If the proof system has the property that given (α, β) it is easy to check if $\beta \in \mathsf{BAD}_\alpha$, then it would be easy to construct such a hash function that makes the Fiat-Shamir paradigm sound. This is done as follows: h would simply enumerate over the first $Q+1$ possible β's, and output the first β such that $\beta \notin \mathsf{BAD}_\alpha$. Unfortunately, such proof systems are often not interesting, since in a sense, they are not really using the power of interaction.

The fundamental observation of [17] is that for many interesting proof systems, this set BAD_α is easy to compute, *given some trapdoor*. Namely, there exists a trapdoor td and an efficient function f_{td} such that $f_{\mathsf{td}}(\alpha) = \mathsf{BAD}_\alpha$. It is tempting to use the hash function h_{td} that (as above) enumerates over the first $Q+1$ possible β's and outputs the first $\beta \notin \mathsf{BAD}_\alpha$. However, the trapdoor td is often unknown, or at least needs to remain unknown in some applications (such as applications to NIZKs).

Their beautiful observation is that for such proof systems, we can take a hash function that does not use td but rather uses $\mathsf{Enc}(\mathsf{td})$, where Enc is the encryption function of a homomorphic encryption scheme. Given $\mathsf{Enc}(\mathsf{td})$, the hash function on input α outputs a value not in BAD_α with probability 1. In the real scheme the hash function will use $\mathsf{Enc}(0)$, but in the analysis we can consider the case where the hash function uses $\mathsf{Enc}(\mathsf{td})$ instead of $\mathsf{Enc}(0)$, relying on semantic security to argue that these two settings are indistinguishable.

Such hash functions have recently been constructed for the case where $|\mathsf{BAD}_\alpha| = 1$ for all α. Canetti *et al.* [17] construct them from a FHE scheme that is circular secure, and the followup work of Peikert and Shienhian [60] construct them assuming the hardness of LWE.

Applying Fiat-Shamir using these hash functions. In [17, 60], these hash functions were applied (via the Fiat-Shamir transform) to special three round protocols to construct NIZKs from LWE. Since these protocols consist of only three rounds, the verifier sends a single message, and hence only one hash function is needed for the Fiat-Shamir transform. In this work, we apply the Fiat-Shamir transform to multi-round protocols, specifically, the sum-check protocol and the GKR protocol. We first discuss some issues with directly applying the hash functions from [17, 60] to these multi-round protocols.

When trying to directly apply their hash functions to the interactive sum-check protocol, we run into several issues. First, the set of BAD verifier's messages in each round has size larger than one. In fact, this set is of size d, where d is the univariate degree of the sum-check polynomial. This problem is easy to overcome by simply considering d functions $f_{\mathsf{td},1}, \ldots, f_{\mathsf{td},d}$ (per round) where $f_{\mathsf{td},j}(\alpha)$ outputs the j'th bad challenge (according to some fixed ordering). In the analysis, we assume that there is a cheating prover P^* who successfully cheats with non-negligible probability ϵ. By a standard hybrid argument, we argue that there must exist a round $i \in [m]$ (where m denotes the number of rounds) and an index $j \in [d]$ such that his partial transcript in the first i rounds hashes to the j'th bad challenge with probability ϵ/md. Thus, we can focus on a single bad challenge.

However, there is a much more substantial problem: the set BAD_i of bad verifier messages in the i'th round depends on all the previous verifier messages. Thus, any trapdoor function td_i computing BAD_i would have to take as input all these previous messages. The problem is that evaluating this trapdoor function is

intractable for the bounded verifier. In [17], they circumvent this by using a different type of hash function construction, similar in spirit to the construction proposed in the prior works of [19, 44], at the price of relying on perfect circular security.[5]

In this work, instead of computing td_i from all previous messages (which is a hard computational task), our hash function guesses these messages, and precomputes and hardwires the corresponding trapdoor. This is done in the analysis, whereas in the scheme itself, we hardwire a *fake* trapdoor, which is the all zero string. The idea is then to prove soundness by relying on semantic security. To do this, we need semantic security to hold even with probability of guessing correctly all the verifier's messages. To this end, we use a complexity leveraging technique, which we elaborate on below. We mention that this leveraging technique causes a blowup to the communication complexity: the communication complexity of our non-interactive sum-check protocol is $\kappa^{2^{O(m)}}$, where m is the round complexity of the interactive sum-check protocol (which is equal to the number of variables in the multivariate sum-check polynomial). As a result, we must take m to be constant, or very slightly super-constant such as $m = \log\log\log(n)$. This means that the communication complexity in our non-interactive GKR protocol is $D \cdot S^{\epsilon}$, whereas [17] achieves a communication complexity of $D \cdot \mathrm{polylog}(S)$.

2.2 Applying Fiat-Shamir to the Sum-Check Protocol

We first give a brief description of the interactive sum-check protocol, then discuss applying the Fiat-Shamir paradigm and our leveraging technique in doing so.

The interactive sum-check protocol. In the sum-check protocol, the prover convinces the verifier that

$$\sum_{h_1,\ldots,h_m \in H} g(h_1, \ldots, h_m) = v,$$

for some known polynomial g of degree at most d in each variable over some large field $\mathbb{F}$, and $H \subset \mathbb{F}$ is a subset. The first message from the prover is the univariate polynomial

$$g_1(\cdot) := \sum_{h_2,\ldots,h_m \in H} g(\cdot, h_2, \ldots, h_m)$$

of degree at most d. The verifier checks that indeed g_1 is of degree $\leq d$ and that $\sum_{h \in H} g_1(h) = v$. If this is not the case, it rejects. Otherwise, it sends to the prover a random $t_1 \leftarrow \mathbb{F}$, and the task is reduced to proving that

$$\sum_{h_2,\ldots,h_m \in H} g(t_1, h_2, \ldots, h_m) = g_1(t_1).$$

In the next round, the prover sends

$$g_2(\cdot) := \sum_{h_3,\ldots,h_m \in H} g(t_1, \cdot, h_3, \ldots, h_m).$$

The verifier checks that g_2 is a univariate polynomial of degree $\leq d$ and that $\sum_{h \in H} g_2(h) = g_1(t_1)$. If this is not the case, it reject. Otherwise, it sends a random $t_2 \leftarrow \mathbb{F}$, and the task is reduces to proving that

$$\sum_{h_3,\ldots,h_m \in H} g(t_1, t_2, h_3, \ldots, h_m) = g_2(t_2).$$

This continues for m rounds at the end of which the verifer checks on its own that $g_m(t_m) = g(t_1, \ldots, t_m)$ for a random $t_m \leftarrow \mathbb{F}$. We refer the reader to Section 3.2 for more details.

[5] The prior works of [19, 44] also rely on perfect security and even on stronger primitives, such as obfuscation or KDM w.r.t. arbitrary (unbounded) functions.

Applying the Fiat-Shamir paradigm to the sum-check protocol. Fix any cheating prover P^* that wishes to prove a false statement of the form $\sum_{h_1,\ldots,h_m \in H} g(h_1,\ldots,h_m) = v$. Denote P^*'s first message by $\tilde{g}_1$, and let

$$\mathsf{BAD}_1 = \{t \in \mathbb{F} : \tilde{g}_1(t) = g_1(t)\},$$

where

$$g_1(\cdot) = \sum_{h_2,\ldots,h_m \in H} g(\cdot, h_2, \ldots, h_m)$$

is the correct first polynomial. If P^* successfully proves a false statement then it must be the case that $\tilde{g}_1 \neq g_1$. If the verifier sends $t_1 \in \mathsf{BAD}_1$ then it is now easy for P^* to cheat: he can simply follow the honest prover strategy and convince the verifier of the (true) statement $\sum_{h_2,\ldots,h_m} g(t_1, h_2, \ldots, h_m) = \tilde{g}_1(t_1)$.

Note that given $\tilde{g}_1$, computing BAD_1 is inefficient, unless we are given the true g_1 as a trapdoor. Given g_1, it becomes easy to compute BAD_1 since BAD_1 is the set of roots of $g_1 - \tilde{g}_1$, and factoring can be done efficiently. So, for this first verifier message, we can use a hash function $h_{\mathsf{Enc}(\mathsf{td},j)}$ that on input $\tilde{g}_1$, with probability 1, outputs a value that is not the j'th root of $g_1 - \tilde{g}_1$, where $\mathsf{td} = g_1$. In the actual scheme, we can set $\mathsf{td}^* = 0$ (and say $j^* = 1$), and rely on semantic security to argue that if the prover cheats in the case that the CRS contains $h_{\mathsf{Enc}(\mathsf{td}^*,j^*)}$ by outputting the j'th root of $g - \tilde{g}$, then the prover can still cheat by outputting the j'th root of $g - \tilde{g}$ even if the CRS contains $h_{\mathsf{Enc}(\mathsf{td},j)}$, and thus reach a contradiction.

So far, replacing the first verifier message by a hash function has worked the same as in the NIZK constructions in [17, 60]. In the second round, we begin to run into problems. The trapdoor is now g_2, where

$$g_2(\cdot) = \sum_{h_3,\ldots,h_m \in H} g(t_1, \cdot, h_3, \ldots, h_m).$$

Note that g_2 depends on t_1, the first verifier message. Unfortunately, we do not know a priori what t_1 will be, since after applying the Fiat-Shamir transformation, t_1 depends on the value of $\tilde{g}_1$ chosen by the cheating prover, which in turn depends on the CRS. We could guess t_1 and hardwire the CRS with the corresponding trapdoor g_2. However, this strategy will only be profitable when t_1 is guessed correctly, which happens with probability $1/|\mathbb{F}|$. Thus, in order to prove soundness, we need that the underlying homomorphic encryption scheme cannot be broken with probability $1/|\mathbb{F}|$. Unfortunately, $|\mathbb{F}|$ is at least as large as the image of the hash function, which in turn is at least as large as 2^κ, where κ is the security parameter of the encryption scheme, hence no encryption scheme can have such strong security.

Complexity leveraging to the rescue. The key idea is to use different security parameters for the different hash functions. Namely, the i'th hash function h_i uses a security parameter κ_i such that $\kappa_1 < \kappa_2 < \cdots < \kappa_m$. As a result, each t_i, which is chosen from the image space of h_i, is selected from a space much larger than that for t_{i-1}. Specifically, the first hash function h_1 selects t_1 from a (small) set $H_1 \subset \mathbb{F}$, of size approximately 2^{κ_1}. This means that we can guess t_1, and hence g_2, correctly with probability roughly $2^{-\kappa_1}$. Since we need semantic security of the ciphertext hardwired in the second hash function h_2 to hold with probability roughly $2^{-\kappa_1}$, we will choose $\kappa_2 \geq {\kappa_1}^{1/\delta}$ for a small constant $\delta > 0$, and thus can argue soundness based on the sub-exponential security of the underlying encryption scheme. As a result, the second hash function h_2 selects t_2 from a larger set $H_2 \subset \mathbb{F}$ of size roughly $2^{\kappa_2} = 2^{{\kappa_1}^{1/\delta}}$. More generally, the ciphertext used in the i'th hash function h_i uses security parameter $\kappa_i \geq {\kappa_{i-1}}^{1/\delta}$, and t_i is selected from a set $H_i \subset \mathbb{F}$ of size roughly 2^{κ_i}. Using this setting of parameters, we prove soundness of the resulting non-interactive sum-check protocol based on the assumption that the underlying homomorphic encryption scheme has sub-exponential security. We refer the reader to Section 4 for more details.

Unfortunately, the m'th security parameter is $\kappa_m \geq \kappa_1^{1/\delta^m}$, which implies that $\log|\mathbb{F}| \geq \kappa_1^{1/\delta^m}$, so the communication complexity is $\geq \kappa_1^{1/\delta^m}$, which makes the scheme useless even for $m = \log\log n$ as we will need to take $\kappa_1 \geq \mathrm{poly}(m \log|H|)$. Nevertheless, as we argue this result is interesting (and useful) even for $m = O(1)$ or $m = O(\log\log\log n)$. Indeed, we manage to use this result (with this setting of parameters) to construct our SNARG (Corollary 1.3), which is obtained by applying the Fiat-Shamir to (a variant of) the GKR protocol.

2.3 Applying Fiat-Shamir to the GKR Protocol

The GKR protocol consists of a series of sum-check protocols to reduce a claim about one layer of a circuit to that of another. For a circuit C of size S, we choose $|H| = S^{1/m}$, where $m = O(1)$ or $m = \log \log \log S$ as is needed for the sum-check protocol to be tractable. The resulting m-variate polynomials that the sum-checks are performed on have degree $\text{poly}(|H|) = \text{poly}(S^{1/m})$ in each variable. This is why the communication complexity and verifier runtime are both S^{ϵ} for a slight sub-constant ϵ, as opposed to $\text{polylog}(S)$ as in the original GKR protocol.

It is tempting to replace each interactive sum-check protocol in GKR with the non-interactive one (obtained by applying the Fiat-Shamir transform), and thereby make GKR non-interactive. At first sight, it may seem that the soundness of the non-interactive sum-check protocol would imply the soundness of the non-interactive GKR protocol.

Unfortunately, when we tried to analyze the soundness of the resulting non-interactive GKR protocol, we failed for the following reason: In the original GKR protocol, the random coin-tosses of the verifier in one sum-check determine the polynomial used in the next sum-check. In the Fiat-Shamir setting, these coin tosses are replaced with a hash of the transcript so far, and thus can be chosen *adaptively* by the cheating prover. This adaptivity prevents us from proving soundness, as we do not manage to prove full adaptive soundness for the underlying non-interactive sum-check protocol.

Combating adaptivity. We manage to solve the problem of adaptivity by both strengthening the soundness requirement of the non-interactive sum-check protocol and modifying the GKR protocol to limit adaptivity. Specifically, we show that our non-interactive sum-check protocol is sound against provers who have a limited choice over g and v, a property we call *semi-adaptive soundness*. Semi-adaptivity means that v can be chosen completely adaptively, while g must be chosen from a $\text{poly}(|H|^m) = \text{poly}(S)$-size set. On the GKR protocol side, we manage to convert each sum-check statement $\sum_{h_1,\ldots,h_m \in H} g(h_1, \ldots, h_m) = v$ to *many* ($\text{poly}(|H|)$) statements $\sum_{h_1,\ldots,h_m \in H} g'(h_1, \ldots, h_m) = v'$, all of which have g' that belongs to a relatively small (size $\text{poly}(S)$) predetermined set Δ of m-variate polynomials. Thus, we limit adaptivity, at the price of converting each sum-check statement to many statements.

At first it may seem that this tradeoff is too costly, as it may seem that the number of sum-check statements will blow up exponentially. Fortunately, this is not the case. We use a recent idea from [40], which shows that if these sum-checks are run in parallel, where the verifier uses the *same randomness* in all these executions, then no blowup is incurred. We discuss our modified GKR scheme in more detail in Section 5.

Since our modified GKR protocol runs the sum-check protocol many times in parallel, where the verifier uses the same randomness in all executions, we need to argue that the Fiat-Shamir paradigm is sound when applied to the parallel sum-check protocol. This is done in Section 4.3.

2.4 Obtaining PPAD-Hardness

The soundness of the Fiat-Shamir transform applied to the sum-check protocol has implications to the hardness of the complexity class PPAD. This was observed in [25], which proved, informally, the following:

Theorem 2.1 (Informal)**.** *[25] Suppose that the Fiat-Shamir transform applied to the sum-check protocol is both adaptively sound and adaptively unambiguous. Then, the complexity class* CLS $\subseteq$ PPAD *is hard on average (assuming* #SAT *is hard on average).*

They prove this by reducing #SAT instances to instances of the Relaxed-Sink-of-Verfiable-Line (rSVL) problem, which is in CLS (we refer the reader to Definition 7.5 in Section 7 for the definition of rSVL). These rSVL instances use the non-interactive sum-check protocol, obtained by applying the Fiat-Shamir transform, and satisfy the guarantee that if one could solve these rSVL instances, then one could either solve #SAT, or break the adaptive soundness or adaptive unambiguity of the non-interactive sum-check protocol.

At first, one may hope that Theorem 2.1, together with Theorem 1.2, immediately imply PPAD hardness assuming the sub-exponential hardness of LWE (and assuming #SAT is hard on average). Indeed, the unambiguity of the Fiat-Shamir transform when applied to the sum-check protocol follows straightforwardly

from the proof of soundness (under the sub-exponential hardness of LWE). However, as we noted above, we do not guarantee adaptive soundness, rather we only guarantee adaptivity over the choice of v and some limited semi-adaptivity over the choice of g.

Nevertheless, upon inspection of the proof of Theorem 2.1 in [25], we noted that adaptivity over g wasn't needed at all. Instead, they needed adaptivity over the first j verifier messages, and required the resulting $(m-j)$-variate sum-check to be unambiguously sound. Our leveraging technique allows us to prove precisely this!

In [25], they reduce a #SAT instance corresponding to a Boolean formula with polylog(n) variables to an rSVL instance, and make the assumption that the Fiat-Shamir transformation applied to the sum-check protocol with $|H| = 2$ and $m = \text{polylog}(n)$ is sound. In our case, due to the leveraging technique we use, our non-interactive sum-check protocol requires that m is much smaller, and thus $|H|$ is much larger. In fact, we take $|H| = n$ and $m = o(\log\log n)$. Then, in the reduction from #SAT to rSVL, the Boolean formula corresponding to the #SAT instance has $o(\log n \cdot \log\log n)$ variables. Thus, to obtain PPAD-hardness (on average), we need to assume that #SAT with $o(\log n \cdot \log\log n)$ variables is hard (on average), which, when the number of variables is $\omega(\log n)$, follows from the (average case variant of the) Exponential Time Hypothesis. We obtain the following corollary:

Corollary 2.2 (Informal)**.** *Assuming* LWE *is sub-exponentially hard and* #SAT *with* $o(\log n \cdot \log\log n)$ *variables is hard (on average),* PPAD *is hard (on average).*

We discuss this result further in Section 7, and refer the interested reader to the original result proved in [25], which contains most of the ideas.

