Finding a Nash Equilibrium Is No Easier Than Breaking Fiat-Shamir

Arka Rai Choudhuri[*] Pavel Hubáček[†] Chethan Kamath[‡]

Krzysztof Pietrzak[§] Alon Rosen[¶] Guy N. Rothblum[‖]

May 22, 2019

Abstract

The Fiat-Shamir heuristic transforms a public-coin interactive proof into a non-interactive argument, by replacing the verifier with a cryptographic hash function that is applied to the protocol's transcript. Constructing hash functions for which this transformation is sound is a central and long-standing open question in cryptography.

We show that solving the End-of-Metered-Line problem is no easier than breaking the soundness of the Fiat-Shamir transformation when applied to the sumcheck protocol. In particular, if the transformed protocol is sound, then *any* hard problem in $\#\mathsf{P}$ gives rise to a hard distribution in the class CLS, which is contained in PPAD.

Our main technical contribution is a stateful *incrementally verifiable* procedure that, given a SAT instance over n variables, counts the number of satisfying assignments. This is accomplished via an exponential sequence of small steps, each computable in time $\mathsf{poly}(n)$. Incremental verifiability means that each intermediate state includes a sumcheck-based proof of its correctness, and the proof can be updated and verified in time $\mathsf{poly}(n)$.

Combining our construction with a hash family proposed by Canetti *et al.* [STOC 2019] gives rise to a distribution in the class CLS, which is provably hard under the assumption that any one of a class of fully homomorphic encryption (FHE) schemes has almost-optimal security against *quasi*-polynomial time adversaries, and under the additional worst-case assumption that there is no polynomial time algorithm for counting the number of satisfying assignments for formulas over a polylogarithmic number of variables.

[*]Johns Hopkins University, Baltimore, USA. Email: `achoud@cs.jhu.edu`. Supported in part by a DARPA/ARL Safeware Grant W911NF-15-C-0213, and a subaward from NSF CNS-1414023.

[†]Charles University, Prague, Czech Republic. Email: `hubacek@iuuk.mff.cuni.cz`. Supported by the project 17-09142S of GA ČR, Charles University project UNCE/SCI/004, and Charles University project PRIMUS/17/SCI/9. This work was done under financial support of the Neuron Fund for the support of science.

[‡]IST Austria, Klosterneuburg, Austria. Email: `ckamath@ist.ac.at`. Supported by the European Research Council, ERC consolidator grant (682815-TOCNeT).

[§]IST Austria, Klosterneuburg, Austria. Email: `pietrzak@ist.ac.at`. Supported by the European Research Council, ERC consolidator grant (682815-TOCNeT).

[¶]Efi Arazi School of Computer Science, IDC Herzliya, Israel. Email: `alon.rosen@idc.ac.il`. supported by ISF grant No. 1399/17 and via Project PROMETHEUS (Grant 780701).

[‖]Weizmann Institute of Science, Rehovot, Israel. Email: `rothblum@alum.mit.edu`. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819702).

Contents

1 Introduction **1**
1.1 Our Results . . . 1
1.2 Technical Overview . . . 4
1.3 Related Work . . . 9

2 Search Problems **11**
2.1 Total Search Problems . . . 11
2.2 The Sink-of-Verifiable-Line Problem . . . 12
2.3 The relaxed-Sink-of-Verifiable-Line Problem . . . 13

3 The Sumcheck Protocol **14**
3.1 Proof Systems . . . 15
3.2 Interactive Sumcheck Protocol . . . 16
3.3 Non-Interactive Sumcheck Protocol . . . 20

4 The Reduction **23**
4.1 The Recursive Construction . . . 24
4.2 The rSVL Instance . . . 29
4.3 Hardness . . . 29

A Instantiating Fiat-Shamir **37**
A.1 Definitions . . . 37
A.2 Fiat-Shamir for the Sumcheck Protocol . . . 40

1 Introduction

The complexity class PPAD, defined by Papadimitriou [47], consists of all total search problems that are polynomial-time reducible to the End-of-Line problem: given a source in a directed graph where every vertex has both in-degree and out-degree at most one, find a sink or another source. The End-of-Line problem can be solved in linear time when the graph is given explicitly, but there is no known algorithm solving it in polynomial time when the input is an implicit representation of the graph describing the successor and predecessor of every vertex.

The class PPAD became a subject of intensive study due to its relation to the problem Nash, of finding a Nash equilibrium in bimatrix games. Papadimitriou showed that Nash is reducible to End-of-Line, and thus belongs to PPAD. A reduction in the opposite direction was later established in a sequence of works by Daskalakis, Goldberg and Papadimitriou [21], and Chen, Deng and Teng [19].

Currently, no PPAD-complete problem is known to admit a sub-exponential-time worst-case algorithm. This, together with the increasingly large number of reductions amongst PPAD complete problems, supports the belief that they are not solvable in polynomial time.[1] Still, even if we do believe that no PPAD complete problem is solvable in polynomial time in the worst-case, it is of great interest to understand whether these problems admit efficient heuristics that perform well on the average, let alone in the worst-case.

A natural approach for arguing average-case PPAD hardness, which was already advocated in Papadimitriou's original paper [47], is to reduce from cryptographic problems. Bitansky, Paneth and Rosen [6] were the first to do so (building on [1]), by demonstrating that the task of breaking sub-exponentially secure *indistinguishability obfuscation* (iO) is reducible to solving End-of-Line. This result was subsequently extended by Hubáček and Yogev [35] to hardness in CLS, a subclass of PPAD, under the same assumptions.[2]

While recent advances in the study of obfuscation support our belief in its attainability, the notion of iO still lies within the domain of speculation: many candidate schemes have been broken, and surviving ones are yet to undergo extensive evaluation by the cryptographic community. Moreover, existing iO schemes tend to be highly complex, hindering any possible attempt to sample reasonably-sized candidate hard instances of Nash.

1.1 Our Results

We widen the basis upon which PPAD hardness can be based. To this end, we rely on the well known *Fiat-Shamir heuristic* [25], which transforms a public-coin interactive proof into a non-interactive argument, by replacing the verifier in the proof with a cryptographic hash function that is applied to the protocol's transcript.

The particular protocol to which we apply the Fiat-Shamir transformation is the *sumcheck protocol* by Lund et al. [43], which is an n-round interactive proof for counting the number of satisfying assignments to a given SAT instance over n variables. We show that solving the End-of-Line problem is no easier than breaking the soundness of the non-interactive argument obtained by applying the Fiat-Shamir transformation to the sumcheck protocol.

Theorem 1 (informal). *Solving the* End-of-Line *problem is no easier than breaking the (adaptive) soundness of the Fiat-Shamir transformation, when applied to the sumcheck protocol.*

We prove this theorem by constructing an efficiently sampleable distibution of End-of-Line instances, where solving this distribution requires either breaking the soundness of the Fiat-Shamir transformation, applied to the sumcheck protocol or solving a #P complete problem

[1]We do know of worst-case hard instances for a *specific algorithm* for finding a Nash equilibrium [52] or of *oracle-based* worst-case hard instances of PPAD-complete problems [33, 3].

[2]The underlying assumptions were further weakened by Garg, Pandey and Srinivasan [27] and then by Komargodski and Segev [41], though still to assumptions that appear to be closely related to iO (see §1.3).

(and thus any problem in #P). Since breaking the soundness of Fiat-Shamir is reducible to #SAT (in fact to SAT) it follows that efficiently solving the above distribution is no easier than breaking Fiat-Shamir. We note that our theorem in fact constructs a distribution on instances of End-of-Metered-Line, a problem that belongs to CLS and hence reduces to the PPAD-complete problems End-of-Line and Nash [35].

On the soundness of Fiat-Shamir. The Fiat-Shamir heuristic is widely used in practice, and constructing hash functions for which the transformation is sound is a central and long-standing open question in cryptography. Empirical evidence in support of the soundness of Fiat-Shamir is the fact that it has been successfully "field tested" for over three decades, though it should be mentioned that the context in which the transformation was originally proposed focused on constant-round protocols, whereas the sumcheck protocol has n rounds.

From a foundational perspective, Goldwasser and Kalai [31] demonstrated theoretical barriers towards the instantiation of Fiat-Shamir in the case of certain computationally sound protocols (a.k.a. "arguments"). Secure instantiations for information theoretically sound protocols (i.e. "proofs"), such as the sumcheck protocol, are an active area of recent research. Several recent works have shown that, under strong cryptographic assumptions, the heuristic is sound when it is applied to certain interactive proofs [38, 16, 14, 17, 48]. For our specific purposes it is sufficient that there exists a specific hash family for which the transformation is sound *for the sumcheck protocol*. Thus, the family can be "tailored" to the protocol. As far as we know, the application of Fiat-Shamir to sumchecks has only been considered recently, most notably in a "sumcheck style" protocol by Pietrzak [49] for proving $y = x^{2^i} \bmod N$ and in very recent work of Canetti *et al.* [14].

The Fiat-Shamir hash function can be instantiated with a random oracle, or with the recent construction of Canetti *et al.* [14].

Random oracle instantiation. We give supporting evidence that the Fiat-Shamir transformation may retain soundness of the sumcheck protocol by proving that this indeed is the case when the hash function in the transformation is modeled as a Random Oracle. What we show is in fact stronger, namely that the transformed protocol satisfies *unambiguous soundness*, which is what our reduction actually requires (see §1.2 for further discussion). One important consequence is the following.

Theorem 2. *Relative to a random oracle, finding a Nash equilibrium is no easier than solving* #SAT *(and in particular no easier than inverting any one-way function).*

FHE-based instantiation. Canetti *et al.* [14] construct a hash family for which, under the assumption that any one of a broad class of fully homomorphic encryption (FHE) schemes has almost optimal security against polynomial-time adversaries, the Fiat-Shamir transformation is sound when it is applied to (certain instantiations of) the sum-check protocol. Adapting their results to our setting gives rise to a hard distribution in the class CLS.

Theorem 3 (Informal Statement, see Theorem 38)**.** *Assuming that any one of the LWE-based fully homomorphic encryption schemes in the literature (such as [10, 9, 8, 28, 11]) has optimal security against quasi-polynomial-size key-recovery attacks, and assuming further that the* #SAT *problem over* polylog *variables is (worst-case) hard for polynomial time algorithms, there exists an efficiently sampleable hard distribution of* End-of-Line *instances.*

Here and below, by optimal security against quasi-polynomial-size attacks, we mean that every quasi-polynomial-size circuit family breaks the assumption with probability at most $2^{\mathsf{polylog}\lambda}/2^{\lambda}$.

To obtain this result, we need a hash function for which the Fiat-Shamir transformation is sound when it is applied to a sum-check protocol for a hard language. Specifically, we consider a sumcheck protocol counting the number of satisfying assignments for a formula with polylogarithmically many variables. By the random self-reducability of $\#\mathsf{P}$ [42, 30], the assumption that the #SAT problem for formulas with $\mathsf{polylog}$ variables is (worst-case) hard for polynomial time algorithms, gives rise to a hard #SAT distribution over such formulas.[3]

The results of Canetti *et al.* [14] do not immediately give a hash function as needed. This is because they consider applying the Fiat-Shamir transformation to *doubly-efficient* interactive proofs, where the honest prover runs in polynomial time.[4] We, on the other hand, need to apply the transformation to a sumcheck over a quasi-polynomial number of assignments, where the honest prover runs in quasi-polynomial time. Adapting their results to our setting, we show that assuming almost-optimal security of the FHE scheme for *quasi-polynomial* adversaries implies that the transformation is sound for the sum-check protocol we consider. See Appendix A for an exposition on this result and a formal statement of Theorem 3.

Sampling hard instances of Nash. By reducing appropriately chosen one-way functions to #SAT, our result opens up the possibility of sampling hard instances of End-of-Line, whose size is significantly smaller than the ones potentially obtained by reducing from iO and related assumptions. First, the random oracle can be instantiated heuristically with a concrete practical hash function (e.g., SHA). The reduction from #SAT is best instantiated with a small SAT instance in which each variable appears in a small constant number of clauses. Hard distributions of such SAT instances arise for example from Goldreich's candidate one-way function [29]. This opens up the possibility, given an efficient reduction from End-of-Line to Nash, of sampling reasonably-sized distributions of games for which solving Nash is heuristically hard and against which existing heuristics (such as the Lemke-Howson algorithm) can be tested.

Unambiguous soundness. An interactive proof system is *unambiguously sound* [50] if the following holds. The proof system specifies a *prescribed* strategy for the honest prover to follow on YES instances. If, given a YES instance, the prover first deviates from its prescribed strategy in some round i, then no matter what strategy it follows in subsequent rounds, the verifier will reject with high probability over its subsequent coin tosses. Note that this is a type of soundness requirement for YES instances. Similarly, we say that a non-interactive argument system is (adaptively) unambiguously sound if there is a prescribed proof for every YES instance, and no PPTM cheating prover can come up with a pair $(x, \widetilde{\pi})$ that is accepted by the verifier unless x is a YES instance and $\widetilde{\pi}$ is its prescribed proof.

The sumcheck protocol is known to be unambiguously sound [50]. For our results, we need to assume that when the Fiat-Shamir transformation is applied to it, the resulting non-interactive argument is adaptively unambiguously sound. We find the assumption that unambiguous soundness is preserved by the Fiat-Shamir transformation to be a natural one. We present supporting evidence for this assumption by demonstrating that it is true in the random oracle model, see Lemma 19. We also show that for a particular instantiation of the Fiat-Shamir transformation (which suffices for PPAD-hardness), adaptive unambiguous soundness reduces to standard adaptive soundness. See Claim 4.

Relationship to Valiant's work on incremental computation. With a similar motivation in mind, Valiant [55] constructed a general-purpose incrementally verifiable computation scheme, where *any* long (exponential) computation can be performed via a sequence of

[3]Specifically, there is a distribution over #SAT instances with $\mathsf{polylog}$ variables and a polynomial-time reduction from solving this distribution with non-negligible probability to solving the problem in the worst case (with high probability). Such a result follows similarly to the worst-case to rare-case reductions in the recent work of Goldreich and Rothblum [30].

[4]In fact, they need a stronger *efficient sampleability* property. A sum-check for a $poly(n)$-sized function over m variables is sampleable in time $\mathsf{poly}(n, 2^m)$.

polynomial-time steps. In between consecutive steps, this process maintains an intermediate state, and a proof of the intermediate state's validity. The state and proof are both of polynomial length. Our work is inspired by this approach. We note, however, that in Valiant's construction the proofs are not unambiguous, and thus it is not clear how to use his scheme to obtain hard instances in PPAD.[5] Putting aside the consequences to PPAD hardness, another distinction between Valiant's work and the direction we take in ours is in the strength of the cryptographic assumptions made. The construction in [55] requires strong non-interactive CS proofs of knowledge, with very efficient (e.g. linear-time) knowledge extractors. Constructing such proof systems is a notoriously hard proposition, see for example the work of Bitansky et al. [5]. Constructions usually rely on knowledge assumptions or are presented in the random oracle model. We, on the other, only assume standard (adaptive) soundness of the concrete and natural cryptographic protocol obtained by applying the Fiat-Shamir transformation to the interactive sumcheck protocol. One consequence of this distinction is that we prove our construction is sound in the random oracle model, whereas no such proof is known for Valiant's construction.[6]

1.2 Technical Overview

Our main technical contribution is a stateful *incrementally verifiable* procedure that, given a SAT instance over n variables, counts the number of satisfying assignments. The counting is performed via an exponential sequence of polynomial-time computation steps, where we maintain an intermediate state between consecutive steps. Incremental verifiability means that each intermediate state includes proof of its correctness, and the proof can be updated and verified in time $\mathsf{poly}(n)$. The proofs are based on a non-interactive sumcheck protocol obtained via the Fiat-Shamir transformation. The main technical challenge is efficient incremental updates to these proofs. We use this incrementally verifiable counting procedure to construct, given a #SAT instance, an instance of the RELAXED-SINK-OF-VERIFIABLE-LINE (rSVL) problem (see below), a promise problem which can be reduced to total search problems in the class CLS (thus also in PPAD). We show that finding a solution to the rSVL instance requires either breaking the unambiguous soundness of the non-interactive sumcheck, or solving the original #SAT instance. We proceed with a high level overview.

Sums and sumcheck proofs. Our incrementally verifiable construction computes sums of low-degree polynomials over exponential numbers of terms. Fix a finite field $\mathbb{F}$ and an n-variate polynomial $f : \mathbb{F}^n \to \mathbb{F}$ over the field $\mathbb{F}$, where f has degree at most d in each variable (think of d as a constant). We are interested in computing sums of the form:

$$\sum_{\boldsymbol{z} \in \{0,1\}^n} f(\boldsymbol{z}).$$

We are also interested in *sumcheck* proofs, proving that y is the correct value of such a sum. More generally, we consider the tasks of computing, proving and verifying sums where a prefix of the variables are fixed to values $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_j) \in \mathbb{F}^j$. We refer to these as *prefix sums*, or the sum with prefix $\boldsymbol{\beta}$. A sumcheck proof can prove statements of the form:

$$\sum_{\boldsymbol{z} \in \{0,1\}^{n-j}} f(\boldsymbol{\beta}, \boldsymbol{z}) = y,$$

which we refer to as a statement *of size* 2^{n-j}.

[5]A key ingredient in Valiant's construction is a CS proof [46] obtained via a Merkel Hash applied to a PCP. This is not unambiguous because small changes to a correct PCP string will change the proof, but will only be noticed by the verifier with small probability.

[6]The issue is that Valiant's proof system cannot be composed to prove statements about a non-explicit oracle.

Given a SAT formula Φ over n variables, a claim about the number of satisfying assignments can be expressed as a sumcheck claim over an appropriately chosen field [43]. The polynomial f is derived from Φ, and the individual degree can be as low as 4. For this, we first transform Φ into a 3SAT-4 formula (a 3CNF where each variable appears in at most 4 clauses), using the (Karp) reduction from counting satisfiable assignments of general CNFs to counting satisfying assignments of 3SAT-4 formulae [54]. A standard arithmetization yields an appropriate polynomial f over the field. In what follows, we use the numbers $\{0, 1, \ldots, d\}$ to refer to the "first" $(d + 1)$ field elements.

Incrementally verifiable counting. Our incrementally verifiable counting procedure is given as input the field $\mathbb{F}$ and an n-variate polynomial f over this field (described as an arithmetic circuit of size $\mathsf{poly}(n)$ and degree d). We also consider giving the procedure, as part of its input, a prefix $\boldsymbol{\beta} \in \mathbb{F}^j$. The goal of the procedure is computing the value y of the sum with prefix $\boldsymbol{\beta}$, and a sumcheck proof for this value.

This computation is performed in a sequence of incremental steps. Towards this, we specify two $\mathsf{poly}(n)$-time algorithms: S and V. The procedure S performs single steps, receiving as input the current state, and computing the next state. The *completeness* requirement is that applying S sequentially for $T = T(n)$ steps, starting at a fixed known initial state s_0, leads to a final state s_T comprised of the correct value y of the sum with prefix $\boldsymbol{\beta}$, as well as a proof π of y's correctness. We use s_t to denote the t-th state along the path from s_0 to s_T. In our construction, each intermediate state s_t includes its index $t \in [T]$. Since we are computing the value of an exponential sum, we expect the number of steps T to be exponential. We use $M = M(n)$ to denote a bound on the size of the state (the memory used by this process), and $P = P(n)$ to denote a bound on the size (and verification time) of the final proof π.

Soundness is guaranteed using the verification procedure V, which takes as input a state and accepts or rejects. The *unambiguous soundness* requirement is that it should be intractable for an adversary who is given the input to compute a state s' with index t s.t. $s' \neq s_t$ but V accepts s'. We note that this is a strong soundness requirement, and we use the strength of this guarantee to reduce to the rSVL problem.

An incrementally verifiable counting procedure as described above directly gives rise to an instance of the rSVL problem, where any solution either specifies the correct count, or describes a state $s' \neq s_t$ that V accepts. We first overview the verifiable counter construction, and close by elaborating on the rSVL problem and on the reduction to it.

A recursive construction. Suppose that $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$ can compute sums of size 2^{n-j} in an incrementally verifiable manner. Suppose further that this computation takes T steps, uses M memory, and has a final proof of size P. We want to recursively construct an incrementally verifiable procedure $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$ for computing sums of size 2^{n-j+1}, which takes $O(T)$ steps, uses $M + O(P) + \mathsf{poly}(n)$ memory, and has a final proof of size $P + \mathsf{poly}(n)$. If we could do so, then unwinding the recursion would give a procedure for computing sums of size 2^n with $2^{O(n)}$ steps, $\mathsf{poly}(n)$ space and $\mathsf{poly}(n)$ proof size (at the base of the recursion, the trivial procedure $(\mathsf{S}_0, \mathsf{V}_0)$ for computing sums of size 1 takes a single step, uses $\mathsf{poly}(n)$ memory and has an "empty" proof). In this overview we ignore the time needed to verify the proof, but we note that it is closely tied to the size P of the final proof. We note that a similar recursive structure underlies Valiant's incrementally verifiable computation procedure [55].

To construct $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$, given a prefix $\boldsymbol{\beta} \in \mathbb{F}^{j-1}$, the naive idea is to use S_{n-j} to sequentially compute two sums of size 2^{n-j}. We refer to the process of running S_{n-j} for T steps to compute a prefix sum as *a full execution* of (S_{n-j}). In the naive approach, a full execution of S_{n-j+1} is comprised of two sequential full executions of S_{n-j}: a first execution for computing the sum for prefix $\boldsymbol{\beta}^0 = (\boldsymbol{\beta}, 0) \in \mathbb{F}^j$, and a second execution computing the sum for prefix $\boldsymbol{\beta}^1 = (\boldsymbol{\beta}, 1) \in \mathbb{F}^j$. The first full execution yields a sum y^0 and a proof π^0.

These are carried through in the second full execution, which yields a sum y^1 and a proof π^1. The final result is $y = (y^0 + y^1)$, and a naive proof for this result is the concatenated proof (y^0, π^0, y^1, π^1). We can construct $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$ to implement and verify the above execution, and it follows that if the base procedure was unambiguously sound, then the new procedure will also be unambiguously sound. The number of steps and the memory grow exactly as we would like them to: in particular, the space complexity of the new procedure is indeed roughly $M + P$, since we only need to "remember" the proof and end result from the first execution while performing the second execution. The main issue is that the proof length and verification time have doubled. If we repeat this recursion many times, they will become super-polynomial.

