Lossy Correlation Intractability and PPAD Hardness from Sub-exponential LWE

Ruta Jawale* Dakshita Khurana*

Abstract

We introduce a new cryptographic primitive, a *lossy correlation-intractable hash function*, and use it to soundly instantiate the Fiat-Shamir transform for the general interactive sumcheck protocol, assuming sub-exponential hardness of the Learning with Errors (LWE) problem. By combining this with the result of Choudhuri et al. (STOC 2019), we show that #SAT reduces to end-of-metered line, which is a PPAD-complete problem, assuming the sub-exponential hardness of LWE.

*University of Illinois, Urbana-Champaign. This material is based upon work supported in part by DARPA under Contract No. HR001120C0024. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

1 Introduction

The hardness of computing a Nash equilibrium is an important open question in algorithmic game theory. This problem is known to be complete for the complexity class PPAD [Pap94, DGP09, CDT09]. The class PPAD consists of all total search problems that are polynomial-time reducible to the end-of-line (EOL) 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. This problem becomes easy when given an explicit graph, but is not known to be solvable in polynomial time when only given a description of the successor and predecessor functions for every vertex.

Naturally, there has been significant interest [AKV04, BPR15, HY17, GPS16, KS17, CHK+19a, CHK+19b, EFKP20, Pie19, LV20] in reducing the hardness of PPAD to that of various cryptographic assumptions. The most relevant to us is the recent work of Choudhuri et. al. [CHK+19a] who showed a reduction from #SAT to EOL, assuming adaptive soundness of the Fiat-Shamir transform applied to the sumcheck protocol. But despite these exciting works, establishing PPAD hardness under standard, well-studied cryptographic assumptions has remained open.

The Sumcheck Protocol. The interactive sumcheck protocol is an important building block for many seminal results in interactive proofs, including the IP=PSPACE theorem [LFKN92, Sha92] and the bounded-depth delegation scheme of Goldwasser, Kalai and Rothblum [GKR15]. These seminal results allow an inefficient prover to interactively convince a relatively efficient verifier about the validity of a computational statement. These protocols are unconditionally sound, meaning that no adaptive, unbounded cheating prover can convince a verifier to accept the proof of a false statement.

An important question, which has received wide-spread attention in recent years, is whether it is possible to obtain succinct non-interactive proof systems with comparable strong efficiency guarantees. The importance of this problem has become evident in recent years due to the increasing popularity of cloud computing. A client (or verifier), would like to offload the computation of a function f to a server (or prover). The client may not trust the server and would therefore want the server to send a non-interactive "proof" that the computation was performed correctly. Clearly, the complexity of verifying such a proof should be significantly lower than the complexity of running f and, at the same time, should not significantly blow up the running time of the prover. The applicability of succinct proof systems goes beyond cloud computing; for example, variants of these are now widely used in cryptocurrencies [SCG+14, BSBHR18]. Despite a recent sequence of beautiful works [Kil92, Mic94, BGL+15, CHJV15, CH16, CCHR15, KLW15, CCC+16, ACC+15, PR17, KRR17, CCRR18, HL18, CCH+18, KPY19], the problem of obtaining publicly-verifiable succinct arguments for meaningful classes of computation based on standard and well-studied cryptographic hardness assumptions has remained open so far[1].

This Work. We show how to obtain a publicly-verifiable succinct non-interactive argument for sumcheck. Like many prior approaches, we rely on the Fiat-Shamir paradigm [FS86], which suggests a method for converting general public-coin interactive proofs to non-interactive arguments

[1]Most recently, the work of Canetti et. al. [CCH+18] obtained delegation for bounded depth assuming "optimally-secure" fully homomorphic encryption and Kalai, Paneth and Yang [KPY19] obtained delegation for P based on a new decisional assumption on groups with bilinear maps. We discuss additional related work in Section 1.3.

in the common reference string (CRS) model. Namely, the CRS contains a key for an appropriately chosen hash function, and the prover non-interactively emulates the verifier's messages by hashing its transcript so far. In fact, recent fascinating works due to Canetti et. al., and Peikert and Shiehian [CCH+19, PS19] designed a special type of hash function, *a correlation-intractable hash function* [CGH04] (based on the hardness of learning with errors (LWE)), and used this to obtain the first NIZK (non-interactive zero-knowledge) for NP from LWE. On the other hand, correlation-intractable hash functions that support *succinct* non-interactive arguments are so far known only under strong assumptions, such as the optimal hardness of fully homomorphic encryption (FHE) [CCH+19].

Our work designs special hash functions that can be used to obtain a succinct argument for the sumcheck protocol. Specifically, we develop *lossy correlation intractable hash functions* based on the hardness of the learning with errors (LWE) assumption. We then show that the Fiat-Shamir transform applied to the interactive sumcheck protocol is sound when using any (sub-exponentially secure) lossy correlation intractable hash function. Combined with the results of Choudhuri et. al. [CHK+19a], this implies the hardness of Nash based on a standard cryptographic assumption: sub-exponential LWE. Prior to this work, establishing PPAD hardness required stronger and non-standard cryptographic assumptions, which we discuss in detail in Section 1.3.

1.1 Our Results

We now provide an overview of our results. Our first main result is a construction, based on the sub-exponential learning with errors (LWE) assumption, of lossy correlation intractable hash functions. At a high level, a hash function family H is correlation intractable (CI) for a relation $\mathcal{R}(x, y)$ if it is computationally hard, given a random key k, to find any input x such that $(x, H_k(x)) \in \mathcal{R}$.

Very recent beautiful constructions of (somewhere-statistical) CI hash functions for restricted classes of relations[2] led to the first realization of non-interactive zero-knowledge (NIZK) from LWE [CLW18, PS19]. The CI hash functions constructed in these works generate keys in two modes: a Gen mode and a $\mathsf{StatGen}$ mode. The Gen mode samples a random key from a fixed distribution, while the $\mathsf{StatGen}$ mode samples a key dependent on the relation against which correlation-intractability is desired. Both modes are indistinguishable, and the $\mathsf{StatGen}$ mode with key sampled according to any function f computable in a-priori bounded time $b(\lambda)$, guarantees that it is not possible to find any x such that $H(k, x) = f(x)$, except with negligible probability. In addition, the time required to compute the hash (in either mode) is proportional to $b(\lambda)$.

Lossy (Somewhere-Statistical) Correlation Intractable Hash Functions. We define and construct *lossy* (somewhere-statistical) correlation intractable hash functions that, in addition to satisfying the properties above, have a *third, lossy mode*. Keys generated in lossy mode are indistinguishable from keys generated in both the other (Gen and $\mathsf{StatGen}$) modes. In addition, for an overwhelming fraction of keys generated in lossy mode, the output of the CI hash function lies within a (subexponentially) small space of outcomes. We construct lossy hash functions by combining (somewhere-statistical) CI hash functions with lossy trapdoor functions [PW08], which can be obtained from LWE.

[2]Specifically, these works build CI hash functions for the class of *efficiently searchable relations*, and, for our purposes, it suffices to consider functions computable in a-priori bounded time (instead of such relations).

Theorem 1.1. *(Informal) Assuming sub-exponential hardness of learning with errors (LWE), there exist lossy correlation intractable hash functions.*

We believe that lossy correlation intractable hash functions may be of independent interest. In this work, we use them to soundly instantiate Fiat-Shamir for the sumcheck protocol.

Fiat-Shamir for Sumcheck. We directly state an informal version of our theorem. The formal theorem and proof can be found in Section 4.