3 Preliminaries

Notation. In this paper, we use the following notation:

$$X \lesssim Y \iff X = O(Y)$$
$$X \lesssim_{\infty} Y \iff X = o(Y)$$
$$X \approx Y \iff X = \theta(Y).$$

3.1 Low Degree Extension

Let $\mathbb{F}$ be a field and let $H \subseteq \mathbb{F}$ be a set. We always assume (without loss of generality) that field operations can be performed in time that is poly-logarithmic in the field size. Fix an integer $m \in \mathbb{N}$.

A basic fact is that for any function $W : H^m \to \mathbb{F}$, there exists a unique extension of W into a function $\hat{W} : \mathbb{F}^m \to \mathbb{F}$ (which agrees with W on H^m, i.e. $\hat{W}|_{H^m} \equiv W$), such that $\hat{W}$ is an m-variate polynomial of degree at most $|H|-1$ in each variable. This function $\hat{W}$ is called the *low degree extension* of W. Moreover, the function $\hat{W}$ can be expressed as

$$\hat{W}(t_1, \ldots, t_m) = \sum_{h_1,\ldots,h_m \in H} \hat{\text{EQ}}(t_1, \ldots, t_m; h_1, \ldots, h_m) \cdot W(h_1, \ldots, h_m), \tag{1}$$

where $\hat{\text{EQ}} : \mathbb{F}^m \times \mathbb{F}^m \to \mathbb{F}$ is the low degree extension of the function $\text{EQ} : H^m \times H^m \to \{0,1\}$ that, on input $(t_1, \ldots, t_m; h_1, \ldots, h_m)$, compares $t = (t_1, \ldots, t_m)$ with $h = (h_1, \ldots, h_m)$ and outputs 1 if and only if $t = h$. We can explicitly write $\hat{\text{EQ}}$ as follows:

$$\hat{\text{EQ}}(t_1, \ldots, t_m; h_1, \ldots, h_m) = \prod_{i=1}^{m} \sum_{\nu \in H} \prod_{\sigma \in H \setminus \{\nu\}} \frac{(t_i - \nu)(h_i - \nu)}{(\sigma - \nu)^2}.$$

It's not hard to check that $\hat{\text{EQ}}$ can be evaluated in $\text{poly}(m, |H|, \log|\mathbb{F}|)$ time, is degree at most $|H|-1$ in each variable, and, restricted to the domain $H^m \times H^m$, $\hat{\text{EQ}}(t; h) = 1$ if $t = h$ and 0 otherwise.

Proposition 3.1. *Given H, $\mathbb{F}$, and an integer m, the $2m$-variate polynomial $\hat{\mathrm{EQ}} : \mathbb{F}^m \times \mathbb{F}^m \to \mathbb{F}$ can be constructed in time $\mathrm{poly}(m, |H|, \log |\mathbb{F}|)$. Furthermore, $\hat{\mathrm{EQ}}$ can be evaluated in time $\mathrm{poly}(m, |H|, \log |\mathbb{F}|)$.*

Evaluating the low degree extension of a function can be done via Equation (1) by summing over $|H|^m$ values.

Proposition 3.2. *Given H, $\mathbb{F}$, an integer m, and the values of a function $W|_{H^m}$, for any $t \in \mathbb{F}^m$, the value of $\hat{W}(t)$ can be computed in $|H|^m \cdot \mathrm{poly}(m, |H|, \log|\mathbb{F}|)$ time.*

3.2 The Sum-Check Protocol

Let $\mathbb{F}$ be a field, and let $H \subseteq \mathbb{F}$ be a subset. Let $g(x_1, \ldots, x_m)$ be a multivariate polynomial of degree at most d in each variable and with coefficients in $\mathbb{F}$, and let $v \in \mathbb{F}$.

In the sum-check protocol, a (not necessarily efficient) prover takes as input an m-variate polynomial $g : \mathbb{F}^m \to \mathbb{F}$ of degree $\leq d$ in each variable (think of d as significantly smaller than $|\mathbb{F}|$). Its goal is to convince a verifier that $\sum_{h \in \mathbb{H}^m} g(h) = v$ for some constant $v \in \mathbb{F}$. The verifier only has oracle access to g, and is given the constant $v \in \mathbb{F}$. It is required to be efficient in both its running time and its number of oracle queries. In Figure 1, we review the standard sum-check protocol from [51, 64]. We denote this protocol by $(P_{\mathsf{SC}}(g), V_{\mathsf{SC}}^g(v))$.

Sum-Check Protocol for $\sum_{h_1,\ldots,h_m \in H} g(h_1, \ldots, h_m) = v$

1. Set $i = 1$.
2. P_{SC} computes the univariate polynomial $g_i : \mathbb{F} \to \mathbb{F}$ defined by
$$g_i(x) := \sum_{h_{i+1},\ldots,h_m \in H} g(t_1, \ldots, t_{i-1}, x, h_{i+1}, \ldots, h_m),$$
and sends g_i to V_{SC}.
3. V_{SC} checks that g_i is a univariate polynomial of degree at most d, and that
$$\sum_{h \in H} g_i(h) = \begin{cases} v & i = 1 \\ g_{i-1}(t_{i-1}) & i > 1. \end{cases}$$
If not, V_{SC} rejects.
4. If $i < m$, V_{SC} chooses a random element $t_i \in_R \mathbb{F}$ and sends t_i to P_{SC}. Set $i = i + 1$ and go back to step 2.
5. If $i = m$, V_{SC} chooses a random element $t_m \in_R \mathbb{F}$ and checks that
$$g_m(t_m) = g(t_1, \ldots, t_m)$$
by querying the oracle at the point $(t_1, \ldots, t_m)$.

Figure 1: Sum-check protocol $(P_{\mathsf{SC}}(g), V_{\mathsf{SC}}^g(v))$ [51, 64]

Lemma 3.3. *[52, 64] Let $g : \mathbb{F}^m \to \mathbb{F}$ be an m-variate polynomial of degree d in each variable. The sum-check protocol $(P_{\mathsf{SC}}, V_{\mathsf{SC}})$ described in Figure 1 satisfies the following properties.*

- **Completeness:** *If $\sum_{(h_1,\ldots,h_m) \in \mathbb{H}^m} g(h_1, \ldots, h_m) = v$ then*
$$\Pr\left[(P_{\mathsf{SC}}(g), V_{\mathsf{SC}}^g(v)) = 1\right] = 1.$$

- **Soundness:** *If* $\sum_{(h_1,\ldots,h_m)\in\mathbb{H}^m} g(h_1,\ldots,h_m) \neq v$ *then for every (unbounded) interactive prover* P^**,*

$$\Pr\left[(P^*(g), V^g_{\mathsf{SC}}(v)) = 1\right] \leq \frac{md}{|\mathbb{F}|}.$$

- **Efficiency:** V_{SC} *has oracle access to* $g : \mathbb{F}^m \to \mathbb{F}$*. The prover* $P_{\mathsf{SC}}(g)$ *runs in time* $\leq \mathrm{poly}(|\mathbb{H}|^m)$.[6] *The verifier* $V^g_{\mathsf{SC}}(v)$ *runs in time* $\leq \mathrm{poly}(|\mathbb{H}|, \log|\mathbb{F}|, m, d)$*, and queries the oracle* g *at a single point. The communication complexity is* $\leq \mathrm{poly}(|\mathbb{H}|, \log|\mathbb{F}|, m, d)$*, and the total number of bits sent from the verifier to the prover is* $O(m \cdot \log|\mathbb{F}|)$*. Moreover, this protocol is public-coin; i.e., all the messages sent by the verifier are truly random and consist of the verifier's random coin tosses.*

3.3 Homomorphic Encryption

Definition 3.4. *A* homomorphic encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *w.r.t. a circuit family* $\mathcal{C} = \{\mathcal{C}_k\}_{k\in\mathbb{N}}$ *consists of four p.p.t. algorithms such that:*

- $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a public key encryption scheme.*
- $\mathsf{Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ *takes as input a public key* pk*, a circuit* $C \in \mathcal{C}_k$*, and* k *ciphertexts* $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ *(each encrypting a single bit), and outputs another ciphertext* ct' *which has size polynomial in the security parameter* λ *and in the output length of* C *(independent of the size of* C*).*
- *For any* $x = (x_1, \ldots, x_k) \in \{0,1\}^k$*, and any circuit* $C \in \mathcal{C}_k$ *there exists a negligible function* μ *such that,*

$$\Pr[\mathsf{Dec}\left(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, C, \mathsf{Enc}(\mathsf{pk}, x_1), \ldots, \mathsf{Enc}(\mathsf{pk}, x_k))\right) = C(x_1, \ldots, x_k)] \geq 1 - \mu(\lambda),$$

where the probability is over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *and over the randomness of the encryption algorithm.*

Definition 3.5. *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to have* sub-exponential security *if there exists a constant* $\delta > 0$ *such that for every (large enough) security parameter* $\lambda \in \mathbb{N}$*, and for every* 2^{λ^δ}*-size adversary* $\mathcal{A}$*,*

$$|\Pr[\mathcal{A}(\mathsf{Enc}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 0))) = 1] - \Pr[\mathcal{A}(\mathsf{Enc}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 1))) = 1]| \leq 2^{-\lambda^\delta},$$

where the probability is over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *and over the randomness of* Enc*.*

In this work we rely on the homomorphic encryption scheme of Gentry, Sahai and Waters [34] (GSW), which is known to be homomorphic w.r.t. the class of bounded depth circuits denoted by $\mathcal{C} = \{\mathcal{C}_k\}_{k\in\mathbb{N}}$ (where the length of the evaluated ciphertexts grow with the depth of the evaluated circuit), assuming the (standard) LWE assumption. Moreover, it is known to be sub-exponentially secure under the sub-exponential LWE assumption.

Theorem 3.6. *[34] The* GSW *encryption scheme has sub-expoenential security under the sub-exponential* LWE *assumption.*

The recent work of Peikert and Shiehian [60] showed that the GSW encryption scheme has the following additional property.

Lemma 3.7. *[60] The* GSW *homomorphic encryption scheme has the following property: There is an efficient algorithm* $\mathsf{InertEval}$*, that takes as input a public key* pk*, which is a matrix* $A \in \mathbb{Z}_q^{n\times M}$ *(where* n, M, q *correspond to the security parameter* λ *and* $M = M(\lambda) \in [\lambda, \lambda^2]$*),*[7] *a circuit* $C \in \mathcal{C}_k$ *such that* $C : \{0,1\}^k \to \{0,1\}^M$ *(for the same* M *as above), and* k *ciphertexts* $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}, x_i)$*, and outputs a special (inert) commitment*

[6] Here we assume the prover's input is a description of the function g, from which g can be computed (on any input) in time $\leq \mathrm{poly}(|\mathbb{H}^m|)$.

[7] Typically in the literature M is denoted by m, however throughout this work m denotes an m-variate polynomial, and hence we use M instead of m in the encryption scheme.

$\mathsf{Com}(\mathsf{pk}, C(x_1, \ldots, x_k)) \in \{0,1\}^M$, *such that under the LWE assumption, for every poly-size adversary* $\mathcal{A}$ *and every* $x = (x_1, \ldots, x_k) \in \{0,1\}^k$,

$$\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k) = C : \ C \in \mathcal{C}_k \ \wedge \ \mathsf{InertEval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k) = C(x)] = \mathrm{negl}(\lambda), \tag{2}$$

where $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i)$ *for every* $i \in [k]$*, and the probability is over* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *and over the randomness of* Enc*.*

Moreover, under the sub-exponential LWE assumption, there exists a constant $\delta > 0$ *such that Equation* (2) *holds with probability at most* $2^{-\lambda^\delta}$ *for any* 2^{λ^δ}*-size adversary* $\mathcal{A}$ *and any* $x \in \{0,1\}^k$*.*

Remark 3.8. *In the above lemma we deviate from the notion of Peikert and Shiehian [60]. In particular, their inert commitment of a string* $y \in \{0,1\}^M$ *is of the form* $Ar + G(y)$ *where* $A \in \mathbb{Z}_q^{n \times M}$ *is the* GSW *public key,* $G : \{0,1\}^M \to \mathbb{Z}_q^n$ *is a fixed linear function with an inverse* $G^{-1} : \mathbb{Z}_q^n \to \{0,1\}^M$*. Their main technical contribution is to show how to convert a GSW ciphertext* $\mathsf{Enc}(y)$ *into an inert commitment* $Ar + G(y)$*.*

In the lemma above we let the inert commitment be $G^{-1}(Ar + G(y))$*. We deviate from their notation for the sake of treating their work in a modular (and black-box) way. We emphasize that for us, the inert commitment of an* M*-bit string is itself an* M*-bit string.*

4 Our Non-Interactive Sum-Check Protocol

In this section we show that the Fiat-Shamir paradigm is sound when applied to the sum-check protocol (Figure 1) with a sufficiently large field size, assuming that LWE is sub-exponentially secure. Specifically, for our security proof to go through we need, roughly, the property that $\log |\mathbb{F}| \geq (m \log |H|)^{(1/\delta)^m}$, where m is the number of variables (corresponding to the number of rounds) in the sum-check protocol and $\delta > 0$ is a small enough constant related to the sub-exponential security of the homomorphic encryption scheme. This result is interesting only in the setting where $|H|$ is quite large and m is very small, for example, the setting where $|H| = n^\epsilon$ where $\epsilon > 0$ is a small constant and m is a large enough constant (or $\epsilon = 1/\log\log\log(n)$ and $m = O(\log\log\log n)$). This should be the setting of parameters the reader should have in mind.

At a high-level, we obtain this result, by using the recent works of [17, 60], together with a leveraging technique that allows us to relax their perfect security assumption to a sub-exponential one.

4.1 The Hash Function Family

In this section we describe the hash function family we use for instantiating the Fiat-Shamir paradigm. This hash family is very similar to the hash family of Peikert and Shiehian [60], which in turn is inspired by the hash family from [17].

Roughly speaking, in order for a cheating prover to convince the verifier of a false statement in the sum-check protocol in Figure 1, there must be some round i such that the polynomial $\tilde{g}_i$ sent by the prover in the i'th round is incorrect (i.e., $\tilde{g}_i \neq g_i$, where g_i is the polynomial that would have been sent by the honest prover), and yet the element t_i sent by the verifier satisfies that $\tilde{g}_i(t_i) = g_i(t_i)$. The hash function we create will need to avoid these "bad" values of t_i that allow the prover to cheat. Note that if $\tilde{g}_i \neq g_i$, then there are at most d bad values of t_i, where d is the degree of g_i.

More generally, suppose that there are d' such bad values that we need to avoid. Let $\varphi_{a,1}, \ldots, \varphi_{a,d'}$ be poly-sized functions that are given a non-uniform advice a (which, in the case of the sum-check protocol, is the description of the true g_i) and each output a "bad" value in $\mathbb{F}$ that we want to avoid. We will construct a hash family $\mathcal{H}$, such that given a random $h \leftarrow \mathcal{H}$, it is hard to find an input x such that $h(x) \in \{\varphi_{a,j}(x)\}_{j \in [d']}$.

The seed to the hash family is a ciphertext. In the analysis, this ciphertext will be an encryption of a bad function $\varphi_{a,j}$. However, in the scheme itself, it will be an encryption of the all zero function. Note that the non-uniform advice a is often hard to compute (and in the case of the sum-check it is not even a priori determined, since the polynomial g_i depends on the messages of the verifier in the first $i-1$ rounds), and hence cannot be used in the CRS. We will rely on the semantic security of the encryption scheme to argue

that if there exists a cheating prover that breaks soundness when the ciphertext encrypts the all zero function, then the prover also breaks soundness when the ciphertext encrypts one of the bad functions $\varphi_{a,j}$.

Our hash function will satisfy a special property that its output will belong to a subset $H_\lambda \subset \mathbb{F}$ of size 2^M, where $M = M(\lambda)$ is polynomially related to λ, the security parameter of the GSW encryption scheme that we use. More specifically, our hash function will always output an inert commitment (see Lemma 3.7), which belongs to $\{0,1\}^M$. We identify H_λ both as a subset of $\mathbb{F}$ and as $\{0,1\}^M$, and switch freely between the two notions. We will need to choose λ so that $2^{M(\lambda)} > d'$.

As we will be working with output space $H_\lambda = \{0,1\}^M$, we will need to confine the output of $\varphi_{a,j}$ to H_λ. This will be done by the function ξ_λ that, on input y, outputs y if $y \in H_\lambda$, and $0^{M(\lambda)} \in H_\lambda$ otherwise.

We next construct our hash family, denoted by $\mathcal{H}_\varphi$. This hash family consists of a seed generation algorithm $\mathcal{H}_\varphi.\mathsf{Gen}$, and an evaluation algorithm $\mathcal{H}_\varphi.\mathsf{Hash}$. The seed generation algorithm takes as input a security parameter 1^λ, and outputs a seed k, whereas the evaluation algorithm takes as input a seed k and an input x and outputs $h_k(x)$. In what follows we slightly abuse notation, and allow the seed generation algorithm $\mathcal{H}_\varphi.\mathsf{Gen}$ to take two additional inputs (a, j). These additional inputs are useful for the analysis, but in the actual scheme these additional inputs are always instantiated with fixed constants (say, $a = 0^{d'+1}$ and $j = 1$).