A natural approach is to try and *merge* the proofs: given (y^0, π^0) and (y^1, π^1), to construct a proof π for the total count $y = (y_0 + y_1)$. Ideally, the merge would be performed in $\mathsf{poly}(n)$ time, and π would be of similar length to π^0 and π^1. This was the approach used in [55], who used strong extractability assumptions to achieve efficient proof merging (we recall that this construction does not have unambiguous proofs, see above). Our approach is different: we use a (long) *incrementally verifiable proof merging procedure*, which is constructed recursively (and is unambiguously sound). Proof merging cannot be performed in $\mathsf{poly}(n)$ time, indeed it requires $O(T)$ steps, but this is fine so long as the merge itself is incrementally verifiable and does not use too much memory or proof size. To obtain an incrementally verifiable proof merging procedure, we show that the proof system we use supports a reduction from proof merging to incrementally verifiable counting. In particular, given the counts $\{y^\gamma\}_{\gamma=0}^d$ for the $(d+1)$ prefix sums with prefixes $\{\boldsymbol{\beta}^\gamma = (\boldsymbol{\beta}, \gamma)\}_{\gamma=0}^d$ (sums of size 2^{n-j}), computing a proof π for the count $y = (y^0 + y^1)$ of the sum with prefix β (a sum of size 2^{n-j+1}) reduces to computing a single additional prefix sum of size 2^{n-j}. This merge procedure relies heavily on the structure of the non-interactive sumcheck proof system, we give an overview below.

Given the merge procedure, we can detail the recursive construction of $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$. Given a prefix $\boldsymbol{\beta} \in \mathbb{F}^{j-1}$, a full execution of S_{n-j+1} runs $(d+1)$ full executions of S_{n-j}, computing the prefix sums (and proofs) for sums with prefixes $\{\boldsymbol{\beta}^\gamma = (\boldsymbol{\beta}, \gamma)\}_{\gamma=0}^d$ (these are sums of size 2^{n-j}). Let $\{(y^\gamma, \pi^\gamma)\}_{\gamma=0}^d$ be the (respective) prefix sums and proofs. We then run a final full execution of S_{n-j} to compute a "merged" proof π for the sum with prefix $\boldsymbol{\beta}$ (a sum of size 2^{n-j+1}). Once the merged proof is completed we can "forget" the intermediate values $\{(y^\gamma, \pi^\gamma)\}_{\gamma=0}^d$. We end the entire process with a proof that is not much larger than the proofs for sums of size 2^{n-j}. Computing the merged proof boils down to computing an additional sum of size 2^{n-j} with a prefix $\boldsymbol{\beta}^\sigma = (\boldsymbol{\beta}, \sigma)$ for a single field element $\sigma \in \mathbb{F}$. Thus, the number of steps for a full execution of S_{n-j+1} is $O(d \cdot T)$ (recall that d is constant), and the memory used is $M + O(d \cdot P)$. Unwinding the recursion, we obtain an incrementally verifiable counting procedure which takes $2^{O(n)}$ steps, with $\mathsf{poly}(n)$ memory and proof size.

We proceed to detail the proof system we use, and then describe the reduction from proof-merging to incrementally verifiable counting.

The non-interactive sumcheck. In the interactive Sumcheck protocol, an untrusted (and not necessarily efficient) prover wants to convince a verifier that:

$$\sum_{\boldsymbol{z} \in \{0,1\}^n} f(\boldsymbol{z}) = y.$$

The protocol proceeds in n rounds. In the first round, the prover sends a univariate polynomial $\widetilde{g}_1$ obtained by leaving the first variable in f free, and summing over all 2^{n-1} assignments to the remaining variables. Note this univariate polynomial is of degree d, and thus it can be specified by sending its valuations over the first $(d+1)$ field elements, and the prover sends these valuations $\alpha_1 = \{\alpha_{1,\gamma} = \widetilde{g}_1(\gamma)\}_{\gamma=0}^d$ as its message. On receiving these valuations, the verifier interpolates to recover $\widetilde{g}_1$, and checks that this polynomial is consistent with the prover's past claims, i.e., that $\widetilde{g}_1(0)+\widetilde{g}_1(1)=y$ (otherwise the verifier rejects immediately). The verifier then picks a random field element β_1 and sends it to the prover.

More generally, the first i rounds fix polynomials $\widetilde{g}_1,\ldots,\widetilde{g}_i$ and a prefix of field elements $\beta_1,\ldots,\beta_j$. In the $(i+1)$-th round the parties run the same two-message protocol described above to verify the i-th claim:

$$\sum_{\boldsymbol{z}\in\{0,1\}^{n-i}} f(\beta_1,\ldots,\beta_j,z_{i+1},\ldots,z_n)=\widetilde{g}_i(\beta_i).$$

Note that this i-th claim is about the sum with the prefix $\boldsymbol{\beta}=(\beta_1,\ldots,\beta_j)$, which is of size 2^{n-i}. After n rounds, the verifier can simply verify the n-th claim on its own using a single evaluation of f on the point $(\beta_1,\ldots,\beta_n)$. Soundness of the protocol follows from the fact that if the i-th claim is false, then for any $\widetilde{g}_{i+1}$ sent by a cheating prover, w.h.p. over the choice of β_{i+1} the $(i+1)$-th claim will also be false (because of the Schwartz-Zippel Lemma). Unambiguity means that even if the i-th claim is true, then if a cheating prover sends any polynomial $\widetilde{g}_{i+1}$ that is not equal to the "prescribed" polynomial $g_{i+1}(x)$ that would have been sent by the honest prover, then w.h.p. over the choice of β_{i+1} the $(i+1)$-th claim will be false (even though the i-th claim was true!). More generally, we can use the same protocol to verify sums with a fixed prefix $\boldsymbol{\beta}\in\mathbb{F}^j$ for any $j\in\{0,\ldots,n\}$. This requires only $(n-j)$ rounds of interaction.

To make this protocol non-interactive, we use the Fiat-Shamir tranformation. Given a hash function h, the prescribed proof for an instance $(\mathbb{F},y,f)$ specifies the prescribed prover's messages $(\alpha_1,\ldots,\alpha_n)$ in a particular execution of the sumcheck protocol. The particular execution is specified by computing for each i the verfier's challenge $\beta_i=h(\mathbb{F},y,f,\alpha_1,\beta_1,\ldots,\beta_{i-1},\alpha_i)$. To verify such a proof, the verifier: (i) computes each β_i (using the hash function, as above), and (ii) checks that the sumcheck verifier would accept the input $(\mathbb{F},y,f)$ given the transcript $(\alpha_1,\beta_1,\ldots,\alpha_n,\beta_n)$. We assume that this non-interactive protocol is adaptively unambiguously sound: given the hash function h, no polynomial-time prover can find a false statement $(\mathbb{F},\widetilde{y},f)$ and an accepting proof for that statement. Nor can a cheating prover find a true statement $(\mathbb{F},y,f)$ and an accepting proof that differs from the prescribed one. Similarly to the interactive sumcheck, we can also use the non-interactive sumcheck to verify sums with a fixed prefix $\boldsymbol{\beta}\in\mathbb{F}^j$. In fact, if we define the language appropriately, adaptive soundness of this protocol directly implies adaptive *unambiguous* soundness. We elaborate on this below, see Claim 4.

Merging proofs by computing a (small) sum. Recall our setting: for a fixed prefix $\boldsymbol{\beta}\in\mathbb{F}^{j-1}$, we have computed the sums with prefixes $\{\boldsymbol{\beta}^\gamma=(\boldsymbol{\beta},\gamma)\in\mathbb{F}^j\}_{\gamma=0}^d$, which have values $\{y^\gamma\}_{\gamma=0}^d$, together with corresponding proofs $\{\pi^\gamma\}_{\gamma=0}^d$ for those values. We want now to compute the proof π for the sum with prefix $\boldsymbol{\beta}$. This proof corresponds to a larger sum, and so it should contain an additional (collapsed) round of interaction. What should the prover's first message in the protocol for this statement be? The first message is comprised of the values of the polynomial g_j, where $g_j(\gamma)$ equals the sum with prefix $\boldsymbol{\beta}^\gamma$. Thus, the first message is simply the values $\alpha_1=\{y^\gamma\}_{\gamma=0}^d$. Once we know the prover's first message α_1, we can compute the verifier's random challenge σ by applying the Fiat-Shamir hash function to the instance and to α_1. To complete a transcript for the non-interactive sumcheck protocol (and a proof for the sum with prefix $\boldsymbol{\beta}$), we now need a proof for the next claim, i.e., a proof that:

$$\sum_{\boldsymbol{z}\in\{0,1\}^{n-j}} f(\boldsymbol{\beta},\sigma,\boldsymbol{z})=g_1(\sigma).$$

In particular, all we need is to compute a sum of size 2^{n-j} with prefix $(\boldsymbol{\beta},\sigma)$, and a proof for this sum. Once the value and proof for this larger sum are computed, we can "forget" the values and proofs $\{y^\gamma,\pi^\gamma\}_{\gamma=0}^d$ that were computed for the prefixes. This completes the reduction from incrementally verifiable proof merging to incrementally verifiable counting.

More generally, we have shown that the sum and proof for a statement of size 2^{n-j+1} can be obtained by computing $(d+1)$ proofs for statements of size 2^{n-j} (with a common prefix of length 2^{j-1}, followed by each one of the first $(d+1)$ field elements), and an additional proof for a final statement of size 2^{n-j} (with the same common prefix, but a different j-th element that depends on the Fiat-Shamir hash function). Note that while sums with boolean prefixes correspond to the counting the number of satisfying assignments in subcubes of the hypercube $\{0,1\}^n$, the sums with prefixes that include elements outside $\{0,1\}$ have no such correspondence and in particular the summands can be arbitrary field elements.

From soundness to unambiguous soundness. We show that adaptive soundness of the non-interactive sum-check protocol applied to a particular language $\mathcal{L}_{\text{SC}}$ implies adaptive unambiguous soundness for the same language. Moreover, the protocol's adaptive unambiguous soundness for $\mathcal{L}_{\text{SC}}$ suffices for (unambiguous) soundness of our incrementally verifiable computation scheme.

We begin by defining $\mathcal{L}_{\text{SC}}$. The language is defined over tuples that include an instance to the sumcheck protocol, a fixed prefix $\beta_1, \dots, \beta_j \in \mathbb{F}$, and a fixed partial transcript of the sumcheck protocol. A tuple is of the form $(\mathbb{F}, y, f, \beta_1, \dots, \beta_j, \widetilde{\alpha}_{j+1}, \beta_{j+1}, \dots, \widetilde{\alpha}_{i+j})$, where $i, j \in \{0, \dots, n\}$ and their sum is at most n. The language is defined as follows:

- When $i = j = 0$, this is simply a standard input for the sumcheck protocol, and the tuple is in the language if and only if indeed the sum of f over all 2^n inputs equals y.
- For $j \geq 1$ and $i = 0$, the tuple is in the language if and only if the sum over all 2^{n-i} assignments with prefix $\beta_1, \dots, \beta_j$ equals y.
- For general j and $i \geq 1$, the tuple is in the language if and only if the final prover message $\widetilde{\alpha}_{i+j}$ is consistent with the prescribed (honest) prover's message, given the fixed prefix $\beta_1, \dots, \beta_j$ and the verifier's messages $\beta_{i+1}, \dots, \beta_{i+j-1}$ (there is no condition on y or on the prior prover messages, only the last one matters).[7]

With this language in mind, we can view each round of the sumcheck as reducing from a claim that a tuple is in $\mathcal{L}_{\text{SC}}$, to a claim that a longer tuple is in $\mathcal{L}_{\text{SC}}$. Soundness means that if the initial claim is false, then w.h.p. the new claim is also false. Unambiguity means that even if the initial instance was in the language, if the prover doesn't follow the prescribed strategy, then w.h.p. over the verifier's choice of β_i, the new instance is not in the language.

For our incrementally verifiable computation scheme, it suffices to assume that the non-interactive sumcheck is an adaptively unambiguously sound non-interactive argument system for the language $\mathcal{L}_{\text{SC}}$ (see the full construction in Section 3). The following Claim shows that in fact it suffices to assume adaptive soundness, which itself implies unambiguity.

Claim 4. *If the sumcheck protocol is an adaptively sound argument system for the language $\mathcal{L}_{\text{SC}}$, then it is also an adaptively unambiguously sound argument system for $\mathcal{L}_{\text{SC}}$.*

Proof. Assume for contradiction that there exists an adversary $\mathcal{A}$ that, given a hash function h, can find with noticeable probability an instance $x \in \mathcal{L}_{\text{SC}}$, whose prescribed proof is π, and an accepting proof $\widetilde{\pi} \neq \pi$. Let

$$x = (\mathbb{F}, y, f, \beta_1, \dots, \beta_j, \alpha_{j+1}, \beta_{j+1}, \dots, \alpha_{i+j}),$$

and $\widetilde{\pi} = (\widetilde{\alpha}_{i+j+1}, \dots, \widetilde{\alpha}_n)$. We can use $\mathcal{A}$ to break the adaptive soundness of the same argument system, by picking a random index $\ell \in \{i+j+1, n\}$, and computing the challenges:

$$\{\widehat{\beta}_k = h(x, \widehat{\beta}_{i+j}, \widetilde{\alpha}_{i+j+1}, \widehat{\beta}_{i+j+1}, \dots, \widetilde{\alpha}_k)\}_{k=i+j}^{\ell-1}.$$

[7]Here, when we refer to the prescribed prover, we are ignoring the fact that the actual claim being proved (i.e. the value of y) might be false, as the prescribed prover in the sum check protocol does not need to use the value y to compute its messages.

The new instance is:

$$x' = (x, \widehat{\beta}_{i+j}, \widetilde{\alpha}_{i+j+1}, \widehat{\beta}_{i+j+1}, \ldots, \widetilde{\alpha}_\ell).$$

The proof for this new instance is $\pi' = (\widetilde{\alpha}_{\ell+1}, \ldots, \widetilde{\alpha}_n)$. By construction, if $\widetilde{\pi}$ is an accepting proof for x, then also π' will be an accepting proof for x'. We claim that with probability at least $1/n$, however, x' is a NO instance of $\mathcal{L}_{\mathrm{SC}}$. To see this, let the prescribed proof be $\pi = (\alpha_{i+j+1}, \ldots, \alpha_n)$. Finally, let ℓ^* be the *smallest* index s.t. $\widetilde{\alpha}_{i+\ell^*} \neq \alpha_{i+\ell^*}$ (such a ℓ^* must exist because $\widetilde{\pi} \neq \pi$). It follows that the instance $x' = (\mathbb{F}, y, f, \alpha_1, \beta_1, \ldots, \alpha_i, \widehat{\beta}_i, \widetilde{\alpha}_{i+1}, \ldots, \widetilde{\alpha}_{i+\ell^*})$ is a NO instance for $\mathcal{L}_{\mathrm{SC}}$. Thus, in the above reduction, when $\ell = \ell^*$ the adversary finds an accepting proof for a NO instance, and this happens with probability at least $1/n$ over the choice of ℓ. □

From incrementally verifiable counting to a problem in CLS. Our ultimate goal is to construct an average-case hard distribution in CLS (and thus in PPAD). To this end, we reduce our incrementally verifiable counting procedure applied to any #SAT instance to a problem in CLS. The blueprint for such a reduction comes from the work of Bitansky, Paneth and Rosen [6], who (building on [1]) introduced the promise search problem Sink-of-Verifiable-Line (SVL). This problem corresponds to an incrementally verifiable sequential computation where S gives the next state and V allows to verify that s_i lies i steps from the initial state s_0. Given a $T \in \mathbb{N}$, the goal is to find a state $s_T = \mathsf{S}^T(s_0)$.

Hubáček and Yogev [35] showed that any S and V with perfect soundness (i.e., such that V accepts a pair s_i and i if and only if s_i lies i consecutive steps of S from s_0) can be reduced to End-of-Metered-Line (EOML), a total search problem which lies in CLS. Our crucial observation is that *unambiguous* soundness is sufficient for the correctness of their reduction. Specifically, we show that an adaptively unambiguously sound non-interactive argument for the sumcheck language gives rise to a hard-on-average distribution of EOML instances.

We define relaxed-Sink-of-Verifiable-Line (rSVL), a generalization of SVL that captures breaking unambiguous soundness and show that the reduction from [35] is robust enough so that when applied to any rSVL instance, any solution for the resulting EOML instance either corresponds to finding the target state s_T or breaking the unambiguous soundness of V. We give the formal definition of relaxed-Sink-of-Verifiable-Line and the reduction to End-of-Metered-Line in §2.3.

1.3 Related Work

The systematic study of total search problems (i.e., with the guaranteed existence of a solution) was initiated by Megiddo and Papadimitriou [45], who defined a corresponding complexity class, called TFNP. They observed that unless NP = co-NP, a "semantic" class such as TFNP is unlikely to have complete problems. Motivated by this observation, Papadimitriou [47] defined "syntactic" subclasses of TFNP with the goal of clustering search problems based on the various (non-constructive) existential theorems used to argue their totality (see Figure 1). Perhaps the best known such class is PPAD [47] which captures the computational complexity of finding Nash equilibria (Nash) in bimatrix games [21, 19], amongst other natural problems [39].

Other subclasses of TFNP include PPA [47], which captures computational problems related to the Borsuk-Ulam theorem (Borsuk-Ulam), Tucker's lemma (Tucker) [24] or fair division [26], the class PLS [37] that was defined to capture the computational complexity of problems amenable to local search such as Local-Maxcut, and the class CLS [22], which captures finding approximate local optima of continuous functions (CLO) or finding Banach's fixed points [23] and contains finding Nash equilibria in congestion games or solving the simple stochastic games of Condon or Shapley. Finally, the classes PPP [47] and PWPP [36] are motivated by the pigeonhole principle and contain important problems related to finding collisions

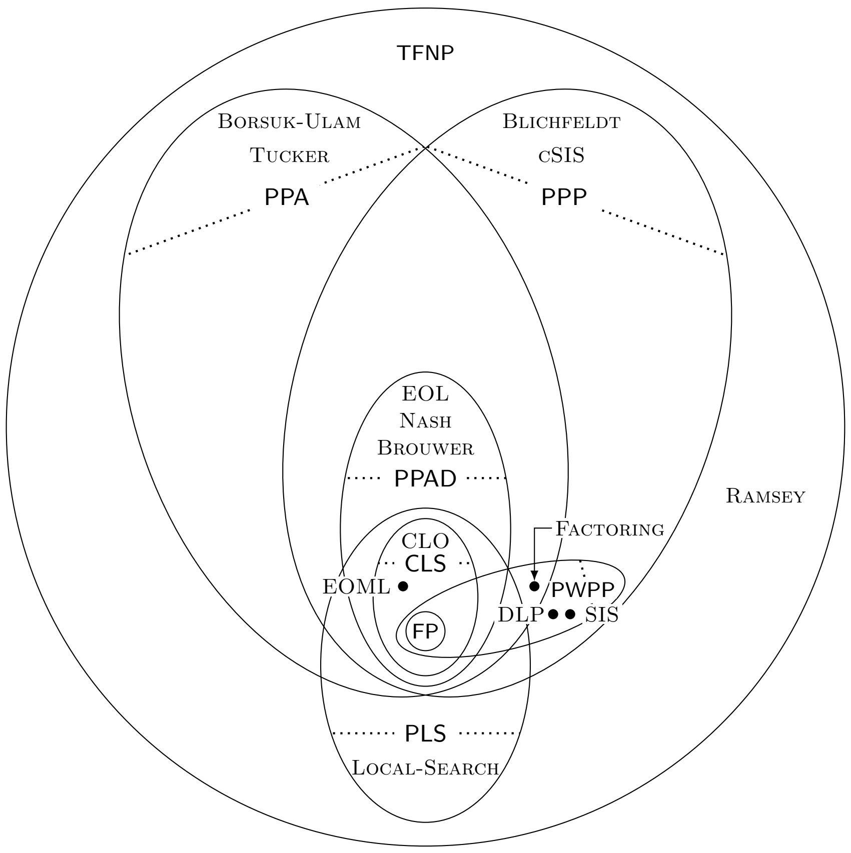

Figure 1: The TFNP landscape.

in functions. Recently, Sotiraki, Zampetakis and Zirdelis [53] introduced a PPP-complete problem related to Blichfeldt's theorem in the theory of lattices (Blichfeldt) and showed that a constrained variant of the short integer solution problem (cSIS) is PPP-complete.

On the face of it, all TFNP problems could potentially be solvable in polynomial time without defying our understanding of the broader landscape of complexity theory (e.g. no surprising collapse of any important complexity classes seems to be implied by assuming TFNP $\subset$ FP). In light of this, it is natural to seek "extrinsic" evidence supporting TFNP hardness, for instance based on computational problems originating in cryptography. This approach would also have the added benefit of establishing average-case hardness, indicating resistance against general heuristic algorithms.

The approach of basing average-case TFNP-hardness on relatively established cryptographic assumptions has already been successfully applied in the context of complexity classes other than PPAD. For instance, Papadimitriou [47] showed that one-way permutations imply average-case hardness in PPP,[8] and Jeřábek [36] showed that the undirected version of End-of-Line, which is complete for the class PPA, is no easier than FACTORING. It is currently not known whether any of these results can be extended to PPAD.

As mentioned in §1, Bitansky, Paneth and Rosen, building on [1], showed that the task of breaking sub-exponentially secure *indistinguishability obfuscation* (*iO*) is reducible to solving the End-of-Line problem [6]. The Bitansky et al. technique was extended by Hubáček and Yogev [35], who established hardness in CLS, a subclass of PPAD, under the same assumptions. Both results were subsequently strengthened. First, by Garg, Pandey and Srinivasan [27], who showed that it is sufficient to assume the existence of *iO* with polynomial (instead of sub-exponential) hardness (or alternatively compact public-key functional encryption) and one-

[8]It is also known (folklore) that any assumption that implies the existence of collision-resistant hashing (e.g. hardness of Factoring, SIS or DLP) implies average-case hardness in PWPP.

way permutations. Second, by Komargodski and Segev [41], who weakened the assumptions to quasi-polynomially secure private-key functional encryption and sub-exponentially-secure injective one-way functions.

Hubáček, Naor and Yogev [34] recently constructed hard TFNP problems from one-way functions (in fact from any average-case hard NP language) under complexity theoretic assumptions used in the context of derandomization. It is not known whether their distribution gives rise to average-case hardness in any of the syntactic subclasses of TFNP. Komargodski, Naor and Yogev [40] demonstrated a close connection between the Ramsey problem (Ramsey) and the existence of collision-resistant hashing.

The relatively small progress on showing average-case hardness of total search problems from weak general assumptions motivated a line of works focusing on limits for proving average-case hardness. The implausibility of using worst-case NP hardness [37, 45] was later strengthened to show that it is unlikely to base average-case TFNP hardness even on problems in the polynomial hierarchy [12], and to show that any randomized reduction from a worst-case NP language to an average-case TFNP problem would imply that SAT is checkable [44]. A recent result [51] applies to the whole of TFNP and shows that any attempt to base average-case TFNP hardness on (trapdoor) one-way functions in a black-box manner must result in instances with exponentially many solutions. Previously to our work, all known constructions of average-case hard PPAD problems resulted in instances with small numbers of solutions. In contrast, our construction yields instances with exponentially many solutions (where all but one of the solutions correspond to breaking unambiguous soundness).

Orthogonally to the above works, the smoothed complexity approach was recently used to identify natural distributions of PLS-complete problems (such as the Local-Maxcut or the problem of finding pure Nash equilibria in network coordination games) that admit polynomial time algorithms [2, 7].