Theorem 1.2. *(Informal) There exists a constant $c > 1$ such that the Fiat-Shamir transform applied to any sumcheck protocol with $\leq \kappa$ variables, with a field $\mathbb{F}$ of size $\geq 2^{\kappa^c}$, sub-field $B \subset \mathbb{F}$ of size $\leq \mathsf{polylog}(\kappa)$, and individual degree $d \leq \mathsf{polylog}(\kappa)$ has soundness error $\mathsf{negl}(\kappa)$ when instantiated with subexponentially-secure lossy (somewhere-statistical) correlation intractable hash functions.*

The Hardness of Finding a Nash Equilibrium. We build on the work of [CHK+19a] who argued that soundness of the Fiat-Shamir transform for sumcheck, combined with the hardness of #SAT implies PPAD hardness. Specifically, they showed that #SAT reduces to the PPAD complete problem end-of-metered-line (EOL), assuming (adaptive) soundness of Fiat-Shamir for sumcheck. While we cannot argue full-fledged adaptive soundness of our non-interactive sumcheck protocol, we show in Section 5 that it satisfies a weaker notion of soundness that suffices for the reduction in [CHK+19a]. As such, we obtain the following corollary.

Corollary 1.1. *(Informal)* #SAT *Reduces to* EOL*, assuming the sub-exponential hardness of LWE.*

1.2 Independent and Concurrent Work [KZ20]

In a beautiful independent and concurrent result, Kalai and Zhang [KZ20] also instantiate Fiat-Shamir for sumcheck from sub-exponential LWE, but via a different approach than ours. In their case, the efficiency of the resulting non-interactive sumcheck degrades double-exponentially with the number of rounds in the underlying interactive sumcheck protocol. They then use the resulting non-interactive sumcheck to compress a modification of the GKR protocol. As a result, they achieve publicly verifiable succinct non-interactive arguments for log-space uniform circuits C of size S, depth D, and input size λ, where the prover runs in time $\mathsf{poly}(S)$ and the verifier runs in time $(D + \lambda) \cdot S^{\epsilon}$, for slightly sub-constant ϵ. They also prove (average-case) PPAD hardness assuming (average-case) hardness of #SAT over $O(\log \lambda \log\log \lambda)$ variables.

On the other hand, our techniques allow for polynomially many rounds (and hence, variables in sumcheck), resulting in optimal parameters for sumcheck and PPAD hardness. We believe that our techniques also directly enable optimal parameters for succinct non-interactive delegation for bounded-depth, and are in touch with [KZ20] about the optimization.

1.3 Related Work

The complexity class PPAD [Pap94] consists of all total search problems that are polynomial-time reducible to the End-of-Line problem. Papadimitriou showed that Nash is reducible to End-of-Line and, thus, belongs to PPAD. As such, the question of establishing PPAD hardness based on cryptographic assumptions has received significant attention in the last few years. A series

of recent works showed that the hardness of PPAD follows from strong cryptographic assumptions: specifically, indistinguishability obfuscation (iO) and related primitives such as functional encryption [AKV04, BPR15, HY17, GPS16, KS17].

Subsequently, Choudhuri et al. [CHK+19a] showed a reduction from #SAT to rSVL, assuming adaptive soundness of the Fiat-Shamir transform. Using the work of [CCH+19] which instantiated Fiat-Shamir for sumcheck under optimally-secure FHE, [CHK+19a] demonstrated PPAD hardness assuming hardness of #SAT for $\mathsf{polylog}(n)$ variables. More recently, [CHK+19b, EFKP20] established PPAD hardness assuming #SAT hardness and soundness of Fiat-Shamir for an interactive proof of repeated squaring due to Pietrzak [Pie19]. Very recently [LV20] established soundness of Fiat-Shamir for this repeated squaring protocol, assuming $2^{-n^{1-\epsilon}}$-hardness of LWE and $2^{\lambda^{\epsilon}}$-hardness of repeated squaring, and therefore also PPAD hardness under the same assumption combined with #SAT hardness.

1.4 Technical Overview

We now outline our technical approach. We begin by recalling the sumcheck protocol, which was first proposed by [LFKN92, Sha92].

Background: Sumcheck. The sumcheck protocol is an interactive proof of the following claim:

$$\sum_{x_1,\ldots,x_\nu \in B} p(x_1,\ldots,x_\nu) = a,$$

where p is a multivariate polynomial with individual degree d in ν variables over field $\mathbb{F}$, $a \in \mathbb{F}$, and field $B \subset \mathbb{F}$. The protocol reduces this claim over ν variables to a related claim with only $\nu - 1$ variables. This is done by having the prover send the univariate polynomial

$$p_1(\cdot) := \sum_{x_2,\ldots,x_\nu \in B} p(\cdot, x_2,\ldots,x_\nu).$$

On receiving the polynomial $p_1(\cdot)$, the verifier checks that $p_1(\cdot)$ is of degree at most d and that

$$\sum_{x_1 \in B} p_1(x_1) = a.$$

If these checks pass, the verifier sends a random challenge $r_1 \leftarrow \mathbb{F}$. The prover upon receiving the challenge fixes variable $x_1 = r_1$ and next proves the following claim:

$$\sum_{x_2,\ldots,x_\nu \in B} p(r_1, x_2,\ldots,x_\nu) = p_1(r_1).$$

This round also proceeds similarly to the previous round, i.e. the prover sends the univariate polynomial

$$p_2(\cdot) := \sum_{x_3,\ldots,x_\nu \in B} p(r_1, \cdot, x_3,\ldots,x_\nu).$$

and on receiving $p_2(\cdot)$, the verifier checks that $p_2(\cdot)$ is of degree at most d and checks that

$$\sum_{x_2 \in B} p_2(x_2) = p_1(r_1).$$

If these checks pass, the verifier sends a random challenge $r_2 \leftarrow \mathbb{F}$. This recursive process continues for ν rounds until the verifier has verified that for $p_\nu(\cdot)$ summed over inputs in B the result will yield $p_{\nu-1}(r_{\nu-1})$. The verifier finally samples a random challenge $r_\nu \leftarrow \mathbb{F}$ and checks that

$$p(r_1, \ldots, r_\nu) = p_\nu(r_\nu).$$

Soundness. To see why this protocol is sound, consider a malicious prover that convinces the verifier to accept an incorrect claim of the form

$$\sum_{x_1,\ldots,x_\nu \in B} p^*(x_1, \ldots, x_\nu) = a^*,$$

where a^* is *not* the correct value of $\sum_{x_1,\ldots,x_\nu \in B} p^*(x_1, \ldots, x_\nu)$. Let us say that this malicious prover sends some polynomial $p_1^*(\cdot)$ as its first message. For convenience, we will define the correct evaluation of the prover's first message $q_1^*(\cdot)$ as

$$q_1^*(\cdot) := \sum_{x_2,\ldots,x_\nu \in B} p^*(\cdot, x_2, \ldots, x_\nu).$$

Now since that $p_1^*(\cdot)$ passes the verifier's next check, this implies that $p_1^*(\cdot)$ is of degree at most d and that

$$\sum_{x_1 \in B} p_1^*(x_1) = a^*.$$

Next, the prover must prove that

$$\sum_{x_2,\ldots,x_\nu \in B} p^*(r_1, x_2, \ldots, x_\nu) = p_1^*(r_1).$$

If this second claim is true, then it must be the case that $p_1^*(r_1) = q_1^*(r_1)$, equivalently r_1 is a root of the polynomial $p_1^*(\cdot) - q_1^*(\cdot)$. Since r_1 is sampled uniformly at random in the field $\mathbb{F}$, the probability that this happens is $\frac{d}{|\mathbb{F}|}$. Looking ahead, we will call such roots of $p_i^*(\cdot) - q_i^*(\cdot)$ in any round i, "bad" verifier challenges, where $p_i^*(\cdot)$ is the claimed polynomial and $q_i^*(\cdot)$ is the true polynomial. Moreover, when we attempt to obtain a non-interactive version of the sumcheck protocol, we will aim to ensure that "bad" challenges do not occur in the non-interactive proof generated by a computationally bounded prover, except with negligible probability.