Construction 4.1. *Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be the GSW public key homomorphic encryption scheme. We define the following hash family $\mathcal{H}_\varphi$:*

- $\mathcal{H}_\varphi.\mathsf{Gen}(1^\lambda, a, j)$ *generates* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*, computes* $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, \xi_\lambda \circ \varphi_{a,j})$*, and outputs* $k = (\mathsf{pk}, \mathsf{ct})$.
- $\mathcal{H}_\varphi.\mathsf{Hash}(k, x)$ *interprets k as* $(\mathsf{pk}, \mathsf{ct})$ *and outputs* $\mathsf{InertEval}(\mathsf{pk}, U_x, \mathsf{ct})$*, where U_x denotes the universal circuit that takes as input any circuit $C : \{0,1\}^n \to \{0,1\}^M$ of size $\mathsf{Size}(\xi_\lambda \circ \varphi_{a,j})$, where $\mathsf{Size}(\xi_\lambda \circ \varphi_{a,j})$ denotes the size the circuit computing $\xi_\lambda \circ \varphi_{a,j}$,*[8] *and outputs*

$$U_x(C) = C(x).$$

Lemma 4.2. *Under the sub-exponential LWE assumption, there exists a constant $\delta > 0$ such that for every (sufficiently large) $\lambda \in \mathbb{N}$, any 2^{λ^δ}-size adversary $\mathcal{A}$, and any non-uniform advice (a, j),*

$$\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{ct}) = x \ : \ \mathcal{H}_\varphi.\mathsf{Hash}(k, x) = \varphi_{a,j}(x)] \leq 2^{-\lambda^\delta}$$

where the probability is over $k = (\mathsf{pk}, \mathsf{ct}) \leftarrow \mathcal{H}_\varphi.\mathsf{Gen}(1^\lambda, a, j)$.

The proof of this lemma follows directly from the definition of

$$\mathcal{H}_\varphi.\mathsf{Hash}(k, x) = \mathsf{InertEval}(\mathsf{pk}, U_x, \mathsf{ct}),$$

together with Lemma 3.7, and noting that if $\varphi_{a,j}(x) \notin H_\lambda$, then $\mathcal{H}_\varphi.\mathsf{Hash}(k, x) \neq \varphi_{a,j}(x)$ trivially.

4.2 Non-Interactive Sum-Check Protocol

In what follows, we present the non-interactive sum-check protocol, obtained by applying the Fiat-Shamir transform to the interactive sum-check protocol (Figure 1) with the hash family from Construction 4.1. In particular, let H be a subset of a field $\mathbb{F}$, and let $N = |H|^m$. Suppose the prover wishes to prove that

$$\sum_{h_1,\ldots,h_m \in \mathbb{F}} g(h_1, \ldots, h_m) = v,$$

where g is a m-variate polynomial with individual degree d and coefficients in $\mathbb{F}$, and $v \in \mathbb{F}$ is a field element.

[8] Clearly, there are many circuits computing $\xi_\lambda \circ \varphi_{a.j}$. We choose one to use in the construction, and it is this circuit size that we refer to above.

We use the hash functions from Construction 4.1 with the following d functions: $\{\varphi_{a,\ell}\}_{\ell\in[d]}$, where on input x, $\varphi_{a,\ell}$ outputs the ℓ'th root of $x-a$, using the Cantor-Zassenhaus [22] algorithm to factor the polynomial $x-a$. Here, the advice a and the input x are both degree d univariate polynomials represented by a tuple of $d+1$ coefficients. We assume an arbitrary ordering over $\mathbb{F}$ so that "the ℓ'th root" is well-defined, and as a convention, if the number of roots of $x^{(j)}-a$ is smaller than d, say, $c<d$, then for every $\ell\in[c+1,d]$, $\varphi_{a,\ell}(x)=0$.

Our non-interactive sum-check protocol employs a leveraging technique. More specifically, we choose security parameters $\kappa=\kappa_1<\kappa_2<\cdots<\kappa_m$ such that in round i we use the Fiat-Shamir hash function $\mathcal{H}_\varphi$ as defined in Construction 4.1, setting the security parameter λ to be κ_i. Importantly, we assume that the GSW scheme with security parameter κ_i is such that Equation (2) holds with probability at most $2^{-L_{i-1}}$, where $L_{i-1}=\sum_{j=1}^{i-1}M_j$ denotes the total number of bits sent by the verifier in the first $i-1$ rounds, and $M_j=M(\kappa_j)$ is the length of the inert commitment (see Lemma 3.7). This allows us in the analysis to guess all the messages of the verifier in the first $i-1$ rounds, and still argue that, conditioned on guessing correctly, one cannot break semantic security of the i'th encryption scheme.

We set the security parameters as follows: let $\kappa=\kappa_1\geq(\log N)^{2/\delta}$, where $N=|H|^m$, and for every $r>1$, let

$$\kappa_r=(2M(\kappa_{r-1}))^{1/\delta},$$

where $\delta>0$ is the constant related to the sub-exponential LWE assumption (see Lemma 3.7). Let $\mathbb{F}$ be a field that has size at least $2^{M(\kappa_m)}$, and take subsets $H_i\subset\mathbb{F}$, where $|H_i|=2^{M(\kappa_i)}$.

Since $M(\lambda)\leq\lambda^2$ by Lemma 3.7, we have that

$$\kappa_r\leq 2^{1/\delta}\cdot\kappa_{r-1}^{2/\delta}\leq\kappa_{r-1}^{3/\delta},$$

so $\kappa_m\leq\kappa^{(3/\delta)^{m-1}}$, meaning that $M(\kappa_m)\leq\kappa^{2\cdot(3/\delta)^{m-1}}$. So it will suffice to take $\mathbb{F}$ to so that

$$\log|\mathbb{F}|\geq\kappa^{2\cdot(3/\delta)^{m-1}}. \tag{3}$$

The non-interactive version of the sum-check protocol is given in Figure 2.

Theorem 4.3. *Let $N=|H|^m$. Assuming $\kappa\geq(\log N)^{2/\delta}$ and $\mathbb{F}$ is a field such that $\log|\mathbb{F}|\geq\kappa^{2\cdot(3/\delta)^{m-1}}$, the protocol in Figure 2 has the following properties.*

- **Completeness:** *For any m-variate polynomial g of individual degree d, if*

$$\sum_{h_1,\ldots,h_m\in H}g(h_1,\ldots,h_m)=v,$$

then

$$\Pr\left[(P_{\mathsf{niSC}}(g),V^g_{\mathsf{niSC}}(v))=1\right]=1.$$

- **Non-adaptive soundness:**[9] *For $v\in\mathbb{F}$ and m-variate polynomial g of degree $\leq d$ with coefficients over $\mathbb{F}$, if*

$$\sum_{h_1,\ldots,h_m\in H}g(h_1,\ldots,h_m)\neq v,$$

then for any $\mathrm{poly}(N)$-size cheating prover P^*,

$$\Pr\left[P^*(\mathrm{CRS})=\pi:V^g_{\mathsf{niSC}}(v,\mathrm{CRS},\pi)=1\right]=\mathrm{negl}(N),$$

assuming the sub-exponential hardness of LWE.

[9] Actually, stronger forms of soundness hold for the non-interactive sum-check protocol, such as *semi-adaptive soundness* (see Theorem 4.4) or *prefix-adaptive soundness* (see Theorem 7.3). We state here non-adaptive soundness for the sake of simplicity.

Non-Interactive Sum-Check Protocol

In this non-interactive protocol, P_{niSC} convinces V_{niSC} that

$$\sum_{h_1,\ldots,h_m \in H} g(h_1,\ldots,h_m) = v,$$

where g is a m-variate polynomial of individual degree d whose coefficients are in $\mathbb{F}$.

- **Setup:** Sample $k_i = (\mathsf{pk}_i, \mathsf{ct}_i) \leftarrow \mathcal{H}_\varphi.\mathsf{Gen}(1^{\kappa_i}, 0^{d+1}, 1)$ and set CRS $= \{k_i\}_{i\in[m]}$.
- **Prover:** P_{niSC} iteratively computes

  $$g_i(\cdot) = \sum_{h_{i+1},\ldots,h_m \in H} g(t_1,\ldots,t_{i-1},\cdot,h_{i+1},\ldots,h_m)$$

  and

  $$t_i = \mathcal{H}_\varphi.\mathsf{Hash}(k_i, g_i) \in H_i.$$

  P_{niSC} sends $\{(g_i, t_i)\}_{i\in[m]}$ to V_{niSC}.
- **Verifier:** For each $i \in [m]$, V_{niSC} checks that g_i is a polynomial of degree at most d and that

  $$\sum_{h\in H} g_i(h) = \begin{cases} v & i = 1 \\ g_{i-1}(t_{i-1}) & i > 1. \end{cases}$$

  V_{niSC} also checks that

  $$g_m(t_m) = g(t_1,\ldots,t_m),$$

  which is done by querying the oracle for g. Finally, he checks that all t_i's are computed correctly, i.e. for all i,

  $$t_i = \mathcal{H}_\varphi.\mathsf{Hash}(k_i, g_i).$$

  If any of these are false, V_{niSC} rejects. Otherwise, he accepts.

Figure 2: Non-interactive sum-check protocol $(P_{\mathsf{niSC}}, V^g_{\mathsf{niSC}})$

- **Efficiency:** *The prover's runtime is* $\mathrm{poly}(N, d^m, \log|\mathbb{F}|)$*, the communication complexity is* $\mathrm{poly}(m, d, \log|\mathbb{F}|)$*, and the verifier's runtime (given oracle access to* g*) is* $\mathrm{poly}(m, d, |H|, \log|\mathbb{F}|)$.

Rather than prove Theorem 4.3, we prove a more general theorem for a parallel version of this non-interactive sum-check protocol in Section 4.3 (see Theorem 4.4). This non-interactive parallel sum-check protocol will be used as a subroutine in the non-interactive GKR protocol in Section 6.

4.3 Non-Interactive Parallel Sum-Check Protocol

A key component in our non-interactive GKR protocol is our non-interactive sum-check protocol. We will need to run *many* non-interactive sum-checks *in parallel* where the verifier uses the *same* random coin-tosses in all these parallel executions. In what follows, we present and analyze such a non-interactive parallel sum-check protocol, obtained by applying the Fiat-Shamir paradigm with the hash functions from Construction 4.1.

Let H be a subset of a field $\mathbb{F}$, and let $N = |H|^m$. Suppose that we have a list of $e \le \mathrm{poly}(N)$[10] claims of the form

$$\sum_{h_1,\ldots,h_m \in H} g^{(j)}(h_1,\ldots,h_m) = v^{(j)},\ j \in [e],$$

[10] We will only need $e = \mathrm{poly}(|H|)$, but Theorem 4.4 holds for e up to $\mathrm{poly}(N)$.

where each $g^{(j)}$ is a m-variate polynomial over $\mathbb{F}$ of individual degree $d \leq \text{poly}(|H|)$. We use the notation $\overline{g} = \{g^{(j)}\}_{j\in[e]}$ and $\overline{v} = \{v^{(j)}\}_{j\in[e]}$. (Similarly, it should be understood, for instance, that $\overline{g}_i = \{g_i^{(j)}\}_{j\in[e]}$.)

We use the hash functions from Construction 4.1 with the following $d' = ed$ functions: $\varphi_{a,j,\ell} := \varphi_{a,(j-1)d+\ell}$, which on input $\overline{x}$, outputs the ℓ'th root of $x^{(j)} - a$, using the Cantor-Zassenhaus [22] algorithm to factor polynomials. Here, the advice a is a degree d univariate polynomial represented by a tuple of $d+1$ coefficients, and the input $\overline{x}$ is a set of e univariate polynomials $\{x^{(j)}\}_{j\in[e]}$, each of degree d. As in Section 4.2, we assume an arbitrary ordering over $\mathbb{F}$ so that "the ℓ'th root" is well-defined, and as a convention, if for $j \in [e]$ the number of roots of $x^{(j)} - a$ is smaller than d, say, $c_j < d$, then for every $\ell \in [c_j + 1, d]$, $\varphi_{a,j,\ell}(x) = 0$.

We use the same leveraging technique as in Section 4.2, where we choose security parameters $\kappa = \kappa_1 < \kappa_2 < \cdots < \kappa_m$ such that in round i we use the Fiat-Shamir hash function $\mathcal{H}_\varphi$ as defined in Construction 4.1, setting the security parameter λ to be κ_i. We set $\kappa = \kappa_1 \geq (\log N)^{2/\delta}$, where $N = |H|^m$, and for every $r > 1$, we set

$$\kappa_r = (2M(\kappa_{r-1}))^{1/\delta},$$

where $\delta > 0$ is the constant related to the sub-exponential LWE assumption (see Lemma 3.7). We let $\mathbb{F}$ be a field of size at least $2^{M(\kappa_m)}$, and take subsets $H_i \subset \mathbb{F}$, where $|H_i| = 2^{M(\kappa_i)}$. As noted in the previous section in Equation (3), we can take any field $\mathbb{F}$ that contains H such that $\log|\mathbb{F}| \geq \kappa^{2\cdot(3/\delta)^{m-1}}$.

Our non-interactive, publicly verifiable parallel sum-check procedure for the e claims

$$\sum_{h_1,\ldots,h_m\in H} g^{(j)}(h_1,\ldots,h_m) = v^{(j)}, \quad j \in [e],$$

where each $g^{(j)}$ is a m-variate polynomial of individual degree at most d, is described in Figure 3.

In what follows, let Δ denote a predetermined set of size $\text{poly}(N)$, where the elements of Δ are m-variate polynomials of individual degree $\leq d$ with coefficients in $\mathbb{F}$. We will show that our non-interactive parallel sum-check protocol satisfies a soundness that is *semi-adaptive*, meaning that the protocol is sound against a cheating prover that selects the multi-varaiate polynomials $g^{(j)}$ adaptively, so long as $g^{(j)} \in \Delta$ for all $j \in [e]$. This semi-adaptive soundness property will be important in proving soundness of our non-interactive GKR protocol in Section 6. Note that this semi-adaptive soundness guarantee for the parallel sum-check immediately implies non-adaptive soundness for the (single) sum-check in Figure 2 by letting $|\Delta| = 1$ and not allowing the prover a choice of v. (In fact, semi-adaptive soundness implies non-adaptive soundness for the *parallel* sum-check protocol as well by letting Δ contain all polynomials $g^{(j)}$, and rejecting if the prover picks values of $g^{(j)}$ other than the correct ones.)

Theorem 4.4. *Let $N = |H|^m$. Assuming $\kappa \geq (\log N)^{2/\delta}$ and $\mathbb{F}$ is a field such that $\log|\mathbb{F}| \geq \kappa^{2\cdot(3/\delta)^{m-1}}$, the protocol in Figure 3 has the following properties.*

- **Completeness:** *For every $\bar{g} = (g^{(1)},\ldots,g^{(e)})$ such that each $g^{(j)}$ is an m-variate polynomial of individual degree d, if*

  $$\sum_{h_1,\ldots,h_m\in H} g^{(j)}(h_1,\ldots,h_m) = v^{(j)}$$

  for all $j \in [e]$, then

  $$\Pr\left[(P_{\mathsf{niPSC}}(\overline{g}), V^{\overline{g}}_{\mathsf{niPSC}}(\overline{v})) = 1\right] = 1.$$

- **Semi-adaptive soundness:** *For any $\text{poly}(N)$-size cheating prover P^* and any set Δ of size $\text{poly}(N)$ of m-variate polynomials of individual degree $\leq d$,*

  $$\Pr\left[P^*(\text{CRS}) = (\overline{g},\overline{v},\pi) : \begin{cases} g^{(j)} \in \Delta \ \forall j \in [e] \\ \wedge\ \exists j \in [e] \ s.t. \sum_{h\in H^m} g^{(j)}(h) \neq v^{(j)} \\ \wedge\ V^{\overline{g}}_{\mathsf{niPSC}}(\overline{v}, \text{CRS}, \pi) = 1 \end{cases}\right] = \text{negl}(N),$$

  assuming the sub-exponential hardness of LWE.

Non-Interactive Parallel Sum-Check Protocol

In this non-interactive protocol, P_{niPSC} convinces V_{niPSC} of the e equations

$$\sum_{h_1,\ldots,h_m \in H} g^{(j)}(h_1,\ldots,h_m) = v^{(j)}, \qquad j \in [e].$$

- **Setup:** Sample $k_i = (\mathsf{pk}_i, \mathsf{ct}_i) \leftarrow \mathcal{H}_\varphi.\mathsf{Gen}(1^{\kappa_i}, 0^{d+1}, 1)$ and set $\mathrm{CRS} = \{k_i\}_{i\in[m]}$.
- **Prover:** P_{niPSC} iteratively computes

$$g_i^{(j)}(\cdot) = \sum_{h_{i+1},\ldots,h_m \in H} g^{(j)}(t_1,\ldots,t_{i-1},\cdot,h_{i+1},\ldots,h_m) \qquad \forall j \in [e]$$

  and

$$t_i = \mathcal{H}_\varphi.\mathsf{Hash}(k_i, \overline{g}_i) \in H_i.$$

  P_{niPSC} sends $\{(\overline{g}_i, t_i)\}_{i\in[m]}$ to V_{niPSC}.
- **Verifier:** For each $i \in [m]$, V_{niPSC} checks for all $j \in [e]$ that $g_i^{(j)}$ is a polynomial of degree at most d, and that

$$\sum_{h\in H} g_i^{(j)}(h) = \begin{cases} v^{(j)} & i = 1 \\ g_{i-1}^{(j)}(t_{i-1}) & i > 1. \end{cases}$$

  V_{niPSC} also checks for all $j \in [e]$ that

$$g_m^{(j)}(t_m) = g^{(j)}(t_1,\ldots,t_m),$$

  when is done by querying the oracle for $g^{(j)}$. Finally, he checks that all t_i's are computed correctly, i.e. for all i,

$$t_i = \mathcal{H}_\varphi.\mathsf{Hash}(k_i, \overline{g}_i).$$

  If any of these are false, V_{niPSC} rejects. Otherwise he accepts.

Figure 3: Non-interactive parallel sum-check protocol $(P_{\mathsf{niPSC}}, V_{\mathsf{niPSC}}^{\overline{g}})$

- **Efficiency:** *The prover's runtime is* $\mathrm{poly}(N, d^m, e, \log|\mathbb{F}|)$*, the communication complexity is* $\mathrm{poly}(m, d, e, \log|\mathbb{F}|)$*, and the verifier's runtime (given oracle access to* g*) is* $\mathrm{poly}(m, d, e, |H|, \log|\mathbb{F}|)$*.*

Proof. Completeness again follows from the correctness of the sum-check protocol. Efficiency follows straightforwardly from an analysis of the non-interactive parallel sum-check protocol, and noting that the hash functions take time $\mathrm{poly}(d, \log|\mathbb{F}|)$ to compute (since they homomorphically factor a degree d polynomial via the Cantor-Zassenhaus algorithm). We will focus on proving semi-adaptive soundness against a cheating prover.