Prior to our work, arguably the most natural average-case hard distribution of structured TFNP problems followed from the randomized reduction from Factoring to PPA developed in the works of Buresh-Oppenheim and Jeřábek [13, 36].

2 Search Problems

In this section, we recall definitions for total search problems and promise search problems from previous work. In §2.3, we define our relaxed version of Sink-of-Verifiable-Line.

2.1 Total Search Problems

An efficiently-verifiable search problem is described via a pair $(\mathcal{L}, \mathcal{R})$, where $\mathcal{L} \subseteq \{0,1\}^*$ is an efficiently-recognizable set of instances, and $\mathcal{R}$ is an efficiently-computable binary relation. Given $v \in \mathcal{L}$, the task is to find a w such that $\mathcal{R}(v, w)$ — the class that contains all such search problems is known as *functional* NP (FNP). An efficiently-verifiable search problem is *total* if for every instance $v \in \mathcal{L}$ there exists a witness w of length $\mathsf{poly}(|v|)$ such that $\mathcal{R}(v, w) = 1$. The class *total* FNP (TFNP) consists of all efficiently-verifiable search problems that are total.

The class *polynomial parity argument over directed graphs* (PPAD) is a *syntactical* sub-class of TFNP which consists of all problems that are polynomial-time reducible to the End-of-Line problem (also known as the Source-or-Sink problem) [47].

Definition 5. An End-of-Line (EOL) instance (S, P) consists of a pair of circuits $\mathsf{S}, \mathsf{P} : \{0,1\}^M \to \{0,1\}^M$ such that $\mathsf{P}(0^M) = 0^M$ and $\mathsf{S}(0^M) \neq 0^M$. The goal is to find a vertex $v \in \{0,1\}^M$ such that $\mathsf{P}(\mathsf{S}(v)) \neq v$ or $\mathsf{S}(\mathsf{P}(v)) \neq v \neq 0^M$.

Intuitively, the circuits S and P can be viewed as implementing the successor and predecessor functions of a directed graph over $\{0,1\}^M$, where for each pair of vertices v and u there exists an edge from v to u if and only if $\mathsf{S}(v) = u$ and $\mathsf{P}(u) = v$ (note that the in-degree and out-degree of every vertex in this graph is at most one, and the in-degree of 0^M is 0). The goal is to find any vertex, other than 0^M, with either no incoming edge or no outgoing edge. Such a vertex must always exist by a parity argument.

The class *continuous local search* (CLS) lies in the intersection of PPAD and PLS and consists of all problems that are polynomial-time reducible to the Continuous-Local-Optimum problem (cf. [22] for the formal definition). Another problem that is known to lie in CLS (but not known to be complete) is End-of-Metered-Line [35].

Definition 6. An End-of-Metered-Line (EOML) instance $(\mathsf{S}, \mathsf{P}, \mathsf{M})$ consists of circuits $\mathsf{S}, \mathsf{P} \colon \{0,1\}^m \to \{0,1\}^m$ and $\mathsf{M} \colon \{0,1\}^m \to \{0, \ldots, 2^m\}$ such that $\mathsf{P}(0^m) = 0^m \neq \mathsf{S}(0^m)$ and $\mathsf{M}(0^m) = 1$. The goal is to find a vertex $v \in \{0,1\}^m$ satisfying one of the following:

(i) **End of line:** either $\mathsf{P}(\mathsf{S}(v)) \neq v$ or $\mathsf{S}(\mathsf{P}(v)) \neq v \neq 0^m$,

(ii) **False source:** $v \neq 0^m$ and $\mathsf{M}(v) = 1$,

(iii) **Miscount:** either $\mathsf{M}(v) > 0$ and $\mathsf{M}(\mathsf{S}(v)) - \mathsf{M}(v) \neq 1$ or $\mathsf{M}(v) > 1$ and $\mathsf{M}(v) - \mathsf{M}(\mathsf{P}(v)) \neq 1$.

The goal in EOML is the same as in EOL, but now the task is made easier as one is also given an "odometer" circuit M. On input a vertex v, this circuit M outputs the number of steps required to reach v from the source. Since the behaviour of M is not guaranteed syntactically, any vertex that attests to deviation in the correct behaviour of M also acts as a solution (and thus puts End-of-Metered-Line in TFNP).

2.2 The Sink-of-Verifiable-Line Problem

The Sink-of-Verifiable-Line problem is a *promise* search problem introduced by Abbot, Kane and Valiant [1] and further developed in [6]. It is defined as follows:

Definition 7. A Sink-of-Verifiable-Line instance $(\mathsf{S}, \mathsf{V}, T, v_0)$ consists of $T \in \{1, \ldots, 2^M\}$, $v_0 \in \{0,1\}^M$, and two circuits $\mathsf{S} : \{0,1\}^M \to \{0,1\}^M$ and $\mathsf{V} : \{0,1\}^M \times \{1, \ldots, T\} \to \{0,1\}$ with the guarantee that for every $v \in \{0,1\}^M$ and $i \in \{1, \ldots, T\}$, it holds that $\mathsf{V}(v, i) = 1$ if and only if $v = \mathsf{S}^i(v_0)$. The goal is to find a vertex $v \in \{0,1\}^M$ such that $\mathsf{V}(v, T) = 1$ (i.e., the sink).

Intuitively, the circuit S can be viewed as implementing the successor function of a directed graph over $\{0,1\}^M$ that consists of a single line starting at v_0. The circuit V enables to efficiently test whether a given vertex v is of distance i from v_0 on the line, and the goal is to find the vertex at distance T from v_0. Note that not every tuple $(\mathsf{S}, \mathsf{V}, T, v_0)$ is a *valid* SVL instance since V might not satisfy the promise about its behaviour. Moreover, there may not be an efficient algorithm for verifying whether a given tuple $(\mathsf{S}, \mathsf{V}, T, v_0)$ is a valid instance, hence this problem lies outside of TFNP.

Remark 8. The definition of SVL with an arbitrary source vertex v_0, as above, is equivalent to the definition in [6] where the source is 0^M. First, any SVL instance $(\mathsf{S}, \mathsf{V}, T)$ where the source is 0^M can be trivially transformed to an instance $(\mathsf{S}, \mathsf{V}, T, v_0 = 0^M)$. Second, we can reduce in the opposite direction by shifting the main line by v_0 as follows. Given an SVL instance $(\mathsf{S}, \mathsf{V}, T, v_0)$, define the new SVL instance as $(\mathsf{S}', \mathsf{V}', T)$ with source 0^M, where $\mathsf{S}'(v) := \mathsf{S}(v \oplus v_0)$ and $\mathsf{V}'(v, i) := \mathsf{V}(v \oplus v_0, i)$, where $\oplus$ denotes the bitwise XOR operation. Note that this general technique can be applied in the context of TFNP to any search problem where part of the instance is some significant vertex (e.g. the trivial source at 0^M in End-of-Line).

2.3 The relaxed-Sink-of-Verifiable-Line Problem

In this section, we introduce a variant of the Sink-of-Verifiable-Line problem, which we call relaxed-Sink-of-Verifiable-Line problem. The main difference from Definition 7 is that the promise about the behaviour of the verifier circuit V is relaxed so that V can also accept vertices off the line starting at the vertex v_0. However, any vertex off the main line accepted by V is an additional solution.

Definition 9. A relaxed-Sink-of-Verifiable-Line (rSVL) instance $(\mathsf{S}, \mathsf{V}, T, v_0)$ consists of $T \in \{1, \ldots, 2^M\}$, $v_0 \in \{0,1\}^M$, and circuits $\mathsf{S} : \{0,1\}^M \to \{0,1\}^M$ and $\mathsf{V} : \{0,1\}^M \times \{1, \ldots, T\} \to \{0,1\}$ with the guarantee that for every $(v,i) \in \{0,1\}^M \times \{1, \ldots, T\}$ such that $v = \mathsf{S}^i(v_0)$, it holds that $\mathsf{V}(v,i) = 1$. The goal is to find one of the following:

(i) **The sink:** a vertex $v \in \{0,1\}^M$ such that $\mathsf{V}(v,T) = 1$; or

(ii) **False positive:** a pair $(v,i) \in \{0,1\}^M \times \{0, \ldots 2^M\}$ such that $v \neq \mathsf{S}^i(v_0)$ and $\mathsf{V}(v,i) = 1$.

We show in the following lemma that, despite the relaxed promise, rSVL reduces to EOML and, thus, average-case hardness of rSVL is sufficient to imply average-case hardness of EOML.

Lemma 10. relaxed-Sink-of-Verifiable-Line *is reducible to* End-of-Metered-Line.

Proof. The proof of the lemma follows by inspection of the reduction from Sink-of-Verifiable-Line to End-of-Metered-Line from [35] when applied to a relaxed-Sink-of-Verifiable-Line instance. We start by giving an overview of the reduction and then argue that its correctness is preserved even when applied to an rSVL instance.

Consider an SVL instance $(\mathsf{S}, \mathsf{V}, T, 0^m)$.[9] In order to reduce it to an End-of-Metered-Line instance, the main technical issue is implementing the predecessor circuit P' and meter circuit M'. Notice that it is easy to construct the predecessor circuit in an *inefficient* way. One can simply modify the labels of the vertices to contain the entire history of the previous steps on the SVL line. Given such labels, implementing the predecessor is easy: simply remove the last element from the label. However, the obvious issue of this transformation is that the size of the labels becomes eventually exponentially large which would render the resulting circuits $\mathsf{S}', \mathsf{P}', \mathsf{M}'$ inefficient relative to the size of the SVL instance. In order to give an efficient reduction, it was observed in [1, 6] that it is possible to reduce the size of the labels by utilizing techniques used for implementing *reversible computation* [4], where only a small number of states is stored in order to be able to revert previous steps in the computation. The general approach can be explained via a simple *pebbling game* that we describe next.

There are ℓ pebbles that can be placed on positions indexed by positive integers. The rules of the game are as follows: a pebble can be placed in or removed from position i if and only if either there is a pebble in position $i-1$ or $i = 1$. The goal of the game is to place a pebble in position $2^\ell - 1$. As shown by Chung, Diaconis and Graham [20], the optimal efficient strategy achieves the goal of the game in a recursive manner. Their main idea is to exploit the symmetry of the rules for placing and removing pebbles. Specifically, that it is always possible to reverse any sequence of moves. Suppose there is a way to get to $2^{\ell-1} - 1$ using only $\ell - 1$ pebbles. Then, place an additional pebble at $2^{\ell-1}$. Next, free the first $\ell - 1$ pebbles by reversing the original sequence of moves performed in the first part. Finally, perform the same sequence starting from $2^{\ell-1}$. This strategy will end with a pebble at position $2^\ell - 1$ while using only ℓ pebbles.

The predecessor circuit in the reduction from SVL to EOML is implemented by simulating the optimal strategy in the above pebbling game. Each pair (v,i) such that $\mathsf{V}(v,i) = 1$ corresponds to a position on which a pebble can be placed. Subsequently, a vertex in the EOML instance has a label representing the states of the $\ell = \log_2(T)$ pebbles, i.e., a tuple of pairs $((v_1, i_1), \ldots, (v_\ell, i_\ell))$ where each pebble corresponds to a pair (v,i) such that $\mathsf{V}(v,i) = 1$

[9]We can assume w.l.o.g. that the starting vertex for the SVL instance is $v_0 = 0^m$ (see Remark 8).

– tuples containing pairs that do not verify become self-loops. The efficient pebbling strategy demonstrates that by storing only ℓ intermediate states we can implement S' and P' that traverse the exponential SVL line from 0^m to $v_T = \mathsf{S}^T(0^m)$. For every valid configuration of the pebbles (i.e., a vertex that is not a self-loop), the meter M' simply outputs the number of steps taken in the pebbling strategy so far plus one (which can be computed efficiently just from the configuration itself) and the self-loops are given value 0. The resulting End-of-Metered-Line instance $(\mathsf{S}', \mathsf{P}', \mathsf{M}')$ corresponds to a graph with a single line traversing the sequence of all the configurations visited by the optimal pebbling strategy. In particular, every vertex corresponding to an intermediate state of the pebbling strategy is followed by the subsequent state, and the final step of the pebbling strategy is a self-loop under S'. Any state describing an illegal configuration of the pebbling game is defined to be a self loop both under S' and P'. Therefore, the resulting instance has a unique solution, a sink that identifies a solution to the original SVL instance. For a formal description of the reduction see [35].

We now analyze the reduction when applied to a relaxed-Sink-of-Verifiable-Line instance $(\mathsf{S}, \mathsf{V}, T, 0^m)$. The main issue is that due to the relaxed guarantee, we might have introduced additional solutions besides the sink corresponding to $v_T = S^T(0^m)$. We claim that the resulting EOML instance $(\mathsf{S}', \mathsf{P}', \mathsf{M}')$ has the following main properties.

1. Every vertex is labeled by a tuple of $\ell = \log_2(T)$ pairs of the form $(v, i) \in \{0,1\}^M \times \{1, \dots, T\}$.

2. Every vertex u that is not a self-loop (i.e., with either $\mathsf{S}'(u) \neq u$ or $\mathsf{P}'(u) \neq u$) contains in its label only pairs $(v, i) \in \{0,1\}^M \times \{1, \dots, T\}$ such that $\mathsf{V}(v, i) = 1$ and is given a non-zero value by M'.

3. A vertex u lies on the main directed line starting at the trivial source corresponding to the initial pebbling configuration if and only if it contains in its label only pairs $(v, i) \in \{0,1\}^M \times \{1, \dots, T\}$ such that $v = \mathsf{S}^i(0^m)$.

Items 1 and 2 are immediate from the description of the reduction. Items 3 follows since any vertex u which is not a self-loop corresponds to a valid pebbling configuration. Thus, if the label of u contains only pairs $(v, i) \in \{0,1\}^M \times \{1, \dots, T\}$ such that $v = \mathsf{S}^i(0^m)$ then the successor (resp. predecessor) of u on the main directed EOML line is the successive (resp. preceding) pebbling configuration. Note that the converse implication is straightforward.

Consider any vertex u that is a solution of the resulting EOML instance (of any type from Definition 6). If the label of u contains a pair of the form (v, T) then we have found a solution to the relaxed SVL instance. By item 2, $\mathsf{V}(v, T) = 1$ and either v is the sink of the rSVL instance (when $v = \mathsf{S}^T(0^m)$) or (v, T) is a false positive (when $v \neq \mathsf{S}^T(0^m)$).

Otherwise, we show that the solution u must contain a false positive in its label. First, notice that there are no other solutions on the main directed line besides the sink containing (v_T, T) in its label (this sink falls into the previous case we already handled). Therefore, u lies off the main path and, by items 2 and 3 above, its label must contain a pair (v, i) such that $\mathsf{V}(v, i) = 1$ but $v \neq \mathsf{S}^i(0^m)$, i.e., a false positive. Since there are $\log_2(T)$ such pairs in the label, we can select the false positive (v, i) with a noticeable probability simply by picking one of the pairs in the label uniformly at random. □

3 The Sumcheck Protocol

The sumcheck protocol was introduced by Lund et al. [43] to show that $\#\mathsf{P}$ is contained in IP. In this section, we recall the original protocol (§3.2) and then describe the non-interactive protocol that is obtained by applying the Fiat-Shamir transformation to the interactive protocol (§3.3). But first, we give the necessary definitions pertaining to proof systems (§3.1).

3.1 Proof Systems

Interactive protocols. An interactive protocol consists of a pair $(\mathcal{P}, \mathcal{V})$ of interactive Turing machines that are run on a common input x. The first machine, which is deterministic, is called the prover and is denoted by $\mathcal{P}$, and the second machine, which is probabilistic, is called the verifier and is denoted by $\mathcal{V}$. In an ℓ-round interactive protocol, in each round $i \in [\ell]$, first $\mathcal{P}$ sends a message $\alpha_i \in \Sigma^a$ to $\mathcal{V}$ and then $\mathcal{V}$ sends a message $\beta_i \in \Sigma^b$ to $\mathcal{P}$, where Σ is a finite alphabet. At the end of the interaction, $\mathcal{V}$ runs a (deterministic) Turing machine on input $(x, (\beta_1, \ldots, \beta_\ell), (\alpha_1, \ldots, \alpha_\ell))$. The interactive protocol is *public-coin* if β_i is a uniformly distributed random string in Σ^b. The *communication complexity* of an interactive protocol is the total number of bits transmitted, namely, $(\ell \cdot (b + a) \cdot \log_2(|\Sigma|))$.

Interactive proofs (IPs). The classical notion of an interactive proof for a language $\mathcal{L}$ is due to Goldwasser, Micali and Rackoff [32].

Definition 11. An interactive protocol $(\mathcal{P}, \mathcal{V})$ is a δ-*sound* interactive proof (IP) for $\mathcal{L}$ if:

- **Completeness:** For every $x \in \mathcal{L}$, if $\mathcal{V}$ interacts with $\mathcal{P}$ on common input x, then $\mathcal{V}$ accepts with probability 1.

- **Soundness:** For every $x \notin \mathcal{L}$ and every (computationally unbounded) cheating prover strategy $\widetilde{\mathcal{P}}$, the verifier $\mathcal{V}$ accepts when interacting with $\widetilde{\mathcal{P}}$ with probability less than $\delta(|x|)$, where $\delta = \delta(n)$ is called the *soundness error* of the proof system.

Unambiguous IPs. Reingold, Rothblum and Rothblum [50] introduced a variant of interactive proofs, called *unambiguous* interactive proofs, in which the *honest* prover strategy is defined for *every* x (i.e., also for $x \notin \mathcal{L}$) and the verifier is required to reject when interacting with any cheating prover that deviates from the prescribed honest prover strategy at any point of the interaction.

More formally, if $(\mathcal{P}, \mathcal{V})$ is an interactive protocol, and $\widetilde{\mathcal{P}}$ is some arbitrary (cheating) strategy, we say that $\widetilde{\mathcal{P}}$ *deviates* from the protocol at round i^* if the message sent by $\widetilde{\mathcal{P}}$ in round i^* differs from the message that $\mathcal{P}$ would have sent given the transcript of the protocol thus far. In other words, if the verifier sent the messages $\beta_1, \ldots, \beta_{i^*-1}$ in rounds $1, \ldots, (i^* - 1)$ respectively, we say that $\widetilde{\mathcal{P}}$ deviates from the protocol at round i^* if

$$\widetilde{\mathcal{P}}\left(x, i^*, (\beta_1, \ldots, \beta_{i-1})\right) \neq \mathcal{P}\left(x, i^*, (\beta_1, \ldots, \beta_{i-1})\right).$$

We consider a slightly different formulation, where the unambiguity is required to hold *only* for $x \in \mathcal{L}$. Therefore for $x \notin \mathcal{L}$, we need to reinstate the standard soundness condition.

Definition 12. An interactive protocol $(\mathcal{P}, \mathcal{V})$, in which we call $\mathcal{P}$ the *prescribed prover*, is a (δ, ϵ)-unambiguosly sound (or simply δ-unambiguosly sound if $\epsilon = \delta$) IP for $\mathcal{L}$ if the following three properties hold:

- **Prescribed Completeness:** For every $x \in \{0,1\}^*$, if $\mathcal{V}$ interacts with $\mathcal{P}$ on common input x, then $\mathcal{V}$ outputs $\mathcal{L}(x)$ with probability 1.

- **Soundness:** For every $x \notin \mathcal{L}$ and every (computationally unbounded) cheating prover strategy $\widetilde{\mathcal{P}}$, the verifier $\mathcal{V}$ accepts when interacting with $\widetilde{\mathcal{P}}$ with probability less than $\delta(|x|)$, where $\delta = \delta(n)$ is called the *soundness error* of the proof system.

- **Unambiguity:** For every $x \in \mathcal{L}$, every (computationally unbounded) cheating prover strategy $\widetilde{\mathcal{P}}$, every round $i^* \in [\ell]$, and for every $\beta_1, \ldots, \beta_{i^*-1}$, if $\widetilde{\mathcal{P}}$ first deviates from the protocol in round i^* (given the messages $\beta_1, \ldots, \beta_{i^*-1}$ in rounds $1, \ldots, (i^*-1)$ respectively), then at the end of the protocol $\mathcal{V}$ accepts with probability at most $\epsilon(|x|)$, where the probability is over $\mathcal{V}$'s coin tosses in rounds $i^*, \ldots, \ell$.

Non-interactive proof systems. A non-interactive proof system involves the prover sending a single message to the verifier. To give this proof system additional power, we assume that both prover and verifier have access to a common reference string (CRS). We focus on *adaptive* proof systems where a cheating prover gets to see the CRS *before* forging a proof for a statement of its choice. As for the case of interactive proofs, we consider unambiguous non-interactive proof systems (instead of the standard "sound" non-interactive proof systems).

Definition 13. A pair of machines $(\mathcal{P}, \mathcal{V})$, where $\mathcal{P}$ is the prescribed prover, is a (δ, ϵ)-unambiguosly sound (or, if $\epsilon = \delta$, simply δ-unambiguosly sound) *adaptive* non-interactive proof system for a language $\mathcal{L}$ if $\mathcal{V}$ is probabilistic polynomial-time, and taking R to be a uniformly random CRS, the following three properties hold:

- **Prescribed Completeness:** For every $x \in \{0,1\}^*$,

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

- **Soundness:** For every (computationally unbounded) cheating prover strategy $\widetilde{\mathcal{P}}$,

$$\Pr\left[\mathcal{V}(x, \widetilde{\pi}, R) = 1 | (x, \widetilde{\pi}) \leftarrow \widetilde{\mathcal{P}}(R), x \notin \mathcal{L}\right] \leq \delta(|x|).$$

- **Unambiguity:** For every (computationally unbounded) cheating prover strategy $\widetilde{\mathcal{P}}$,

$$\Pr\left[\mathcal{V}(x, \widetilde{\pi}, R) = 1 \;\middle|\; \begin{array}{l} (x, \widetilde{\pi}) \leftarrow \widetilde{\mathcal{P}}(R), \pi \leftarrow \mathcal{P}(x,R) \\ \widetilde{\pi} \neq \pi, x \in \mathcal{L} \end{array}\right] \leq \epsilon(|x|).$$

Remark 14. A non-interactive proof system is called an *argument* if the soundness and unambiguity properties hold only against computationally-bounded (i.e., $\mathsf{poly}(n)$) cheating prover strategy $\widetilde{\mathcal{P}}$.