Background: Correlation Intractable Hash Functions [CGH04]. At a high level, a hash function family H is correlation intractable (CI) for a relation $\mathcal{R}(x, y)$ if it is computationally hard, given a random key k, to find any input x such that $(x, H_k(x)) \in \mathcal{R}$. As observed in [HMR08], CI hash functions for the broad class of sparse relations suffice to prove soundness of the Fiat-Shamir transform, whenever the initial protocol is a statistically sound proof.

Very recent beautiful constructions of CI hash functions for restricted classes of relations, and their applications to instantiating Fiat-Shamir for Sigma protocols, led to the first realization of non-interactive zero-knowledge (NIZK) for NP from LWE [CLW18, PS19]. The CI hash functions specifically constructed in these works generate keys in two modes: a Gen mode and a StatGen mode. The Gen mode samples a random key from a fixed distribution, while the StatGen mode

samples a key dependent on the function f (computable in a-priori bounded time) against which correlation-intractability is desired. Both modes are indistinguishable, and the StatGen mode with key sampled according to f has the following additional informal guarantee:

$$\Pr_{k \leftarrow \mathsf{StatGen}(1^\lambda, f)}[\exists x \ : \ H(k,x) = f(x)] = \mathsf{negl}(\lambda).$$

Since Gen and StatGen modes are indistinguishable, this implies that for polynomial-sized $\mathcal{A}$,

$$\Pr_{k \leftarrow \mathsf{Gen}(1^\lambda)}[\mathcal{A}(k) \to x \ : \ H(k,x) = f(x)] = \mathsf{negl}(\lambda).$$

Non-Interactive Sumcheck via Fiat-Shamir: First Attempt. We consider applying the Fiat-Shamir transform [FS86] instantiated with CI hash functions to the sumcheck protocol, in order to get a non-interactive argument in the common reference string (CRS) model. Specifically, the CRS contains a hash function key, and the prover non-interactively computes verifier challenges as the outcome of the hash function applied to the transcript (or part of the transcript) so far.

As discussed previously, we would need to ensure that the prover cannot non-interactively produce transcripts that contain "bad" verifier challenges. The CI hash functions that we just discussed can be used to avoid exactly such challenges, as long as they are (relatively) efficiently computable. First, we consider using an independent hash function key k_i for each round $i \in [\nu(\lambda)]$ of the interactive sumcheck protocol. For $i \in [\nu(\lambda)]$, the prover creates a partial transcript τ_i of interactive sumcheck until round i and emulates the i^{th} verifier message by computing $H(k_i, \tau_i)$[3]. In the i^{th} round, we would like to argue that $H(k_i, \tau_i)$ is not a "bad" challenge, which we recall is any of the $\leq d$ roots of $p_i^*(\cdot) - q_i^*(\cdot)$.

If roots of $p_i^*(\cdot) - q_i^*(\cdot)$ were efficiently computable given the transcript, then we would almost be done. We say *almost*, because CI hash functions from LWE as defined and constructed in [CLW18, PS19] rule out the possibility of finding inputs for which the output of the hash corresponds to a *single, efficiently computable* "bad" challenge – and on the other hand, we have d possible "bad" challenges and we must rule them all out. Luckily, we observe (via a simple hybrid argument) that *any somewhere-statistical* CI hash function with Gen and StatGen modes as described above, readily extends to rule out d possible "bad" challenges, with a corresponding loss in security, that is, for any set of d efficiently computable functions $f_1, \ldots, f_d$ and any polynomial sized adversary $\mathcal{A}$,

$$\Pr_{k \leftarrow \mathsf{Gen}(1^\lambda)}[\exists j \in [d] \ : \ (\mathcal{A}(k) \to x) \wedge (H(k,x) = f_j(x))] = d \cdot \mathsf{negl}(\lambda).$$

What turns out to be a more serious issue, is the fact that roots of $p_i^*(\cdot) - q_i^*(\cdot)$ may *not* be efficiently computable. Recall that

$$q_i^*(\cdot) := \sum_{x_{i+1}, \ldots, x_\nu \in B} p^*(r_1^*, \ldots, r_{i-1}^*, \cdot, x_{i+1}, \ldots, x_\nu).$$

which, if done naively, takes time $O(|B|^\nu)$ time to compute.

[3] In our actual construction of non-interactive sumcheck, we will not need to hash all of τ_i, only the most recent prover message.

A solution to this problem could be to define functions f_i for $i \in [\nu]$ that are somehow hardwired with correctly computed functions $q_i^*(\cdot)$. But even restricting ourselves to a selective attacker, while p^* may be known prior to generating the CRS, the values $r_1^*, \ldots, r_{i-1}^*$ will be controlled by the prover and cannot possibly be known prior to generating the CRS. This means that f_i would be required to guess these values. Denoting these guesses by $r_1', \ldots, r_i'$, we could hope to hardwire $q_i'(\cdot)$ where

$$q_i'(\cdot) := \sum_{x_{i+1},\ldots,x_\nu \in B} p^*(r_1', \ldots, r_{i-1}', \cdot, x_{i+1}, \ldots, x_\nu).$$

Now on input $p_i^*(\cdot)$, the function f_i can efficiently compute the roots of the polynomial $p_i^*(\cdot) - q_i'(\cdot)$ by the Cantor-Zassenhaus algorithm [CZ81]. In the case that our guesses were correct, that is, if $r_j' = r_j^*$ for all $j \in [i-1]$, f_i will have computed the correct evaluation $q_i^*(\cdot)$ where

$$q_i^*(\cdot) := \sum_{x_{i+1},\ldots,x_\nu \in B} p^*(r_1^*, \ldots, r_{i-1}^*, \cdot, x_{i+1}, \ldots, x_\nu),$$

for transcript $\tau_i = (p_1^*, r_1^*, \ldots, p_i^*)$, and f_i will correctly output the "bad" challenge for round i. In this case, when using $\mathsf{StatGen}$ key k_i associated with function f_i, our CI property will guarantee that the probability that the i^{th} hash output equals the i^{th} function output will be negligible.

Unfortunately, the function f_i for $i \in [\nu]$ can only guess the correct hash function outputs, that is $r_j^* = r_j'$ for $j \in [i-1]$, with probability $1/|\mathbb{F}|^{i-1} \geq 1/|\mathbb{F}|^\nu$. In order for our guesswork to be meaningful, we must have a hash function with an impossibly stronger security guarantee, on the order of $|\mathbb{F}|^{-\nu}$ (whereas the outputs of the hash function are of size $\log |\mathbb{F}|$). Clearly, such a hash function cannot exist. One could alternatively try to increase the space of challenges in each round and rely on complexity leveraging to argue that it is possible to guess previous challenges while still meaningfully contradicting CI in the current step. However, this growth in parameters becomes unsustainable quickly: it limits ν to being at most $c \log \log \lambda$ for some constant $c < 1$ [PW10].

Resolution via Lossy Correlation Intractable Hashing. In order to overcome this problem, we will make it possible to *artificially* decrease the output space of the hash function family. To this end, we will develop and use lossy correlation-intractable hash functions, which in addition to satisfying the CI properties discussed above, allow for an *extra lossy mode*. In this mode, the space of outcomes (or image) of the hash function will be restricted to being much smaller than its co-domain. It may at first appear that making a hash function lossy may interfere with correlation intractability. But we show that using these three modes, one can perform a careful sequence of hybrid experiments to argue soundness.

Constructing Lossy Correlation Intractable Hash Functions. Our construction of lossy correlation intractable hash functions will combine sufficiently lossy trapdoor functions [PW08] with CI hash functions [PS19], both of which can be obtained based on (sub-exponential) LWE.

A lossy trapdoor function (introduced in [PW08]) is, roughly, a keyed family of functions where keys can be generated in two modes: an injective mode and a lossy mode. The injective mode has a corresponding trapdoor which can be used to perform efficient inversions. The lossy mode, in contrast, information theoretically loses information about its input. In other words,

this mode restricts the size of the function's output space to a space bounded by 2^{λ^ϵ} for some $0 < \epsilon < 1$[4]. Additionally, both modes are computationally indistinguishable.