We will prove in fact that for every $\mathrm{poly}(N)$-size cheating prover P^* and every set Δ of size $\mathrm{poly}(N)$,

$$\Pr\left[P^*(\mathrm{CRS}) = (\overline{g}, \overline{v}, \pi) : \begin{cases} g^{(j)} \in \Delta \ \forall j \in [e] \\ \wedge\ \exists j \in [e] \text{ s.t. } \sum_{z\in H^m} g^{(j)}(z) \neq v^{(j)} \\ \wedge\ V_{\mathsf{niPSC}}^{\overline{g}}(\overline{v}, \mathrm{CRS}, \pi) = 1 \end{cases}\right] \leq 2mde|\Delta|/2^{\kappa^\delta}.$$

For $\kappa \geq (\log N)^{2/\delta}$, the right hand side is negligible in N, since $d \leq \mathrm{poly}(|H|)$ and $e, |\Delta| = \mathrm{poly}(N)$.

Suppose for the sake of contradiction that

$$\Pr\left[P^*(\text{CRS}) = (\overline{g}, \overline{v}, \pi) : \begin{cases} g^{(j)} \in \Delta \ \forall j \in [e] \\ \wedge \ \exists j \in [e] \text{ s.t. } \sum_{z \in H^m} g^{(j)}(z) \neq v^{(j)} \\ \wedge \ V^{\overline{g}}_{\mathsf{niPSC}}(\overline{v}, \text{CRS}, \pi) = 1 \end{cases}\right] = \epsilon > 2mde|\Delta|/2^{\kappa^\delta}.$$

Let

$$g_i^{(j)} = \sum_{h_{i+1},\ldots,h_m \in H} g^{(j)}(t_1, \ldots, t_{i-1}, \cdot, h_{i+1}, \ldots, h_m)$$

denote the correct value of the i'th round polynomial for the j'th claim, and let $\tilde{g}_i^{(j)}$ denote the polynomial that the prover claims is $g_i^{(j)}$.

P^* successfully convincing V_{niPSC} of the set of claims $\{\sum_{h_1,\ldots,h_m} g^{(j)}(h_1, \ldots, h_m) = v^{(j)}\}_{j\in[e]}$, at least one of which is incorrect, means that there is at least one round r and one claim c for which $\tilde{g}_r^{(c)} \neq g_r^{(c)}$, and t_r is a root of $\tilde{g}_r^{(c)} - g_r^{(c)}$. By an averaging argument, there is some round $r \in [m]$, some claim $c \in [e]$, some $\ell \in [d]$, some polynomial $a \in \Delta$, and values $\{t'_i \in H_i\}_{i\in[r-1]}$ for which with probability at least $\frac{\epsilon}{m\cdot d\cdot e\cdot|\Delta|\cdot|H_1|\cdot\cdots\cdot|H_{r-1}|}$, P^* produces a transcript such that the following four things hold:

1. $g^{(c)} = a$,
2. P^*'s transcript uses $t_i = t'_i$ for all $i \in [r-1]$,
3. $\tilde{g}_r^{(c)} \neq g_r^{(c)}$,
4. the ℓ'th root of $\tilde{g}_r^{(c)} - g_r^{(c)}$ is t_r.

Call these simultaneous four conditions event A. Note that event A is checkable in $\text{poly}(N) = o(2^{\kappa^\delta}) = o(2^{\kappa_r^\delta})$ time, since the correct value of $g_r^{(c)}$ can be computed in $\text{poly}(N)$ time. Let a_r denote the correct value of $g_r^{(e)}$ when $g^{(e)} = a$ and $t_i = t'_i$ for all $i \in [r-1]$.

We look at the probability of event A occurring. In the original hybrid, we have that event A occurs with probability at least $\frac{\epsilon}{m\cdot d\cdot e\cdot|\Delta|\cdot|H_1|\cdot\cdots\cdot|H_{r-1}|}$, and the r'th hash key is $k_r = (\mathsf{pk}_r, \mathsf{ct}_r)$, where $\mathsf{ct}_r = \mathsf{Enc}(\mathsf{pk}_r, \xi_{\kappa_r} \circ \varphi_{0^{d+1},1,1})$ is the encryption of the function that outputs the first root of $\tilde{g}_r^{(1)} - 0$ if it is in H_r, else it outputs $0 \in H_r$. We replace k_r by $k'_r = (\mathsf{pk}_r, \mathsf{ct}'_r)$, where $\mathsf{ct}'_r = \mathsf{Enc}(\mathsf{pk}_r, \varphi_{a'_r,c,\ell})$ is the encryption of the function that outputs the ℓ'th root of $x^{(c)} - a_r$ if it is in H_r, else it outputs $0 \in H_r$. The δ-sub-exponential security of the encryption scheme and the fact that event A is checkable in $2^{\kappa_r^\delta}$ time imply that the difference in probabilities of event A occurring under k_r and k'_r is at most $\frac{1}{2^{\kappa_r^\delta}}$. Moreover,

$$\frac{1}{2^{\kappa_r^\delta}} \leq \frac{1}{2^{\kappa^\delta}\cdot\prod_{i=1}^{r-1} 2^{M(\kappa_i)}} = \frac{1}{2^{\kappa^\delta}\cdot|H_1|\cdot\cdots\cdot|H_{r-1}|} < \frac{\epsilon}{2m\cdot d\cdot e\cdot|\Delta|\cdot|H_1|\cdot\cdots\cdot|H_{r-1}|},$$

where we defer the proof of the first inequality to the end of this proof, and the last holds by the definition of ϵ. Thus, event A should still occur with with probability greater than $\frac{\epsilon}{2m\cdot d\cdot e\cdot|\Delta|\cdot|H_1|\cdot\cdots\cdot|H_{r-1}|}$.

Note that k'_r is identical to a key generated by $\mathcal{H}_\varphi.\mathsf{Gen}(1^{\kappa_r}, a'_r, (c-1)d + \ell)$. But by Lemma 4.2, any $2^{\kappa_r^\delta}$-sized adversary (and in particular any $\text{poly}(N)$-sized cheating prover) cannot find $\tilde{g}_r$ such that $t_r = \mathcal{H}_\varphi.\mathsf{Hash}(k'_r, \tilde{g}_r)$ is the ℓ'th root of $\tilde{g}_r - a_r$ with probability greater than $2^{-\kappa_r^\delta}$. This means that property A can only hold with probability at most $2^{-\kappa_r^\delta} < \frac{\epsilon}{2m\cdot d\cdot e\cdot|\Delta|\cdot|H_1|\cdot\cdots\cdot|H_{r-1}|}$, giving a contradiction.

Finally, we check that $\frac{1}{2^{\kappa_r^\delta}} \leq \frac{1}{2^{\kappa^\delta}\cdot\prod_{i=1}^{r-1} 2^{M(\kappa_i)}}$, or equivalently, that

$$\kappa_r^\delta \geq \kappa^\delta + \sum_{i=1}^{r-1} M(\kappa_i), \tag{4}$$

which we claimed earlier. The proof of this follows from a straightforward induction on r: Inequality (4) clearly holds for $r = 1$, as it simply states that $\kappa_1^\delta \geq \kappa^\delta$. Now, consider $r > 1$. Substituting $(2M(\kappa_{r-1}))^{1/\delta}$ for κ_r, it suffices to show that

$$2M(\kappa_{r-1}) \geq \kappa^\delta + \sum_{i=1}^{r-1} M(\kappa_i),$$

or equivalently, that

$$M(\kappa_{r-1}) \geq \kappa^\delta + \sum_{i=1}^{r-2} M(\kappa_i).$$

This follows from the inductive hypothesis, which asserts that the right-hand-side is at most k_{r-1}^δ, which in turn is at most the left-hand side (since $M(\kappa_{r-1}) \geq \kappa_{r-1}$). □

Remark 4.5. *By adjusting the values of $|\Delta|$ and κ, we can in fact prove Theorem 4.4 for larger values of $|\Delta|$, even $|\Delta|$ that is super-polynomial in N. However, we will not need this fact in our application to* GKR.

Remark 4.6. *Under the sub-exponential LWE assumption, the* CRS *used in the construction of our non-interactive (parallel) sum-check protocol is indistinguishable from a random string of the same length by a 2^{κ^δ}-sized adversary with advantage greater than $2^{-\kappa^\delta}$. In particular, this means that no* $\mathrm{poly}(N)$*-sized cheating prover can distinguish the generated* CRS *from a random string. We can therefore replace the* CRS *by a uniformly random string of the same length, giving a sound protocol in the common* random *string model.*

5 The Modified Interactive GKR Protocol

In this section we present a modification to the GKR protocol, which we call mGKR, for delegating bounded depth computations. Jumping ahead, we remark that we can prove the soundness of the Fiat-Shamir paradigm when applied to mGKR (under sub-exponential circular security of the LWE assumption), whereas we do not know how to prove soundness w.r.t the original GKR protocol.

We first recall the original GKR protocol in Section 5.1. Then in Section 5.2 we explain the technical obstacles we face when trying to prove soundness of the Fiat-Shamir paradigm when applied to the original GKR protocol. Finally, in Section 5.3 and present our modified GKR protocol which overcomes these obstacles.

5.1 The GKR Protocol

The GKR protocol is a publicly verifiable *interactive* proof for delegating log-space uniform bounded depth computations. Fix any log-space uniform circuit $C : \{0,1\}^n \to \{0,1\}$ of size S and depth D. Let $H \subseteq \mathbb{F}$, where $\mathbb{F}$ is an extension field of GF[2] for which addition and multiplication in $\mathbb{F}$ can be done in time $\mathrm{polylog}(|\mathbb{F}|)$, and where

$$\max\{D, \log S\} \leq |H| \leq S,$$

Let $m \in \mathbb{N}$ be such that $S \leq |H|^m \leq \mathrm{poly}(S)$.

Remark 5.1. *We note that in [37], they set $|H| = \mathrm{poly}(D, \log S)$ and $\mathbb{F}$ to be a field of size that is polynomially related to H. In this work we take $|H| = S^\epsilon$ (for a slightly sub-constant $\epsilon = o(1)$), and take $\mathbb{F}$ to be exponentially larger than H.*

Notations and Assumptions. We make the following assumptions (without loss of generality):

1. C is an arithmetic circuit over $\mathbb{F}$.
2. C is a layered circuit of fan-in 2. Namely, the gates in C can be (uniquely) partitioned into layers such that the inputs to a gate in layer i are the outputs of gates in layer $i-1$. Layer 0 is the input layer, and layer D is the output layer.

3. Each layer has exactly S gates (we pad all the layers which consist of less than S gates, including the input and output layers).

For each layer $i \in [D]$, we assign each of the S gates a distinct label in H^m. We associate with each layer $i \in [D]$ two functions

$$\mathsf{add}_i, \mathsf{mult}_i : H^{3m} \to \{0,1\},$$

where, for $\omega_1, \omega_2, \omega_3 \in H^m$, $\mathsf{add}_i(\omega_1, \omega_2, \omega_3) = 1$ if and only if ω_1 is an addition gate in layer i applied to gates ω_2 and ω_3, which are in layer $i-1$. Similarly, $\mathsf{mult}_i(\omega_1, \omega_2, \omega_3) = 1$ if and only if ω_1 is a multiplication gate in layer i applied to gates ω_2 and ω_3 in layer $i-1$. In particular, if at least one of $\omega_1, \omega_2, \omega_3$ does not refer to a gate, then $\mathsf{add}_i(\omega_1, \omega_2, \omega_3) = \mathsf{mult}_i(\omega_1, \omega_2, \omega_3) = 0$.

It will be convenient to work with circuits for which there are (not necessarily lowest degree) extensions $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{mult}}_i : \mathbb{F}^{3m} \to \mathbb{F}$ that agree with add_i and mult_i on H^{3m}, and which can be evaluated quickly (in time $\mathrm{poly}(\log S, \log|\mathbb{F}|)$). It is not known if such extensions of add_i and mult_i exist for any given log-space uniform circuit C. But any log-space uniform circuit C can be converted into a slightly larger circuit C' deciding the same language, for which such extensions do exist.

Proposition 5.2. *[35] For any log-space uniform circuit C of size S and depth D, there is a circuit C' of size $S' = \mathrm{poly}(S)$ and depth $D' = D \cdot \mathrm{polylog} S$ such that $C(x) = C'(x)\ \forall x \in \{0,1\}^n$, for which there are size $\mathrm{poly}(m, \log|H|)$ formulas computing add'_i and mult'_i that are constructible in $\mathrm{poly}(m, \log|H|)$ time. In particular, this means that for all $i \in [D']$, there are polynomials*

$$\widetilde{\mathsf{add}'}_i, \widetilde{\mathsf{mult}'}_i : \mathbb{F}^{3m'} \to \mathbb{F}$$

such that $\widetilde{\mathsf{add}'}_i|_{H'^{3m'}} \equiv \mathsf{add}'_i$ and $\widetilde{\mathsf{mult}'}_i|_{H'^{3m'}} \equiv \mathsf{mult}'_i$, and both these polynomials are of degree $\leq \mathrm{poly}(m', |H'|)$ and are evaluable in $\mathrm{poly}(m', |H'|, \log|\mathbb{F}|)$ time.

In what follows, we consider circuits C for which $\widetilde{\mathsf{add}}_i$ and $\widetilde{\mathsf{mult}}_i$ are both evaluable in $\mathrm{poly}(m, |H|, \log|\mathbb{F}|)$ time, knowing that it is easy to convert any log-space uniform circuit into such a circuit.

Fix an input $x \in \{0,1\}^n$. For each $i \in \{0, 1, \ldots, D\}$, we associate a function

$$V_i : H^m \to \{0,1\}$$

so that $V_i(\omega)$ is the value of the gate ω in the i'th layer of the circuit C on input x. So, if ω_{out} is the output gate, we have that $V_D(\omega_{\mathrm{out}}) = C(x)$. Let

$$\hat{V}_i : \mathbb{F}^m \to \mathbb{F}$$

be the low-degree extension of V_i.

Overview of the GKR protocol. Suppose the prover wishes to prove that $C(x) = 0$, or equivalently, that $\hat{V}_D(\omega_{\mathrm{out}}) = 0$. This is done in D phases (where D is the depth of C). In the original GKR protocol, in the i'th phase ($1 \leq i \leq D$) the prover reduces the task of proving that $\hat{V}_i(w) = r$ to the task of proving two equations: $\hat{V}_{i-1}(w_1) = r_1$ and $\hat{V}_{i-1}(w_2) = r_2$, where w_1, w_2 are random elements in $\mathbb{F}^m$ determined by the random coin tosses of the verifier. This is done by running the sum-check protocol.

In more detail, to go from layer i to layer $i-1$, notice that for every $p \in H^m$,

$$\hat{V}_i(p) = \sum_{\omega_1, \omega_2 \in H^m} \widetilde{\mathsf{add}}_i(p, \omega_1, \omega_2) \cdot \left(\hat{V}_{i-1}(\omega_1) + \hat{V}_{i-1}(\omega_2)\right) + \widetilde{\mathsf{mult}}_i(p, \omega_1, \omega_2) \cdot \hat{V}_{i-1}(\omega_1) \cdot \hat{V}_{i-1}(\omega_2).$$

Thus, by Equation (1), for every $w \in \mathbb{F}^m$,

$$\hat{V}_i(w) = \sum_{p, \omega_1, \omega_2 \in H^m} \hat{\mathrm{EQ}}(w; p) \cdot \begin{pmatrix} \widetilde{\mathsf{add}}_i(p, \omega_1, \omega_2) \cdot \left(\hat{V}_{i-1}(\omega_1) + \hat{V}_{i-1}(\omega_2)\right) \\ + \widetilde{\mathsf{mult}}_i(p, \omega_1, \omega_2) \cdot \left(\hat{V}_{i-1}(\omega_1) \cdot \hat{V}_{i-1}(\omega_2)\right) \end{pmatrix}.$$

For every $w \in \mathbb{F}^m$, let $f_{i,w} : (\mathbb{F}^m)^3 \to \mathbb{F}$ be the function defined by

$$f_{i,w}(p,\omega_1,\omega_2) := \hat{\mathrm{EQ}}(w;p) \cdot \begin{pmatrix} \widetilde{\mathsf{add}}_i(p,\omega_1,\omega_2) \cdot \left(\hat{V}_{i-1}(\omega_1) + \hat{V}_{i-1}(\omega_2)\right) \\ + \widetilde{\mathsf{mult}}_i(p,\omega_1,\omega_2) \cdot \left(\hat{V}_{i-1}(\omega_1) \cdot \hat{V}_{i-1}(\omega_2)\right) \end{pmatrix}. \tag{5}$$

This means that for every $w \in \mathbb{F}^m$,

$$\hat{V}_i(w) = \sum_{p,\omega_1,\omega_2 \in H^m} f_{i,w}(p,\omega_1,\omega_2).$$

Note that $f_{i,w}$ is a $3m$-variate polynomial of size $\leq \mathrm{poly}(S)$ and degree $\leq \mathrm{poly}(m,|H|)$ (we will asssume m is much smaller than $|H|$, so the degree will be $\mathrm{poly}(|H|)$). In the GKR protocol, the prover and verifier run the sum-check protocol, and thus reduce the task of verifying the statement $\hat{V}_i(w) = r$ to verifying a statement of the form $f_{i,w}(p,w_1,\omega_2) = r'$ for $p,\omega_1,\omega_2 \in \mathbb{F}^m$ which are determined by the random coin tosses of the verifier.

