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

Arka Rai Choudhuri
Johns Hopkins University
Baltimore, USA
achoud@cs.jhu.edu

Pavel Hubáček
Charles University
Prague, Czech Republic
hubacek@iuuk.mff.cuni.cz

Chethan Kamath
IST Austria
Klosterneuburg, Austria
ckamath@ist.ac.at

Krzysztof Pietrzak
IST Austria
Klosterneuburg, Austria
pietrzak@ist.ac.at

Alon Rosen
IDC Herzliya
Herzliya, Israel
alon.rosen@idc.ac.il

Guy N. Rothblum
Weizmann Institute of Science
Rehovot, Israel
rothblum@alum.mit.edu

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 #P gives rise to a hard distribution in the class CLS, which is contained in PPAD. Our result opens up the possibility of sampling moderately-sized games for which it is hard to find a Nash equilibrium, by reducing the inversion of appropriately chosen one-way functions to #SAT.

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 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 poly(n).

CCS CONCEPTS

• **Theory of computation → Complexity classes**; **Problems, reductions and completeness**.

KEYWORDS

TFNP, PPAD, Nash Equilibrium, Fiat-Shamir transformation

ACM Reference Format:
Arka Rai Choudhuri, Pavel Hubáček, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. 2019. Finding a Nash Equilibrium Is No Easier Than Breaking Fiat-Shamir. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19), June 23–26, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3313276.3316400

STOC '19, June 23–26, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6705-9/19/06…$15.00
https://doi.org/10.1145/3313276.3316400

1 INTRODUCTION

The complexity class PPAD, defined by Papadimitriou [39], 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 [16], and Chen, Deng and Teng [14].

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 [39], 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

[1] We do know of worst-case hard instances for a *specific algorithm* for finding a Nash equilibrium [44] or of *oracle-based* worst-case hard instances of PPAD-complete problems [3, 26].

was subsequently extended by Hubáček and Yogev [28] 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* [20], 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. [35], 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 (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 [28].

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 [24] 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 [10–12, 31, 40]. 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 [41] for proving $y = x^{2^i} \bmod N$ and in very recent work of Canetti *et al.* [10].

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).*

Secure Instantiations. The results in [10] imply a hash family for which, under suitably strong assumptions on the security of known lattice-based fully homomorphic encryption schemes, the Fiat-Shamir transformation is sound when it is applied to (certain instantiations of) the sum-check protocol. In particular, we observe that assuming optimally secure (Regev-extractable) FHE against quasi-polynomial adversaries (see [10]), the transformation is unambiguously sound when applied to the sumcheck protocol over polylogarithmically many variables. By our main result, the same assumption implies average-case hardness of PPAD. We elaborate on these corollaries in the full version.

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 [23]. 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* [42] 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

[2] The underlying assumptions were further weakened by Garg, Pandey and Srinivasan [22] and then by Komargodski and Segev [34], though still to assumptions that appear to be closely related to *iO* (see §1.3).

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 [42]. 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 16. 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 3.

Relationship to Valiant's work on incremental computation. With a similar motivation in mind, Valiant [47] constructed a general-purpose incrementally verifiable computation scheme, where *any* long (exponential) computation can be performed via a sequence of 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.[3] 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 [47] 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.[4]

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 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}.

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 [35]. 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 [46]. 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 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 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

[3] A key ingredient in Valiant's construction is a CS proof [38] 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.

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

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 $M+O(P)+\mathrm{poly}(n)$ memory, and has a final proof of size $P + \mathrm{poly}(n)$. 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, $\mathrm{poly}(n)$ space and $\mathrm{poly}(n)$ 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 $\mathrm{poly}(n)$ 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 [47].

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 $\mathrm{poly}(n)$ time, and π would be of similar length to π^0 and π^1. This was the approach used in [47], 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 $\mathrm{poly}(n)$ 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 $\mathrm{poly}(n)$ 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:

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

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 $\beta_1, \ldots, \beta_i$. In the $(i+1)$-th round the parties run the same two-message protocol described above to verify the i-th claim:

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

Note that this i-th claim is about the sum with the prefix $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_i)$, 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 verifier'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 3.