We construct lossy CI hash functions by concatenating CI hash functions with lossy trapdoor functions. The lossy CI hash key will consist of (k, ik) where k is a key for a (regular) CI hash function and ik is a key for the lossy trapdoor function. In more detail, our lossy CI hash function will have keys generated in three modes as follows:

- In Gen mode, the lossy CI hash outputs (k, ik) for k sampled according to the CI Gen algorithm, and ik sampled as an injective key for the lossy trapdoor function.
- In $\mathsf{StatGen}$ mode corresponding to an efficiently computable function f, the lossy CI hash outputs (k, ik) for k sampled according to the CI $\mathsf{StatGen}$ algorithm for function $f_{\mathsf{td}} = f(\mathsf{Inv}(\mathsf{td}), \cdot)$, and $(\mathsf{ik}, \mathsf{td})$ sampled as an injective key and trapdoor pair for the lossy trapdoor function. Here, Inv denotes the (trapdoor) inversion algorithm for the lossy trapdoor function.
- In $\mathsf{LossyGen}$ mode, the lossy CI hash outputs (k, ik) for k sampled according to the CI Gen algorithm, and ik sampled as a *lossy* key for the lossy trapdoor function.

Evaluating the lossy CI hash function on input x simply involves first computing y as the output of the lossy trapdoor function with key ik on input x, and then evaluating the CI hash function with key k on input y. More formally, $\mathcal{H}'.\mathsf{Hash}((k, \mathsf{ik}), x) = \mathcal{H}.\mathsf{Hash}(k, \mathsf{Eval}(\mathsf{ik}, x))$ where $\mathcal{H}$ and $\mathcal{H}'$ denote the underlying CI hash function and resulting lossy CI hash function respectively, and Eval denotes the evaluation algorithm for the lossy trapdoor function.

These modes are easily seen to be indistinguishable from each other by key indistinguishability of the underlying CI hash function family and the lossy trapdoor function family. Moreover, the $\mathsf{StatGen}$ mode continues to satisfy statistical correlation intractability, as before, due to the correctness of trapdoor inversion. To see this, note that on input x, the evaluator computes $\mathcal{H}.\mathsf{Hash}(k, y)$ where $y = \mathsf{Eval}(\mathsf{ik}, x)$. The function $f_{\mathsf{td}}(y)$ is therefore equal to $f(\mathsf{Inv}(\mathsf{td}, y))$, which is simply $f(x)$. Finally, in $\mathsf{LossyGen}$ mode, the underlying lossy function is first applied to the input x, which restricts the space of possible outcomes to 2^{λ^ϵ} for some $0 < \epsilon < 1$.

Fiat-Shamir for Sumcheck via Lossy CI Hash Functions. Our proposed protocol is exactly the same as discussed before, except we use *lossy CI hash functions* instead of (regular) CI hash functions to perform Fiat-Shamir. Specifically, the CRS consists of a sequence of keys $\{k_i\}_{i \in [\nu(\lambda)]}$ for the lossy CI hash function family, with hash output size $\log |\mathbb{F}| = \lambda$. The non-interactive prover proceeds in $\nu(\lambda)$ steps, where it generates a partial sumcheck transcript τ_i (as before) and then computes $\mathsf{Hash}(k_i, x_i)$ to obtain the next verifier challenge. We will assume that:

- In lossy mode, the output space of the lossy CI hash function is compressed to T,
- All $\mathsf{poly}(T)$ size adversaries have $\mathsf{negl}(T)$ advantage in distinguishing keys generated in Gen or $\mathsf{StatGen}$ modes or in breaking statistical correlation intractability, and
- All $\mathsf{poly}(T')$ size adversaries have $\mathsf{negl}(T')$ advantage in distinguishing keys generated in $\mathsf{LossyGen}$ mode from those generated in any other mode,

[4] As described in [PW08], the output space is only compressed to $2^{c\lambda}$ for $0 < c < 1$, however, stronger compression can be obtained by relying on sub-exponential hardness of LWE.

where $T' << T$, and for simplicity in the rest of this discussion, we set $T = 2^{\lambda^\epsilon}$ and $T' = 2^{\lambda^{\epsilon^2}}$ for some constant $0 < \epsilon < 1$. These properties can be simultaneously achieved assuming sub-exponential hardness of LWE (where the exact hardness parameter will determine ϵ).

Next, we rely on *round-by-round* soundness of the sumcheck protocol: very roughly, this property states that there exists a deterministic, public State function that can be used to keep track of the exact step at which a malicious prover switches from trying to prove false claims to proving true claims. It is clear that, for any prover $\mathcal{P}^*$ that breaks soundness of non-interactive sumcheck, such a step would exist. For convenience, we denote this by step i in the discussion below, where $i \in [\nu]$. Our goal will be to show that any polynomial-size prover that switches from false to true claims in step i contradicts correlation intractability of the i^{th} hash function family. This will establish that such a prover cannot exist: which in turn (by the structure of sumcheck) implies that our compressed sumcheck protocol is sound.

Recall that our main technical bottleneck was the following: to contradict statistical correlation intractability, we need to have an *efficiently computable* function f_i that computes "bad" challenges given the transcript, and this turns out to be inefficient. Specifically, f_i needs to output a root of the polynomial $p_i^*(\cdot) - q_i^*(\cdot)$, where

$$q_i^*(\cdot) := \sum_{x_{i+1},\ldots,x_\nu \in B} p^*(r_1^*, \ldots, r_{i-1}^*, \cdot, x_{i+1}, \ldots, x_\nu).$$

and this polynomial is *not efficiently computable* given $r_1^*, \ldots, r_{i-1}^*$. To overcome this problem, we previously suggested *guessing* $r_1^*, \ldots, r_{i-1}^*$ and using them to hardwire q_i^* into the function f_i. Unfortunately since $r_1^*, \ldots, r_{i-1}^*$ are dynamically generated by the prover depending on the CRS, a naive guess is only correct with probability $\frac{1}{|\mathbb{F}|^{i-1}}$. Guessing with such a small probability is meaningless, as it gives us no advantage in contradicting correlation intractability of the i^{th} hash function.

However, we note that when contradicting correlation intractability of the i^{th} hash function family, we only need to guess the outcomes of *the previous* $i-1$ hash functions. As such, we can first (T'-indistinguishably) switch to generating the first $i-1$ keys in lossy mode[5]. This makes it so that each r_j^* for $j \in [i-1]$ takes at most $T = 2^{\lambda^\epsilon}$ values, and $r_1^*, \ldots, r_{i-1}^*$ can collectively be guessed with probability $2^{(i-1)\cdot\lambda^\epsilon}$, which is upper-bounded by $2^{\nu\lambda^\epsilon}$. This means that we can correctly guess and hardwire $q_i^*(\cdot)$ with probability $2^{\nu\lambda^\epsilon}$. Given a correct guess, the Cantor-Zassenhaus algorithm [CZ81] can now be applied to sample the d roots of $p_i^*(\cdot) - q_i^*(\cdot)$.

Next, recall that it suffices to break statistical correlation intractability of the hash function with advantage better than $\mathsf{negl}(T)$. Therefore, this argument goes through as long as $\mathsf{poly}(T) \geq 2^{\nu\lambda^\epsilon}$. (To better understand the type of parameters we achieve, the reader could imagine setting $\nu = \lambda^\epsilon$ and recall that $T = 2^\epsilon$. We emphasize that these parameters remain consistent across *all* ν hash functions and do not increase from one step to the next.) Additionally, for convenience, the output space of the hash function, which is $\log|\mathbb{F}|$, will be restricted to be equal to λ. This leads to ν (the number of variables) being restricted to sublinear in λ. We note that we can actually tolerate $\nu = w(\lambda)$ for any polynomial $w(\cdot)$ by scaling parameters appropriately such that $\log|\mathbb{F}| = w'(\lambda)$ for a different (larger) polynomial $w'(\cdot)$.