3.2 Interactive Sumcheck Protocol

Fix a finite field $\mathbb{F}$ and a subset $\mathbb{H} \subseteq \mathbb{F}$ (usually $\mathbb{H} = \{0,1\}$). In the original sumcheck protocol, a (not necessarily efficient) prover takes as input an n-variate polynomial $f : \mathbb{F}^n \to \mathbb{F}$ of degree at most d in each variable (think of d as a constant significantly smaller than $|\mathbb{F}|$). The prover's goal is to convince a verifier that

$$\sum_{\boldsymbol{z} \in \mathbb{H}^n} f(\boldsymbol{z}) = y,$$

for some value $y \in \mathbb{F}$. The verifier only has oracle access to f, and is given the constant $y \in \mathbb{F}$. The verifier is allowed a single oracle query to f, and runs in time $\mathsf{poly}(n, d, \log_2(|\mathbb{F}|))$. In Figure 2, we review the standard sumcheck protocol from [43], denoted by

$$\left(\mathcal{P}_{\mathrm{SC}}(y, f), \mathcal{V}_{\mathrm{SC}}^{f}(y)\right).$$

$\mathcal{P}_{\mathrm{SC}}$ is an interactive Turing machine, and $\mathcal{V}_{\mathrm{SC}}$ is a probabilistic interactive Turing machine with oracle access to $f : \mathbb{F}^n \to \mathbb{F}$. The prover $\mathcal{P}_{\mathrm{SC}}(y, f)$ runs in time $\mathsf{poly}(|\mathbb{F}|^n)$.[10] The verifier $\mathcal{V}_{\mathrm{SC}}^{g}(y)$ runs in time $\mathsf{poly}(n, \log_2(|\mathbb{F}|), d)$ and queries the oracle f at a single point. The communication complexity is $\mathsf{poly}(n, \log_2(|\mathbb{F}|), d)$, and the total number of bits sent from the verifier to the prover is $O(n \cdot \log_2 |\mathbb{F}|)$. 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.

[10]Here we assume the prover's input is a description of the function f, from which f can be computed (on any input) in time $\mathsf{poly}(n)$.

Interactive Sumcheck Protocol for $\sum_{z_1,\ldots,z_n \in \mathbb{H}} f(z_1,\ldots,z_n) = y$

Parameters:

1. $\mathbb{F}$ (field), n (dimension), d (individual degree)
2. $\mathbb{H} \subset \mathbb{F}$

Protocol:

1. Set $y_0 = y$
2. For $i \leftarrow 1,\ldots,n$:
   (at the beginning of round i, both $\mathcal{P}_{\text{SC}}$ and $\mathcal{V}_{\text{SC}}$ know y_{i-1} and $\beta_1,\ldots,\beta_{i-1} \in \mathbb{F}$)
   (a) $\mathcal{P}_{\text{SC}}$ computes the degree-d univariate polynomial
   $$g_i(x) := \sum_{z_{i+1},\ldots,z_n \in \mathbb{H}} f(\beta_1,\ldots,\beta_{i-1},x,z_{i+1},\ldots,z_n),$$
   $\mathcal{P}_{\text{SC}}$ sends this polynomial to $\mathcal{V}_{\text{SC}}$ by specifying its values on the first $d+1$ field elements, i.e. by sending $\{\alpha_{i,\gamma} = g_i(\gamma)\}_{\gamma=0}^{d}$.
   (b) $\mathcal{V}_{\text{SC}}$ receives $d+1$ field elements $\{\widetilde{\alpha}_{i,\gamma}\}_{\gamma=0}^{d}$, and interpolates the (unique) degree-d polynomial $\widetilde{g}_i$ s.t. $\forall \gamma \in \{0,\ldots,d\}, \widetilde{g}_i(\gamma) = \widetilde{\alpha}_{i,\gamma}$
   $\mathcal{V}_{\text{SC}}$ then checks that:
   $$\sum_{x \in \mathbb{H}} \widetilde{g}_i(x) = y_{i-1}.$$
   If not, then $\mathcal{V}_{\text{SC}}$ rejects.
   (c) $\mathcal{V}_{\text{SC}}$ chooses a random element $\beta_i \in_R \mathbb{F}$, sets $y_i = \widetilde{g}_i(\beta_i)$, and sends β_i to $\mathcal{P}_{\text{SC}}$.
3. $\mathcal{V}_{\text{SC}}$ uses a single oracle call to f to check that $y_n = f(\beta_1,\ldots,\beta_n)$.

Figure 2: Sumcheck protocol $(\mathcal{P}_{\text{SC}}(y,f), \mathcal{V}_{\text{SC}}^{f}(y))$ from [43]

Sumcheck protocol for $\mathcal{L}_{\text{SC}}$. Recall the definition of the language $\mathcal{L}_{\text{SC}}$ from §1.2. Although the sumcheck protocol described above works for "plain" $\mathcal{L}_{\text{SC}}$ — i.e., $\mathcal{L}_{\text{SC}}$ without a prefix and partial transcript and hence $i = j = 0$, — it can be easily adapted for "full" $\mathcal{L}_{\text{SC}}$ where i and j can be both greater than zero, which is required for our application.[11] As a first step, we describe in Figure 3 the protocol for "prefixed" $\mathcal{L}_{\text{SC}}$ — i.e., $\mathcal{L}_{\text{SC}}$ with $j > 0$ but $i = 0$. We show in Theorem 15 that this protocol is an unambiguously-sound interactive proof system (for prefixed $\mathcal{L}_{\text{SC}}$). The sketch of the protocol and the corresponding theorem for full $\mathcal{L}_{\text{SC}}$ then follows Theorem 15. We remark that in both the protocols the verifier is given the polynomial f, unlike oracle-access as in the original Sumcheck protocol in Figure 2. Thus, the verifier's run-time can be $\mathsf{poly}(n, d, \log_2(|\mathbb{F}|), |f|)$, where $|f|$ refers to the size of the polynomial f under some appropriate representation (e.g., arithmetic circuits).

Theorem 15. *Let $f : \mathbb{F}^n \to \mathbb{F}$ be an n-variate polynomial of degree at most $d < |\mathbb{F}|$ in each variable. The sumcheck protocol described in Figure 3 is a $(d(n-j)/|\mathbb{F}|)$-unambiguously sound interactive proof system for prefixed $\mathcal{L}_{\text{SC}}$ (i.e., with $j > 0$ and $i = 0$). That is, it satisfies the following three properties.*

[11]We remark that the language $\mathcal{L}_{\text{SC}}$ in its most general form (i.e., with $i, j > 0$) is used only in Claim 4. In the discussions that follow, the prefixed language suffices.

Interactive $\boldsymbol{\beta}$-Prefix Sumcheck Protocol for $\sum_{z_{j+1},\dots,z_n\in\mathbb{H}} f(\boldsymbol{\beta}, z_{j+1},\dots,z_n) = y$

Parameters:

1. $\mathbb{F}$ (field), n (dimension), d (individual degree)
2. $\mathbb{H} \subset \mathbb{F}$

Protocol:

1. Let $\boldsymbol{\beta} = (\beta_1,\dots,\beta_j)$ and set $y_j = y$ and
2. For $i \leftarrow j+1,\dots,n$:
   (at the beginning of round i, both $\mathcal{P}_{\mathrm{SC}}$ and $\mathcal{V}_{\mathrm{SC}}$ know y_{i-1} and $\beta_1,\dots,\beta_{i-1} \in \mathbb{F}$)

   (a) $\mathcal{P}_{\mathrm{SC}}$ computes the degree-d univariate polynomial
   $$g_i(x) := \sum_{z_{i+1},\dots,z_n\in\mathbb{H}} f(\beta_1,\dots,\beta_{i-1},x,z_{i+1},\dots,z_n),$$
   $\mathcal{P}_{\mathrm{SC}}$ sends this polynomial to $\mathcal{V}_{\mathrm{SC}}$ by specifying its values on the first $d+1$ field elements, i.e. by sending $\{\alpha_{i,\gamma} = g_i(\gamma)\}_{\gamma=0}^{d}$.

   (b) $\mathcal{V}_{\mathrm{SC}}$ receives $d+1$ field elements $\{\widetilde{\alpha}_{i,\gamma}\}_{\gamma=0}^{d}$, and interpolates the (unique) degree-d polynomial $\widetilde{g}_i$ s.t. $\forall\gamma \in \{0,\dots,d\}, \widetilde{g}_i(\gamma) = \widetilde{\alpha}_{i,\gamma}$
   $\mathcal{V}_{\mathrm{SC}}$ then checks that:
   $$\sum_{x\in\mathbb{H}} \widetilde{g}_i(x) = y_{i-1}.$$
   If not, then $\mathcal{V}_{\mathrm{SC}}$ rejects.

   (c) $\mathcal{V}_{\mathrm{SC}}$ chooses a random element $\beta_i \in_R \mathbb{F}$, sets $y_i = \widetilde{g}_i(\beta_i)$, and sends β_i to $\mathcal{P}_{\mathrm{SC}}$.
3. $\mathcal{V}_{\mathrm{SC}}$ checks that $y_n = f(\beta_1,\dots,\beta_n)$.

Figure 3: $\boldsymbol{\beta}$-prefix Sumcheck protocol $(\mathcal{P}_{\mathrm{SC}}(y,f,\boldsymbol{\beta}), \mathcal{V}_{\mathrm{SC}}(y,f,\boldsymbol{\beta}))$

- ***Prescribed Completeness:*** *For every $y \in \mathbb{F}$ and $\boldsymbol{\beta} \in \mathbb{F}^j$,*
$$\Pr\left[(\mathcal{P}_{\mathrm{SC}}(y,f,\boldsymbol{\beta}), \mathcal{V}_{\mathrm{SC}}(y,f,\boldsymbol{\beta})) = \mathcal{L}_{\mathrm{SC}}(y,f,\boldsymbol{\beta},\emptyset)\right] = 1.$$

- ***Soundness:*** *If $\sum_{\boldsymbol{z}\in\mathbb{H}^{n-j}} f(\boldsymbol{\beta},\boldsymbol{z}) \neq y$ then for every (computationally unbounded) interactive Turing machine $\widetilde{\mathcal{P}}$,*
$$\Pr\left[\left(\widetilde{\mathcal{P}}(y,f,\boldsymbol{\beta}), \mathcal{V}_{\mathrm{SC}}(y,f,\boldsymbol{\beta})\right) = 1\right] \leq \frac{d(n-j)}{|\mathbb{F}|}.$$

- ***Unambiguity:*** *If $\sum_{\boldsymbol{z}\in\mathbb{H}^{n-j}} f(\boldsymbol{\beta},\boldsymbol{z}) = y$ then for every (computationally unbounded) interactive Turing machine $\widetilde{\mathcal{P}}$ that deviates from the protocol, $\mathcal{V}_{\mathrm{SC}}$ accepts with probability at most $d(n-j)/|\mathbb{F}|$.*

Proof. We show below that completeness, soundness and unambiguity holds for $j = 0$ — the argument when $j > 0$ follows similarly by fixing the prefix $\boldsymbol{\beta}$.

Prescribed completeness follows from the protocol description. As for the soundness, let $f : \mathbb{F}^n \to \mathbb{F}$ be a polynomial of degree at most d in each variable, such that $\sum_{\boldsymbol{z}\in\mathbb{H}^n} f(\boldsymbol{z}) \neq y$. Assume for the sake of contradiction that there exists a cheating prover $\widetilde{\mathcal{P}}$ for which
$$s := \Pr\left[\left(\widetilde{\mathcal{P}}(y,f), \mathcal{V}_{\mathrm{SC}}(y,f)\right) = 1\right] > \frac{dn}{|\mathbb{F}|}.$$

Recall that in the sumcheck protocol the prover sends n univariate polynomials $g_1, \ldots, g_n$, and the verifier sends $n-1$ random field elements $\beta_1, \ldots, \beta_{n-1} \in \mathbb{F}$. For every $i \in [1, n]$, let A_i denote the event that

$$g_i(x) = \sum_{z_{i+1},\ldots,z_n \in \mathbb{H}} f(\beta_1, \ldots, \beta_{i-1}, x, z_{i+1}, \ldots, z_n).$$

Let S denote the event that $\left(\widetilde{\mathcal{P}}(y, f), \mathcal{V}_{\mathrm{SC}}(y, f)\right) = 1$. Notice that $\Pr[S|A_1 \wedge \ldots \wedge A_n] = 0$. We will reach a contradiction by proving that

$$\Pr[S|A_1 \wedge \ldots \wedge A_n] \geq s - \frac{dn}{|\mathbb{F}|}.$$

To this end, we prove by (reverse) induction that for every $j \in [1, n]$,

$$\Pr[S|A_j \wedge \ldots \wedge A_n] \geq s - \frac{(n-j+1)d}{|\mathbb{F}|}. \tag{1}$$

For $j = n$,

$$s = \Pr[S] \leq \Pr[S|\neg(A_n)] + \Pr[S|A_n] \leq \frac{d}{|\mathbb{F}|} + \Pr[S|A_n],$$

where the latter inequality follows by the Schwartz-Zippel lemma, i.e. the fact that every two distinct univariate polynomials of degree $\leq d$ over $\mathbb{F}$ agree in at most d points. Thus,

$$\Pr[S|A_n] \geq s - \frac{d}{|\mathbb{F}|}.$$

Assume that Equation (1) holds for j, and we will show that it holds for $j-1$.

$$\begin{aligned} s - \frac{(n-j+1)d}{|\mathbb{F}|} &\leq \Pr[S|A_j \wedge \ldots \wedge A_n] \\ &\leq \Pr[S|\neg(A_{j-1}) \wedge A_j \wedge \ldots \wedge A_n] + \Pr[S|A_{j-1} \wedge A_j \wedge \ldots \wedge A_n] \\ &\leq \frac{d}{|\mathbb{F}|} + \Pr[S|A_{j-1} \wedge \ldots \wedge A_n], \end{aligned}$$

which implies that

$$\Pr[S|A_{j-1} \wedge \ldots \wedge A_n] \geq s - \frac{(n-(j-1)+1)d}{|\mathbb{F}|},$$

as desired.

Finally, to prove unambiguity, we describe the proof in [50]. Let $f : \mathbb{F}^n \to \mathbb{F}$ be a polynomial of degree at most d in each variable, such that $\sum_{\boldsymbol{z} \in \mathbb{H}^n} f(\boldsymbol{z}) = y$, and let $\widetilde{\mathcal{P}}$ be a cheating prover. Consider $i^* \in [n]$ and $\beta_1, \ldots, \beta_{i^*-1} \in \mathbb{F}$ be the first round that $\widetilde{\mathcal{P}}$ deviates from the prescribed strategy. i.e. $\forall i < i^*$, $g_i \equiv \widetilde{g}_i$ and $g_{i^*} \not\equiv \widetilde{g}_{i^*}$, where $g_i \equiv \widetilde{g}_i$ denotes that two polynomials g_i and $\widetilde{g}_i$ are equivalent. Here g_i is the output of the prescribed prover $\mathcal{P}_{\mathrm{SC}}$ on input $\beta_1, \ldots, \beta_{i-1}$ and $\widetilde{g}_i$ is the output of $\widetilde{\mathcal{P}}$ on the same input.

Claim 16. *Let $i \in \{i^*, \cdots, n-1\}$ and $\beta_{i^*}, \ldots, \beta_{i-1} \in \mathbb{F}$ and suppose $g_i \not\equiv \widetilde{g}_i$. Then with probability at least $\left(1 - \frac{d}{|\mathbb{F}|}\right)$ over the choice of β_i, either $\mathcal{V}_{\mathrm{SC}}$ rejects or $g_{i+1} \not\equiv \widetilde{g}_{i+1}$.*

Proof. By the Schwartz-Zippel lemma, since g_i and $\widetilde{g}_i$ have degree at most d, $g_i(\beta_i) \neq \widetilde{g}_i(\beta_i)$ with probability $\left(1 - \frac{d}{|\mathbb{F}|}\right)$ over the choice of β_i. Suppose this is the case, and yet $g_{i+1} \equiv \widetilde{g}_{i+1}$, then

$$\sum_{x \in \mathbb{H}} \widetilde{g}_{i+1}(x) = \sum_{x \in \mathbb{H}} g_{i+1}(x) = g_i(\beta_i) \neq \widetilde{g}_i(\beta_i) = y_i.$$

Thus $\mathcal{V}_{\mathrm{SC}}$ would reject when it makes the check to see if $\sum_{x \in \mathbb{H}} \widetilde{g}_{i+1}(x) = y_i$. □

Applying union bound over all $i \in \{i^*, \cdots, n-1\}$ for the above claim, we have that with probability at least $\left(1-(n-1)\cdot\frac{d}{|\mathbb{F}|}\right)$ over the choice of $\beta_{i^*}, \cdots, \beta_{n-1}$ either $\mathcal{V}_{\mathrm{SC}}$ rejects or $g_n \not\equiv \widetilde{g}_n$.

If $g_n \not\equiv \widetilde{g}_n$, by the Schwartz-Zippel lemma, with probability $\left(1-\frac{d}{|\mathbb{F}|}\right)$ over the choice of β_n,

$$\widetilde{g}_n(\beta_n) \neq g_n(\beta_n) = f(\beta_1, \cdots, \beta_n).$$

Thus when $\mathcal{V}_{\mathrm{SC}}$ makes the oracle call to check if $y_n = f(\beta_1, \cdots, \beta_n)$, it will reject. Therefore, by a further application of the union bound, with probability at least $\left(1-\frac{nd}{|\mathbb{F}|}\right)$ either $\mathcal{V}_{\mathrm{SC}}$ rejects or $y_n = \widetilde{g}_n(\beta_n) \neq g_n(\beta_n) = f(\beta_1, \cdots, \beta_n)$. □

The full Sumcheck protocol. When $i=0$, the sumcheck protocol for full $\mathcal{L}_{\mathrm{SC}}$ works exactly as described in Figure 3. Since $i>0$ corresponds to the case where the sumcheck protocol has been executed partially to the point where the prover sends the message $\tilde{\alpha}_{i+j}$, the full sumcheck protocol just continues the protocol from that point onwards. Hence the first message in the full sumcheck protocol is from the verifier to the prover, unlike the normal sumcheck protocol. Let's denote this protocol by

$$(\mathcal{P}_{\mathrm{SC}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}), \mathcal{V}_{\mathrm{SC}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}))\,,$$

where $\boldsymbol{\tau} = \tilde{\alpha}_{j+1}, \beta_{j+1}, \ldots, \tilde{\alpha}_{j+i}$ is the fixed partial transcript. In Theorem 17 below, we capture the fact that the protocol just described is an unambiguously sound interactive proof for $\mathcal{L}_{\mathrm{SC}}$ — the proof is similar to that of Theorem 15 and is omitted.

Theorem 17. *Let $f : \mathbb{F}^n \to \mathbb{F}$ be an n-variate polynomial of degree at most $d < |\mathbb{F}|$ in each variable. The sumcheck protocol $(\mathcal{P}_{\mathrm{SC}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}), \mathcal{V}_{\mathrm{SC}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}))$ described above is a $(d(n-i-j)/|\mathbb{F}|)$-unambiguously sound interactive proof system for $\mathcal{L}_{\mathrm{SC}}$.*

3.3 Non-Interactive Sumcheck Protocol

We consider the non-interactive version of the sumcheck protocol obtained by applying the Fiat-Shamir transformation to the protocols from Figure 3 and Theorem 17. To be exact, the verifier's "challenges" β_i in the non-interactive protocol are obtained by applying a hash function $h : \{0,1\}^* \to \mathbb{F}$ to the transcript thus far, which is comprised of the instance and the prover's messages up to that round.

The non-interactive sumcheck protocol $(\mathcal{P}_{\mathrm{FS}}(y, f, \boldsymbol{\beta}), \mathcal{V}_{\mathrm{FS}}(y, f, \boldsymbol{\beta}))$ corresponding to Figure 3 is given in Figure 4. The algorithm $\mathcal{P}_{\mathrm{FS}}$ runs in time $\mathsf{poly}(|\mathbb{F}|^n)$. The verification algorithm $\mathcal{V}_{\mathrm{FS}}$ runs in time $\mathsf{poly}(n, |f|)$ and space $O(n \cdot \log_2(|\mathbb{F}|))$. The size of the proof is $O(n \cdot \log_2(|\mathbb{F}|))$. The non-interactive sumcheck protocol $(\mathcal{P}_{\mathrm{FS}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}), \mathcal{V}_{\mathrm{FS}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}))$ corresponding to the protocol from Theorem 17 works similar to the protocol in Figure 3 and hence incurs similar costs.

Assumption on Fiat-Shamir. The assumption that underlies our main theorem pertains to the soundness of the collapsed protocol $(\mathcal{P}_{\mathrm{FS}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}), \mathcal{V}_{\mathrm{FS}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}))$. In particular, we assume that the Fiat-Shamir Transform is adaptively *sound* when applied to the interactive sumcheck protocol and, as a result, that the collapsed protocol $(\mathcal{P}_{\mathrm{FS}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}), \mathcal{V}_{\mathrm{FS}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}))$ is an adaptively sound non-interactive proof system for the language $\mathcal{L}_{\mathrm{SC}}$. By Claim 16, it follows that the collapsed protocol is adaptively unambiguously sound as required for the application in Section 4. In Lemma 19 below, we show that the assumption holds for the protocol described in Figure 4 *with respect to random oracles*. This also holds for the protocol from Theorem 17 as claimed in Lemma 20. In fact we directly show the stronger property of unambiguous soundness.

Non-Interactive $\boldsymbol{\beta}$-Prefix Sumcheck Protocol for $\sum_{z_{j+1},\ldots,z_n \in \mathbb{H}} f(\boldsymbol{\beta}, z_{j+1}, \ldots, z_n) = y$

Parameters:

1. $\mathbb{F}$ (field), n (dimension), d (individual degree)
2. $\mathbb{H} \subset \mathbb{F}$
3. hash function $h : \{0,1\}^* \to \mathbb{F}$

$\mathcal{P}_{\mathrm{FS}}(y, f, \boldsymbol{\beta})$:

1. Let $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_j)$ and set $y_j = y$.
2. For $i \leftarrow j+1, \ldots, n$:
   (a) Compute the degree-d univariate polynomial
   $$g_i(x) := \sum_{z_{i+1},\ldots,z_n \in \mathbb{H}} f(\boldsymbol{\beta}, \beta_{j+1}, \ldots, \beta_{i-1}, x, z_{i+1}, \ldots, z_n).$$
   Let $\{\alpha_{i,\gamma} = g_i(\gamma)\}_{\gamma=0}^{d}$ be the values of g_i on the first $d+1$ field elements.
   (b) Compute
   $$\beta_i = h(\mathbb{F}, y, f, \boldsymbol{\beta}, \{\alpha_{j+1,\gamma}\}_{\gamma=0}^{d}, \beta_{j+1}, \ldots, \beta_{i-1}, \{\alpha_{i,\gamma}\}_{\gamma=0}^{d})$$
   and set $y_i = g_i(\beta_i)$.
3. Output $\pi = (\{\alpha_{j+1,\gamma}\}_{\gamma=0}^{d}, \ldots, \{\alpha_{n,\gamma}\}_{\gamma=0}^{d})$

$\mathcal{V}_{\mathrm{FS}}((y, f, \boldsymbol{\beta}), \{\widetilde{\alpha}_{j+1,\gamma}\}_{\gamma=0}^{d}, \ldots, \{\widetilde{\alpha}_{n,\gamma}\}_{\gamma=0}^{d})$:

1. For $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_j)$, set $y_j = y$.
2. For $i \leftarrow j+1, \ldots, n$:
   (a) Use the $d+1$ field elements $\{\widetilde{\alpha}_{i,\gamma}\}_{\gamma=0}^{d}$ to interpolate the (unique) degree-d polynomial $\widetilde{g}_i$ s.t. $\forall \gamma \in \{0, \ldots, d\}, \widetilde{g}_i(\gamma) = \widetilde{\alpha}_{i,\gamma}$.
   Check that:
   $$\sum_{x \in \mathbb{H}} \widetilde{g}_i(x) = y_{i-1}.$$
   If not, then reject.
   (b) Compute
   $$\beta_i = h(\mathbb{F}, y, f, \boldsymbol{\beta}, \{\widetilde{\alpha}_{j+1,\gamma}\}_{\gamma=0}^{d}, \beta_{j+1}, \ldots, \beta_{i-1}, \{\widetilde{\alpha}_{i,\gamma}\}_{\gamma=0}^{d})$$
   and set $y_i = \widetilde{g}_i(\beta_i)$.
3. If $y_n = f(\beta_1, \ldots, \beta_n)$ then accept and otherwise reject.