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:

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

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 $\mathcal{L}_{\mathrm{SC}}$ implies adaptive unambiguous soundness for the same language. Moreover, the protocol's adaptive unambiguous soundness for $\mathcal{L}_{\mathrm{SC}}$ suffices for (unambiguous) soundness of our incrementally verifiable computation scheme.

We begin by defining $\mathcal{L}_{\mathrm{SC}}$. The language is defined over tuples that include an instance to the sumcheck protocol, a fixed prefix $\beta_1, \ldots, \beta_i \in \mathbb{F}$, and a fixed partial transcript of the sumcheck protocol. A tuple is of the form $(\mathbb{F}, y, f, \beta_1, \ldots, \beta_i, \widetilde{\alpha}_{i+1}, \beta_{i+1}, \ldots, \widetilde{\alpha}_{i+j})$, where $i, j \in \{0, \ldots, n\}$ 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 $i \geq 1$ and $j = 0$, the tuple is in the language if and only if the sum over all 2^{n-i} assignments with prefix $\beta_1, \ldots, \beta_i$ equals y.
- For general i and $j \geq 1$, 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 $\beta_1, \ldots, \beta_i$ and the verifier's messages $\beta_{i+1}, \ldots, \beta_{i+j-1}$ (there is no condition on y or on the prior prover messages, only the last one matters).[5]

With this language in mind, we can view each round of the sumcheck as reducing from a claim that a tuple is in $\mathcal{L}_{\mathrm{SC}}$, to a claim that a longer tuple is in $\mathcal{L}_{\mathrm{SC}}$. 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

[5] 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.

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 3. *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

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

and $\widetilde{\pi} = (\widetilde{\alpha}_{i+j+1}, \ldots, \widetilde{\alpha}_n)$. 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:

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

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 $\mathcal{L}_{\text{SC}}$. 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 $\mathcal{L}_{\text{SC}}$. 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 [28] 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 [28] 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 [37], 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 [39] 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 . Perhaps the best known such class is PPAD [39] which captures the computational complexity of finding Nash equilibria (Nash) in bimatrix games [14, 16], amongst other natural problems [32].

Other subclasses of TFNP include PPA [39], which captures computational problems related to the Borsuk-Ulam theorem (Borsuk-Ulam), Tucker's lemma (Tucker) [19] or fair division [21], the class PLS [30] that was defined to capture the computational complexity of problems amenable to local search such as Local-Maxcut, and the class CLS [17], which captures finding approximate local optima of continuous functions (CLO) or finding Banach's fixed points [18] and contains finding Nash equilibria in congestion games or solving the simple stochastic games of Condon or Shapley. Finally, the classes PPP [39] and PWPP [29] are motivated by the pigeonhole principle and contain important problems related to finding collisions in functions. Recently, Sotiraki, Zampetakis and Zirdelis [45] 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 [39] showed that one-way permutations imply average-case hardness in PPP,[6] and Jeřábek [29]

[6] 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.

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 [28], who established hardness in CLS, a subclass of PPAD, under the same assumptions. Both results were subsequently strengthened. First, by Garg, Pandey and Srinivasan [22], 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-way permutations. Second, by Komargodski and Segev [34], who weakened the assumptions to quasi-polynomially secure private-key functional encryption and sub-exponentially-secure injective one-way functions.

Hubáček, Naor and Yogev [27] 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 [33] 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 [30, 37] was later strengthened to show that it is unlikely to base average-case TFNP hardness even on problems in the polynomial hierarchy [8], 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 [36]. A recent result [43] 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 [9, 29].

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 $\text{poly}(|v|)$ 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) [39].

Definition 4. An End-of-Line (EOL) instance (S, P) consists of a pair of circuits $S, P : \{0,1\}^M \to \{0,1\}^M$ such that $P(0^M) = 0^M$ and $S(0^M) \neq 0^M$. The goal is to find a vertex $v \in \{0,1\}^M$ such that $P(S(v)) \neq v$ or $S(P(v)) \neq v \neq 0^M$.