[5]This involves a somewhat subtle argument: specifically, we must establish that the prover continues to go from a false to a true claim in step i with non-negligible probability, even when the first $i-1$ keys are generated in lossy mode. This requires a reduction that efficiently checks if the prover went from a false to a true claim in any given step, which takes time $|B|^\nu$. Therefore, we will need to set T' such that $\mathsf{poly}(T') \geq |B|^\nu$ for some polynomial $\mathsf{poly}(\cdot)$

This completes our discussion on applying lossy CI hash functions to compress the sumcheck protocol. We now show how this can be used to establish PPAD hardness.

PPAD Hardness. To establish the hardness of Nash under sub-exponential LWE, we rely on the beautiful work of Choudhuri et. al. [CHK+19a]. Specifically, [CHK+19a] showed that *adaptive unambiguous soundness of Fiat-Shamir for sumcheck* helps reduce #SAT instances to rSVL instances. Roughly, unambiguity requires that it be (computationally) hard to find *two accepting proofs*, even for a true statement. While our non-interactive sumcheck protocol does satisfy unambiguity due to the special structure of sumcheck, we unfortunately fall short of proving full-fledged adaptive soundness. This is because the polynomial p^* needs to be known in advance in order to precompute true claims q_i^* at each step i for $i \in [\nu]$.

However, we observe that the proof in [CHK+19a] does not require full adaptivity over the choice of p^*. Specifically, they only require partial adaptive unambiguous soundness – where they allow a prefix $r_1^*, \ldots, r_j^*$ to be chosen adaptively, and they require non-interactive sumchecks proofs of the form

$$\sum_{x_{j+1},\ldots,x_\nu \in B} p^*(r_1^*, \ldots, r_j^*, x_{j+1}, \ldots, x_\nu)$$

to satisfy soundness and unambiguity. We also observe that in their construction, the prefix $r_1^*, \ldots, r_j^*$ is generated as a result of a series of recursive computation steps, more specifically by either evaluating the hash function on some input and appending the result, or selecting a value in $[d+1]$. As such, the proof is only required to be sound for prefixes that take values in the union of $[d+1]$ and the support of outputs of the hash function.

The proof techniques we developed so far for sumcheck allow us to prove soundness for p^* fixed selectively and any prefixes that take values in the union of $[d+1]$ and the space of outputs of the hash function. In more detail, we note that the space from which prefixes are chosen can be indistinguishably restricted to be relatively small (by switching the relevant keys in the CRS to lossy mode). This allows our reduction to guess these values, and use a prover that breaks partial adaptive unambiguity or partial adaptive soundness to contradict CI with sufficient probability. This completes an overview of our techniques establishing PPAD hardness.

2 Preliminaries

2.1 (Somewhere Statistical) Correlation Intractability

Definition 2.1 (T-Correlation Intractability)**.** [CLW18] For a given function ensemble $F = \{F_\lambda : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}\}_{\lambda\in\mathbb{N}}$, a hash family $\mathcal{H} = \{h_\lambda : \{0,1\}^{s(\lambda)} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}\}_{\lambda\in\mathbb{N}}$ is said to be *T-correlation intractable with respect to F* if for every $\mathsf{poly}(T)$-size $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$,

$$\Pr_{\substack{k\leftarrow\mathcal{H}.\mathsf{Gen}(1^\lambda)\\ x\leftarrow\mathcal{A}(k)}}[\mathcal{H}.\mathsf{Hash}(k,x) = F(x)] = \mathsf{negl}(T(\lambda)).$$

If $\mathcal{F}$ is a collection of function ensembles, then $\mathcal{H}$ is said to be *uniform correlation intractable with respect to the collection $\mathcal{F}$* if for every $\mathsf{poly}(T)$-size $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, there exists a negligible function

$\nu(\cdot)$ such that for every $F \in \mathcal{F}$,

$$\Pr_{\substack{k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \\ x \leftarrow \mathcal{A}(k)}}[\mathcal{H}.\mathsf{Hash}(k, x) = F(x)] = \nu(T(\lambda)).$$

Definition 2.2 (T-Somewhere Statistical Correlation Intractability (SSCI))**.** [CLW18] Given a collection $\mathcal{F}$ of function ensembles, we say that a hash family $\mathcal{H}$ is *somewhere statistically correlation intractable with respect to* $\mathcal{F}$ if there is an additional key generation algorithm $\mathsf{StatGen}$ with the following properties:

- **Syntax.** $\mathsf{StatGen}(1^\lambda, f_\lambda)$ takes as input the security parameter λ as well as the function $f_\lambda \in F$. It outputs a hash key k.
- **Security.** For any function ensemble $F \in \mathcal{F}$, the following two properties hold:
  - **Key Indistinguishability (KI).** For every $\mathsf{poly}(T)$-size adversary $\mathcal{D}$, there exists a negligible function $\nu(\cdot)$ such that

    $$|\Pr_{k \leftarrow \mathcal{H}.\mathsf{StatGen}(1^\lambda, f_\lambda)}[\mathcal{D}(k) = 1] - \Pr_{k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)}[\mathcal{D}(k) = 1]| = \nu(T(\lambda)).$$

  - **T-Statistical Correlation Intractability (SCI).**

    $$\Pr_{k \leftarrow \mathcal{H}.\mathsf{StatGen}(1^\lambda, f_\lambda)}\left[\exists x \in \{0,1\}^{n(\lambda)} \ : \ \mathcal{H}.\mathsf{Hash}(k, x) = F(x)\right] = \mathsf{negl}(T(\lambda)).$$

    That is, with overwhelming probability over the choice of $k \leftarrow \mathsf{StatGen}(1^\lambda, f_\lambda)$, input-output pairs satisfying F do not exist.

Theorem 2.1. *[PS19] For every set of polynomials $n(\cdot), m(\cdot)$ and $\rho(\cdot)$, assuming the sub-exponential hardness of LWE, there exists a constant $0 < \epsilon < 1$ for which there exist T-secure SSCI hash functions, where $T = 2^{\lambda^\epsilon}$, according to Definition 2.1 for all functions that are computable in time $\rho(\lambda)$ and map $n(\lambda)$-bit inputs to $m(\lambda)$-bit outputs.*

2.2 Lossy Trapdoor Functions

Lossy trapdoor functions were first defined and constructed (based on LWE) in an influential work of Pass and Waters [PW08]. We use a variant of lossy trapdoor functions where the output space (in lossy mode) is 2^{λ^ϵ} for some constant $0 < \epsilon < 1$, with key size and security parameter λ, input space $\{0,1\}^\lambda$, and output space $\{0,1\}^{\mathsf{poly}(\lambda)}$. We require sub-exponential indistinguishability between lossy and injective modes. We also assume (for simplicity in our proofs) that function outcomes are invertible for all keys generated in injective mode, and lossy for all keys generated in lossy mode. Lossy trapdoor functions with these properties can be obtained based on rate-1 perfectly correct FHE [BDGM19], via rate-1 OT [DGI+19]. We believe that our techniques also work with lossy trapdoor functions that are *almost-always lossy* and *almost-always injective* (over the randomness of sampling lossy/injective keys, as is true for the LWE-based construction in [PW08]).