Note that the verifier cannot compute on its own the function $f_{i,w}$, though by Proposition 3.1 and Proposition 5.2 he can compute efficiently the functions $\hat{\mathrm{EQ}}$, $\widetilde{\mathsf{add}}_i$ and $\widetilde{\mathsf{mult}}_i$. Thus, to compute the value of $f_{i,w}(p,w_1,\omega_2)$, he only needs to help in computing the values of $\hat{V}_{i-1}(\omega_1)$ and $\hat{V}_{i-1}(\omega_2)$. Indeed, as mentioned above, the sum-check reduces the task of verifying one claim of the form $\hat{V}_i(w) = r$ to verifying *two* claims of the form $\hat{V}_{i-1}(w_1) = r_1$ and $\hat{V}_{i-1}(w_2) = r_2$.

To avoid an exponential blowup in the number of equations that are needed to check, they then reduce the task of verifying that both $\hat{V}_{i-1}(w_1) = r_1$ and $\hat{V}_{i-1}(w_2) = r_2$ to the task of verifying a single statement of the form $\hat{V}_{i-1}(w_{i-1}) = r_{i-1}$ by running a *two-to-one* protocol.[11] Finally, the verifier checks on his own that $\hat{V}_0(w_0) = r_0$, which can be done efficiently since it amounts to computing a single point on the low-degree extension of x.

5.2 Obstacles for Arguing Soundness of the Non-Interactive GKR Protocol

It is tempting to replace each interactive sum-check protocol in GKR with a non-interactive version, and thus make GKR non-interactive (ignoring the two-to-one protocol, as it was shown to be unnecessary in [40]). At first sight, it may seem that the soundness of the non-interactive sum-check protocol would imply the soundness of the non-interactive GKR protocol.

Unfortunately, when we tried to analyze the non-interactive GKR protocol we failed for the following reason: In the original GKR protocol the sum-check statements are determined by the random coin-tosses of the verifier. More specifically, if in the i'th layer the prover proves that $\hat{V}_i(w) = r$ via the sum-check protocol, then the sum-check polynomial $f_{i,w}$ depends on the value of w. In the Fiat-Shamir setting, the value of each such w is determined by hashing the partial transcript, and thus can be chosen adaptively by the cheating prover.

This adaptivity is precisely what prohibits the proof from going through, since we failed to prove adaptive soundness of the non-interactive sum-check protocol (rather, we only managed to prove *semi-adaptive* soundness (see Theorem 4.4)). Indeed, the proof would go through if we could somehow ensure that each w belongs to a small set $\Lambda \subseteq \mathbb{F}^m$. Unfortunately, we do not know how to reduce the task of verifying a statement of the form $\hat{V}_i(w) = r$ for an arbitrary $w \in \mathbb{F}^m$ to verifying a statement of the form $\hat{V}_i(w') = r'$ for $w' \in \Lambda$, where $\Lambda \subseteq \mathbb{F}^m$ is a small predetermined set. However, we do know how to convert it to $e = \mathrm{poly}(|H|)$ statements of the form $\hat{V}_i(z^{(j)}) = v^{(j)}$, where $z^{(j)} \in \Lambda$ for every $j \in [e]$. This will be done via a *one-to-many reduction*.

[11] It was shown in [40] that this two-to-one protocol is unnecessary. We use this fact in our modification to the GKR protocol in Section 5.3.

5.3 Our Modified GKR Protocol

In this work we modify the original GKR protocol, as follows: In each layer i, before running any sum-check procedure, we first convert each claim $\hat{V}_i(w_i) = r_i$ into e claims $\hat{V}_i(z_i^{(j)}) = v_i^{(j)}$ for $j \in [e]$, where $e = \text{poly}(|H|)$. This, *one-to-many* reduction may at first seem counterproductive, and is in contrast with the more intuitive *two-to-one* reduction of the original GKR protocol. However, it has the important property that all the elements $z_i^{(j)}$ belong to a small set $\Lambda = \mathbb{F}_0^m \subseteq \mathbb{F}^m$. The *one-to-many reduction* is defined in Definition 5.3. The basic idea is to find e values $z_i^{(j)}$, such that the values of $\hat{V}_i(z_i^{(j)})$ for every $j \in [e]$ determine the value of $\hat{V}_i(w_i)$.

This one-to-many reduction, together with the sum-check protocol, allows us to reduce the task of verifying a single claim of the form $\hat{V}_i(w_i) = r_i$ for $w_i \in \mathbb{F}^m$ to verifying $2e$ claims of the form $\hat{V}_{i-1}(z_{i-1}^{(j)}) = v_{i-1}^{(j)}$ where each $z_{i-1}^{(j)} \in \Lambda$. This is done by first using our one-to-many reduction to reduce the task of verifying $\hat{V}_i(w_i) = r_i$ to verifying e claims $\hat{V}_i(z_i^{(j)}) = v_i^{(j)}$. Then, we run e sum-checks to reduce verifying these claims to verifying $2e$ claims of the form $\hat{V}_{i-1}(w_{i-1}^{(j,b)}) = r_{i-1,b}^{(j,b)}$ for $j \in [e], b \in \{1,2\}$.

We can now use a $2e$-to-one trick, similar to the two-to-one trick in the original GKR protocol, to reduce the $2e$ claims to one, and continue from there. While we don't see any problems with this approach, we will choose not to implement a $2e$-to-one trick, instead relying on an observation by [40] to prevent an exponential blowup in the number of claims.

The observation of [40] is that the two-to-one trick in GKR is unnecessary. Rather, they show that one can continue with two claims per layer, without incurring an exponential growth in the number of claims. Specifically, their main observation is that this exponential growth can be avoided if the verifier uses the *same random coins* in both sum-checks. Indeed, if the verifier uses the same random coin-tosses in both sum-checks, then the two sum-checks reduce the task of verifying two claims of the form $\hat{V}_i(w_{i,b}) = r_{i,b}$ to verifying *two* claims of the form $\hat{V}_{i-1}(w_{i-1,b}) = r_{i-1,b}$ (as opposed to four claims). This is the case since the values of $w_{i-1,1}$ and $w_{i-1,2}$ are determined only by the verifier's coin-tosses, and if he used the same coins in both executions both claims are reduced to the same claims!

Thus, in our modified GKR protocol, the prover and verifier run $2e$ sum-checks in parallel, where the verifier uses the same random coin tosses in all these $2e$ sum-checks. Thus, these $2e$ sum-checks reduce the task of verifying $2e$ claims of the form $\hat{V}_i(z_i^{(j)}) = v_i^{(j)}$ to the task of verifying *two* claims $\hat{V}_{i-1}(w_{i-1}^{(b)}) = r_{i-1}^{(b)}$, where $w_{i-1}^{(1)}, w_{i-1}^{(2)} \in \mathbb{F}^m$.

In more detail, when running in parallel $2e$ sum-checks for proving that for every $j \in [2e]$,

$$v_i^{(j)} = \sum_{p,\omega_1,\omega_2 \in H^m} f_{i,z_i^{(j)}}(p, \omega_1, \omega_2),$$

at the end, the verifier needs to check that $f_{i,z_i^{(j)}}(p, \omega_1, \omega_2) = \rho_i^{(j)}$ for the *same* values of $(p, \omega_1, \omega_2) \in \mathbb{F}^{3m}$, since these values are determined by the random coin tosses used by the sum-check verifier. In particular, to verify these $2e$ equations, all the verifier needs is $\hat{V}_{i-1}(\omega_1)$ and $\hat{V}_{i-1}(\omega_2)$, since it can compute the rest of the terms on its own. Thus, we reduce the task of verifying $\hat{V}_i(z_i^{(j)}) = v_i^{(j)}$ for every $j \in [2e]$ to the task of verifying $\hat{V}_{i-1}(\omega_b) = r_b$ for $b \in \{1,2\}$, where $\omega_1, \omega_2 \in \mathbb{F}^m$. We set $w_{i-1}^{(b)} = \omega_b$ for the next layer.

We repeat these two steps (applying the one-to-many reduction, then running $2e$ sum-checks in parallel) for each level $i \in [D]$. At the very end of the D rounds, it remains for the verifier to verify that for every $b \in \{0,1\}$, $\hat{V}_0(\omega_b) = r_b$. This amounts to computing two points in the low-degree extension of the input x (with respect to $H, m, \mathbb{F}$), which he can compute on his own in time $n \cdot \text{poly}(m, |H|, \log|\mathbb{F}|)$.

A formal description of our modified GKR protocol can be found in Figure 5, and its analysis can be found in Section 5.3.2. It makes use of a one-to-many reduction which can be found in Section 5.3.1.

5.3.1 One-to-Many Reduction

We define a one-to-many reduction, and give the specific construction we will use in our modified GKR protocol.

Definition 5.3. *A* one-to-many reduction *for layer $i \in [D]$ consists of two deterministic functions* $(\mathsf{Reduce}_i, \mathsf{Verify}_i)$*:*

- $\mathsf{Reduce}_i(w, r)$ *takes as input $w \in \mathbb{F}^m$ and $r \in \mathbb{F}$, and outputs a message $R = \{(z^{(j)}, v^{(j)})\}_{j\in[e]}$.*
- $\mathsf{Verify}_i(M, w, r)$ *takes as input a message $R = \{(z^{(j)}, v^{(j)})\}_{j\in[e]}$, $w \in \mathbb{F}^m$, and $r \in \mathbb{F}$, and outputs 0 or 1.*

These functions should satisfy the following guarantees:

- **Correctness:** *If $\hat{V}_i(w) = r$, then for $R = \{(z^{(j)}, v^{(j)})\}_{j\in[e]} \leftarrow \mathsf{Reduce}_i(w, r)$,*

$$\hat{V}_i(z^{(j)}) = v^{(j)} \ \forall j \in [e],$$

and

$$\mathsf{Verify}_i(R, w, r) = 1.$$

- **Statistical Soundness:** *If $\hat{V}_i(w) \neq r$ is false, then for $R = \{(z^{(j)}, v^{(j)})\}_{j\in[e]} \leftarrow \mathsf{Reduce}_i(w, r)$, either*

$$\mathsf{Verify}_i(R, w, r) = 0$$

or there exists $j \in [e]$ such that

$$\hat{V}_i(z^{(j)}) \neq v^{(j)}.$$

- **Efficiency:** Reduce_i *is computable in* $\mathrm{poly}(S)$ *time, and* Verify_i *is computable in* $\mathrm{poly}(e, \log|\mathbb{F}|)$ *time.*

Our one-to-many reduction depends on the following observation: take $\mathbb{F}_0 \subset \mathbb{F}$ to be a field extension of a finite field. A basic fact is that there is a primitive element, i.e. some element $\zeta \in \mathbb{F}$ such that $\mathbb{F}$ is a vector space over $\mathbb{F}_0$, with basis elements $(1, \zeta, \zeta^2, \ldots, \zeta^\beta)$. In particular, β is such that $|\mathbb{F}_0|^{\beta+1} = |\mathbb{F}|$, and we write $\mathbb{F} = \mathbb{F}_0(\zeta)$. Then for any element $t \in \mathbb{F}$, there is a unique polynomial p of degree at most β and coefficients in $\mathbb{F}_0$ for which $p(\zeta) = t$. In fact, we can choose our field representation of $\mathbb{F}$ so that t is represented by p.

For this protocol, we fix a list of $e = m \cdot \beta \cdot (|H|-1) + 1$ values in $\mathbb{F}_0$, known to both the prover and verifier. We can take, for instance, the first e elements of $\mathbb{F}_0$ lexicographically. Call these elements $y^{(1)}, \ldots, y^{(e)}$.

Lemma 5.4. *The protocol in Figure 4 is a one-to-many reduction.*

Proof. To show completeness, note that $\gamma = \tilde{V}_i \circ p$ is a univariate polynomial of degree at most $\deg \tilde{V}_i \cdot \deg p = m(|H|-1) \cdot \beta = e - 1$, and it satisfies the property that $\gamma(y^{(j)}) = v^{(j)}$ for all $j \in [e]$ and $\gamma(\zeta) = r$.

For soundness, note that e points on a univariate polynomial of degree $\leq e - 1$ uniquely determine the polynomial. If indeed $\tilde{V}_i(z^{(j)}) = v^{(j)}$ for every $j \in [e]$, then the polynomial γ through the e points $(y^{(j)}, v^{(j)})$ must in fact be $\tilde{V}_i \circ p$. But then, $\gamma(\zeta) = \tilde{V}_i(p_1(\zeta), \ldots, p_m(\zeta)) \neq r$, so the verification must fail.

The efficiency claims are clear: the bottleneck for Reduce_i is computing $v^{(j)} = \hat{V}_i(z^{(j)})$, which can be done in $\mathrm{poly}(S)$ time. For Verify_i, there are e points $y^{(j)}$, and computing a polynomial through e points can be done using interpolation in $\mathrm{poly}(e, \log|\mathbb{F}|)$ time. □

Remark 5.5. *A rather trivial one-to-many reduction is as follows: let $e = 1$, and define* Reduce_i *and* Verify_i *so that* $\mathsf{Reduce}_i(w, r) = (w, r)$ *and* $\mathsf{Verify}_i((w', r'), w, r) = [w' = w \ \wedge \ r' = r]$*. The* mGKR *protocol given in Figure 5, using this trivial one-to-many reduction, will reproduce the variant of the* GKR *protocol in [40], in which two claims in layer i are reduced to two claims in layer $i - 1$ by running two sum-checks in parallel.*

One-to-Many Reduction

- $\mathsf{Reduce}_i(t, r)$ takes as input $t = (t_1, \ldots, t_m) \in \mathbb{F}^m$ and $r \in \mathbb{F}$, and does the following:
  1. Let $p_1, \ldots, p_m$ be the unique degree β polynomials with coefficients in $\mathbb{F}_0$ such that $p_i(\zeta) = t_i$ for all $i \in [m]$.
  2. Compute $z^{(j)} = p(y^{(j)}) = (p_1(y^{(j)}), \ldots, p_m(y^{(j)})) \in \mathbb{F}_0^m$ for each $j \in [e]$.
  3. Compute $v^{(j)} = \hat{V}_i(z^{(j)})$ for each $j \in [e]$.
  4. Output $R = \{(z^{(j)}, v^{(j)})\}_{j \in [e]}$.
- $\mathsf{Verify}_i(R, t, r)$ takes as input $R = \{(z^{(j)}, v^{(j)})\}_{j \in [e]}$, $t = (t_1, \ldots, t_m) \in \mathbb{F}^m$, and $r \in \mathbb{F}$, and does the following:
  1. Let $p_1, \ldots, p_m$ be the unique degree β polynomials with coefficients in $\mathbb{F}_0$ such that $p_i(\zeta) = t_i$ for all $i \in [m]$.
  2. Compute $z^{(j)'} = p(y^{(j)}) = (p_1(y^{(j)}), \ldots, p_m(y^{(j)})) \in \mathbb{F}_0^m$ for each $j \in [e]$.
  3. Check that $z^{(j)} = z^{(j)'}$ for all $j \in [e]$, rejecting and outputting 0 if $z^{(j)} \neq z^{(j)'}$ for some j.
  4. Construct the unique degree $\leq e - 1 = m\beta(|H|-1)$ polynomial γ such that $\gamma(y^{(j)}) = v^{(j)}$ for all $j \in [e]$. Check that $\gamma(\zeta) = r$, and output 0 if $\gamma(\zeta) \neq r$, and output 1 otherwise.

Figure 4: One-to-many reduction

5.3.2 Analysis of the mGKR Protocol

Theorem 5.6. *Fix any Boolean circuit $C : \{0,1\}^n \to \{0,1\}$ of size S and depth D for which extensions $\widetilde{\mathsf{add}}_i$ and $\widetilde{\mathsf{mult}}_i$ are evaluable in $\mathrm{poly}(m, |H|, \log|\mathbb{F}|)$ time. The mGKR protocol $(P_{\mathsf{mGKR}}, V_{\mathsf{mGKR}})$, described in Figure 5, has the following guarantees:*

- **Completeness:** *For every $x \in \{0,1\}^n$ such that $C(x) = 0$,*

$$\Pr[(P_{\mathsf{mGKR}}, V_{\mathsf{mGKR}})(C, x) = 1] = 1.$$

- **Soundness:** *For every $x \in \{0,1\}^n$ such that $C(x) \neq 0$, and for every (all powerful) cheating prover P^*,*

$$\Pr[(P^*, V_{\mathsf{mGKR}})(C, x) = 1] \leq \frac{D \cdot 6mde}{|\mathbb{F}|},$$

  where $d \leq \mathrm{poly}(H)$ is the individual degree of the polynomials polynomial $f_{i,z}$.

- **Efficiency:** *The prover $P_{\mathsf{mGKR}}(C)$ runs in time $\leq S \cdot \mathrm{poly}(|\mathbb{H}|^m)$. The communication complexity is $\leq D \cdot \mathrm{poly}(m, d, e, |\mathbb{H}|, \log|\mathbb{F}|)$, and the runtime of the verifier is $D \cdot \mathrm{poly}(m, d, e, |\mathbb{H}|, \log|\mathbb{F}|) + n \cdot \mathrm{poly}(m, |H|, \log|\mathbb{F}|)$. Moreover, this protocol is public-coin; i.e., all the messages sent by the verifier are truly random and consist of the verifier's random coin tosses.*

Remark 5.7. *Recall that by Proposition 5.2, any log-space uniform circuit C can be converted into a circuit C' for which there exist extensions $\widetilde{\mathsf{add}}_i$ and $\widetilde{\mathsf{mult}}_i$ that are evaluable in $\mathrm{poly}(m, |H|, \log|\mathbb{F}|)$ time. This new circuit C' has size $S' = \mathrm{poly}(S)$ and depth $D' = D \cdot \mathrm{polylog}(S)$. The guarantees of Theorem 5.6 therefore hold for any log-space uniform C as well.*

Proof. The completeness condition follows immediately from the completeness of the sum-check protocol. The efficiency guarantees follow from Proposition 5.2, together with the efficiency guarantees of the sum-check protocol. We thus focus on proving soundness.