Figure 4: Non-interactive version of the sumcheck protocol $(\mathcal{P}_{\mathrm{SC}}(y, f, \boldsymbol{\beta}), \mathcal{V}_{\mathrm{SC}}(y, f, \boldsymbol{\beta}))$ from Figure 3.

Assumption 18. *The Fiat-Shamir heuristic is* unambiguously *sound for the sumcheck protocol from Theorem 17. In other words, there exists a family of hash functions $\mathcal{H}$ such that when instantiated with (random) $h : \{0,1\}^* \to \mathbb{F}$ from $\mathcal{H}$, the non-interactive sumcheck protocol $(\mathcal{P}_{\mathrm{FS}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}), \mathcal{V}_{\mathrm{FS}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}))$ from Theorem 17 is (δ, ϵ)-unambiguously sound for the language $\mathcal{L}_{\mathrm{SC}}$ for some δ and ϵ that are negligible in n.*

Lemma 19. *Let $f : \mathbb{F}^n \to \mathbb{F}$ be an n-variate polynomial of degree at most d in each variable and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_j) \in \mathbb{F}^j$ be any prefix. Let $(\mathcal{P}^h_{\text{FS}}(y, f, \boldsymbol{\beta}), \mathcal{V}^h_{\text{FS}}(y, f, \boldsymbol{\beta}))$ denote the non-interactive sumcheck protocol obtained by instantiating the protocol described in Figure 4 with a random oracle h. Then $(\mathcal{P}^h_{\text{FS}}, \mathcal{V}^h_{\text{FS}})$ is a $(Qd/|\mathbb{F}|)$-unambiguously sound non-interactive proof system for prefixed language $\mathcal{L}_{\text{SC}}$, where $Q = Q(n)$ denotes the number of queries made to the random oracle. That is, it satisfies the following three properties.*

- ***Prescribed Completeness:*** *For every y,*

$$\Pr\left[\mathcal{V}^h_{\text{FS}}((y, f, \boldsymbol{\beta}), \mathcal{P}^h_{\text{FS}}(y, f, \boldsymbol{\beta})) = \mathcal{L}_{\text{SC}}(y, f, \boldsymbol{\beta}, \emptyset)\right] = 1.$$

- ***Soundness:*** *For every (computationally unbounded) cheating prover strategy $\widetilde{\mathcal{P}}$ that makes at most Q queries to the random oracle,*

$$\Pr\left[\mathcal{V}^h_{\text{FS}}((y, f, \boldsymbol{\beta}), \widetilde{\pi}) = 1 \;\middle|\; \begin{array}{l} ((y, f, \boldsymbol{\beta}), \widetilde{\pi}) \leftarrow \widetilde{\mathcal{P}}^h \\ \sum_{\boldsymbol{z} \in \mathbb{H}^{n-j}} f(\boldsymbol{\beta}, \boldsymbol{z}) \neq y \end{array}\right] \leq \frac{Qd}{|\mathbb{F}|}.$$

- ***Unambiguity:*** *For every (computationally unbounded) cheating prover strategy $\widetilde{\mathcal{P}}$ that makes at most Q queries to the random oracle,*

$$\Pr\left[\mathcal{V}^h_{\text{FS}}((y, f, \boldsymbol{\beta}), \widetilde{\pi}) = 1 \;\middle|\; \begin{array}{l} ((y, f, \boldsymbol{\beta}), \widetilde{\pi}) \leftarrow \widetilde{\mathcal{P}}^h \\ \pi \leftarrow \mathcal{P}^h_{\text{FS}}(y, f, \boldsymbol{\beta}) \\ \widetilde{\pi} \neq \pi, \sum_{\boldsymbol{z} \in \mathbb{H}^{n-j}} f(\boldsymbol{\beta}, \boldsymbol{z}) = y \end{array}\right] \leq \frac{Qd}{|\mathbb{F}|}.$$

Lemma 20. *Let $f : \mathbb{F}^n \to \mathbb{F}$ be an n-variate polynomial of degree at most d in each variable, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_j) \in \mathbb{F}^j$ be any prefix, and $\boldsymbol{\tau} = \widetilde{\alpha}_{j+1}, \beta_{j+1}, \ldots, \widetilde{\alpha}_{j+i}$ be any fixed partial transcript. Let $(\mathcal{P}^h_{\text{FS}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}), \mathcal{V}^h_{\text{FS}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}))$ denote the non-interactive sumcheck protocol obtained by instantiating the protocol described in Theorem 17 with a random oracle h. Then $(\mathcal{P}^h_{\text{FS}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}), \mathcal{V}^h_{\text{FS}}(y, f, \boldsymbol{\beta}, \boldsymbol{\tau}))$ is a $(Qd/|\mathbb{F}|)$-unambiguously sound non-interactive proof system for the language $\mathcal{L}_{\text{SC}}$, where $Q = Q(n)$ denotes the number of queries made to the random oracle.*

Proof (of Lemma 19). A query is a tuple of the form

$$(\mathbb{F}, y, f, \boldsymbol{\beta}, \widetilde{\alpha}_{j+1}, \widetilde{\beta}_{j+1}, \ldots, \widetilde{\beta}_{\ell-1}, \widetilde{\alpha}_\ell),$$

where f is a polynomial over the field $\mathbb{F}$, $y \in \mathbb{F}$, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_j)$ is a prefix, and $\widetilde{\alpha}_k \in \mathbb{F}^{d+1}$ and $\widetilde{\beta}_k \in \mathbb{F}$ for $k \in [j+1, \ell]$ where $\ell \in [n-j]$. For a statement $(y, f, \boldsymbol{\beta})$ (either in or not in the language $\mathcal{L}_{\text{SC}}$), we consider the output $\alpha_{j+1}, \ldots, \alpha_n$ of the prescribed prover $\mathcal{P}_{\text{FS}}$ when invoked on $(y, f, \boldsymbol{\beta})$ and the associated hash values $\beta_{j+1}, \ldots, \beta_{\ell-1}$.

Let $\beta_\ell := h(\mathbb{F}, y, f, \boldsymbol{\beta}, \alpha_{j+1}, \beta_{j+1}, \ldots, \beta_{\ell-1}, \alpha_\ell)$ and $\widetilde{\beta}_\ell := h(\mathbb{F}, y, f, \boldsymbol{\beta}, \widetilde{\alpha}_{j+1}, \widetilde{\beta}_{j+1}, \ldots, \widetilde{\alpha}_\ell))$. Also, let $g_\ell(x)$ (resp., $\widetilde{g}_\ell(x)$) denote the unique degree-d polynomial obtained by interpolating the field elements in α_ℓ (resp., $\widetilde{\alpha}_\ell$). We say that the query $(y, f, \boldsymbol{\beta}, \widetilde{\alpha}_{j+1}, \widetilde{\beta}_{j+1}, \ldots, \widetilde{\beta}_{\ell-1}, \widetilde{\alpha}_\ell)$ is "bad" if

1. $\widetilde{\alpha}_\ell \neq \alpha_\ell$ (which implies that the polynomial $g_\ell(x) \neq \widetilde{g}_\ell(x)$) and
2. $\widetilde{g}_\ell(\widetilde{\beta}_\ell) = g_\ell(\beta_\ell)$.

Since β_ℓ and $\widetilde{\beta}_\ell$ are outputs of a random oracle and the polynomials $g_\ell(x)$, $\widetilde{g}_\ell(x)$ are different, the probability of a particular query being bad is at most $d/|\mathbb{F}|$ by the Schwartz-Zippel lemma. Therefore by a union bound over all the queries, the probability that the adversary made a bad query during its execution is at most $Qd/|\mathbb{F}|$.

Note that in the absence of bad queries, an adversary cannot break either soundness or unambiguity of the non-interactive sumcheck protocol. It follows that the probability the adversary breaks the soundness or unambiguity is also at most $Qd/|\mathbb{F}|$. More formally, let "break" denote the event that an adversary that makes at most Q queries to the random oracle finds a proof that breaks either the soundness or unambiguity of the non-interactive such check protocol. The probability of "break" can be bounded as follows:

$$\begin{aligned}\Pr[\text{break}] &= \Pr[\text{break} \wedge (\text{bad query} \vee \neg\text{bad query})] \\ &\leq \Pr[\text{break} \wedge \text{bad query}] + \Pr[\text{break} \wedge \neg\text{bad query})] \\ &= \Pr[\text{break}|\text{bad query}] \cdot \Pr[\text{bad query}] \leq Qd/|\mathbb{F}|. \qquad \square\end{aligned}$$

4 The Reduction

In this section, we present an rSVL instance constructed using the non-interactive sumcheck protocol $(\mathcal{P}_{\text{FS}}, \mathcal{V}_{\text{FS}})$ for the language $\mathcal{L}_{\text{SC}}$ from §3.3 as a building block. The proposed rSVL instance counts — *incrementally and verifiably* — the number of satisfying assignments (i.e., the sum) of an n-variate polynomial f with individual degree at most d. To be specific, the main line in the rSVL instance starts at a fixed initial state s_0 (the source) and ends at a final state s_T (the sink) comprised of the sum

$$y = \sum_{\boldsymbol{z} \in \{0,1\}^n} f(\boldsymbol{z}),$$

as well as a proof π of y's correctness. The i-th (intermediate) state along the path from s_0 to s_T, which we denote by s_i, consists of appropriately-chosen prefix sums and associated proofs. (To be precise, each state also includes an index $t \in [d+1]^{\leq n}$ that is determined by its counter i.) The successor S performs single steps, receiving as input the current state s_i, and computing the next state s_{i+1}. The verification procedure V, which takes as input a state s and a counter i and accepts if s is the i-th state.

Since the sink will contain the overall sum y with a proof, any adversary that attempts to solve the rSVL instance by finding a type (i) solution (see, Definition 9) must compute the sum for f, the correctness of which can be verified using the proof. On the other hand, it is intractable for an adversary to find a type (ii) solution (i.e., a false positive (s', i) such that $s' \neq s_i$ but V accepts s' as the i-th vertex on the rSVL line) because of the unambiguous soundness of the non-interactive sumcheck proof system. The above is formally stated in the following theorem.

Theorem 21. *For a parameter n, fix a finite field $\mathbb{F}$ of sufficiently large size p (say $O(2^{\omega(n)})$. Let f be an n-variate polynomial over $\mathbb{F}$ of individual degree at most d. Pick a hash function h uniformly at random from a family $\mathcal{H}$. Let*

$$\mathsf{S} := \mathsf{S}_{f,h} : \{0,1\}^M \to \{0,1\}^M \text{ and } \mathsf{V} := \mathsf{V}_{f,h} : \{0,1\}^M \times [T]$$

be constructed as in Algorithm 3, with $M = M(n,d,p) = (d+1)n\log_2(p)$ and $T = T(n,d) = \sum_{j\in[n]}(d+2)^j$. Given an adversary A that solves instances from the rSVL family

$$\{(\mathsf{S}, \mathsf{V}, (f(0^n), \emptyset), T)\}_{n\in\mathbb{N}} \tag{2}$$

in polynomial time $T_{\mathsf{A}} = T_{\mathsf{A}}(n)$ and a non-negligible probability $\epsilon = \epsilon(n)$, it is possible to either

- *count the number of satisfying assignments to f in time $O(T_{\mathsf{A}})$ with probability ϵ, or*
- *break the unambiguous soundness of the non-interactive sumcheck protocol (Assumption 18) with probability at least $\epsilon/(d+1)\cdot n$.*

The proof of Theorem 21 is given in §4.3. The main technical component of our reduction are the successor circuit S and the verifier circuit V, described in §4.1 and §4.2. S an V, together, implement the incrementally-verifiable counter for statements of size 2^n. They are defined using a sequence of circuits

$$(\mathsf{S}_n, \mathsf{V}_n), \dots, (\mathsf{S}_0, \mathsf{V}_0),$$

where $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$ is an incrementally-verifiable counter for statements of size 2^{n-j+1} and is implemented *recursively* using $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$. At the base of the recursion, $(\mathsf{S}_0, \mathsf{V}_0)$ computes sums of size 1 and is therefore trivial: it takes a single step, uses $\mathsf{poly}(n)$ memory and has an "empty" proof. The circuits (S, V) simply invoke $(\mathsf{S}_n, \mathsf{V}_n)$.

We implement these procedures using circuits and to ensure that the size of these circuits does not blow up, we have to exploit the recursive structure of the sumcheck protocol. In our construction, if $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$ takes T steps, uses M bits of memory, and generates a final proof of size P bits, then $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$ takes $O(dT)$ steps, uses $M+O(dP)+\mathsf{poly}(n)$ memory, and has a final proof of size $P+\mathsf{poly}(n)$. On unwinding the recursion, it can be shown that (S, V) runs for $2^{O(n)}$ steps, uses $\mathsf{poly}(n)$ space and has proof size of $\mathsf{poly}(n)$. But most importantly S and V are polynomial-sized circuits, and therefore each step can be carried out in $\mathsf{poly}(n)$ time. In other words, we get an rSVL instance describing a directed graph with $2^{O(n)}$ vertices each with a label of size $\mathsf{poly}(n)$, where the successor and verifier functions have efficient descriptions.

4.1 The Recursive Construction

The circuits $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$ in our incrementally-verifiable counting procedure have hardwired into them

1. f, an n-variate polynomial over a field $\mathbb{F}$ of individual degree at most d (described as an arithmetic circuit of size $\mathsf{poly}(n)$ and degree d), and
2. h, the description of a hash function from the family $\mathcal{H}$.

It takes as its input a prefix $\boldsymbol{\beta} = (\beta_1, \dots, \beta_{j-1}) \in \mathbb{F}^{j-1}$ (and also the transcript **trans** as explained below). The goal of the procedure is computing the value y of the sum with prefix $\boldsymbol{\beta}$, along with a sumcheck proof for this value.

In order to describe how $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$ is implemented using $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$, we need to take a closer look at the non-interactive sumcheck protocol given in Figure 4. Suppose that the prover $\mathcal{P}_{\text{FS}}$ has been invoked on the $\boldsymbol{\beta} = (\beta_1, \dots, \beta_{j-1})$-prefix sum

$$\sum_{\boldsymbol{z} \in \{0,1\}^{n-j+1}} f(\boldsymbol{\beta}, \boldsymbol{z}) = y_{j-1}, \tag{3}$$

which is a statement of size 2^{n-j+1}. At the end of the first iteration, $\mathcal{P}_{\text{FS}}$ reduces this sumcheck to checking a smaller $(\boldsymbol{\beta}, \sigma)$-prefix sum $y_j = g_j(\sigma)$ of size 2^{n-j}, where $g_j(x)$ is the univariate polynomial

$$\sum_{\boldsymbol{z} \in \{0,1\}^{n-j}} f(\boldsymbol{\beta}, x, \boldsymbol{z})$$

specified by the field elements $\alpha_{j,0} = g_j(0), \dots, \alpha_{j,d} = g_j(d)$, and σ is the "challenge", i.e., a hash value depending on $\alpha_{j,0}, \dots, \alpha_{j,d}$.

Now, suppose we are given incrementally-verifiable procedures $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$ to compute sums of size 2^{n-j}, which takes $T(n-j)$ steps, uses $M(n-j)$ memory, and has a final proof of size $P(n-j)$ bits. Our construction of $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$ takes $(d+2)\cdot T(n-j)$ steps, uses $M(n-j)+(d+1)\cdot P(n-j)$ memory, and has a final proof of size $P(n-j)+(d+1)\log_2(p)$ bits (where, if you recall, p denotes the size of the field $\mathbb{F}$). On unwinding the above recursive expressions for T, M and P, we conclude that $(\mathsf{S}_n, \mathsf{V}_n)$ is procedure for computing sums of size 2^n with $2^{O(n)}$ steps and $\mathsf{poly}(n)$ space, and the final proof is of size $\mathsf{poly}(n)$.

To achieve this construction, we exploit the structure of the sumcheck protocol. Note that the polynomial $g_j(x)$ can itself be recursively computed with proof. To be more precise, for each $\gamma \in [d]$, we sequentially run $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$ to compute the valuations $\alpha_{j+1,\gamma}$ with a proof certifying the sum

$$\sum_{\boldsymbol{z}\in\{0,1\}^{n-j-1}} f(\boldsymbol{\beta}, \gamma, \boldsymbol{z}) = \alpha_{j+1,\gamma} = g_j(\gamma) \tag{4}$$

Once we possess $\alpha_{j+1,0}, \ldots, \alpha_{j+1,d}$ after the $(d+1)$ sequential applications of $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$, the challenge σ can be computed and subsequently the prefix-sum

$$\sum_{\boldsymbol{z}\in\{0,1\}^{n-j-1}} f(\boldsymbol{\beta}, \sigma, \boldsymbol{z}) = y_j = g_j(\sigma) \tag{5}$$

for the next round can also be computed using $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$ in an incrementally-verifiable manner. In other words, we have reduced computing the proof for the $\boldsymbol{\beta}$-prefix sum given in eq.3 to (i) $(d+1)$ *new* sumchecks given in eq.4 concerning the computation of polynomial $g_j(x)$, and (ii) the second iteration of the *original* sumcheck given in eq.5, which serves as an incrementally verifiable proof-merging procedure. Moreover, all the $(d+2)$ sumchecks above involve work proportional to the computation of sumchecks of size 2^{n-j-1}, and therefore they can be computed using $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$.

The working of the procedure $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$ on an input a prefix $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{j-1})$ can therefore be described on a high level as follows.

1. Compute the polynomial $g_j(x)$, represented by the field elements $\{\alpha_{j,\gamma} = g_j(\gamma)\}_{\gamma=0}^{d}$, incrementally and verifiably by invoking $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$ on $(\boldsymbol{\beta}, \gamma)$

2. Compute the $\boldsymbol{\beta}$-prefix sum by adding $(\boldsymbol{\beta}, 0)$- and $(\boldsymbol{\beta}, 1)$-prefix sums $\alpha_{j,0}$ and $\alpha_{j,1}$

3. Calculate the "challenge" σ and compute the partial proof for the original sumcheck: compute the proof for the $(\boldsymbol{\beta}, \sigma)$-prefix sum $g_j(\sigma)$ using $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$

4. Obtain the "merged proof" for the $\boldsymbol{\beta}$-prefix sum by appending $\{\alpha_{j,\gamma}\}_{\gamma=0}^{d}$ to the proof for $(\boldsymbol{\beta}, \sigma)$-prefix sum

5. Return the $\boldsymbol{\beta}$-prefix sum with proof

Keeping the above recursive procedure in mind, we proceed to detail the recursive successor and verifier circuits S_{n-j+1} and V_{n-j+1}.

The recursive successor. Recall that $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$, on input a prefix $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_{j-1})$, calls the procedure $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$ sequentially $d+2$ times. This results in a sequence of states $s_0, \ldots, s_{T(n-j+1)}$, where $s_{T(n-j+1)}$ is comprised of the sum

$$y = \sum_{\boldsymbol{z}\in\{0,1\}^{n-j+1}} f(\boldsymbol{\beta}, \boldsymbol{z})$$

as well as a proof π of y's correctness. Since the invocations of $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$ are sequential, an intermediate state s_i along the path from s_0 to $s_{T(n-j+1)}$, is comprised of at most $(d+2)$ "sub-states", one for each invocation of $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$.

In more details, each state s_i is associated with an index $t \in [d+1]^{\leq(n-j+1)}$ which is determined by the counter i. Loosely speaking, the index t of the i-th state s_i is the i-th vertex in the perfect $(d+2)$-dary tree that the standard depth-first search visits for the last time (see the discussion in §4.2 for more details). If $t = (t_1, \ldots, t_\ell)$ (where $\ell \in [n-j+1]$) then s_i consists of t_1 sub-states, where

- the first $t_1 - 1$ are *final*, i.e., correspond to full executions of $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$, and therefore consist of a single tuple of the form (y, π), and
- the t_1-th sub-state is either final and consists of a single tuple (y, π) as in the previous case *or* is itself *intermediate* (i.e., not corresponding to a full execution of $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$) and therefore consists of a sequence of tuples of the form (y, π).

The successor circuit S_{n-j+1}, on input a prefix $\boldsymbol{\beta}$ and the current state s with index t, computes the next state. Depending on the conditions of sub-states in s, it takes one of the following actions:

- Case A: The state s consists of $d+2$ final sub-states of $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$. Such sub-states contain the information necessary to compute the sum for the prefix $\boldsymbol{\beta}$ and assemble its proof (by merging). As a result, the next state is the final state of $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$.
- Case B: The state s consists of $t_1 < d+2$ final sub-states of $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$. In this case, S_{n-j+1} initiates the next (i.e., $t_1 + 1$-th) execution of $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$.
- Case C: The t_1-th sub-state s' is intermediate. Here, S_{n-j+1} simply calls the successor S_{n-j} to increment s' — and as a result the state s — by one step.

The resulting construction of S_{n-j+1} is formally described in Algorithm 1. There we have also addressed a minor detail (which also applies to V_{n-j+1}) that we have up to now brushed under the rug: in order to compute the challenges σ, the counters need some additional information. To this end, S_{n-j+1} receives as an auxiliary argument the protocol transcript that serves as the input to h, denoted by **trans**. From the description of the non-interactive sumcheck protocol in Figure 4, **trans** should contain the following information:

1. the *original* statement $(\mathbb{F}, f, y, \boldsymbol{\beta})$, left empty if the sum y has not been computed yet, and
2. a *partial* proof π for $\boldsymbol{\beta}$-prefix sum, which consists of all the values α_i and β_i that have been computed up to the current iteration (as specified in the description of $\mathcal{P}_{\text{FS}}$).

The recursive verifier. Given as input the prefix $\boldsymbol{\beta}$, the state s and an index t, the verifier V_{n-j+1} ensures that s equals the intermediate state s_i for $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$, where i is the counter that is associated with t.

If the state s is final for $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$ then $t = \varepsilon$ and s is a single tuple of the form (y, π). This can be verified directly by invoking $\mathcal{V}_{\text{FS}}$.