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 $S(v) = u$ and $P(u) = v$ (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. [17] for the formal definition). Another problem that is known to lie in CLS (but not known to be complete) is End-of-Metered-Line [28].

Definition 5. An End-of-Metered-Line (EOML) instance (S, P, M) consists of circuits $S, P\colon \{0,1\}^m \to \{0,1\}^m$ and $M\colon \{0,1\}^m \to \{0,\ldots,2^m\}$ such that $P(0^m) = 0^m \neq S(0^m)$ and $M(0^m) = 1$. The goal is to find a vertex $v \in \{0,1\}^m$ satisfying one of the following:

(i) **End of line:** either $P(S(v)) \neq v$ or $S(P(v)) \neq v \neq 0^m$,
(ii) **False source:** $v \neq 0^m$ and $M(v) = 1$,
(iii) **Miscount:** either $M(v) > 0$ and $M(S(v)) - M(v) \neq 1$ or $M(v) > 1$ and $M(v) - M(P(v)) \neq 1$.

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 6. A Sink-of-Verifiable-Line instance (S, V, T, v_0) consists of $T \in \{1,\ldots,2^M\}$, $v_0 \in \{0,1\}^M$, and two circuits 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 7. 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 6 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 8. 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 9. *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 [28] when applied to a relaxed-Sink-of-Verifiable-Line instance. We refer the readers to the full version of the paper for the details. □

3 THE SUMCHECK PROTOCOL

The sumcheck protocol was introduced by Lund et al. [35] to show that #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 [25].

Definition 10. 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 [42] 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 11. 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-unambiguously 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 12. A pair of machines $(\mathcal{P}, \mathcal{V})$, where $\mathcal{P}$ is the prescribed prover, is a (δ, ϵ)-unambiguously sound (or, if $\epsilon = \delta$, simply δ-unambiguously 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 $x \in \{0,1\}^*$,
$$\Pr[\mathcal{V}(x, \mathcal{P}(x, R), R) = \mathcal{L}(x)] = 1.$$
- **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 $\widetilde{\mathcal{P}}$,
$$\Pr\left[\begin{array}{l|l} \mathcal{V}(x, \widetilde{\pi}, R) = 1 & (x, \widetilde{\pi}) \leftarrow \widetilde{\mathcal{P}}(R), \pi \leftarrow \mathcal{P}(x, R) \\ & \widetilde{\pi} \neq \pi, x \in \mathcal{L} \end{array}\right] \leq \epsilon(|x|).$$

Remark 13. A non-interactive proof system is called an *argument* if the soundness and unambiguity properties hold only against computationally-bounded (i.e., 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 a 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

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

for some value $y \in \mathbb{F}$. Thus the language $\mathcal{L}_{\mathrm{SC}}$ is defined[7] as

$$\mathcal{L}_{\mathrm{SC}} := \{(\mathbb{F}, f, y) : \sum_{z \in \mathbb{H}^n} f(z) = y\}.$$

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 poly$(n, d, \log_2(|\mathbb{F}|))$. In Figure 1, we review the standard sumcheck protocol from [35], denoted by

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

$\mathcal{P}_{\mathrm{SC}}$ 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 $\mathcal{P}_{\mathrm{SC}}(f)$ runs in time poly$(|\mathbb{F}|^n)$.[8] The verifier $V_{\mathrm{SC}}^{g}(y)$ runs in time poly$(\log_2(|\mathbb{F}|), d, n)$ and queries the oracle f at a single point. The communication complexity is poly$(\log_2(|\mathbb{F}|), d, n)$, 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.

We show in Theorem 14 that the protocol is an unambiguously-sound interactive proof system for the language $\mathcal{L}_{\mathrm{SC}}$. Therefore we have a strong guarantee that it is hard to come up with a proof different from the one generated by the prescribed prover.