The Modified GKR Protocol $(P_{\mathsf{mGKR}}, V_{\mathsf{mGKR}})(C, x)$

In this protocol, P_{mGKR} proves to V_{mGKR} that $\hat{V}_D(\omega_{\text{out}}) = 0$, where $\omega_{\text{out}} \in H^m$ corresponds to the output wire. This is done in D phases. For every $i \in [D]$, in the i'th phase, the task of proving that $\hat{V}_i(w_i^{(b)}) = r_i^{(b)}$ for $b \in \{1,2\}$ is reduced to the task of proving that $\hat{V}_{i-1}(w_{i-1}^{(b)}) = r_{i-1}^{(b)}$ for $b \in \{1,2\}$. This is done as follows:

1. Set $w_D^{(1)} = w_D^{(2)} = \omega_{\text{out}}$ and $r_D^{(1)} = r_D^{(2)} = 0$, and set $i = D$.
2. P_{mGKR} and V_{mGKR} use a one-to-many reduction on each of the claims $\hat{V}_i(w_i^{(b)}) = r_i^{(b)}$ to reduce to a list of $2e$ claims $\hat{V}_i(z_i^{(j)}) = v_i^{(j)}$, $j \in [2e]$. That is, P_{mGKR} computes $R_i^{(b)} = (\{(z_i^{(j+(b-1)e)}, r_i^{(j+(b-1)e)})\}_{j\in[e]}) \leftarrow \mathsf{Reduce}_i(w_i^{(b)}, r_i^{(b)})$ for $b \in \{1,2\}$ and sends $R_i^{(b)}$ to V_{mGKR}. The verifier then runs $\mathsf{Verify}_i(R_i^{(b)}, w_i^{(b)}, r_i^{(b)})$ for each b and rejects if either check fails. Otherwise, P_{mGKR} and V_{mGKR} replace the two claims $\hat{V}_i(w_i^{(b)}) = r_i^{(b)}$ with the $2e$ claims $\hat{V}_i(z_i^{(j)}) = v_i^{(j)}$ and continue.
3. P_{mGKR} and V_{mGKR} run $2e$ sum-check protocols in parallel: $(P_{\mathsf{SC}}(f_{i,z_i^{(j)}}), V_{\mathsf{SC}}^{f_{i,z_i^{(j)}}}(v_i^{(j)}))$ for $j \in [2e]$, where the verifier uses the same randomness in all sum-checks, and where for all $i \in [D]$ and $z \in \mathbb{F}^m$, $f_{i,z}$ is defined as follows:
$$f_{i,z}(p,\omega_1,\omega_2) = \hat{\mathrm{EQ}}(z,p) \cdot \begin{pmatrix} \widetilde{\mathsf{add}}_i(p,\omega_1,\omega_2) \cdot \left(\hat{V}_{i-1}(\omega_1) + \hat{V}_{i-1}(\omega_2)\right) \\ + \widetilde{\mathsf{mult}}_i(p,\omega_1,\omega_2) \cdot \left(\hat{V}_{i-1}(\omega_1) \cdot \hat{V}_{i-1}(\omega_2)\right) \end{pmatrix},$$
where $p, \omega_1, \omega_2 \in \mathbb{F}^m$. Namely, the prover proves that
$$\hat{V}_i(z_i^{(j)}) = \sum_{p,\omega_1,\omega_2 \in H^m} f_{i,z_i^{(j)}}(p,\omega_1,\omega_2) = v_i^{(j)} \qquad \forall j \in [2e].$$
At the end of this sum-check protocol, V_{mGKR} needs to check that for all $j \in [2e]$,
$$f_{i,z_i^{(j)}}(p,\omega_1,\omega_2) = \rho_i^{(j)},$$
where $p, \omega_1, \omega_2 \in \mathbb{F}^m$ are random elements chosen by V_{SC} in the sum-check protocol (which are the same in all protocols), and $\rho_i^{(j)}$ are values determined by the sum-check protocol.
4. V_{mGKR} asks the prover for the values of $\hat{V}_{i-1}(\omega_1)$ and $\hat{V}_{i-1}(\omega_2)$, and obtains values r_1, r_2.
5. V_{mGKR} checks that indeed for every $j \in [2e]$ it holds that
$$\rho_i^{(j)} = \hat{\mathrm{EQ}}(z_i^{(j)}, p) \cdot \left(\widetilde{\mathsf{add}}_i(p,\omega_1,\omega_2) \cdot (r_1 + r_2) + \widetilde{\mathsf{mult}}_i(p,\omega_1,\omega_2) \cdot (r_1 \cdot r_2)\right),$$
by computing $\hat{\mathrm{EQ}}(z_i^{(j)}, p)$, $\widetilde{\mathsf{add}}_i(p,\omega_1,\omega\_2)$ and $\widetilde{\mathsf{mult}}_i(p,\omega_1,\omega_2)$ himself. If this is not the case it rejects. Otherwise, if $i \geq 1$, then go back to Item 2 with $i = i-1$, with $w_i^{(b)} = \omega_b$ and $r_i^{(b)} = r_b$ for $b \in \{1,2\}$.
6. If $i = 0$, V_{mGKR} checks on his own that indeed for every $b \in \{1,2\}$ it holds that $\hat{V}_0(\omega_b) = r_b$, by evaluating
$$\hat{V}_0(\omega_b) = \sum_{p \in [n]} \hat{\mathrm{EQ}}(\omega_b, p) \cdot x_p,$$
where $[n]$ is the set of input wires.

Figure 5: The Modified GKR delegation protocol $(P_{\mathsf{mGKR}}, V_{\mathsf{mGKR}})$

To this suppose for contradiction that there exists $x \in \{0,1\}^n$ such that $C(x) \neq 0$, and a (non-uniform, all powerful) cheating prover P^*, such that
$$\Pr[(P^*, V_{\mathsf{mGKR}})(C,x) = 1] = p > \frac{D \cdot 6mde}{|\mathbb{F}|}.$$

We assume without loss of generality that P^* is deterministic.[12] Denote by $r_i^{(1)}$ and $r_i^{(2)}$ the two values sent by P^* as supposedly corresponding to $\hat{V}_i(w_i^{(1)})$ and $\hat{V}_i(w_i^{(2)})$, where the values $w_i^{(1)}, w_i^{(2)} \in \mathbb{F}^m$ are determined by the randomness of the verifier in the sumcheck protocol (as described in Figure 5). In particular, with probability p,

$$(r_D^{(1)}, r_D^{(2)}) \neq (\hat{V}_D(w_D^{(1)}), \hat{V}_D(w_D^{(2)})) \text{ and } (r_0^{(1)}, r_0^{(2)}) = (\hat{V}_0(w_0^{(1)}), \hat{V}_0(w_0^{(2)})).$$

By a standard hybrid argument this implies that there exists $i \in [D]$ such that probability at least $\frac{p}{D}$,

$$(r_i^{(1)}, r_i^{(2)}) \neq (\hat{V}_i(w_i^{(1)}), \hat{V}_i(w_i^{(2)})) \text{ and } (r_{i-1}^{(1)}, r_{i-1}^{(2)}) = (\hat{V}_{i-1}(w_{i-1}^{(1)}), \hat{V}_{i-1}(w_{i-1}^{(2)})).$$

P^* and V_{mGKR} apply the one-to-many reduction to the claims $\hat{V}_i(w_i^{(1)}) = r_i^{(1)}$ and $\hat{V}_i(w_i^{(2)}) = r_i^{(2)}$ to obtain $2e$ claims $\hat{V}_i(z_i^{(j)}) = v_i^{(j)}$, $j \in [2e]$. By the statistical soundness of the one-to-many reduction, since V_{mGKR} does not reject during the verification check, there exists $j \in [2e]$ for which $v_i^{(j)} \neq \hat{V}_i(z_i^{(j)})$. In other words, there exists $i \in [D]$ and $j \in [2e]$ such that with probability at least $\frac{p}{2e \cdot D}$,

$$v_i^{(j)} \neq \hat{V}_i(z_i^{(j)}) \text{ and } (r_{i-1}^{(1)}, r_{i-1}^{(2)}) = (\hat{V}_{i-1}(w_{i-1}^{(1)}), \hat{V}_{i-1}(w_{i-1}^{(2)})). \tag{6}$$

We argue that this contradicts the soundness of the sum-check protocol w.r.t. the multivariate polynomial

$$\hat{V}_i(z_i^{(j)}) = \sum_{p,\omega_1,\omega_2 \in \mathbb{H}^m} f_{i,z_i^{(j)}}(p, \omega_1, \omega_2),$$

where recall that

$$f_{i,z}(p,\omega_1,\omega_2) = \hat{\mathrm{EQ}}(z,p) \cdot \left[\widetilde{\mathsf{add}}_i(p,\omega_1,\omega_2) \cdot \left(\hat{V}_{i-1}(\omega_1) + \hat{V}_{i-1}(\omega_2)\right) + \widetilde{\mathsf{mult}}_i(p,\omega_1,\omega_2) \cdot \hat{V}_{i-1}(\omega_1) \cdot \hat{V}_{i-1}(\omega_2)\right].$$

Indeed, consider the following cheating prover P^*_{SC} that breaks the soundness of the sum-check protocol. Let trans_i be an emulation of a prefix of the interaction between (P^*, V_{mGKR}), until the point where the values $z_i^{(j)}$ are determined, such that the following two conditions hold:

1. $v_i^{(j)} \neq \hat{V}_i(z_i^{(j)})$.
2. $\Pr[(r_{i-1}^{(1)}, r_{i-1}^{(2)}) = (\hat{V}_{i-1}(w_{i-1}^{(1)}), \hat{V}_{i-1}(w_{i-1}^{(2)})) \mid \mathsf{trans}_i] \geq \frac{p}{2e \cdot D}$.

The prover P^*_{SC} will break soundness of the sum-check protocol by convincing the verifier V_{SC} to accept the false statement $v_i^{(j)} = \sum_{p,\omega_1,\omega_2 \in \mathbb{H}^m} f_{i,z_i^{(j)}}(p,\omega_1,\omega_2)$ with probability at least $\frac{3md}{|\mathbb{F}|}$.

The prover P^*_{SC} has trans_i hard-wired, and emulates the mGKR cheating prover P^*, as follows: in round $\iota \in [3m]$, given the prefix $t_1, \ldots, t_\iota \leftarrow \mathbb{F}$ sent by verifier V_{SC} in the first ι rounds, emulate $P^*(\mathsf{trans}_i, t_1, \ldots, t_\iota)$ to obtain $(g_\iota^{(1)}, \ldots, g_\iota^{(2e)})$, and send $g_\iota^{(j)}$ to V_{SC}. The fact that Equation (6) holds with probability at least $\frac{p}{2e \cdot D} > \frac{3md}{\mathbb{F}}$ contradicts the soundness guarantee of the sum-check protocol. □

6 Our Non-Interactive mGKR Protocol

In this section, we apply the Fiat-Shamir paradigm to the mGKR protocol described in Section 5, by using the non-interactive parallel sum-check procedure described in Section 4.3.

Recall that mGKR is an interactive protocol to delegate the computation of $C(x)$, where $C : \{0,1\}^n \to \{0,1\}$ is a circuit of size S and depth D, and $x \in \{0,1\}^n$ is an input. At a high level, the protocol has the following structure: for each layer i of the circuit, we first reduce two claims of the form $\hat{V}_i(w_i^{(b)})$, $b \in \{1,2\}$ to $2e$ claims $\{\hat{V}_i(z_i^{(j)})\}_{j \in [2e]}$ using two applications of our one-to-many reduction, and then we run $2e$ sum-checks

[12] This assumption is without loss of generality since if P^* was randomized, one can simply hard-wire its random coin tosses.

in parallel. At the end of these $2e$ sum-checks, we once again are left with two claims $\hat{V}_{i-1}(w_{i-1}^{(b)}) = r_{i-1}^{(b)}$, $b \in \{1,2\}$, for the next layer.

In this section, we present a non-interactive version of the mGKR protocol. Our non-interactive mGKR protocol is similar to the interactive one, except that the parallel sum-check protocols are replaced with their non-interactive version, as defined in Figure 3.

Let ϵ and m be such that $1/\epsilon \lesssim m \lesssim \log\log S$, and let $\kappa = (\log S)^{2/\delta}$. Let $\mathbb{F}$ be a field of size $2^{(\log S)^{2\cdot(3/\delta)^{3m}}} \geq 2^{\kappa^{2\cdot(3/\delta)^{3m-1}}}$, and let $H \subset \mathbb{F}$ be a subset of size $S^{1/m}$, so $|H|^m = S$. Furthermore, take $\mathbb{F}_0 \subset \mathbb{F}$ to be a subfield of size $\mathrm{poly}(|H|)$.

Our non-interactive mGKR protocol has the following structure: for each layer $i \in [D]$, the prover begins with two claims $\hat{V}_i(w_i^{(b)}) = r_i^{(b)}$. He will do two things. First, he'll apply the one-to-many reduction given in Section 5.3.1 to reduce each of the two claims $\hat{V}_{i-1}(w_{i-1}^{(b)}) = r_{i-1}^{(b)}$ to $e = 3m\beta(|H|-1)+1 \leq 3m\cdot|H|\cdot\log|\mathbb{F}|$ new claims $\hat{V}_{i-1}(z_{i-1}^{(j+(b-1)e)}) = v_{i-1}^{(j+(b-1)e)}$, each of which satisfies $z_{i-1}^{j} \in \mathbb{F}_0^{3m}$. Next, he'll generate an argument for the $2e$ parallel sum-checks

$$\hat{V}_i(z_i^{(j)}) = \sum_{p,\omega_1,\omega_2 \in H^m} f_{i,z_i^{(j)}}(p,\omega_1,\omega_2) = v_i^{(b)},$$

using the non-interactive parallel sum-check procedure given in Figure 3. Each of the polynomials $f_{i,z}$ is a $3m$-variate polynomial with individual degree $\leq d = \mathrm{poly}(|H|)$. Finally, at the end of D layers, he sends the verifier the messages of the one-to-many reductions, as well as the D arguments for the parallel sum-checks.

Our non-interactive, publicly verifiable mGKR protocol is described in Figure 6.

Remark 6.1. *The* CRS *for our non-interactive* mGKR *protocol is the* D CRS*'s for the* D *parallel sum-checks. As noted in Remark 4.6, each of these* D CRS*'s are indistinguishable from a random string under the sub-exponential LWE assumption. Therefore, the* CRS *for our non-interactive* mGKR *protocol is also distinguishable from a random string of the same length.*

Theorem 6.2. *For* $\epsilon = \epsilon(S) = \omega(1/\log\log S)$*, there are* m*,* $|H|$*, and* $|\mathbb{F}|$ *such that for any Boolean circuit* $C : \{0,1\}^n \to \{0,1\}$ *of size* S *and depth* D *for which* $\widetilde{\mathsf{add}}_i$ *and* $\widetilde{\mathsf{mult}}_i$ *are evaluable in* $\mathrm{poly}(m, |H|, \log|\mathbb{F}|)$ *time, the non-interactive* mGKR *protocol* $(P_{\mathsf{niGKR}}, V_{\mathsf{niGKR}})$*, described in Figure 6, has the following guarantees:*

- **Completeness:** *For any* $x \in \{0,1\}^n$ *such that* $C(x) = 0$*,*

$$\Pr[(P_{\mathsf{niGKR}}, V_{\mathsf{niGKR}})(C,x) = 1] = 1.$$

- **Soundness:** *For any* $\mathrm{poly}(S)$*-size cheating prover* P^* *and* $x \in \{0,1\}^n$ *such that* $C(x) \neq 0$*,*

$$\Pr[(P^*, V_{\mathsf{niGKR}})(C,x) = 1] = \mathrm{negl}(S),$$

  assuming the sub-exponential hardness of LWE*.*

- **Efficiency:** *The prover* $P_{\mathsf{niGKR}}(C)$ *runs in time* $\leq \mathrm{poly}(S)$*. The communication complexity is* $o(D\cdot S^{\epsilon})$*, and the runtime of the verifier is* $o((D+n)\cdot S^{\epsilon})$*.*

Proof. In what follows, let m be so that $1/\epsilon \lesssim m \lesssim \log\log S$, and let $\mathbb{F}$ be a field of size $2^{(\log S)^{2\cdot(3/\delta)^{3m}}}$ and $H \subset \mathbb{F}$ be a set of size $S^{1/m}$. We split our proof into the three components.

Completeness - Completeness follows from the correctness of the mGKR protocol described in Figure 5 and the correctness of our parallel sum-check protocol in Figure 3.

Soundness - Next, we prove soundness. Suppose that there is a $\mathrm{poly}(S)$-size cheating prover P^* such that

$$\Pr_{\pi \leftarrow P^*(\mathrm{CRS})}[V_{\mathsf{niGKR}}(\mathrm{CRS}, C, x, \pi) = 1] > p(S),$$

Non-Interactive mGKR $(P_{\mathsf{niGKR}}, V_{\mathsf{niGKR}})(C, x)$

- **Setup:** For $i \in [D]$, generate CRS_i as in the setup of the non-interactive parallel sum-check protocol $(P_{\mathsf{niPSC}}, V_{\mathsf{niPSC}})$ described in Figure 3. Set $\mathrm{CRS} = (\mathrm{CRS}_1, \ldots, \mathrm{CRS}_D)$.
- **Prover:** P_{niGKR} follows the following procedure to generate the proof string:
  1. Set $w_D^{(1)} = w_D^{(2)} = \omega_{\text{out}}$ and $r_D^{(1)} = r_D^{(2)} = 0$, and set $i = D$.
  2. For $b \in \{1, 2\}$, compute $R_i^{(b)} = \{(z_i^{(j+(b-1)e)}, v_i^{(j+(b-1)e)})\}_{j \in [e]} \leftarrow \mathsf{Reduce}_i(w_i^{(b)}, r_i^{(b)})$, where Reduce_i is given by the one-to-many reduction described in Figure 4.
  3. Recall that
  $$\hat{V}_i(z_i^{(j)}) = \sum_{p, \omega_1, \omega_2 \in H} f_{i, z_i^{(j)}}(p, \omega_1, \omega_2) \quad \forall j \in [2e],$$
  where
  $$f_{i,z}(p, \omega_1, \omega_2) = \hat{\mathrm{EQ}}(z, p) \cdot \begin{pmatrix} \widetilde{\mathsf{add}}_i(p, \omega_1, \omega_2) \cdot \left(\hat{V}_{i-1}(\omega_1) + \hat{V}_{i-1}(\omega_2)\right) \\ + \widetilde{\mathsf{mult}}_i(p, \omega_1, \omega_2) \cdot \left(\hat{V}_{i-1}(\omega_1) \cdot \hat{V}_{i-1}(\omega_2)\right) \end{pmatrix}.$$
  Generate a proof string for the $2e$ claims $\{\hat{V}_i(z_i^{(j)}) = v_i^{(j)}\}_{j \in [2e]}$ by running the non-interactive parallel sum-check procedure $(P_{\mathsf{niPSC}}, V_{\mathsf{niPSC}})$ from Figure 3 on the $2e$ claims
  $$\sum_{p, \omega_1, \omega_2 \in H} f_{i, z_i^{(j)}}(p, \omega_1, \omega_2) = v_i^{(j)},$$
  using CRS_i. P_{niGKR} thus obtains a proof string π_i.
  4. At the end of $2e$ parallel sum-checks, there will be a specific point $(p, \omega_1, \omega_2) \in \mathbb{F}^{3m}$ determined by the outputs of the hash functions used to simulate verifier queries. P_{niGKR} computes $r_{i-1}^{(1)} = \hat{V}_{i-1}(\omega_1)$ and $r_{i-1}^{(2)} = \hat{V}_{i-1}(\omega_2)$, and sets $w_{i-1}^{(1)} = \omega_1$ and $w_{i-1}^{(2)} = \omega_2$. Set $i = i - 1$, and if $i \geq 1$, return to Item 2.