Otherwise s consists of at most $(d+2)$ (final or intermediate) sub-states of $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$, and V_{n-j+1} verifies each of these sub-states by invoking V_{n-j}. To be precise, for each sub-state s' in s, V_{n-j+1} first computes

1. the prefix $\boldsymbol{\beta}'$ for s', which is either $(\boldsymbol{\beta}, \gamma)$ for $\gamma \in [d]$, or $(\boldsymbol{\beta}, \sigma)$ for a challenge σ, and
2. the index t' for $\boldsymbol{\beta}'$, which is either ε in case s' is final for $(\mathsf{S}_{n-j}, \mathsf{V}_{n-j})$, or $(t_2, \ldots, t_\ell)$ otherwise.

Next, it checks the validity of the sub-state s' recursively by invoking V_{n-j}.

The formal description of V_{n-j+1} is given in Algorithm 2. Similarly to the successor S_{n-j+1}, V_{n-j+1} also receives the transcript as input to ensure that it possesses the necessary information to compute the challenges.

Recursive successor circuit $\mathsf{S}_{n-j+1}(\boldsymbol{\beta}, t, s, \mathbf{trans})$

```
Hardwired
    1. an $n$-variate polynomial $f$ of individual degree $d$ over $\mathbb{F}$
    2. the description of a hash function $h \in \mathcal{H}$
Input
    1. a prefix $\boldsymbol{\beta} = (\beta_1, \dots, \beta_{j-1}) \in \mathbb{F}^{j-1}$
    2. an index $t = (t_1, \dots, t_\ell) \in [d+1]^{\leq(n-j+1)}$
    3. a state $s \in \mathbb{F}^*$ parsed as a set of pairs of prefix sums and proofs $\{(y_0, \pi_0), (y_1, \pi_1), \dots, \}$
    4. a transcript $\mathbf{trans}$ containing the statement and partial proofs
Output the next state
Base Case $\mathsf{S}_0(\boldsymbol{\beta}, \varepsilon, \emptyset, \emptyset)$: Return $(f(\boldsymbol{\beta}), \emptyset)$
Recursion
    1. If $t = \varepsilon$ and $s \neq \emptyset$ return $s$ (already in final state: self-loop)
    2. If $t = d+1$ then return $\{y_\gamma\}_{\gamma=0}^{d}$ appended to $\pi_{d+1}$ (Case A: merge)
    3. Compute sub-state $s'$ by truncating $\{(y_\gamma, \pi_\gamma)\}_{\gamma=0}^{t_1-1}$ from $s$
    4. Set $t' := (t_2, \dots, t_\ell)$ as the index of the sub-state $s'$
    5. If $t = d$ or $t_1 = d+1$: increment/initialise $d+2$-th sub-state
       (a) If $\mathbf{trans} = \emptyset$ then initialise with statement $\mathbf{trans} := (\mathbb{F}, y_0 + y_1, f, \boldsymbol{\beta})$ (Case B)
       (b) Compute updated transcript $\mathbf{trans}'$ by appending $\{y_\gamma\}_{\gamma=0}^{d}$ to $\mathbf{trans}$ (Case C)
       (c) Compute the challenge $\sigma := h(\mathbf{trans})$ and append $\sigma$ to $\mathbf{trans}$
       (d) Increment/initialise $d+2$-th sub-state: $s' := \mathsf{S}_{n-j}((\boldsymbol{\beta}, \sigma), t', s', \mathbf{trans})$
       (e) Append $\{(y_\gamma, \pi_\gamma)\}_{\gamma=0}^{d}$ back to $s'$ to update $s$ and return it
       Else $t \neq d$ and $t_1 \neq d+1$
       (a) If $t \in [d-1]$: initialise $t_1+1$-th sub-state (Case B)
           i. Return $\{(y_\gamma, \pi_\gamma)\}_{\gamma=0}^{t_1}$ appended to $\mathsf{S}_{n-j}((\boldsymbol{\beta}, t_1+1), \varepsilon, \emptyset, \emptyset)$
       (b) Else: increment $t_1$-th sub-state (Case C)
           i. Return $\{(y_\gamma, \pi_\gamma)\}_{\gamma=0}^{t_1-1}$ appended to $\mathsf{S}_{n-j}((\boldsymbol{\beta}, t_1), t', s', \emptyset)$
```

Algorithm 1: The recursive successor circuit S_{n-j+1}.

Recursive verifier circuit $\mathsf{V}_{n-j+1}(\boldsymbol{\beta}, t, s, \mathbf{trans})$

```
Hardwired
    1. an $n$-variate polynomial $f$ of individual degree $d$ over $\mathbb{F}$
    2. the description of a hash function $h \in \mathcal{H}$
Input
    1. a prefix $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{j-1}) \in \mathbb{F}^{j-1}$
    2. an index $t = (t_1, \ldots, t_\ell) \in [d+1]^{\leq(n-j+1)}$
    3. a state $s \in \mathbb{F}^*$ parsed as a set of pairs of prefix sums and proofs $\{(y_0, \pi_0), (y_1, \pi_1), \ldots, \}$
    4. a transcript $\mathbf{trans}$ containing the statement and partial proofs
Output a bit indicating accept (1) or reject (0)
Base Case $\mathsf{V}_0(\boldsymbol{\beta}, \varepsilon, (y, \emptyset), \emptyset)$: Accept if $y = f(\boldsymbol{\beta})$ and reject otherwise
Recursion
    1. If $t = \varepsilon$ return the bit $b \leftarrow \mathcal{V}_{\mathrm{FS}}((y_\varepsilon, f, \boldsymbol{\beta}), \pi_\varepsilon)$ (final state)
    2. For $\gamma \in [t_1 - 1]$: verify all final sub-states
           Reject if $\mathsf{V}_{n-j}((\boldsymbol{\beta}, \gamma), \varepsilon, (y_\gamma, \pi_\gamma), \emptyset)$ rejects
    3. Compute $t_1$-th sub-state $s'$ by truncating $\{(y_\gamma, \pi_\gamma)\}_{\gamma=0}^{t_1-1}$ from $s$
    4. Set $t' := (t_2, \ldots, t_\ell)$ as the index of the sub-state $s'$
    5. If $t_1 = d + 1$
       (a) If $\mathbf{trans} = \emptyset$ then initialise with statement $\mathbf{trans} := (\mathbb{F}, y_0 + y_1, f, \boldsymbol{\beta})$
       (b) Compute updated transcript $\mathbf{trans}'$ by appending $\{y_\gamma\}_{\gamma=0}^{d}$ to $\mathbf{trans}$
       (c) Compute the challenge $\sigma := h(\mathbf{trans})$ and append $\sigma$ to $\mathbf{trans}$
       (d) Reject if $\mathsf{V}_{n-j}((\boldsymbol{\beta}, \sigma), t', s', \mathbf{trans}')$ rejects
       Otherwise $t_1 < d + 1$
       (a) Reject if $\mathsf{V}_{n-j}((\boldsymbol{\beta}, t_1), t', s', \emptyset)$ rejects
    6. Accept
```

Algorithm 2: The recursive verifier circuit V_{n-j+1}.

4.2 The rSVL Instance

The label of the i-th vertex v_i in the proposed rSVL instance is a tuple (t, s_i), where $s_i \in \mathbb{F}^*$ is a state and $t \in [d+1]^{\leq n}$ its index determined by the counter i. To be precise, t is the (address of) i-th vertex in the perfect $(d+2)$-dary tree[12] that the depth-first search *leaves* (i.e. visits for the final time). To map a counter $i \in [T]$ to an index $t \in [d+1]^{\leq n}$, we use a bijective map $\mathit{DFS}(\cdot)$ (Note that this makes sense only if $T = |[d+1]^{\leq n}|$), which we show below.). Its inverse is denoted by $\mathit{DFS}^{-1}(\cdot)$. Thus, the main rSVL line consists of the sequence of labels

$$(0^n, s_1), (0^{n-1}1, s_2), (0^{n-1}, s_3), \ldots, (11, s_{T-2}), (1, s_{T-1}), (\varepsilon, s_T).$$

The successor and verifier circuits (S, V) for the rSVL instance can now be implemented using $(\mathsf{S}_n, \mathsf{V}_n)$ as shown in Algorithm 3. In short, on input a counter i and a label (t, s), the verifier circuit V simply checks if t matches $\mathit{DFS}(i)$ and then invokes the recursive verifier circuit V_n on the state s and the index t. On the other hand, on input a label (t, s), the successor S first ensures that s is indeed the i-th intermediate state by checking its correctness using V. Then it increments the state by calling the recursive successor function S_n. It returns this new state along with an incremented index as the next label.

Efficiency. We argue that S and V are both $\mathsf{poly}(n)$-sized circuits. Observe that for S_{n-j+1} at most one recursion S_{n-j} is active at any given point. The rest of the operations within S_{n-j+1} (viz., append, truncate, compute the hash etc.) are all efficient. Therefore $|\mathsf{S}_{n-j+1}| < |\mathsf{S}_{n-j}| + \mathsf{poly}(n)$ with $|\mathsf{S}_0| = \mathsf{poly}(n)$, and consequently $|\mathsf{S}| = \mathsf{poly}(n)$. A similar argument holds for the verifier circuit V, taking into account the fact that even though there are multiple active recursive calls within V_{n-j+1}, *all but one* are of depth 1. This is true as the verification of the final sub-states in V_{n-j} is carried out by a single call to $\mathcal{V}_{\mathrm{FS}}$.

Parameters. Recall that $T(n-j+1)$, $M(n-j+1)$ and $P(n-j+1)$ denote the number of steps, amount of memory and the final proof size for $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$, respectively.

Since S_{n-j+1} runs S_{n-j} $(d+2)$ times and then takes one step for merging, we have $T(n-j+1) = (d+2)T(n-j) + 1$. By unwinding the recursion with $T(0) = 1$, we get

$$T = T(n,d) = \sum_{j \in [n]} (d+2)^j = |[d+1]^{\leq n}|.$$

For simplicity, assume that $t \in \mathbb{F}^{\leq n}$. From the description of $(\mathsf{S}_0, \mathsf{V}_0)$, it is clear that $M(0) = P(0) = \log_2(p)$ (where p, if you recall, denotes the size of the finite field $\mathbb{F}$). From the description of $(\mathsf{S}_{n-j+1}, \mathsf{V}_{n-j+1})$, we have $M(n-j+1) \leq M(n-j) + (d+1)\log_2(p)$ and $P(n-j+1) = P(n-j) + (d+1)\log_2(p)$. On solving the recursion, we get $M = M(n) \leq (d+1)n\log_2(p)$ and $P = P(n) \leq (d+1)n\log_2(p)$.

4.3 Hardness

In this section we restate and prove Theorem 21.

Theorem 21. *For a parameter n, fix a finite field $\mathbb{F}$ of sufficiently large size p (say $O(2^n)$). Let f be an n-variate polynomial over $\mathbb{F}$ of individual degree at most d. Pick a hash function h uniformly at random from a family $\mathcal{H}$. Let*

$$\mathsf{S} := \mathsf{S}_{f,h} : \{0,1\}^M \to \{0,1\}^M \text{ and } \mathsf{V} := \mathsf{V}_{f,h} : \{0,1\}^M \times [T]$$

[12] A $(d+2)$-ary tree is called *perfect* if all its interior nodes have $(d+2)$ children and all leaves have the same depth.

Verifier circuit $\mathsf{V}_{f,h}(v,i)$

Hardwired

1. an n-variate polynomial f of individual degree d over $\mathbb{F}$
2. the description of a hash function $h \in \mathcal{H}$

Input

1. a label v parsed as $(t,s) \in [d+1]^{\leq n} \times \mathbb{F}^*$ where t is the index and s the state
2. a counter $i \in [T]$

Output a bit indicating accept (1) or reject (0)
Procedure

1. Reject if $t \neq \mathit{DFS}(i)$
2. Return the bit $b \leftarrow \mathsf{V}_n(\varepsilon, t, s, \emptyset)$

Successor circuit $\mathsf{S}_{f,h}(v)$

Hardwired see V
Input a label v parsed as $(t,s) \in [d+1]^{\leq n} \times \mathbb{F}^*$
Output the next label
Procedure

1. Set the index $i := \mathit{DFS}^{-1}(t)$
2. Return v if $\mathsf{V}(v,i)$ rejects (self-loop)
3. Return $(\mathit{DFS}(i+1), \mathsf{S}_n(\varepsilon, t, s, \emptyset))$

Algorithm 3: The verifier V and successor S for the rSVL instance.

be constructed as in Algorithm 3 with $M = M(n,d,p) = (d+1)n\log_2(p)$ *and* $T = T(n,d) = \sum_{j\in[n]}(d+2)^j$*. Given an adversary* A *that solves instances from the* rSVL *family*

$$\{(\mathsf{S},\mathsf{V},(0^n,f(0^n),\emptyset),T)\}_{n\in\mathbb{N}} \tag{6}$$

in polynomial time $T_{\mathsf{A}} = T_{\mathsf{A}}(n)$ *and a non-negligible probability* $\epsilon = \epsilon(n)$*, it is possible to either*

- *find the sum* $\sum_{\boldsymbol{z}\in\{0,1\}^n} f(\boldsymbol{z})$ *in time* $O(T_{\mathsf{A}})$ *with probability* ϵ*, or*
- *break the unambiguous soundness of the non-interactive sumcheck protocol (Assumption 18) with probability* $\epsilon/(d+1)\cdot n$*.*

Corollary 22. *Relative to a random oracle,* #SAT *reduces to* End-of-Metered-Line*.*

Proof. Given a SAT formula Φ over n variables, a claim about the number of satisfying assignments can be expressed as a sumcheck claim over $\mathbb{F}$. The polynomial f is derived from Φ, and the individual degree can be as low as 4. For this, we first transform Φ into a 3SAT-4 formula, a 3CNF where each variable appears in at most 4 clauses. A standard arithmetization yields an appropriate polynomial f_Φ over the field. A reduction from #SAT to rSVL (relative to a random oracle) follows Theorem 21 with $f = f_\Phi$, and Lemma 19 with, for example, $p = |\mathbb{F}| = O(2^n)$ and $Q \in \mathsf{poly}(n)$. The reduction from rSVL to EOML given in Lemma 10 completes the corollary. □

Proof (of Theorem 21). Recall that by Definition 9 the adversary A can solve an rSVL instance in two ways: find either (i) the real sink *or* (ii) a false positive i.e., a pair (v,i) s.t. $\mathsf{V}(v,i)=1$ while $\mathsf{S}^i((0^n,f(0^n),\emptyset)) \neq v$.

Finding a type (i) solution is tantamount to solving the #SAT instance f since the sink of the rSVL instance defined in eq.6 is $(\varepsilon, s_T = (y_T,\pi_T))$ and contains the number of solutions to f in the form of y_T. In the discussion below we rule out solutions of type (ii) under Assumption 18. Taken together, the theorem follows.

Let v be of the form $(t,\{(\widetilde{y}_1,\widetilde{\pi}_1),\ldots,(\widetilde{y}_\ell,\widetilde{\pi}_\ell)\}$ and let

$$v_i = \mathsf{S}^i(0^n,f(0^n),\emptyset) = (t,\{(y_1,\pi_1),\ldots,(y_\ell,\pi_\ell)\})$$

be the correctly computed vertex. Also, let $\{\widetilde{\boldsymbol{\beta}}_1,\ldots,\widetilde{\boldsymbol{\beta}}_\ell\}$ and $\{\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_\ell\}$ denote the associated prefixes. We first establish that there exists at least one index $k^* \in [1,\ell]$ such that the proof $\widetilde{\pi}_{k^*}$ breaks the unambiguous soundness of the non-interactive sumcheck protocol.

Assume for contradiction that A violated neither soundness nor unambiguity in the process of finding the type (ii) solution (v,i) s.t. $\mathsf{V}(v,i)=1$ but $\mathsf{S}^i((0^n,f(0^n),\emptyset)) \neq v$. We show that (v,i) *could not* have been a type (ii) solution. To this end, we establish iteratively from $k=1$ to $k=\ell$ that $(\widetilde{y}_k,\widetilde{\pi}_k) = (y_k,\pi_k)$.

When V_n is invoked by V on $(\varepsilon,t,s,\emptyset)$ it recurses until $(\widetilde{y}_1,\widetilde{\pi}_1)$ is a final sub-state for some $(\mathsf{S}_{n-j},\mathsf{V}_{n-j})$. At this point the validity of $(\widetilde{y}_1,\widetilde{\pi}_1)$ is checked using a single call to $\mathcal{V}_{\mathrm{FS}}$. Recall that it is assumed that neither soundness nor unambiguity was broken. Since $(\widetilde{y}_1,\widetilde{\pi}_1)$ passes verification and $\widetilde{\boldsymbol{\beta}}_1 = \boldsymbol{\beta}_1$, it is guaranteed that the proofs $(\widetilde{y}_1,\widetilde{\pi}_1)$ (y_1,π_1) are for the *same* statement. Therefore $(\widetilde{y}_1,\widetilde{\pi}_1) = (y_1,\pi_1)$ is the correct sub-state.

Assuming that the first $k-1$ sub-states in v are correct, we will infer that the k-th sub-state is also correct. The first step is to show that $\boldsymbol{\beta}_k$ is correctly computed, for which there are two possibilities. If $\boldsymbol{\beta}_k$ corresponds to a challenge then, since $y_1,\ldots,y_{k-1}$ have been validated, $\boldsymbol{\beta}_k$ is also guaranteed to be computed by hashing the same arguments as in the protocol specification; otherwise, $\boldsymbol{\beta}_k$ is computed by a simple increment, which the verifier again checks for. Therefore, by the same argument as for $k=1$, we get $\widetilde{\pi}_k = \pi_k$ and $\widetilde{y}_k = y_k$.

Consequently, all the labels in v are as prescribed by the successor circuit S, contradicting the premise of the lemma that $v \neq \mathsf{S}^i((0^n,f(0^n),\emptyset))$. One therefore concludes that there exists at least one index $k^* \in [1,\ell]$ such that either

- $\mathcal{V}_{\mathrm{FS}}((\widetilde{y}_{k^*}, f, \widetilde{\boldsymbol{\beta}}_{k^*}), \widetilde{\pi}_{k^*}) = 1$, and $\sum_{\boldsymbol{z}\in\{0,1\}^{n-j_{k^*}}} f(\widetilde{\boldsymbol{\beta}}_{k^*}, \boldsymbol{z}) \neq \widetilde{y}_{k^*}$; *or*

- $\mathcal{V}_{\mathrm{FS}}((\widetilde{y}_{k^*}, f, \widetilde{\boldsymbol{\beta}}_{k^*}), \widetilde{\pi}_{k^*}) = 1$, and $\widetilde{\pi}_{k^*} \neq \mathcal{P}_{\mathrm{FS}}(\widetilde{y}_{k^*}, f, \widetilde{\boldsymbol{\beta}}_{k^*})$ where $\mathcal{P}_{\mathrm{FS}}$ is the prescribed prover.

Here, the first case corresponds to the setting that there is an accepting proof for an incorrect statement, while the second case corresponds to an accepting proof different from that output by the prescribed prover.

Given an adversary A that finds such a vertex i with probability ϵ we can build an adversary A' that, depending on the case we're in, breaks soundness or unambiguity. The strategy for A' in either case is identical and is described below:

1. Run A on the rSVL instance $(\mathsf{S}, \mathsf{V}, (0^n, f(0^n), \emptyset), T)$.
2. Let the output returned by A be of the form $(t, \{(\widetilde{y}_1, \widetilde{\pi}_1), \ldots, (\widetilde{y}_\ell, \widetilde{\pi}_\ell)\})$.
3. Sample a random index $k^* \overset{\$}{\leftarrow} [1, \ell]$.
4. Return $\left(\left(f, \widetilde{\boldsymbol{\beta}}_{k^*}, \widetilde{y}_{k^*}\right), \widetilde{\pi}_{k^*}\right)$, where $\widetilde{\boldsymbol{\beta}}_{k^*}$ is the prefix associated to $(\widetilde{y}_{k^*}, \widetilde{\pi}_{k^*})$.

A' runs for a time that is roughly the same as that of A (i.e., T_{A}). The analysis for A''s success probability is simple and we describe it only for the first case (as the other case is identical). Informally, since there exists an index k^* that breaks soundness, A' succeeds as long as it is able to guess this index correctly. Formally,

$$\begin{aligned}\Pr\left[\mathsf{A}' \text{ succeeds}\right] &\geq \Pr\left[\mathsf{A} \text{ succeeds}\right] \cdot \Pr\left[k^* \text{ is a correct guess} | \mathsf{A}' \text{ succeeds}\right] \\ &\geq \epsilon \cdot \frac{1}{(d+1)\cdot n}\end{aligned}$$

The first inequality follows from the fact that if A succeeds, there is at least one index k^* such that $\mathcal{V}_{\mathrm{FS}}((\widetilde{y}_{k^*}, f, \widetilde{\boldsymbol{\beta}}_{k^*}), \widetilde{\pi}_{k^*}) = 1$, and $\sum_{\boldsymbol{z}\in\{0,1\}^{n-j_{k^*}}} f(\widetilde{\boldsymbol{\beta}}_{k^*}, \boldsymbol{z}) \neq \widetilde{y}_{k^*}$. The second inequality follows from the fact that a label contains at most $M(n) \leq (d+1)\cdot n$ tuples of the form (y, π). $\square$

References

[1] Abbot, T., Kane, D., and Valiant, P. On algorithms for Nash equilibria. Unpublished manuscript, 2004. `http://web.mit.edu/tabbott/Public/final.pdf`. (Cited on pages 1, 9, 10, 12 and 13.)

[2] Angel, O., Bubeck, S., Peres, Y., and Wei, F. Local max-cut in smoothed polynomial time. In *49th Annual ACM Symposium on Theory of Computing* (Montreal, QC, Canada, June 19–23, 2017), H. Hatami, P. McKenzie, and V. King, Eds., ACM Press, pp. 429–437. (Cited on page 11.)

[3] Babichenko, Y. Query complexity of approximate Nash equilibria. *J. ACM 63*, 4 (2016), 36:1–36:24. (Cited on page 1.)

[4] Bennett, C. H. Time/space trade-offs for reversible computation. *SIAM J. Comput. 18*, 4 (1989), 766–776. (Cited on page 13.)

[5] Bitansky, N., Canetti, R., Chiesa, A., and Tromer, E. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012* (2012), pp. 326–349. (Cited on page 4.)

[6] Bitansky, N., Paneth, O., and Rosen, A. On the cryptographic hardness of finding a Nash equilibrium. In *56th Annual Symposium on Foundations of Computer Science* (Berkeley, CA, USA, Oct. 17–20, 2015), V. Guruswami, Ed., IEEE Computer Society Press, pp. 1480–1498. (Cited on pages 1, 9, 10, 12 and 13.)

[7] Boodaghians, S., Kulkarni, R., and Mehta, R. Nash equilibrium in smoothed polynomial time for network coordination games. *CoRR abs/1809.02280* (2018). (Cited on page 11.)