Theorem 14. *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 1 is a $(dn/|\mathbb{F}|)$-unambiguously sound interactive proof system for the language $\mathcal{L}_{\mathrm{SC}}$. That is, it satisfies the following three properties.*

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

Proof. We refer the readers to the full version of the paper for the proof. □

[7] We note that we overload $\mathcal{L}_{\mathrm{SC}}$ to mean different, but related, languages in different sections, but the meaning will be clear from the context.

[8] 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 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 1: Sumcheck protocol $(\mathcal{P}_{\text{SC}}(f), \mathcal{V}_{\text{SC}}^{f}(y))$ **[35]**

3.3 Non-Interactive Sumcheck Protocol

We consider the non-interactive version of the sumcheck protocol obtained by applying the Fiat-Shamir transformation to protocol from Figure 1. To be exact, the verifier's "challenges" β_i in the non-interactive protocol are obtained by applying a hash function $h : \mathbb{F}^* \to \mathbb{F}$ 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 $(\mathcal{P}_{\text{FS}}, \mathcal{V}_{\text{FS}})$ is given in Figure 2. It works for a more general language than in §3.2 that allows sums with an arbitrary prefix $\boldsymbol{\beta} = (\beta_1,\ldots,\beta_j) \in \mathbb{F}^{\le n}$. That is we consider proof systems for the "prefixed" language $\mathcal{L}_{\text{SC}}$ defined as

$$\mathcal{L}_{\text{SC}} := \{(\mathbb{F}, f, y, \boldsymbol{\beta}) : \sum_{\boldsymbol{z}\in\mathbb{H}^{n-j}} f(\boldsymbol{\beta}, \boldsymbol{z}) = y\}.$$

The algorithm $\mathcal{P}_{\text{FS}}$ runs in time poly$(|\mathbb{F}|^n)$. The verification algorithm $\mathcal{V}_{\text{FS}}$ runs in time poly(n) and space $O(\log_2(|\mathbb{F}|)\cdot n)$, and queries the oracle f at a single point. The size of the proof is $O(n\cdot\log_2(|\mathbb{F}|))$.

Assumption on Fiat-Shamir. The assumption that underlies our main theorem pertains to the soundness of the collapsed protocol $(\mathcal{P}_{\text{FS}}, \mathcal{V}_{\text{FS}})$. In particular, we assume that the Fiat-Shamir Transform is *unambiguously* sound when applied to the interactive sumcheck protocol and, as a result, that the collapsed protocol $(\mathcal{P}_{\text{FS}}, \mathcal{V}_{\text{FS}})$ is an unambiguously sound (adaptive) non-interactive proof system for

Non-Interactive β-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 $h : \mathbb{F}^* \to \mathbb{F}$

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

(1) For $\boldsymbol{\beta} = (\beta_1,\ldots,\beta_j)$, 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

$$\beta_i = h((f, y, \boldsymbol{\beta}), \{\alpha_{j+1,\gamma}\}_{\gamma=0}^{d}, \beta_{j+1}, \ldots, \{\alpha_{i-1,\gamma}\}_{\gamma=0}^{d}, \beta_{i-1})$$

and set $y_i = g_i(\beta_i)$.
(3) Output $\pi = (\{\alpha_{j+1,\gamma}\}_{\gamma=0}^{d}, \ldots, \{\alpha_{n,\gamma}\}_{\gamma=0}^{d})$

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

(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

$$\beta_i = h((f, y, \boldsymbol{\beta}), \{\widetilde{\alpha}_{j+1,\gamma}\}_{\gamma=0}^{d}, \beta_{j+1}, \ldots, \{\widetilde{\alpha}_{i,\gamma}\}_{\gamma=0}^{d}\beta_{i-1})$$

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 2: Non-interactive version of the sumcheck protocol $(\mathcal{P}_{\text{SC}}(f), \mathcal{V}_{\text{SC}}^{f}(y))$ **from [35]**

the language $\mathcal{L}_{\text{SC}}$. In Lemma 16 below, we show that the assumption holds *with respect to random oracles.*