Definition 2.3 ((ϵ, T')-secure Lossy Trapdoor Function)**.** Let $\eta = \eta(\cdot)$ and $m = m(\cdot)$ denote polynomial functions. A function family $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda \in \mathbb{N}}$ is a (ϵ, T')-lossy trapdoor function family if $\mathcal{G}$ satisfies the following properties:

- **Syntax.** The following algorithms exist:
  - $\mathsf{InjGen}(1^\lambda)$ is a probabilistic polynomial time algorithm that on input security parameter λ outputs an injective functions' index key ik, and a corresponding trapdoor td.
  - $\mathsf{LossyGen}(1^\lambda)$ is a probabilistic polynomial time algorithms that takes input security parameter λ and outputs a lossy functions' index key ik.
  - $\mathsf{Eval}(\mathsf{ik}, x)$ is a polynomial time algorithm which takes as input index key ik and function input $x \in \{0,1\}^{\eta(\lambda)}$ and outputs function output $y \in \{0,1\}^{m(\lambda)}$.
  - $\mathsf{Inv}(\mathsf{ik}, y)$ is a polynomial time algorithm which takes as input an injective index key ik and function output $y \in \{0,1\}^{m(\lambda)}$, and outputs function input $x \in \{0,1\}^{\eta(\lambda)}$.
- **Security.** The following properties hold:
  - **Key Indistinguishability (KI).** For every $\mathsf{poly}(T'(\lambda))$-size adversary $\mathcal{D}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that:
$$|\Pr_{\mathsf{ik}\leftarrow\mathcal{G}.\mathsf{LossyGen}(1^\lambda)}[\mathcal{D}(\mathsf{ik})=1] - \Pr_{\mathsf{ik}\leftarrow\mathcal{G}.\mathsf{InjGen}(1^\lambda)}[\mathcal{D}(\mathsf{ik})=1]| = \mathsf{negl}(T'(\lambda))$$
  - **Injectivity and Trapdoor Inversion.** For every $\mathsf{ik} \in \mathsf{Supp}(\mathsf{InjGen})$, $g_{\mathsf{ik}} \in G_\lambda : \{0,1\}^{n(\lambda)} \to \{0,1\}^{\eta(\lambda)}$ is injective. Furthermore, for every $x \in \{0,1\}^{n(\lambda)}$, $\mathsf{Inv}(\mathsf{td}, \mathsf{Eval}(\mathsf{ik}, x)) = x$.
  - **Lossiness.** For every $s \in \mathsf{Supp}(\mathsf{LossyGen})$, $g_{\mathsf{ik}} \in G_\lambda : \{0,1\}^{n(\lambda)} \to \{0,1\}^{\eta(\lambda)}$ is lossy, so its range is of size at most 2^{λ^ϵ}.
  - **Evaluation.** For every $\mathsf{ik} \in \mathsf{Supp}(\mathsf{InjGen}) \cup \mathsf{Supp}(\mathsf{LossyGen})$ and every $x \in \{0,1\}^{n(\lambda)}$, $\mathsf{Eval}(\mathsf{ik}, x) = g_{\mathsf{ik}}(x)$.

Theorem 2.2. *[PW08, BDGM19] Assuming sub-exponential hardness of LWE, there exists a constant $0 < \epsilon < 1$ for which there exist $(\epsilon, 2^{\lambda^{\epsilon^2}})$ lossy trapdoor functions according to Definition 2.3.*

2.3 Round-by-Round Soundness

In what follows, we define round-by-round soundness [CCH+18].

Definition 2.4 (Round-by-Round Soundness)**.** [CCH+18] Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a $\nu(\lambda)$-message public coin interactive proof system for language $\mathcal{L}$. For any $x \in \{0,1\}^*$, and any prefix τ of a protocol transcript, let $\Pi_{\mathcal{V}}(x, \tau)$ denote the next message function of $\mathcal{V}$ on input x when the partial transcript is τ. We say that Π is round-by-round sound if there exists a deterministic (not necessarily efficiently computable) function State that takes as input an instance x and a transcript prefix τ and outputs either accept or reject such that the following properties hold:

1. If $x \notin \mathcal{L}$, then $\mathsf{State}(x, \emptyset) = \mathsf{reject}$, where $\emptyset$ denotes the empty transcript.
2. There exists a polynomial $d(\cdot)$ such that if $\mathsf{State}(x, \tau) = \mathsf{reject}$ for a transcript prefix τ, then for every potential prover message α it holds that[6]
$$\Pr_{\beta\leftarrow\Pi_{\mathcal{V}}(x,\tau|\alpha)}[\mathsf{State}(x, \tau|\alpha|\beta) = \mathsf{accept}] = d(\lambda) \cdot 2^{-|\beta|}.$$

[6]Here, we point out that we modify the definition in [CCH+18] to replace $\mathsf{negl}(\lambda)$ with $d(\lambda) \cdot 2^{-|\beta|}$ on the right hand side of the following equation.

3. For every transcript τ, if $\mathsf{State}(x, \tau) = \mathsf{reject}$ then $\Pi_{\mathcal{V}}(x, \tau) = 0$.

3 Lossy Somewhere-Statistical Corelation Intractable Hash Functions

Definition 3.1 ((ϵ, T, T')-Lossy Statistical Correlation Intractability (Lossy SSCI))**.** Given a collection $\mathcal{F}$ of function ensembles, we say that a hash family $\mathcal{H}$ is *lossy somewhere statistically correlation intractable with respect to* $\mathcal{F}$ if it is T-SSCI for $\mathcal{F}$ according to Definition 2.2 and there is an additional key generation algorithm $\mathsf{LossyGen}$ with the following properties:

- **Syntax.** $\mathsf{LossyGen}(1^\lambda)$ takes as input security parameter λ. It outputs hash key k.
- **Security.** For any function ensemble $F \in \mathcal{F}$, the following two properties hold:
  - **Key Indistinguishability (KI).** For every $\mathsf{poly}(T')$-size adversary $\mathcal{D}$, there exists a negligible function $\nu(\cdot)$ such that

$$|\Pr_{k \leftarrow \mathcal{H}.\mathsf{LossyGen}(1^\lambda)}[\mathcal{D}(k) = 1] - \Pr_{k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)}[\mathcal{D}(k) = 1]| = \nu(T'(\lambda)).$$

  - **Lossiness.** There exists a constant $0 < \epsilon < 1$, such that for every $k \in \mathcal{H}.\mathsf{LossyGen}(1^\lambda)$,

$$|\{\mathcal{H}.\mathsf{Hash}(k, x)\}_{x \in \{0,1\}^{n(\lambda)}}| \leq 2^{\lambda^\epsilon}.$$

Theorem 3.1. *For every polynomial $\rho(\cdot)$, every $T'(\cdot) < T(\cdot)$, if there exists a polynomial $\rho'(\cdot)$ such that there exists a $T(\lambda)$-secure SSCI hash function family $\mathcal{H}$ for the family of functions computable in time $(\rho + \rho')(\lambda)$ according to Definition 2.2 and $(\epsilon, T'(\lambda))$-secure lossy trapdoor function family $\mathcal{G}$ according to Definition 2.3 where trapdoor inversion takes time $\rho'(\lambda)$ in the worst case, then there exists an $(\epsilon, T(\lambda), T'(\lambda))$-secure lossy SSCI hash function family $\mathcal{H}'$ for the family of functions computable in time $\rho(\lambda)$ according to Definition 3.1.*

Proof. We describe a construction which we prove satisfies the properties in Definition 3.1.

Construction 3.1. *Let $\mathcal{G} = \{\mathcal{G}_\lambda : \{0,1\}^{s'(\lambda)} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{\eta(\lambda)}\}_{\lambda \in \mathbb{N}}$ be a $(\epsilon, T'(\lambda))$-secure lossy trapdoor function and $\mathcal{H} = \{h_\lambda : \{0,1\}^{s(\lambda)} \times \{0,1\}^{\eta(\lambda)} \to \{0,1\}^{m(\lambda)}\}_{\lambda \in \mathbb{N}}$ be a $T(\lambda)$-secure SSCI hash function family for function family $\mathcal{F}_{\mathsf{td}}$ family as defined in the theorem statement. We define our $T(\lambda)$-secure lossy SSCI hash function family $\mathcal{H}' = \{h'_\lambda : \{0,1\}^{s(\lambda)+s'(\lambda)} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}\}_{\lambda \in \mathbb{N}}$ as follows:*

- $\mathcal{H}'.\mathsf{Gen}(1^\lambda)$*:*
  - *Computes* $k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$.
  - *Computes* $(\mathsf{ik}, \mathsf{td}) \leftarrow \mathcal{G}.\mathsf{InjGen}(1^\lambda)$.
  - *Outputs* $k' = (k, \mathsf{ik})$.
- $\mathcal{H}'.\mathsf{StatGen}(1^\lambda, f_\lambda)$*:*
  - *Computes* $(\mathsf{ik}, \mathsf{td}) \leftarrow \mathcal{G}.\mathsf{InjGen}(1^\lambda)$.
  - *Sets* $f_{\mathsf{td},\lambda}(\cdot) = f_\lambda(\mathcal{G}.\mathsf{Inv}(\mathsf{td}, \cdot))$.