[8] Brakerski, Z. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology – CRYPTO 2012* (Santa Barbara, CA, USA, Aug. 19–23, 2012), R. Safavi-Naini and R. Canetti, Eds., vol. 7417 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 868–886. (Cited on page 2.)

[9] Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS 2012: 3rd Innovations in Theoretical Computer Science* (Cambridge, MA, USA, Jan. 8–10, 2012), S. Goldwasser, Ed., Association for Computing Machinery, pp. 309–325. (Cited on page 2.)

[10] Brakerski, Z., and Vaikuntanathan, V. Efficient fully homomorphic encryption from (standard) LWE. In *52nd Annual Symposium on Foundations of Computer Science* (Palm Springs, CA, USA, Oct. 22–25, 2011), R. Ostrovsky, Ed., IEEE Computer Society Press, pp. 97–106. (Cited on page 2.)

[11] Brakerski, Z., and Vaikuntanathan, V. Lattice-based FHE as secure as PKE. In *ITCS 2014: 5th Conference on Innovations in Theoretical Computer Science* (Princeton, NJ, USA, Jan. 12–14, 2014), M. Naor, Ed., Association for Computing Machinery, pp. 1–12. (Cited on page 2.)

[12] Buhrman, H., Fortnow, L., Koucký, M., Rogers, J. D., and Vereshchagin, N. Does the polynomial hierarchy collapse if onto functions are invertible? *Theory of Computing Systems 46*, 1 (Dec 2008), 143. (Cited on page 11.)

[13] Buresh-Oppenheim, J. On the TFNP complexity of factoring. Unpublished, http://www.cs.toronto.edu/~bureshop/factor.pdf, 2006. (Cited on page 11.)

[14] Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G. N., and Rothblum, R. D. Fiat-shamir from simpler assumptions. Cryptology ePrint Archive, Report 2018/1004, 2018. https://eprint.iacr.org/2018/1004. (Cited on pages 2, 3, 37, 39, 40, 41 and 42.)

[15] Canetti, R., Chen, Y., and Reyzin, L. On the correlation intractability of obfuscated pseudorandom functions. In *TCC 2016-A: 13th Theory of Cryptography Conference, Part I* (Tel Aviv, Israel, Jan. 10–13, 2016), E. Kushilevitz and T. Malkin, Eds., vol. 9562 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 389–415. (Cited on page 37.)

[16] Canetti, R., Chen, Y., Reyzin, L., and Rothblum, R. D. 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* (2018), pp. 91–122. (Cited on pages 2, 37, 38 and 40.)

[17] Canetti, R., Lombardi, A., and Wichs, D. Non-interactive zero knowledge and correlation intractability from circular-secure fhe. Cryptology ePrint Archive, Report 2018/1248, 2018. https://eprint.iacr.org/2018/1248. (Cited on pages 2 and 37.)

[18] Cantor, D. G., and Zassenhaus, H. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation 36* (1981), 587–592. (Not cited.)

[19] Chen, X., Deng, X., and Teng, S. Settling the complexity of computing two-player Nash equilibria. *J. ACM 56*, 3 (2009). (Cited on pages 1 and 9.)

[20] Chung, F., Diaconis, P., and Graham, R. Combinatorics for the east model. *Advances in Applied Mathematics 27*, 1 (2001), 192–206. (Cited on page 13.)

[21] Daskalakis, C., Goldberg, P. W., and Papadimitriou, C. H. The complexity of computing a Nash equilibrium. *SIAM J. Comput. 39*, 1 (2009), 195–259. (Cited on pages 1 and 9.)

[22] Daskalakis, C., and Papadimitriou, C. H. Continuous local search. In *22nd Annual ACM-SIAM Symposium on Discrete Algorithms* (San Francisco, CA, USA, Jan. 23–25, 2011), D. Randall, Ed., ACM-SIAM, pp. 790–804. (Cited on pages 9 and 12.)

[23] Daskalakis, C., Tzamos, C., and Zampetakis, M. A converse to Banach's fixed point theorem and its CLS-completeness. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018* (2018), pp. 44–50. (Cited on page 9.)

[24] Deng, X., Edmonds, J. R., Feng, Z., Liu, Z., Qi, Q., and Xu, Z. Understanding PPA-completeness. In *31st Conference on Computational Complexity (CCC 2016)* (Dagstuhl, Germany, 2016), R. Raz, Ed., vol. 50 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 23:1–23:25. (Cited on page 9.)

[25] Fiat, A., and Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO'86* (Santa Barbara, CA, USA, Aug. 1987), A. M. Odlyzko, Ed., vol. 263 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 186–194. (Cited on page 1.)

[26] Filos-Ratsikas, A., and Goldberg, P. W. Consensus halving is PPA-complete. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018* (2018), pp. 51–64. (Cited on page 9.)

[27] Garg, S., Pandey, O., and Srinivasan, A. Revisiting the cryptographic hardness of finding a Nash equilibrium. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II* (2016), pp. 579–604. (Cited on pages 1 and 10.)

[28] Gentry, C., Sahai, A., and Waters, B. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology – CRYPTO 2013, Part I* (Santa Barbara, CA, USA, Aug. 18–22, 2013), R. Canetti and J. A. Garay, Eds., vol. 8042 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 75–92. (Cited on page 2.)

[29] Goldreich, O. Candidate one-way functions based on expander graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman.* 2011, pp. 76–87. (Cited on page 3.)

[30] Goldreich, O., and Rothblum, G. N. Worst-case to average-case reductions for subclasses of P. *Electronic Colloquium on Computational Complexity (ECCC) 24* (2017), 130. (Cited on page 3.)

[31] Goldwasser, S., and Kalai, Y. T. On the (in)security of the Fiat-Shamir paradigm. In *44th Annual Symposium on Foundations of Computer Science* (Cambridge, MA, USA, Oct. 11–14, 2003), IEEE Computer Society Press, pp. 102–115. (Cited on page 2.)

[32] Goldwasser, S., Micali, S., and Rackoff, C. The knowledge complexity of interactive proof systems. *SIAM J. Comput. 18*, 1 (1989), 186–208. (Cited on page 15.)

[33] Hirsch, M. D., Papadimitriou, C. H., and Vavasis, S. A. Exponential lower bounds for finding Brouwer fix points. *J. Complexity 5*, 4 (1989), 379–416. (Cited on page 1.)

[34] Hubáček, P., Naor, M., and Yogev, E. The journey from NP to TFNP hardness. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA* (2017), pp. 60:1–60:21. (Cited on page 11.)

[35] Hubáček, P., and Yogev, E. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19* (2017), pp. 1352–1371. (Cited on pages 1, 2, 9, 10, 12, 13 and 14.)

[36] Jeřábek, E. Integer factoring and modular square roots. *J. Comput. Syst. Sci. 82*, 2 (2016), 380–394. (Cited on pages 9, 10 and 11.)

[37] Johnson, D. S., Papadimitriou, C. H., and Yannakakis, M. How easy is local search? *Journal of Computer and System Sciences 37*, 1 (1988), 79 – 100. (Cited on pages 9 and 11.)

[38] Kalai, Y. T., Rothblum, G. N., and Rothblum, R. D. From obfuscation to the security of fiat-shamir for proofs. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II* (2017), pp. 224–251. (Cited on pages 2 and 37.)

[39] Kintali, S., Poplawski, L., Rajaraman, R., Sundaram, R., and Teng, S. Reducibility among fractional stability problems. *SIAM Journal on Computing 42*, 6 (2013), 2063–2113. (Cited on page 9.)

[40] Komargodski, I., Naor, M., and Yogev, E. White-box vs. black-box complexity of search problems: Ramsey and graph property testing. In *58th Annual Symposium on Foundations of Computer Science* (2017), IEEE Computer Society Press, pp. 622–632. (Cited on page 11.)

[41] Komargodski, I., and Segev, G. From minicrypt to obfustopia via private-key functional encryption. In *Advances in Cryptology – EUROCRYPT 2017, Part I* (Paris, France, Apr. 30 – May 4, 2017), J. Coron and J. B. Nielsen, Eds., vol. 10210 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 122–151. (Cited on pages 1 and 11.)

[42] Lipton, R. J. New directions in testing. In *Distributed Computing And Cryptography, Proceedings of a DIMACS Workshop, Princeton, New Jersey, USA, October 4-6, 1989* (1989), pp. 191–202. (Cited on page 3.)

[43] Lund, C., Fortnow, L., Karloff, H., and Nisan, N. Algebraic methods for interactive proof systems. *J. ACM 39*, 4 (Oct. 1992), 859–868. (Cited on pages 1, 5, 14, 16 and 17.)

[44] Mahmoody, M., and Xiao, D. On the power of randomized reductions and the checkability of SAT. In *2010 IEEE 25th Annual Conference on Computational Complexity* (June 2010), pp. 64–75. (Cited on page 11.)

[45] Megiddo, N., and Papadimitriou, C. H. On total functions, existence theorems and computational complexity. *Theor. Comput. Sci. 81*, 2 (1991), 317–324. (Cited on pages 9 and 11.)

[46] Micali, S. Computationally sound proofs. *SIAM Journal on Computing 30*, 4 (2000), 1253–1298. Preliminary version appeared in FOCS '94. (Cited on page 4.)

[47] Papadimitriou, C. H. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci. 48*, 3 (1994), 498–532. (Cited on pages 1, 9, 10 and 11.)

[48] Peikert, C., and Shiehian, S. Noninteractive zero knowledge for np from (plain) learning with errors. Cryptology ePrint Archive, Report 2019/158, 2019. https://eprint.iacr.org/2019/158. (Cited on pages 2 and 37.)

[49] Pietrzak, K. Simple Verifiable Delay Functions. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)* (Dagstuhl, Germany, 2018), A. Blum, Ed., vol. 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 60:1–60:15. (Cited on page 2.)

[50] Reingold, O., Rothblum, G. N., and Rothblum, R. D. Constant-round interactive proofs for delegating computation. In *48th Annual ACM Symposium on Theory of Computing* (Cambridge, MA, USA, June 18–21, 2016), D. Wichs and Y. Mansour, Eds., ACM Press, pp. 49–62. (Cited on pages 3, 15 and 19.)

[51] Rosen, A., Segev, G., and Shahaf, I. Can PPAD hardness be based on standard cryptographic assumptions? In *TCC 2017: 15th Theory of Cryptography Conference, Part II* (Baltimore, MD, USA, Nov. 12–15, 2017), Y. Kalai and L. Reyzin, Eds., vol. 10678 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 747–776. (Cited on page 11.)

[52] Savani, R., and von Stengel, B. Exponentially many steps for finding a Nash equilibrium in a bimatrix game. In *45th Annual Symposium on Foundations of Computer Science* (Rome, Italy, Oct. 17–19, 2004), IEEE Computer Society Press, pp. 258–267. (Cited on page 1.)

[53] Sotiraki, K., Zampetakis, M., and Zirdelis, G. PPP-completeness with connections to cryptography. Cryptology ePrint Archive, Report 2018/778, 2018. https://eprint.iacr.org/2018/778. (Cited on page 10.)

[54] Tovey, C. A. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics 8*, 1 (1984), 85–89. (Cited on page 5.)

[55] Valiant, P. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *TCC 2008: 5th Theory of Cryptography Conference* (San Francisco, CA, USA, Mar. 19–21, 2008), R. Canetti, Ed., vol. 4948 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, pp. 1–18. (Cited on pages 3, 4, 5 and 6.)

A Instantiating Fiat-Shamir

There has been exciting recent progress on constructing hash functions for which the Fiat-Shamir transformation (applied to certain protocols) is provably sound [15, 38, 16, 14, 16, 17, 48]. In this section, we discuss the applicability of these results to our setting. We observe that the results in [14] can be extended to our setting, yielding a hash family for which, under quasi-polynomial variants of the (strong) assumptions made in that work, the Fiat-Shamir transform is sound when applied to the sumcheck protocol over polylog variables.

Our starting point is the application of the Fiat-Shamir transform in [14] to construct publicly verifiable succinct arguments (pv-SNARGs). We note that while the assumptions required when the Fiat-Shamir transform is used to construct Non-interactive Zero Knowledge (NIZK) proofs are significantly more standard (plain LWE in very recent work [48]), these results hold limited relevance to our setting. This is because the time required to evaluate those hash functions is as large as the time needed to compute the (honest) prover's messages in the interactive proof protocol. In our context, this means that evaluating the hash function would take super-polynomial time.

The subsequent text is (to a large extent) adapted from [14], with several (important) changes that are needed to obtain results in our setting. We highlight changes in the assumptions and theorem statements. For a comprehensive discussion, we refer the reader to [14].

A.1 Definitions

We begin with definitions, starting with the notion of *correlation intractability*. We often refer to quasi-polynomial functions, and by quasi-polynomial, we mean $2^{\mathsf{polylog}(\lambda)}$.

Definition 23 (Correlation Intractability)**.** For a given relation ensemble $R = \{R_\lambda \subseteq X_\lambda \times Y_\lambda\}$, a hash family $\mathcal{H} = \{h_\lambda : \mathcal{I}_\lambda \times X_\lambda \to Y_\lambda\}_{\lambda \in \mathbb{Z}^+}$ is said to be R-correlation intractable against quasi-polynomial adversaries if for every quasi-polynomial-size $\mathsf{A} = \{\mathsf{A}_\lambda\}$, there exists an ϵ

$$\Pr\left[I \leftarrow \mathcal{I}_\lambda, x \leftarrow \mathsf{A}_\lambda(I) \ : \ (x, h_\lambda(I, x)) \in R_\lambda\right] \leq \epsilon$$

For the context of our work, we will need ϵ to be a negligible function.

We want to guarantee such a property in the standard model. But even in the random oracle model, this only makes sense for relations R that are *sparse*, which we formalize below.

Definition 24 (Sparsity)**.** For any relation ensemble $R = \{R_\lambda \subseteq X_\lambda \times Y_\lambda\}$, we say that R is $\rho(\cdot)$-sparse if for all $\lambda \in \mathbb{Z}^+$ and any $x \in X_\lambda$,

$$\Pr\left[y \leftarrow Y_\lambda : (x, y) \in R_\lambda\right] \leq \rho(\lambda).$$

Remark 25. When we talk about correlation intractability with respect to quasi-polynomial time adversaries, it is not sufficient for ρ to be negligible. We will in fact require ρ to be smaller than any inverse quasi-polynomial function.

We will need the ability to *sample* from the relation R. In fact, it is sufficient to be able to *approximately sample* from the relation. We begin by defining what it means for a distribution to be approximated.

Definition 26. A distribution P multiplicatively ϵ-approximates a distribution Q if for all outcomes ω, it holds that

$$\Pr\left[x \leftarrow P : x = w\right] \geq \epsilon \cdot \Pr\left[x \leftarrow Q : x = w\right].$$

We proceed to formalize the notion of *approximately sampling from a relation R.*

Definition 27 (Approximate Sampleability of Relations)**.** A relation ensemble $R = \{R_\lambda \subseteq X_\lambda \times Y_\lambda\}$ is non-uniformly ϵ-approximately sampleable if there is a circuit ensemble $\{\mathsf{Samp}_\lambda\}$ such that for every $x \in X_\lambda$, the distribution $\mathsf{Samp}_\lambda(x)$ multiplicatively ϵ-approximates the uniform distribution on the (by assumption, non-empty) set $\{y' \in Y_\lambda : (x, y') \in R_\lambda\}$.

We say that R is (non-uniformly) efficiently approximately sampleable if it is non-uniformly ϵ-approximately sampleable for some $\epsilon \geq \frac{1}{\mathsf{poly}(\lambda)}$.

We will focus on relations where the sampling function Samp runs in quasi-polynomial time.

We conclude the definitional discussion of hash families by describing a specific construction given in [16]. We refer to this as the CCRR hash family.

Definition 28 (CCRR Hash Family)**.** Let $\mathcal{E} = \{(\mathcal{K}_\lambda, \mathsf{Enc}_\lambda, \mathsf{Dec}_\lambda)\}_\lambda$ be a secret key encryption scheme with message space $\{0,1\}^\ell$ for $\ell = \ell(\lambda)$. The CCRR hash family associated to this encryption scheme, denoted $\mathcal{H}^{\mathcal{E}}_{\mathsf{CCRR}}$ is

$$\mathcal{H}^{\mathcal{E}}_{\mathsf{CCRR}} = \left\{ h_\lambda : \mathcal{I}_\lambda \times \mathcal{K}_\lambda \to \{0,1\}^\ell \right\}_\lambda$$

where

$$h_\lambda(C, x) \coloneqq \mathsf{Dec}_\lambda(x, C)$$

and the distribution $\mathcal{I}_\lambda$ is a random ciphertext C obtained by sampling $K \leftarrow \mathcal{K}_\lambda$ along with $M \leftarrow \{0,1\}^\ell$ and defining $C \coloneqq \mathsf{Enc}_\lambda(K, M)$. That is, the hashing key is a ciphertext, messages are interpreted as decryption keys, and hashing is performed by decrypting the hash key / ciphertexts under the message / decryption key.

Turning our attention to encryption schemes, we first define what it means for an encryption scheme to have universal ciphertexts.

Definition 29. A secret-key encryption scheme $\mathcal{E} = \{(\mathcal{K}_\lambda, \mathsf{Enc}_\lambda, \mathsf{Dec}_\lambda)\}_\lambda$ with message space $\mathcal{M}_\lambda$ has universal ciphertexts if for any secret key $k \in \mathcal{K}_\lambda$, the distribution $\mathsf{Enc}_\lambda(k, \mathcal{U}_{\mathcal{M}_\lambda})$ multiplicatively $\frac{1}{\mathsf{poly}(\lambda)}$-approximates the distribution $\mathsf{Enc}_\lambda(\mathcal{K}_\lambda, \mathcal{U}_{\mathcal{M}_\lambda})$, where $\mathcal{U}_{\mathcal{M}_\lambda}$ denotes the uniform distribution over $\mathcal{M}_\lambda$.

Moving to security, we will define the security of certain primitives with respect to quasi-polynomial adversaries, parameterized by a class of functions $\Delta_{\mathsf{quasi-poly}}$. For every quasi-polynomial adversary, there exists a function $\delta \in \Delta_{\mathsf{quasi-poly}}$ such that the success probability of the adversary is bounded by δ. In the context of our work, we will consider $\Delta_{\mathsf{quasi-poly}}$ to contain functions of the form $\frac{\mathsf{quasi-poly}(\kappa)}{2^\kappa}$, where κ will denote the key length.

We now define Key-Dependent Message (KDM) security for a homomorphic encryption scheme. The security definition for the regular encryption scheme follows in a similar manner, with the evaluation key being empty.

Definition 30 (Key-Dependent Message (KDM)-security)**.** Let $\mathcal{E} = \{(\mathcal{K}_\lambda, \mathsf{Enc}_\lambda, \mathsf{Dec}_\lambda, \mathsf{Eval}_\lambda)\}_\lambda$ be a secret-key fully homomorphic bit-encryption scheme with message space $\mathcal{M}_\lambda$, and let $f = \{f_\lambda : \mathcal{K}_\lambda \to \mathcal{M}_\lambda\}$ be a (potentially probabilistic) function. $\mathcal{E}$ is said to be $\Delta_{\mathsf{quasi-poly}}$-immune to key recovery by an f-KDM query against quasi-polynomial adversaries if for each quasi-polynomial-sized $\mathsf{A} = \{\mathsf{A}_\lambda\}$, there exists a $\delta \in \Delta_{\mathsf{quasi-poly}}$ such that:

$$\Pr\left[\begin{array}{c} (K, E) \leftarrow \mathcal{K}_\lambda \\ (M_1, \cdots, M_\ell) \leftarrow f_\lambda(K) \\ \{C_i \leftarrow \mathsf{Enc}_\lambda(K, M_i)\}_{i \in [\ell]} \end{array} : \mathsf{A}_\lambda(E, C_1, \cdots, C_\ell) = K \right] \leq \delta(\lambda)$$

Throughout this paper we will abbreviate the above by saying that $\mathcal{E}$ is f-KDM $\Delta_{\mathsf{quasi-poly}}$-secure against quasi-polynomial adversaries. If $\mathcal{F}$ is a set of functions then we say that $\mathcal{E}$ is $\mathcal{F}$-KDM $\Delta_{\mathsf{quasi-poly}}$-secure against quasi-polynomial adversaries if $\mathcal{E}$ is f-KDM $\Delta_{\mathsf{quasi-poly}}$-secure against quasi-polynomial adversaries for all $f \in \mathcal{F}$.

This is a modification of the f-KDM security used in [14]. Here, an adversary is allowed to run in quasi-polynomial time. Looking ahead, $\mathcal{F}$ will be the collection of all functions, with ℓ-bits of output, that can be computed in quasi-polynomial time.

The construction and its security hinge on the Regev encryption scheme, which we define below.

Definition 31 (Secret-Key Regev Encryption)**.** For any positive integer $q = q(\lambda) \leq 2^{\mathsf{poly}(\lambda)}$, $n' = n'(\lambda) \leq \mathsf{poly}(\lambda)$, and any $\mathsf{poly}(\lambda)$-time sampleable distribution ensembles $\chi_{\mathsf{sec}} = \{\chi_{\mathsf{sec}}(\lambda)\}$ and $\chi_{\mathsf{err}} = \{\chi_{\mathsf{err}}(\lambda)\}$ over $\mathbb{Z}_{q(\lambda)}$, we define $\mathbf{Regev}_{n',q,\chi_{\mathsf{sec}},\chi_{\mathsf{err}}}$ to be the secret key encryption scheme $\{(\mathcal{K}_\lambda, \mathsf{Enc}_\lambda, \mathsf{Dec}_\lambda)\}$ where:

- $\mathcal{K}_\lambda$ is the distribution $\chi_{\mathsf{sec}}^{n'}$.
- $\mathsf{Enc}_\lambda : \mathbb{Z}_q^{n'} \times \{0,1\} \to \mathbb{Z}_q^{n'} \times \mathbb{Z}_q$ is defined so that $\mathsf{Enc}_\lambda(\mathbf{s}, m)$ is obtained by sampling a uniformly random vector $\mathbf{a} \leftarrow \mathbb{Z}_q^{n'}$, sampling $e \leftarrow \chi_{\mathsf{err}}(\lambda)$, and outputting $\left(\mathbf{a}, \mathbf{s}^t \cdot \mathbf{a} + m \cdot \left\lceil \frac{q}{2} \right\rfloor + e\right)$.
- $\mathsf{Dec}_\lambda : \mathbb{Z}_q^{n'} \times \left(\mathbb{Z}_q^{n'} \times \mathbb{Z}_q\right) \to \{0,1\}$ is defined so that $\mathsf{Dec}_\lambda(\mathbf{s}, (\mathbf{a}, b))$ is the bit m for which $b - \mathbf{s}^t \cdot \mathbf{a}$ is closer to $m \cdot \left\lceil \frac{q}{2} \right\rfloor$ than to $(1-m) \cdot \left\lceil \frac{q}{2} \right\rfloor$.