Assumption 15. *The Fiat-Shamir heuristic is* unambiguously *sound for the sumcheck protocol from Figure 1. In other words, there exists a family of hash functions $\mathcal{H}$ such that when instantiated with (random) $h : \mathbb{F}^* \to \mathbb{F}$ from $\mathcal{H}$, the non-interactive sumcheck protocol $(\mathcal{P}_{\text{FS}}, \mathcal{V}_{\text{FS}})$ given in Figure 2 is (δ, ϵ)-unambiguously sound for the language $\mathcal{L}_{\text{SC}}$ for some δ and ϵ that are negligible in n.*

Lemma 16. *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 $(\mathcal{P}_{\text{FS}}^{h}, \mathcal{V}_{\text{FS}}^{h})$ denote the non-interactive sumcheck protocol obtained by instantiating the protocol described in Figure 2 with a random*

oracle h. Then $(\mathcal{P}^h_{\mathrm{FS}}, \mathcal{V}^h_{\mathrm{FS}})$ is a $(Qd/|\mathbb{F}|)$-unambiguously sound non-interactive proof system for the language $\mathcal{L}_{\mathrm{SC}}$, 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,*

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

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

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

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

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

Proof. We refer the readers to the full version of the paper for the proof. □

4 THE REDUCTION

We construct an rSVL instance using the non-interactive sumcheck protocol $(\mathcal{P}_{\mathrm{FS}}, \mathcal{V}_{\mathrm{FS}})$ 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

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

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 8) 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 main technical components of our reduction are the successor circuit S and the verifier circuit V which, together, implement the incrementally-verifiable counter for statements of size 2^n. They are defined using a sequence of circuits

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

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 $\mathrm{poly}(n)$ 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 $M + O(dP) + \mathrm{poly}(n)$ memory, and has a final proof of size $P + \mathrm{poly}(n)$. On unwinding the recursion, it can be shown that (S, V) runs for $2^{O(n)}$ steps, uses $\mathrm{poly}(n)$ space and has proof size of $\mathrm{poly}(n)$. But most importantly S and V are polynomial-sized circuits, and therefore each step can be carried out in $\mathrm{poly}(n)$ time. In other words, we get an rSVL instance describing a directed graph with $2^{O(n)}$ vertices each with a label of size $\mathrm{poly}(n)$, where the successor and verifier functions have efficient descriptions.

We refer the readers to the full version of the paper for details of the reduction.

ACKNOWLEDGMENTS

The first author is supported in part by a DARPA/ARL Safeware Grant W911NF-15-C-0213, and a subaward from NSF CNS-1414023. The second author is supported by the project 17-09142S of GA ČR, Charles University project UNCE/SCI/004, Charles University project PRIMUS/17/SCI/9, and by the Neuron Fund for the support of science. The third and fourth authors are supported by the European Research Council (ERC) consolidator grant (682815-TOCNeT). The fifth author is supported by ISF grant No. 1399/17 and via Project PROMETHEUS (Grant 780701). The sixth author has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819702).

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.

[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.

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

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

[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.

[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.

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

[8] 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.

[9] Buresh-Oppenheim, J. On the TFNP complexity of factoring. Unpublished, http://www.cs.toronto.edu/~bureshop/factor.pdf, 2006.

[10] 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.

[11] 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.

[12] 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.

[13] Cantor, David G and Zassenhaus, Hans A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 1981.

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

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

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] Jeřábek, E. Integer factoring and modular square roots. *J. Comput. Syst. Sci. 82*, 2 (2016), 380–394.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] Lund, C., Fortnow, L., Karloff, H., and Nisan, N. Algebraic methods for interactive proof systems. *J. ACM 39*, 4 (Oct. 1992), 859–868.

[36] 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.

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

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

[39] 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.

[40] 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.

[41] Pietrzak, K. Simple verifiable delay functions. *IACR Cryptology ePrint Archive 2018* (2018), 627.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

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

[47] 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.