  P_{niGKR} sends $\pi = \left\{ \left(\pi_i, \left(w_i^{(b)}, r_i^{(b)}, R_i^{(b)} \right)_{b \in \{1,2\}} \right) \right\}_{i \in [D]}$ to V_{niGKR}.
- **Verifier:** Upon obtaining $\pi = \left\{ \left(\pi_i, \left(w_i^{(b)}, r_i^{(b)}, R_i^{(b)} \right)_{b \in \{1,2\}} \right) \right\}_{i \in [D]}$, V_{niGKR} does the following checks:
  - Check $w_D^{(1)} = w_D^{(2)} = \omega_{\text{out}}$ and $r_D^{(1)} = r_D^{(2)} = 0$.
  - For each $i \in [D]$ and $b \in \{1, 2\}$, run $\mathsf{Verify}_i(R_i^{(b)}, w_i^{(b)}, r_i^{(b)})$, and reject if either output is 0.
  - Check that $w_i^{(1)}$ and $w_i^{(2)}$ are indeed the outputs of the hash functions used in the parallel sum-check procedure. Verify that each π_i is a valid proof for the $2e$ claims $\hat{V}_i(z_i^{(j)}) = v_i^{(j)}$ by running $V_{\mathsf{niPSC}}(\overline{v}_i, \mathrm{CRS}_i, \pi_i)$, replacing V_{niPSC}'s oracle queries to $f_{i, z_i^{(j)}}$ with the computations
  $$f_{i, z_i^{(j)}}(p, w_i^{(1)}, w_i^{(2)}) = \hat{\mathrm{EQ}}(z^{(j)}, p) \cdot \left(\widetilde{\mathsf{add}}_i(p, w_i^{(1)}, w_i^{(2)}) \cdot (r_i^{(1)} + r_i^{(2)}) + \widetilde{\mathsf{mult}}_i(p, w_i^{(1)}, w_i^{(2)}) \cdot (r_i^{(1)} \cdot r_i^{(2)}) \right).$$
  - Compute $\hat{V}_0(w_0^{(1)})$ and $\hat{V}_0(w_0^{(2)})$ and verify that $r_0^{(b)} = \hat{V}_0(w_0^{(b)})$ for $b \in \{1, 2\}$.

  If any of these checks fail, V_{niGKR} rejects. Otherwise, he accepts.

Figure 6: Non-interactive GKR protocol $(P_{\mathsf{niGKR}}, V_{\mathsf{niGKR}})$

for some non-negligible $p(S)$. Then by a standard hybrid argument, there is some layer $i \in [D]$ for which

with probability at least $\frac{p}{D}$,

$$(r_i^{(1)}, r_i^{(2)}) \neq (\hat{V}_i(w_i^{(1)}), \hat{V}_i(w_i^{(2)})) \text{ and } (r_{i-1}^{(1)}, r_{i-1}^{(2)}) = (\hat{V}_{i-1}(w_{i-1}^{(1)}), \hat{V}_{i-1}(w_{i-1}^{(2)})).$$

By the statistical soundness of the one-to-many protocol, this means that with probability at least $\frac{p}{D}$,

$$\{v_i^{(j)}\}_{j\in[2e]} \neq \{\hat{V}_i(z_i^{(j)})\}_{j\in[2e]} \text{ and } (r_{i-1}^{(1)}, r_{i-1}^{(2)}) = (\hat{V}_{i-1}(w_{i-1}^{(1)}), \hat{V}_{i-1}(w_{i-1}^{(2)})).$$

However, this contradicts the semi-adaptive soundness of the non-interactive parallel sum-check protocol $(P_{\mathsf{niPSC}}, V_{\mathsf{niPSC}})$ from Figure 3: each $z_i^{(j)}$ is chosen from $\mathbb{F}_0^m$, so the polynomials $f_{i,z_i^{(j)}}$ belong to a set

$$\Delta_i = \left\{ f_{i,z} : (p,\omega_1,\omega_2) \mapsto \hat{\mathrm{EQ}}(z,p) \cdot \begin{pmatrix} \widetilde{\mathsf{add}}_i(p,\omega_1,\omega_2) \cdot \left(\hat{V}_{i-1}(\omega_1) + \hat{V}_{i-1}(\omega_2)\right) \\ + \widetilde{\mathsf{mult}}_i(p,\omega_1,\omega_2) \cdot \left(\hat{V}_{i-1}(\omega_1) \cdot \hat{V}_{i-1}(\omega_2)\right) \end{pmatrix} \right\}_{z\in\mathbb{F}_0^m}.$$

In particular, $|\Delta| = |\mathbb{F}_0|^m = (\mathrm{poly}(|H|))^m = \mathrm{poly}(S)$. Thus, noting that $\frac{p}{D}$ is non-negligible in S since $D \leq S$, we get a contradiction on the semi-adaptive soundness guarantee in Theorem 4.4, which states that this should happen with probability negligible in S.

Efficiency - To analyze the efficiency of $(P_{\mathsf{niGKR}}, V_{\mathsf{niGKR}})$, we need the following two lemmas, which bound the size of $|\mathbb{F}|$ and $|H|$, respectively. Recall that $1/\epsilon \lesssim m \lesssim \log\log S$.

Lemma 6.3. *For any constant $\mu > 0$, $(\log|\mathbb{F}|)^\mu = o(S^\epsilon)$.*

Proof. Recall that $\log|\mathbb{F}| = (\log S)^{2\cdot(3/\delta)^{3m}}$. Because $m, 1/\epsilon = o(\log\log S)$, we have that

$$\begin{aligned} &3m\cdot\log(3/\delta) + \log\log\log S + \log(1/\epsilon) \lesssim \log\log S \\ \iff &3m\cdot\log(3/\delta) + \log\log\log S \lesssim \log\epsilon + \log\log S \\ \implies &(3/\delta)^{3m}\cdot\log\log S \lesssim \epsilon\log S \\ \implies &(\log S)^{2\cdot(3/\delta)^{3m}} \lesssim S^\epsilon \\ \iff &\mathrm{polylog}|\mathbb{F}| = \mathrm{poly}((\log S)^{2\cdot(3/\delta)^{3m}}) \lesssim S^\epsilon. \end{aligned}$$

□

Lemma 6.4. *For any constant $\mu > 0$, $|H|^\mu = o(S^\epsilon)$.*

Proof. This is easy to see as $|H| = S^{1/m} = S^{o(\epsilon)}$. □

Now, we analyze the efficiency of $(P_{\mathsf{niGKR}}, V_{\mathsf{niGKR}})$.

First, we look at the prover's runtime. In each of D layers, P_{niGKR} has to do two things. The first is to run $e = O(m\cdot|H|\cdot\log|\mathbb{F}|)$ non-interactive sum-checks in parallel. By Theorem 4.4, the prover runtime for this parallel sum-check is $\mathrm{poly}(S, d^m, e, \log|\mathbb{F}|) = \mathrm{poly}(S, \log|\mathbb{F}|)$, where we use that $d = \mathrm{poly}(|H|)$ and $e = \mathrm{poly}(|H|, \log|\mathbb{F}|)$. The second thing P_{niGKR} has to do is perform two one-to-many reductions, each of which involves computing $\hat{V}_i$ at e points. This can be done in $S\cdot\mathrm{poly}(m, |H|, \log|\mathbb{F}|) = \mathrm{poly}(S, \log|\mathbb{F}|)$ time, by Proposition 3.2. Altogether, the prover runtime is $D\cdot\mathrm{poly}(S, \log|\mathbb{F}|)$. Using from Lemma 6.3 that $\log|\mathbb{F}| = o(S^\epsilon)$, we obtain that P_{niGKR} runs in $\mathrm{poly}(S)$ time.

For the communication complexity, note that for each of the D layers, the prover generates a proof string for the parallel sum-check as well as sends over the outputs of Reduce_i for the one-to-many reduction. By Theorem 4.4, the proof string for the parallel sum-check has length $\mathrm{poly}(m, d, e, \log|\mathbb{F}|) = \mathrm{poly}(|H|, \log|\mathbb{F}|)$. The output of Reduce_i is a list of $O(e) = \mathrm{poly}(|H|, \log|\mathbb{F}|)$ values in $\mathbb{F}$. In total, for each of D layers, the communication complexity is $\mathrm{poly}(|H|, \log|\mathbb{F}|)$. By Lemmas 6.4 and 6.3, $\mathrm{poly}(H) \lesssim S^{\epsilon/2}$ and $\mathrm{poly}(\log|\mathbb{F}|) \lesssim S^{\epsilon/2}$, so the total communication complexity is $D\cdot\mathrm{poly}(|H|, \log|\mathbb{F}|) = o(D\cdot S^\epsilon)$.

Finally, we check the verifier's runtime. The verifier has to check three things: verify each parallel sum-check, verify each one-to-many reduction, and verify the values of $\hat{V}_0(w_0^{(b)})$ for $b \in \{1,2\}$. For the first, each of D parallel sum-checks can be verified in $\text{poly}(d, m, e, |H|, \log|\mathbb{F}|) = \text{poly}(|H|, \log|\mathbb{F}|) = o(S^\epsilon)$ time, by Theorem 4.4 and the arguments above. Each of D one-to-many reductions can be verified by interpolating $e = \text{poly}(|H|, \log|\mathbb{F}|)$ points to get a degree $e-1$ polynomial and evaluating this polynomial at a point. This process takes $\text{poly}(|H|, \log|\mathbb{F}|) = o(S^\epsilon)$ as well. Finally, to compute $\hat{V}_0(w_0^{(b)})$, $b \in \{1,2\}$, recall that $\hat{V}_0$ is the LDE of V_0, which on H^m gives the value of the input corresponding input wire. As a slight abuse of notation, let $[n]$ denote the set of labels in H^m corresponding to input wires. Then, we can write:

$$\hat{V}_0(w) = \sum_{p \in [n] \subset H^m} \hat{\text{EQ}}(w, p) \cdot V_0(p).$$

By Proposition 3.1, $\hat{\text{EQ}}(w, p)$ can be evaluated in $\text{poly}(m, |H|, \log|\mathbb{F}|) = o(S^\epsilon)$ time, so computing $\hat{V}_0(w_0^{(b)})$ takes $o(n \cdot S^\epsilon)$ time. In total, all verifier checks can be performed in $o((D+n) \cdot S^\epsilon)$ time.

□

In light of Proposition 5.2, Theorem 6.2 gives a non-interactive argument for any log-space uniform Boolean circuit $C : \{0,1\}^n \to \{0,1\}$. This is done by running $(P_{\mathsf{niGKR}}, V_{\mathsf{niGKR}})$ on the circuit C', which is of size $S' = \text{poly}(S)$ and depth $D' = D \cdot \text{polylog}(S)$.

Corollary 6.5. *Assuming the sub-exponential hardness of* LWE*, for any $\epsilon = \epsilon(S) = \omega(1/\log\log S)$ and any log-space uniform circuit $C : \{0,1\}^n \to \{0,1\}$ of size S and depth D, there is a non-interactive argument for $\{x : C(x) = 0\}$ with completeness 1 and soundness $\text{negl}(S)$. The prescribed prover strategy runs in $\text{poly}(S)$ time, the communication complexity is bounded by $D \cdot S^\epsilon$, and the verifier runtime is bounded by $(D+n) \cdot S^\epsilon$.*

7 PPAD Hardness

The main result of [25], restated to align with our non-interactive sum-check protocol, is the following theorem:

Theorem 7.1. *[25] The class* CLS *is hard (on average), assuming that* #SAT *with $\log(n) \cdot m$ variables is hard (on average), where $m = o(\log\log n)$, and assuming that the non-interactive sum-check protocol $(P_{\mathsf{niSC}}, V_{\mathsf{niSC}})$ from Figure 2 (obtained by applying the Fiat-Shamir transformation), has the following guarantees for a field $\mathbb{F}$ such that $\log|\mathbb{F}| = \text{poly}(n)$, $|H| = n$, and $d = \text{poly}(n)$. In what follows, H_i denotes the image space of the i'th Fiat-Shamir hash function, and CRS_j denotes the final $m-j$ hash keys $(k_{j+1}, \ldots, k_m)$, where $\text{CRS} = (k_1, \ldots, k_m)$.*

- **Prefix-adaptive soundness:** *For every m-variate polynomial $g : \mathbb{F}^m \to \mathbb{F}$ of individual degree d, every $j \in [0, m]$, and every $\text{poly}(n)$-size cheating prover P^*,*

$$\Pr\left[P^*(\text{CRS}) = (\beta_1, \ldots, \beta_j, v, \pi) : \begin{cases} \beta_i \in H_i \ \forall\ i \in [j] \\ \wedge \sum_{h_{j+1}, \ldots, h_m \in H} g_{\beta_1, \ldots, \beta_j}(h_{j+1}, \ldots, h_m) \neq v \\ \wedge\ V_{\mathsf{niSC}}^{g_{\beta_1, \ldots, \beta_j}}(v, \text{CRS}_j, \pi) = 1 \end{cases} \right] = \text{negl}(n),$$

  where $g_{\beta_1, \ldots, \beta_j} : \mathbb{F}^{m-j} \to \mathbb{F}$ is an $(m-j)$-variate polynomial of individual degree d defined by $g_{\beta_1, \ldots, \beta_j}(h_{j+1}, \ldots, h_m) \triangleq g(\beta_1, \ldots, \beta_j, h_{j+1}, \ldots, h_m)$.

- **Prefix-adaptive unambiguity:** *For every m-variate polynomial $g : \mathbb{F}^m \to \mathbb{F}$ of individual degree d,*

every $j \in [0, m]$*, and every* $\mathrm{poly}(n)$*-size cheating prover* P^**,*

$$\Pr\left[P^*(\mathrm{CRS}) = (\beta_1, \ldots, \beta_j, v, \pi_1, \pi_2) : \begin{cases} \beta_i \in H_i \ \forall\ i \in [j] \\ \wedge\ \pi_1 \neq \pi_2 \\ \wedge\ V_{\mathsf{niSC}}^{g_{\beta_1,\ldots,\beta_j}}(v, \mathrm{CRS}_j, \pi_1) = 1 \\ \wedge\ V_{\mathsf{niSC}}^{g_{\beta_1,\ldots,\beta_j}}(v, \mathrm{CRS}_j, \pi_2) = 1 \end{cases}\right] = \mathrm{negl}(n).$$

- **Efficiency:** V_{niSC} *runs in time* $\mathrm{poly}(n)$*.*

In Section 7.1, we show that our non-interactive sum-check protocol does indeed satisfy the properties of prefix-adaptive soundness and prefix-adaptive unambiguity (Theorem 7.3), and that for $\kappa = (m \log n)^{2/\delta}$ and $|\mathbb{F}| = 2^{(m \log n)^{2\cdot(3/\delta)^{m-1}}}$, the size of $\log|\mathbb{F}|$ and the runtime of V_{niSC} are both $\mathrm{poly}(n)$ (Lemma 7.4). We thus have the following corollary:

Corollary 7.2. *Assuming that* LWE *is sub-exponentially hard and that* $\#\mathsf{SAT}$ *with* $o(\log n \cdot \log\log n)$ *variables is hard (on average), then* CLS *is hard (on average).*

We note that Theorem 7.1, as stated, differs from the theorem in [25]. Specifically, in [25] they require *full adaptivity* of soundness and unambiguity, and in particular, allow the cheating prover to choose the multivariate polynomial g adaptively as well. While we do not guarantee that our non-interactive sum-check protocol has full adaptivity, upon inspecting their proof, we notice that they do not need the adaptivity on g (and probably require it only for the sake of simplicity), and only need the adaptivity of $\beta_1, \ldots, \beta_j$ and v.

The rest of this section is split into two parts. In Section 7.1, we show that our non-interactive sum-check protocol satisfies prefix-adaptive soundness and unambiguity. Then, in Section 7.2, we give a sketch of the proof of Theorem 7.1.

7.1 Analysis of the Non-Interactive Sum-Check Protocol

In what follows, we prove that our non-interactive sum-check protocol in Figure 2 satisfies the prefix-adaptive soundness and unambiguity properties. The proof of these properties follows from the same argument as that of semi-adaptive soundness in Theorem 4.4, so we gloss the outline of the proof without diving too much into details, referring the reader to the proof of Theorem 4.4 for a more detailed argument. In fact, we prove something stronger, that the prefix-adaptive soundness and unambiguity guarantees hold with probability negligible in $N = |H|^m$ against cheating provers that are size polynomial in N, as opposed to with probability negligible in n against provers of size polynomial in n.