- *Computes* $k \leftarrow \mathcal{H}.\mathsf{StatGen}(1^\lambda, f_{\mathsf{td},\lambda})$.
- *Outputs* $k' = (k, \mathsf{ik})$.

- $\mathcal{H}'.\mathsf{LossyGen}(1^\lambda)$*:*
  - *Computes* $k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$.
  - *Computes* $\mathsf{ik} \leftarrow \mathcal{G}.\mathsf{LossyGen}(1^\lambda)$.
  - *Outputs* $k' = (k, \mathsf{ik})$.
- $\mathcal{H}'.\mathsf{Hash}(k' = (k, \mathsf{ik}), x)$*:*
  - *Outputs* $\mathcal{H}.\mathsf{Hash}(k, \mathcal{G}.\mathsf{Eval}(\mathsf{ik}, x))$.

Key Indistinguishability. Informally, the outputs of StatGen can be shown to be indistinguishable from the outputs of Gen by the key indistinguishability property of the underlying hash function family $\mathcal{H}$. Specifically, if there exists an adversary $\mathcal{A}$ that breaks T-indistinguishability of $\mathcal{H}'.\mathsf{Gen}$ and $\mathcal{H}'.\mathsf{StatGen}$, then there exists adversary $\mathcal{B}$ that obtains challenge key k from the challenger for the key indistinguishability game of $\mathcal{H}$, samples index key $s \leftarrow \mathcal{G}.\mathsf{InjGen}(1^\lambda)$, and outputs $\mathcal{A}(k, \mathsf{ik})$. Adversary $\mathcal{B}$ would succeed in distinguishing the two experiments, where $k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$ and $k \leftarrow \mathcal{H}.\mathsf{StatGen}(1^\lambda, f_\lambda)$, with the same advantage as $\mathcal{A}$.

Informally, the outputs of LossyGen can be shown to be indistinguishable from the outputs of Gen (and StatGen) by the key indistinguishability property of the underlying hash function family $\mathcal{H}$. If there exists an adversary $\mathcal{A}$ that breaks T'-indistinguishability of $\mathcal{H}'.\mathsf{Gen}$ and $\mathcal{H}'.\mathsf{LossyGen}$, then there exists adversary $\mathcal{B}$ that obtains challenge index key ik from the challenger for the $\mathcal{G}$ lossy key indistinguishability game, samples key $k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$, and outputs $\mathcal{A}(k, \mathsf{ik})$. Adversary $\mathcal{B}$ would succeed in distinguishing the two experiments, where $\mathsf{ik} \leftarrow \mathcal{G}.\mathsf{InjGen}(1^\lambda)$ and $\mathsf{ik} \leftarrow \mathcal{G}.\mathsf{LossyGen}(1^\lambda)$, with the same advantage as $\mathcal{A}$.

Statistical Correlation Intractability. This property follows from the SCI property of the hash function family $\mathcal{H}$ and the injectivity of the trapdoor function. Towards a contradiction, suppose there exists a function $f = f_\lambda \in \mathcal{F}_\lambda$ and polynomial $w(\cdot)$ such that for infinitely many $\lambda \in \mathbb{N}$,

$$\Pr_{k' \leftarrow \mathcal{H}'.\mathsf{StatGen}(1^\lambda, f_\lambda)}[\exists x \ : \ \mathcal{H}'.\mathsf{Hash}(k', x) = f(x)] \geq \frac{1}{w(T(\lambda))}.$$

Fix any (x, k') such that $\mathcal{H}'.\mathsf{Hash}(k', x) = f(x)$, where $k' \in \mathsf{Supp}(\mathcal{H}'.\mathsf{StatGen}(1^\lambda, f))$ can be parsed as $k' = (k, \mathsf{ik})$, and define

$$y = \mathcal{G}.\mathsf{Eval}(\mathsf{ik}, x). \tag{1}$$

By construction of $\mathcal{H}'.\mathsf{Hash}$ and substituting Equation (1), we have that

$$\mathcal{H}.\mathsf{Hash}(k, y) = \mathcal{H}.\mathsf{Hash}(k, \mathcal{H}.\mathsf{Eval}(\mathsf{ik}, x)) = \mathcal{H}'.\mathsf{Hash}(k', x). \tag{2}$$

Let td denote the trapdoor corresponding to injective index key ik. By injectivity and everywhere-inversion, we have

$$f(x) = f(\mathcal{G}.\mathsf{Inv}(\mathsf{td}, \mathcal{G}.\mathsf{Eval}(\mathsf{ik}, x))). \tag{3}$$

Define function $f_{\mathsf{td}} \in \mathcal{F}_{\mathsf{td}}$ as $f_{\mathsf{td}}(\cdot) = f(\mathcal{G}.\mathsf{Inv}(\mathsf{td}, \cdot))$. Substituting Equation (1) in Equation (3),

$$f(x) = f(\mathcal{G}.\mathsf{Inv}(\mathsf{td}, \mathcal{G}.\mathsf{Eval}(\mathsf{ik}, x))) = f(\mathcal{G}.\mathsf{Inv}(\mathsf{td}, y)) = f_{\mathsf{td}}(y).$$

Combining this with Equation (2) implies

$$\mathcal{H}.\mathsf{Hash}(k, y) = \mathcal{H}.\mathsf{Hash}(k', x) = f(x) = f_{\mathsf{td}}(y). \tag{4}$$

Therefore, for *any* (x, k') where $k' = (k, \mathsf{ik})$, td corresponds to the trapdoor for ik, and f_{td} is as defined above, we have that

$$\mathcal{H}'.\mathsf{Hash}(k', x) = f(x) \implies \mathcal{H}.\mathsf{Hash}(k, y) = f_{\mathsf{td}}(y). \tag{5}$$

Then, $\exists(\mathsf{ik}', \mathsf{td}') \in \mathsf{Supp}(\mathsf{InjGen}(1^\lambda))$ such that

$$\begin{aligned}
\Pr_{k \leftarrow \mathcal{H}.\mathsf{StatGen}(1^\lambda, f_\lambda)}[\exists y \ : \ \mathcal{H}.\mathsf{Hash}(k, y) = f_{\mathsf{td}'}(y)] &\geq \Pr_{\substack{k \leftarrow \mathcal{H}.\mathsf{StatGen}(1^\lambda, f_\lambda), \\ (\mathsf{ik}, \mathsf{td}) \leftarrow \mathsf{InjGen}(1^\lambda)}}[\exists y \ : \ \mathcal{H}.\mathsf{Hash}(k, y) = f_{\mathsf{td}}(y)] \\
&\geq \Pr_{k' \leftarrow \mathcal{H}'.\mathsf{StatGen}(1^\lambda, f_\lambda)}[\exists x \ : \ \mathcal{H}'.\mathsf{Hash}(k', x) = f(x)] \\
&\geq \frac{1}{w(T(\lambda))}
\end{aligned} \tag{6}$$

This contradicts the statistical correlation intractability of $\mathcal{H}$, as desired.