A pair $(\mathbf{a}, b) \in \mathbb{Z}_q^{n'} \times \mathbb{Z}_q$ is a Regev encryption of $m \in \{0,1\}$ under $\mathbf{s} \in \mathbb{Z}_q^{n'}$ with B-bounded noise if $b - \mathbf{s}^t \cdot \mathbf{a} - m \cdot \left\lceil \frac{q}{2} \right\rfloor$ is in the interval $[-B, B)$

We now define a homomorphic encryption scheme that is sufficient for our application. In some sense, these are FHE schemes that have implicit in them a (low-noise) secret-key Regev ciphertext.

Definition 32 (Regev-Extractable Secret-Key Homomorphic Encryption)**.** A secret-key fully homomorphic bit-encryption scheme $\{(\mathcal{K}_\lambda, \mathsf{Enc}_\lambda, \mathsf{Dec}_\lambda, \mathsf{Eval}_\lambda)\}$ is $\mathbf{Regev}_{n',q,\chi_{\mathsf{sec}}}$-extractable with $B(\lambda)$-bounded noise if it satisfies the following structural properties.

- The distribution of $\mathbf{s}$ when sampling $(\mathbf{s}, \mathsf{ek}) \leftarrow \mathcal{K}_\lambda$ is $\chi_{\mathsf{sec}}^{n'}$ where χ_{sec} is a distribution over $\mathbb{Z}_q$.
- There is a $\mathsf{poly}(\lambda)$-time evaluable function $\mathsf{Extract} = \{\mathsf{Extract}_\lambda\}$ such that
  - For any λ, any $\mathbf{s} \in \chi_{\mathsf{sec}}^{n'}$, and any $m \in \{0,1\}$, it holds that $\mathsf{Extract}_\lambda(\mathsf{Enc}_\lambda(\mathbf{s}, m))$ is a Regev encryption $(\mathbf{a}, b)$ of m under $\mathbf{s}$ with B-bounded noise, and where $\mathbf{a}$ is uniformly random in $\mathbb{Z}_q^{n'}$.
  - For any $m_1, \cdots, m_{n'} \in \{0,1\}$, any circuit $C : \{0,1\}^{n'} \to \{0,1\}$, and any $(\mathbf{s}, \mathsf{ek}) \in \mathcal{K}_\lambda$, it holds with probability 1 that
  $$\mathsf{Extract}_\lambda\left(\mathsf{Eval}_\lambda\left(\mathsf{ek}, C, \mathsf{Enc}_\lambda(\mathbf{s}, m_1), \cdots, \mathsf{Enc}_\lambda(\mathbf{s}, m_{n'})\right)\right)$$
  is a Regev encryption $(\mathbf{a}, b)$ of $C(m_1, \cdots, m_{n'})$ under $\mathbf{s}$ with B-bounded noise.

For our applications, we require the Regev extractable schemes to have the following security property.

Definition 33. Let $\mathcal{E}$ be a FHE scheme with key distributions $\{\mathcal{K}_\lambda\}$. For $(\mathsf{sk}, \mathsf{ek}) \in \mathcal{K}_\lambda$, let $\mathsf{bin}(\mathsf{sk})$ denote the binary representation of sk, and let $\kappa = \kappa(\lambda)$ denote the length of such a representation. For any $\ell = \ell(\lambda)$, $\mathcal{E}$ is said to be [ℓ-bit CPA + circular] $\Delta_{\mathsf{quasi-poly}}$-optimally secure with a κ-bit key against quasi-polynomial time adversaries (abbreviated quasi-poly $(\kappa, \ell, \Delta_{\mathsf{quasi-poly}})$-CCO secure) if for every ensemble of ℓ-bit messages $\{m_\lambda\}$, $\mathcal{E}$ is f-KDM $\Delta_{\mathsf{quasi-poly}}$-secure against quasi-polynomial adversaries for the "augmented bit-by-bit circular security function"

$$f = \left\{f_\lambda : \mathcal{K}_\lambda \to \{0,1\}^{\ell+\kappa}\right\}$$

$$f_\lambda(k) = m_\lambda \circ \mathsf{bin}(k) \qquad (\circ \text{ denotes concatenation})$$

We emphasize that we require security against quasi-polynomial time adversaries.

A.2 Fiat-Shamir for the Sumcheck Protocol

Now that we have the relevant definitions, we show these help us achieve the desired result. As indicated while defining the various primitives, we will need to work with adversaries that have quasi-polynomial running time.

A central component of this construction is the following theorem from [14, 16] for the construction of correlation intractable hash functions. We restate the theorem below, with remarks regarding its differences for our setting.

Theorem 34 ([16])**.** *Let $\mathcal{E} = \{(\mathcal{K}_\lambda, \mathsf{Enc}_\lambda, \mathsf{Dec}_\lambda)\}_\lambda$ be a secret key encryption scheme with universal ciphertexts, message space $\{0,1\}^\ell$, and key distribution $\mathcal{K}_\lambda$ equal to the uniform distribution on $\{0,1\}^\kappa$ for some $\kappa = \kappa(\lambda)$. If $\mathcal{E}$ is $\mathcal{F}$-KDM $\Delta_{\mathsf{quasi-poly}}$-secure against quasi-polynomial adversaries and R is a ρ-sparse relation that is $\lambda^{-O(1)}$-approximately $\mathcal{F}$-sampleable, then for every quasi-polynomial time adversary, there is a $\delta \in \Delta_{\mathsf{quasi-poly}}$ such that $\mathcal{H}^{\mathcal{E}}_{\mathsf{CCRR}}$ is R-correlation intractable with $\epsilon := \frac{\delta(\lambda)\cdot\rho(\lambda)}{2^{-\kappa}} \cdot \lambda^{O(1)}$.*

Remark 35. We have defined our underlying primitives to require both correlation intractability, and KDM security, against adversaries running in quasi-polynomial time.

We omit the proof here, as it follows in an identical manner to the proof of the corresponding theorem in [14].

To instantiate the above theorem, we're going to have to require circular (CCO) security against quasi-polynomial time adversaries for fully homomorphic encryption scheme, which is different from the assumption stated in [14].

Assumption 36 (Quasi-poly Dream FHE)**.** *For some $n', q, \chi_{\mathsf{sec}}$, there exists a quasi-poly $(\kappa, \ell, \Delta_{\mathsf{quasi-poly}})$-CCO secure secret key FHE scheme that is $\mathbf{Regev}_{n',q,\chi_{\mathsf{sec}}}$-extractable with B-bounded noise for $\kappa = \lambda^{\Theta(1)}, \ell = \lambda^{\Omega(1)}, B \leq q/\widetilde{\Omega}(\lambda)$ and $\chi_{\mathsf{sec}}^{n'}$ that is sampleable in $\widetilde{O}(n')$ time using $\kappa + O(\log \lambda)$ random bits.*

The following claim states that if the assumption is true, then the Regev encryption scheme has universal ciphertexts and satisfies KDM security. As stated earlier, another difference in our assumption is that the class of function $\mathcal{F}^\ell$ contains all functions, with output size ℓ, computable in quasi-polynomial time.

Claim 37. *If Assumption 36 is true, then there exist parameters $n' = n'(\lambda), q = q(\lambda)$, and $\chi_{\mathsf{sec}} = \chi_{\mathsf{sec}}(\lambda)$ such that for some $\ell = \lambda^{\Omega(1)}$, $\mathbf{Regev}_{n',q,\chi_{\mathsf{sec}},\chi_{\mathsf{err}}}$ is $\mathcal{F}^\ell$-KDM $\Delta_{\mathsf{quasi-poly}}$-secure, where $\mathcal{F}^\ell$ is the class of functions with ℓ output bits computable in $2^{\mathsf{polylog}(\lambda)}$ time, where χ_{err} is the uniform error distribution on $[-q/4, q/4)$, and where κ is the length of the binary representation of an element of $\chi_{\mathsf{sec}}^{n'}$.*

Proof Sketch. Here we briefly sketch the main difference in the proof that necessitates the different assumption from the underlying FHE scheme. For the proof details, we refer the reader to the original proof in [14].

Let $\mathsf{A}_{\mathrm{KDM}}$ be the adversary that breaks the KDM security of the Regev encryption scheme. We will use $\mathsf{A}_{\mathrm{KDM}}$ to build an adversary $\mathsf{A}_{\mathrm{CCO}}$ that breaks Assumption 36. $\mathsf{A}_{\mathrm{CCO}}$ is given as input the challenge $(\mathsf{ek}, c_1, \cdots, c_{\ell+\kappa})$, where $c_{\ell+1}, \cdots, c_{\ell+\kappa}$ is the encryption of the secret key sk.

On initialization, $\mathsf{A}_{\mathrm{KDM}}$ queries the challenger with a function $\widehat{f} \in \mathcal{F}^\ell$ of its choice, and expects an encryption of $\widehat{f}(\mathsf{sk})$. In order to facilitate this, $\mathsf{A}_{\mathrm{CCO}}$ uses the homomorphic evaluation function Eval to homomorphically compute $\widehat{f}(\mathsf{sk})$ as $\mathsf{Eval}\left(\mathsf{ek}, \widehat{f}, c_{\ell+1}, \cdots, c_{\ell+\kappa}\right)$. But given that $\mathcal{F}^\ell$ consists of all functions computable in quasi-polynomial time, the above Eval computation may take quasi-polynomial time. The rest of the reduction remains unchanged.

Therefore, the reduction requires $\mathsf{A}_{\mathrm{CCO}}$ to perform a quasi-polynomial computation, which in turn necessitates that Assumption 36 be secure against quasi-polynomial adversaries. □

We can now state the main theorem. Namely, under circular security assumptions against quasi-polynomial time adversaries for the FHE scheme, the Fiat-Shamir transform when applied to the Sumcheck protocol is an adaptively sound argument.

Theorem 38. *If Assumption 36 is true, then the non-interactive sumcheck protocol in Figure 4, instantiated with the hash family $\mathcal{H}^{\mathcal{E}}_{\mathsf{CCRR}}$, is adaptively unambiguously sound for the language L_{SC} (see section 1.2), with formulas on $\mathsf{polylog}(\lambda)$ variables.*

Remark 39. We note that the size of the field $\mathbb{F}$, defined to be $2^{\omega(n)}$, is larger than any quasi-polynomial in the parameter λ.

This follows from the fact that for correlation intractability to make sense against quasi-polynomial adversaries, we require the relation to be sufficiently sparse. Setting the field size to be quasi-polynomial is not sufficient since a quasi-polynomial time adversary can break correlation intractability even when the hash function is modeled as a random oracle by trying quasi-polynomially many different values of x.

Proof. We proceed in two stages, initially establishing that a correlation intractable hash function for the relevant relation implies that the Fiat-Shamir transformation is adaptively sound. Next, we show that under suitable choices of parameters for the underlying primitives, we can instantiate such a correlation intractable hash function.

Stage I: from correlation intractability to FS-soundness. For the first part, we restate the following claim from [14], removing the efficiency requirements needed for their result, and adapting it to the specific case of the sumcheck protocol.

The following relation specifies whether partial transcript is *bad* or *good*. For any $i \in \{j+1, \cdots, n\}$, an i-th *round partial transcript* consists of the statement $(\mathbb{F}, y, f, \beta_1, \cdots, \beta_j)$, and $\tau_i := \{\alpha_{j+1,\gamma}\}_{\gamma=0}^{d}, \beta_{j+1}, \cdots, \{\alpha_{i,\gamma}\}_{\gamma=0}^{d}, \beta_i$. Recall that $\{\alpha_{i,\gamma}\}_{\gamma=0}^{d}$ defines a unique degree d polynomial g_i. Formally the relation R_{SC} is defined as,

$$R_{\mathrm{SC}} := \left\{ \left(\left(\mathbb{F}, y, f, \beta_1, \cdots, \beta_j, \tau_i, \{\alpha_\gamma\}_{\gamma=0}^{d} \right), \beta \right) : \begin{array}{r} \left(\mathbb{F}, y, f, \beta_1, \cdots, \beta_j, \tau_i \right) \in \mathsf{BAD} \quad \bigwedge \\ \left(\mathbb{F}, y, f, \beta_1, \cdots, \beta_j, \tau_i, \{\alpha_\gamma\}_{\gamma=0}^{d}, \beta \right) \in \mathsf{GOOD} \end{array} \right\}$$

for some i-th *round partial transcript.*

We say that a partial transcript $\left(\mathbb{F}, y, f, \beta_1, \cdots, \beta_j, \tau_i\right) \in \mathsf{BAD}$ if

$$\left(\mathbb{F}, g_i(\beta_i), f, \beta_1, \cdots, \beta_j, \beta_{j+1}, \cdots, \beta_i\right) \notin \mathcal{L}_{\mathrm{SC}}$$

where $\beta_{j+1}, \cdots, \beta_i$ and g_i are obtained from τ_i. Roughly, a partial transcript is *bad* if it leads the verifier in the interactive protocol to reject with high probability. Correspondingly, we say that a partial transcript $\left(\mathbb{F}, y, f, \beta_1, \cdots, \beta_j, \tau_i\right) \in \mathsf{GOOD}$ if

$$\left(\mathbb{F}, g_i(\beta_i), f, \beta_1, \cdots, \beta_j, \beta_{j+1}, \cdots, \beta_i\right) \in \mathcal{L}_{\mathrm{SC}},$$

again roughly translating to a partial transcript that leads the verifier to in the interactive protocol to accept.

We now state the claim.

Claim 40. *Let $\Pi = (\mathcal{P}_{\mathrm{SC}}, \mathcal{V}_{\mathrm{SC}})$ be the $O(\mathsf{polylog}(\lambda))$-round public coin interactive protocol for the language $\mathcal{L}_{\mathrm{SC}}$ with perfect completeness and adaptive soundness. If a hash family $\mathcal{H}$ is R_{SC} correlation intractable, and evaluable in time $\mathsf{poly}(\lambda)$, then the Fiat-Shamir Transform gives an adaptively sound argument for $\mathcal{L}_{\mathrm{SC}}$.*

Proof. This follows in a straightforward manner as in the proof in [14], and is included here for completeness. The completeness of the protocol follows from the completeness of the underlying protocol $(\mathcal{P}, \mathcal{V})$.

For the adaptive soundness, we prove this via contradiction. Suppose there exists a cheating prover $\mathcal{P}^*$ that on input $(1^\lambda, h)$, where h is sampled from $\mathcal{H}_\lambda$, that produces a string $(\mathbb{F}, , y^*, f^*, \beta_1^*, \cdots, \beta_j^*) \notin \mathcal{L}_{\mathrm{SC}}$ (i.e. $\sum_{\boldsymbol{z} \in \{0,1\}^{n-j}} f^*(\beta_1^*, \cdots, \beta_j^*, \boldsymbol{z}) \neq y^*$) and $(\{\alpha^*_{j+1,\gamma}\}_{\gamma=0}^d, \cdots, \{\alpha^*_{n,\gamma}\}_{\gamma=0}^d)$ such that $\mathcal{V}$ accepts the transcript derived using h. We shall use this cheating prover $\mathcal{P}^*$ to create an adversary $\mathsf{A} = \{\mathsf{A}_\lambda\}$ that breaks the R_{SC}-correlation intractability of $\mathcal{H}$.

On receiving $h \in \mathcal{H}_\lambda$, A_λ does the following:

1. Run $\mathcal{P}^*$ on input $(1^\lambda, h)$ to obtain $(\mathbb{F}, y^*, f^*, \beta_1^*, \cdots, \beta_j^*)$ and $(\{\alpha^*_{j+1,\gamma}\}_{\gamma=0}^d, \cdots, \{\alpha^*_{n,\gamma}\}_{\gamma=0}^d)$.
2. Sample a random index $i^* \leftarrow \{j+1, \cdots, n-1\}$.
3. Return $\left(\mathbb{F}, y^*, f^*, \beta_1^*, \cdots, \beta_j^*, \tau_{i^*}, \{\alpha^*_{i^*+1,\gamma}\}_{\gamma=0}^d\right)$, where $\forall k \in [i]$, $\beta_k = h(\mathbb{F}, y^*, f^*, \beta_1^*, \cdots, \beta_j^*, \tau_{k-1}, \{\alpha^*_{k,\gamma}\}_{\gamma=0}^d)$.

From the sumcheck protocol, for every accepting transcript for $(\mathbb{F}, y^*, f^*, \beta_1^*, \cdots, \beta_j^*) \notin \mathcal{L}_{\mathrm{SC}}$, there must exist at least one round k such that $(\mathbb{F}, y^*, f^*, \beta_1^*, \cdots, \beta_j^*, \tau_k) \in \mathsf{BAD}$, but $\left(\mathbb{F}, y^*, f^*, \beta_1^*, \cdots, \beta_j^*, \tau_k, \{\alpha^*_{k+1,\gamma}\}_{\gamma=0}^d, \beta_{k+1}\right) \in \mathsf{GOOD}$.

This follows from the fact that $(\mathbb{F}, f^*, y^*, \beta_1^*, \cdots, \beta_j^*) \notin \mathcal{L}_{\mathrm{SC}}$, and for $\mathcal{V}$ to accept, the complete transcript must be GOOD. Thus with probability $\epsilon/(n-j-1)$, A_λ selects the appropriate index k, and outputs the right partial transcript. This contradicts our assumption of correlation intractability. □

Stage II: Building the relevant correlation intractable function. It remains to show that we can build a correlation intractable function for the relation R_{SC}. Now, we need to establish certain properties from the relation R_{SC} in order to invoke Theorem 34. We start with two simple claims regarding the relation R_{SC}. For simplicity of exposition, let us denote by g_{i+1} the prescribed polynomial (given prefix $\boldsymbol{\beta}$) to be sent in round $i+1$, i.e.

$$g_{i+1}(x) := \sum_{\boldsymbol{z} \in \{0,1\}^{n-i-1}} f(\boldsymbol{\beta}, \beta_{j+1}, \cdots, \beta_i, x, \boldsymbol{z}).$$

Claim 41. *R_{SC} is a $\rho = \frac{d}{|\mathbb{F}|}$-sparse relation.*

Proof. Given $\left(\mathbb{F}, y, f, \beta_1, \cdots, \beta_j, \tau_i, \{\alpha_\gamma\}_{\gamma=0}^d\right)$, we compute the fraction of β such that

$$\left(\left(\mathbb{F}, y, f, \boldsymbol{\beta}, \tau_i, \{\alpha_\gamma\}_{\gamma=0}^d\right), \beta\right) \in R_{\mathrm{SC}}.$$

For $\left(\mathbb{F}, y, f, \beta_1, \cdots, \beta_j, \tau_i, \{\alpha_\gamma\}_{\gamma=0}^d, \beta\right) \in \mathsf{GOOD}$, we require β to be such that $\widetilde{g}(\beta) = g_{i+1}(\beta)$, where $\widetilde{g}(x)$ denotes the polynomial described by $\{\alpha_\gamma\}_{\gamma=0}^d$, and $g_{i+1}(x)$ is as defined above. This follows from the definition of $\mathcal{L}_{\mathrm{SC}}$.

The polynomial $g_{i+1}(x) - \widetilde{g}(x)$ has degree at most d ($g(x)$ has degree at most d), and is non-zero (since $\widetilde{g}$ is not the prescribed polynomial). Thus, from Schwartz-Zippel lemma, there are at most d roots to the above polynomial, and thus d values β such that $\widetilde{g}(\beta) = g_{i+1}(\beta)$.

Thus the fractions of such values are $\frac{d}{|\mathbb{F}|}$. Since, we have set $|\mathbb{F}|$ to be of size $2^{\omega(\mathsf{polylog}(\lambda))}$, ρ is negligible. □

Claim 42. *R_{SC} is sampleable in $2^{\mathsf{polylog}(\lambda)}$-time.*

Proof. The algorithm for sampling a β given $\left((\mathbb{F}, y, f, \beta_1, \cdots, \beta_j, \tau_i, \{\alpha_\gamma\}_{\gamma=0}^{d}\right)$ works in the following manner. As described above, it is sufficient to output a random root of $g_{i+1}(x) - \widetilde{g}(x)$.

We note that from Remark 39, the size of field is larger than any quasi-polynomial, and thus we do not know if we can do this determinisitcally in quasi-polynomial time.

The running time of the above sampling strategy derives from the fact that to compute the polynomial g_{i+1}, we need to compute an exponential sum over $\mathsf{polylog}(\lambda)$ variables. $\square$

Theorem 34 requires R_{SC} to be sampled by a function in $\mathcal{F}$. Thus, in our setting, $\mathcal{F}$ represents the set of all functions computable in $2^{\mathsf{polylog}(\lambda)}$-time.

We additionally make the following simple observations regarding the sumcheck protocol:

- The total number of rounds r in the protocol is $\mathsf{polylog}(\lambda)$. This follows from the structure of the protocol wherein each round corresponds to reducing the claim by a single variable, and we have set the number of variables to be $\mathsf{polylog}(\lambda)$
- The length of each verifier message is $|\beta_i| = \omega(\mathsf{polylog}(\lambda))$ by our choice of parameters. This follows from the fact that each $\beta_i \in \mathbb{F}$.
- The size of the input to the hash function, $\left(\mathbb{F}, y, f, \beta_1, \cdots, \beta_j, \tau_r\right)$, is $\mathsf{poly}(\lambda)$. This follows from the fact that in addition to the description of the function f (of size $\mathsf{poly}(\lambda)$), the input consists of r rounds of prover, and verifier, messages. A prover message consists of only $O(d)$ elements from $\mathbb{F}$. Given that the number of rounds are $\mathsf{polylog}(\lambda)$, this gives an additive overhead of $\omega(\mathsf{polylog}(\lambda))$ to the description of f.

Finally,to instantiate Theorem 34, we need an appropriate encryption scheme with universal ciphertexts, and KDM security for all quasi-polynomial computable functions. Specifically, we require an encryption scheme $\mathsf{SKE} = (\mathsf{SKE.Gen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ with keys of length $\kappa = \kappa(\lambda) \geq \lambda^{\Omega(1)}$ and universal ciphertexts that are $\Delta_{\mathsf{quasi-poly}}$-KDM secure for arbitrary quasi-polynomial computable functions, of output length ℓ.

Assumption 36 implies that secret key Regev encryption satisfies these properties, with secret distribution χ_{sec} that is uniform on $[-B, B)$ for some B, and error distribution χ_{err} that is uniform on $\left[-\frac{q}{4}, \frac{q}{4}\right)$. For the corresponding scheme n' is set to be such that $(2B+1)^{n'} \in \{0,1\}^{|\mathbf{trans}|}$, where $|\mathbf{trans}|$ is the size of the largest input to the hash function, and $\ell = \omega(\mathsf{polylog}(\lambda))$ is the size of a single verifier message.

We now have all the requisite conditions for Theorem 34, and thus invoking the result, the Fiat-Shamir transform gives us an adaptively sound argument system. From Claim 4, an adaptively sound argument system is also an adaptively unambigiously sound argument system, thus completing the proof.

$\square$