Theorem 7.3. *Let* $\kappa = (\log N)^{2/\delta}$ *and* $|\mathbb{F}| = 2^{\kappa^{2\cdot(3/\delta)^{m-1}}}$*, where* $N = |H|^m$*. Assuming that* LWE *is sub-exponentially hard, the protocol in Figure 2 has the following guarantees:*

- **Prefix-adaptive soundness:** *For every* m*-variate polynomial* $g : \mathbb{F}^m \to \mathbb{F}$ *of individual degree* d*, every* $j \in [0, m]$*, and every* $\mathrm{poly}(N)$*-size cheating prover* P^**,*

$$\Pr\left[P^*(\mathrm{CRS}) = (\beta_1, \ldots, \beta_j, v, \pi) : \begin{cases} \beta_i \in H_i \ \forall\ i \in [j] \\ \wedge \sum_{h_{j+1},\ldots,h_m \in H} g_{\beta_1,\ldots,\beta_j}(h_{j+1}, \ldots, h_m) \neq v \\ \wedge\ V_{\mathsf{niSC}}^{g_{\beta_1,\ldots,\beta_j}}(v, \mathrm{CRS}_j, \pi) = 1 \end{cases}\right] = \mathrm{negl}(N),$$

where $g_{\beta_1,\ldots,\beta_j} : \mathbb{F}^{m-j} \to \mathbb{F}$ *is an* $(m-j)$*-variate polynomial of individual degree* d *defined by* $g_{\beta_1,\ldots,\beta_j}(h_{j+1}, \ldots, h_m) \triangleq g(\beta_1, \ldots, \beta_j, h_{j+1}, \ldots, h_m)$*.*

- **Prefix-adaptive unambiguity:** *For every m-variate polynomial $g : \mathbb{F}^m \to \mathbb{F}$ of individual degree d, every $j \in [0, m]$, and every* poly(N)*-size cheating prover P^*,*

$$\Pr\left[P^*(\text{CRS}) = (\beta_1, \ldots, \beta_j, v, \pi_1, \pi_2) : \begin{cases} \beta_i \in H_i \ \forall\ i \in [j] \\ \wedge\ \pi_1 \neq \pi_2 \\ \wedge\ V_{\mathsf{niSC}}^{g_{\beta_1,\ldots,\beta_j}}(v, \text{CRS}_j, \pi_1) = 1 \\ \wedge\ V_{\mathsf{niSC}}^{g_{\beta_1,\ldots,\beta_j}}(v, \text{CRS}_j, \pi_2) = 1 \end{cases}\right] = \text{negl}(N).$$

Proof. Prefix-adaptive unambiguity follows from the proof of prefix-adaptive soundness, and specifically from the fact that no poly(N)-size prover can form a proof π that goes from an incorrect statement to a correct one, except with negligible probability. We focus on proving prefix-adaptive soundness. To this end, fix any m-variate polynomial of individual degree d, an index $j \in [0, m]$, and a poly(N)-size cheating prover P^*. We prove the stronger statement that

$$\Pr\left[P^*(\text{CRS}) = (\beta_1, \ldots, \beta_j, v, \pi) : \begin{cases} \beta_i \in H_i \ \forall\ i \in [j] \\ \wedge \sum_{h_{j+1},\ldots,h_m \in H} g_{\beta_1,\ldots,\beta_j}(h_{j+1}, \ldots, h_m) \neq v \\ \wedge\ V_{\mathsf{niSC}}^{g_{\beta_1,\ldots,\beta_j}}(v, \text{CRS}_j, \pi) = 1 \end{cases}\right] = \epsilon \leq 2(m-j)d/2^{\kappa^\delta} = \text{negl}(N).$$

To this end suppose for contradiction that

$$\epsilon > 2(m-j)d/2^{\kappa^\delta}. \tag{7}$$

For all $i > j$, let

$$g_i(\cdot) = \sum_{h_{i+1},\ldots,h_m} g(\beta_1, \ldots, \beta_j, t_{j+1}, \ldots, t_{i-1}, \cdot, h_{i+1}, \ldots, h_m)$$

denote the correct value of the i'th polynomial, and let $\tilde{g}_i$ denote the polynomial that P^* actually sends. By an averaging argument, there must exist some round $r \in [j+1, m]$, $\ell \in [d]$, and values $\beta'_1, \ldots, \beta'_j, t'_{j+1}, \ldots, t'_{r-1}$ (where $\beta'_i \in H_i$ and $t'_i \in H_i$) for which, with probability at least $\frac{\epsilon}{(m-j)\cdot d\cdot|H_1|\cdot\cdots\cdot|H_{r-1}|}$, the following four things hold: (1) $\tilde{g}_r \neq g_r$, (2) t_r is the ℓ'th root of $\tilde{g}_r - g_r$, (3) $\beta_i = \beta'_i$ for all $i \in [j]$, and (4) $t_i = t'_i$ for all $i \in [j+1, r-1]$. Call these simultaneous four conditions event B, and let a_r denote the correct value of g_r when $\beta_i = \beta'_i$ for all $i \in [j]$ and $t_i = t'_i$ for all $i \in [j+1, r-1]$.

Originally, the r'th hash key is $k_r = (\mathsf{pk}_r, \mathsf{ct}_r)$, where $\mathsf{ct}_r = \mathsf{Enc}(\mathsf{pk}_r, \xi_{\kappa_r} \circ \varphi_{0^{d+1},1})$ is the encryption of the function that outputs the first root of $\tilde{g}_r - 0$ if it is in H_r, else it outputs $0 \in H_r$. When we change k_r to $k'_r = (\mathsf{pk}_r, \mathsf{ct}'_r)$ (where $\mathsf{ct}'_r = \mathsf{Enc}(\mathsf{pk}_r, \xi_{\kappa_r} \circ \varphi_{a_r,\ell})$ is the encryption of the function that outputs the ℓ'th root of $\tilde{g}_r - a_r$ if it is in H_r, else it outputs $0 \in H_r$), the δ-sub-exponential security of the encryption scheme means that the probability of event B occurring cannot change by more than

$$\frac{1}{2^{\kappa_r^\delta}} \leq \frac{1}{2^{\kappa^\delta}\cdot|H_1|\cdot\cdots\cdot|H_{r-1}|} < \frac{\epsilon}{2(m-j)\cdot d\cdot|H_1|\cdot\cdots\cdot|H_{r-1}|}.$$

The proof of the left inequality is in the proof of Theorem 4.4, and the right inequality follows from our contradiction assumption (see Equation (7)). This means that the probability of event B occurring should still be greater than $\frac{\epsilon}{2(m-j)\cdot d\cdot|H_1|\cdot\cdots\cdot|H_{r-1}|}$. However, this yields a contradiction by Lemma 4.2, since no poly(N)-size prover should be able to find the ℓ'th root of $\tilde{g}_r - a_r$ with probability more than $\frac{1}{2^{\kappa_r^\delta}} < \frac{\epsilon}{2(m-j)\cdot d\cdot|H_1|\cdot\cdots\cdot|H_{r-1}|}$. □

Next, we check that for the setting of parameters stated in Theorem 7.1, the size of $\log|\mathbb{F}|$ and the verifier runtime of the non-interactive sum-check protocol are both polynomial in n.

Lemma 7.4. *For $|H|= n$, $m = o(\log\log n)$, $\kappa = (\log N)^{2/\delta}$, and $|\mathbb{F}|= 2^{\kappa^{2\cdot(3/\delta)^{m-1}}}$, it holds that $\log|\mathbb{F}| = \text{poly}(n)$. Furthermore, the runtime of V_{niSC} (in Figure 2) for a m-variate sum-check polynomial of degree $d = \text{poly}(n)$ is $\text{poly}(n)$.*

Proof. Because $m = o(\log\log n)$ and δ is a constant, we have that

$$m \cdot \log(3/\delta) \lesssim \log\log n.$$

This implies that

$$\begin{aligned} m \cdot \log(3/\delta) + \log\log(\log n + \log m) &\lesssim \log\log n \\ \implies 2 \cdot (3/\delta)^m \cdot \log\log N &\lesssim \log n \\ \implies (\log N)^{2\cdot(3/\delta)^m} &\lesssim n. \end{aligned}$$

Then, because $|\mathbb{F}|= 2^{(\log N)^{4\cdot 3^{m-1}/\delta^m}} < 2^{(\log N)^{2\cdot(3/\delta)^m}}$, we have that $\log|\mathbb{F}| < (\log N)^{2\cdot(3/\delta)^m} \lesssim \text{poly}(n)$.

By Theorem 4.3, the runtime of V_{niSC} is $\text{poly}(m, d, |H|, \log|\mathbb{F}|)$, which is equal to $\text{poly}(n)$ since $m = o(\log\log n)$ and $d, |H|, \log|\mathbb{F}| \lesssim \text{poly}(n)$. □

7.2 Sketch of Proof of Theorem 7.1

The proof of Theorem 7.1 in [25] reduces #SAT instances to the *Relaxed-Sink-of-Verifiable-Line problem*, which is known to belong to CLS. We start by defining the rSVL problem.

Definition 7.5. *The* Relaxed-Sink-of-Verifiable-Line (rSVL) *problem $(\mathsf{S}, \mathsf{V}, T, s_0)$ consists of time bound $T \in \mathbb{N}$, starting state $s_0 \in \{0,1\}^L$, and $\text{poly}(n)$-sized functions $\mathsf{S} : \{0,1\}^L \to \{0,1\}^L$ and $\mathsf{V} : [0,T] \times \{0,1\}^L \to \{0,1\}$ with the guarantee that for all $i \in [0,T]$ and $s \in \{0,1\}^L$, if $s = \mathsf{S}^i(s_0)$ then $\mathsf{V}(i,s) = 1$. The goal is to find one of the following:*

- **The sink:** *some $s \in \{0,1\}^L$ such that $\mathsf{V}(T,s) = 1$,*
- **False positive:** *some $i \in [0,T]$ and $s \in \{0,1\}^L$ such that $s \neq \mathsf{S}^i(s_0)$ but $\mathsf{V}(i,s) = 1$.*

Lemma 7.6. *[25]* rSVL *is in* CLS.

Proof sketch of Theorem 7.1. Let $m = o(\log\log n)$. Beginning with a Boolean formula ϕ in $k = \log(n)\cdot m$ variables $x_1, \ldots, x_k$, the task of the #SAT problem is to find the number of satisfying solutions to ϕ, or to evaluate

$$\sum_{x_1,\ldots,x_k \in \{0,1\}} \phi(x_1, \ldots, x_k).$$

Let $H \subset \mathbb{F}$ be a set of size n, and fix a bijection $\alpha : H^m \to \{0,1\}^k$. Then we can define $\phi' = \phi \circ \alpha$, so that $\phi'(h_1, \ldots, h_m) = \phi(x_1, \ldots, x_k)$, where $(x_1, \ldots, x_k) = \alpha(h_1, \ldots, h_m)$. Finally, we arithmetize ϕ' to get a m-variate polynomial g of individual degree $d = \text{poly}(n)$ that can be evaluated in $\text{poly}(n)$ time. Then the number of satisfying solutions to ϕ can be written also as

$$\sum_{h_1,\ldots,h_m \in H} g(h_1, \ldots, h_m).$$

Denote by $N = n^m = |H|^m$ the number of points we're summing over. This computation can be delegated via the sum-check protocol to a $\text{poly}(N)$-size prover who gives a verifiable and unambiguous proof of correctness.

The main technical contribution of [25] is that this proof can be done in an incremental, and efficiently updatable manner. Namely, the summation above can be performed via a $\text{poly}(N)$ sequence of steps, where going from step i to step $i+1$ can be done in $\text{poly}(n)$ time. More specifically, let s_i denote the state after the i'th computation step. Incremental verifiability means that each intermediate state s_i includes a proof

of its correctness, and the proof can be updated and verified in time poly(n). This incrementally verifiable counting procedure is used to construct, given a #SAT instance, an instance of the rSVL problem.

The reduction from #SAT to rSVL, at a high level, is the following: The states correspond to the incrementally updatable states of the computation, where the successor function S takes as input a pair (i, s_i) and generates the next computational state $(i+1, s_{i+1})$. The verification procedure V takes as input a state (i, s_i) and either accepts or rejects.

Suppose that there exists an adversary that solves the rSVL instance corresponding to the computation

$$\sum_{h_1,\ldots,h_m \in H} g(h_1, \ldots, h_m).$$

It either solves the instance by finding a sink, which corresponds to successfully computing v such that $\sum_{h_1,\ldots,h_m \in H} g(h_1, \ldots, h_m) = v$ (i.e., solving #SAT), or by finding false positives, which corresponds to breaking the unambiguous soundness of the underlying non-interactive sum-check proof.

We observe that the latter breaks the *prefix-adaptive* unambiguous soundness. In a nutshell, the polynomial g is fixed, and the adaptivity is in choosing (i, s) such that $\mathsf{V}(i, s) = 1$. Loosely speaking, each state s corresponds to a prefix $(\beta_1, \ldots, \beta_j) \in H_1 \times \cdots \times H_j$, a partial sum v, and a proof π for the fact that

$$\sum_{h_{j+1},\ldots,h_m \in H} g(\beta_1, \ldots, \beta_j, h_{j+1}, \ldots, h_m) = v.$$

Thus, finding a false positive implies choosing $(\beta_1, \ldots, \beta_j, v, \pi)$ as above, such that the proof π accepts, yet $s \neq \mathsf{S}^i(s_0)$. This corresponds to either breaking the prefix-adaptive soundness or breaking the prefix-adaptive unambiguity.

The formal proof follows precisely the proof in [25], and hence is not repeated here. Rather, we refer the reader to [25] for details.

References

[1] Prabhanjan Ananth, Yu-Chi Chen, Kai-Min Chung, Huijia Lin, and Wei-Kai Lin. Delegating RAM computations with adaptive soundness and privacy. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 3–30, 2016.

[2] Saikrishna Badrinarayanan, Yael Tauman Kalai, Dakshita Khurana, Amit Sahai, and Daniel Wichs. Succinct delegation for low-space non-deterministic computation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 709–721, 2018.

[3] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001.

[4] James Bartusek, Liron Bronfman, Justin Holmgren, Fermi Ma, and Ron D. Rothblum. On the (in)security of kilian-based snargs. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 522–551. Springer, 2019.

[5] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[6] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, 2016.

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

[8] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Boneh et al. [13], pages 111–120.

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

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

[11] Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: a paradigm for keyless hash functions. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 671–684. ACM, 2018.

[12] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1480–1498. IEEE Computer Society, 2015.

[13] Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors. *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. ACM, 2013.

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

[15] Zvika Brakerski and Yael Kalai. Witness indistinguishability for any single-round argument with applications to access control. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 97–123. Springer, 2020.

[16] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[17] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1082–1090. ACM, 2019.

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

[19] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-shamir and correlation intractability from strong kdm-secure encryption. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, pages 91–122, 2018.

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

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

[22] David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Compuation*, pages 587–592, 1981.

[23] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player nash equilibria. *J. ACM*, 56(3):14:1–14:57, 2009.

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

[25] Arka Rai Choudhuri, Pavel Hubácek, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. Finding a nash equilibrium is no easier than breaking fiat-shamir. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1103–1114. ACM, 2019.

[26] Arka Rai Choudhuri, Pavel Hubácek, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. Ppad-hardness via iterated squaring modulo a composite. *IACR Cryptol. ePrint Arch.*, 2019:667, 2019.

[27] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *Proceedings of CRYPTOï¿œ91*, pages 445–456, 1992.

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

[29] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009.

[30] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Continuous verifiable delay functions. In Anne Canteaut and Yuval Ishai, editors, *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*, volume 12107 of *Lecture Notes in Computer Science*, pages 125–154. Springer, 2020.

[31] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[32] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 579–604. Springer, 2016.

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

[34] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 75–92, 2013.

[35] Oded Goldreich. On doubly-efficient interactive proof systems. *Foundations and Trends in Theoretical Computer Science*, 13(3):158–246, 2018.

[36] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, pages 102–, 2003.

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

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

[39] Pavel Hubácek and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1352–1371. SIAM, 2017.

[40] Yael Kalai, Omer Paneth, and Lisa Yang. On publicly verifiable delegation from standard assumptions. *IACR Cryptol. ePrint Arch.*, 2018:776, 2018.

[41] Yael Tauman Kalai and Omer Paneth. Delegating RAM computations. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 91–118, 2016.

[42] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1115–1124. ACM, 2019.

[43] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. Ppad-hardness and delegation with unambiguous proofs. CRYPTO, 2020.

[44] Yael Tauman Kalai, Ran Raz, and Oded Regev. On the space complexity of linear programming with preprocessing. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 293–300, 2016.

[45] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In Boneh et al. [13], pages 565–574.

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

[47] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 723–732. ACM, 1992.

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

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

[50] Alex Lombardi and Vinod Vaikuntanathan. Fiat-shamir for repeated squaring with applications to ppad-hardness and vdfs. Cryptology ePrint Archive, Report 2020/772, 2020. `https://eprint.iacr.org/2020/772`.

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

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

[53] Silvio Micali. Cs proofs (extended abstracts). In *FOCS*, pages 436–453, 1994.

[54] Silvio Micali. CS proofs (extended abstracts). In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 436–453, 1994. Full version in [55].

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

[56] Moni Naor. On cryptographic assumptions and challenges. In *Proceedings of the 23rd Annual International Cryptology Conference*, pages 96–109, 2003.

[57] Omer Paneth and Guy N. Rothblum. On zero-testable homomorphic encryption and publicly verifiable non-interactive arguments. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*, pages 283–315, 2017.

[58] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994.

[59] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. 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*, pages 422–439, 2012.

[60] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 89–114. Springer, 2019.

[61] Krzysztof Pietrzak. Simple verifiable delay functions. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 60:1–60:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[62] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 1996.

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

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

[65] Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zksnarks without trusted setup. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 926–943. IEEE Computer Society, 2018.