Lossiness. The lossiness of the resulting hash function $\mathcal{H}'$ follows by the lossiness of the underlying lossy trapdoor function family $\mathcal{G}$. Since $|\mathcal{G}.\mathsf{Eval}(\mathsf{ik}, x)| \leq 2^{\lambda^\epsilon}$ when index key $\mathsf{ik} \leftarrow \mathcal{G}.\mathsf{LossyGen}(1^\lambda)$, we have that $|\{\mathcal{H}.\mathsf{Hash}(k, \mathcal{G}.\mathsf{Eval}(\mathsf{ik}, x))\}_{x \in \{0,1\}^{n(\lambda)}}| \leq 2^{\lambda^\epsilon}$ when hash key $(k, \mathsf{ik}) \leftarrow \mathcal{H}'.\mathsf{LossyGen}(1^\lambda)$. □

Corollary 3.1. *Assuming sub-exponential hardness of LWE, there exists a constant $0 < \epsilon < 1$ for which there exist $(\epsilon, 2^{\lambda^\epsilon}, 2^{\lambda^{\epsilon^2}})$ lossy SSCI hash functions according to Definition 3.1.*

4 Non-Interactive Sumcheck

4.1 The Sumcheck Protocol

Construction 4.1 (Sumcheck). *[LFKN92, Sha92] The sumcheck protocol $\Pi = (\mathcal{P}, \mathcal{V})$ is an interactive proof for language $\mathcal{L}_{B,\mathbb{F},\nu,d} = \{(a, p) : \sum_{x_1,\ldots,x_\nu \in B} p(x_1, \ldots, x_\nu) = a\}$ consisting of field $\mathbb{F}$, subset $B \subset \mathbb{F}$, field elements $a \in \mathbb{F}$, and polynomials $p(\cdot)$ of individual degree d. Π involves the following ν rounds:*

1. *In Round 1,*
   - *$\mathcal{P}$ computes and sends the polynomial $p_1(\cdot) = \sum_{x_2,\ldots,x_\nu \in B} p(\cdot, x_2, \ldots, x_\nu)$.*
   - *$\mathcal{V}$ checks that $p_1(\cdot)$ is of degree at most d and that $a = \sum_{x_1 \in B} p_1(x_1)$. If the check passes, $\mathcal{V}$ samples uniform $r_1 \leftarrow \mathbb{F}$, sets $v_2 = p_1(r_1)$, and sends r_1.*
2. *In Round i for $i \in [\nu]$,*

- *$\mathcal{P}$ computes and sends the polynomial $p_i(\cdot) = \sum_{x_{i+1},\ldots,x_\nu \in B} p(r_1,\ldots,r_{i-1},\cdot,x_{i+1},\ldots,x_\nu)$.*
- *$\mathcal{V}$ checks that $p_i(\cdot)$ is of degree at most d and that $v_i = \sum_{x_i \in B} p_i(x_i)$. If $i = \nu$, it additionally checks that $v_\nu = p(r_1,\ldots,r_\nu)$. If the check passes, $\mathcal{V}$ samples uniform $r_i \leftarrow \mathbb{F}$, sets $v_{i+1} = p_i(r_i)$, and sends r_i to the prover.*

The verifier $\mathcal{V}$ accepts the proof for (a,p) if all checks in each round pass.

Theorem 4.1. *[LFKN92, Sha92] Let $\Pi = (\mathcal{P},\mathcal{V})$ be the interactive sumcheck protocol for language $\mathcal{L}_{B,\mathbb{F},\nu,d}$ in field $\mathbb{F}$ from Construction 4.1 with security parameter λ. For all $(a^*,p^*) \notin \mathcal{L}$ and for all unbounded cheating provers $\mathcal{P}^*$*

$$\Pr_{\mathcal{V}}[\mathsf{Output}_{\mathcal{V}}(\mathcal{P}^*(a^*,p^*) \leftrightarrow \mathcal{V}(a^*,p^*)) = 1] \leq \frac{\nu \cdot d}{|\mathbb{F}|} = \mathsf{negl}(\lambda)$$

where d is the individual degree of each variable in polynomial $p^*(\cdot)$ and $|\mathbb{F}| = \lambda^{\omega(1)}/(\nu \cdot d)$.

4.2 Instantiating Fiat-Shamir for Sumcheck

Definition 4.1 (Selective Soundness)**.** Let $\Pi = (\mathcal{P},\mathcal{V})$ denote a non-interactive sumcheck protocol in the CRS model for language $\mathcal{L}_{B,\mathbb{F},\nu,d}$ over field $\mathbb{F}$ with subset B, ν variables and individual degree d. We say that Π is selectively sound if for all $(a^*,p^*) \notin \mathcal{L}$, where $p^*(\cdot)$ is a ν-variate polynomial of individual degree d in each variable, and for all PPT cheating prover $\mathcal{P}^*$ we have

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Setup}(1^\lambda)\\ \tau^*\leftarrow\mathcal{P}(\mathsf{CRS},p^*,a^*)}}[\mathcal{V}(\mathsf{CRS},(a^*,p^*),\tau^*) = 1] = \mathsf{negl}(\lambda).$$

Theorem 4.2. *Let $B \subset \mathbb{F}$, and $\nu = \nu(\lambda)$. Let $\Pi = (\mathcal{P},\mathcal{V})$ be the interactive, statistically sound sumcheck protocol for $\mathcal{L}_{B,\mathbb{F},\nu,d}$ from Construction 4.1. There exists a polynomial $\rho(\cdot)$, such that if there exists an $(\epsilon, T(\lambda), T'(\lambda))$-secure lossy SSCI hash function family $\mathcal{H} : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$ where $n(\lambda) = d\log(|\mathbb{F}|)$ and $m(\lambda) = \log(|\mathbb{F}|)$ for functions $\mathcal{F}$ computable in time $\rho(d,\lambda)$ according to Definition 3.1, then the Fiat-Shamir paradigm will yield a non-interactive, computationally selectively sound sumcheck protocol satisfying Definition 4.1, where the verifier runs in time $\nu \cdot \mathsf{poly}(|B|,d,\log|\mathbb{F}|)$ and the prover runs in time $\mathsf{poly}(|B|^\nu, d^\nu, \log|\mathbb{F}|)$, as long as: there exists a polynomial $\mathsf{poly}(\cdot)$ such that $\mathsf{poly}(T) \geq \max(2^{\nu\lambda^\epsilon}, |B|^\nu, d^\nu)$, and $\mathsf{poly}(T') \geq \mathsf{max}(|B|^\nu, d^\nu)$.*

Proof. We describe a construction that we prove achieves the desired completeness and soundness.

Construction 4.2. *Let $\Pi = (\mathcal{P},\mathcal{V})$ be the interactive sumcheck proof for language $\mathcal{L}_{B,\mathbb{F},\nu,d} = \{(a,p) : \sum_{x_1,\ldots,x_\nu \in B} p(x_1,\ldots,x_\nu) = a\}$ for $B \subset \mathbb{F}$ and $\mathcal{H}$ be the $T(\lambda)$-secure lossy SSCI hash function family for function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$. We define the non-interactive protocol $\Pi' = (\mathcal{P}',\mathcal{V}')$ for language $\mathcal{L}_{B,\mathbb{F},\nu,d}$. The common reference string CRS consists of ν keys:*

- *$\{k_i \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)\}_{i\in[\nu]}$.*

The prover $\mathcal{P}'$ obtains input $(\mathsf{CRS},(a,p))$. It then performs the following steps:

1. *In Step 1, $\mathcal{P}'$*
   - *Computes the polynomial $p_1(\cdot) = \sum_{x_2,\ldots,x_\nu \in B} p(\cdot,x_2,\ldots,x_\nu)$ by invoking $\mathcal{P}$.*

- *Computes* $r_1 = \mathcal{H}.\mathsf{Hash}(k_1, p_1)$.
- *Sets* $\tau_1 = (p_1, r_1)$

2. *In Step i for $i \in [\nu]$, $\mathcal{P}'$*
   - *Computes* $p_i(\cdot) = \sum_{x_{i+1},\ldots,x_\nu \in B} p(r_1, \ldots, r_{i-1}, \cdot, x_{i+1}, \ldots, x_\nu)$ *by invoking* $\mathcal{P}$ *on* τ_{i-1}.
   - *Computes* $r_i = \mathcal{H}.\mathsf{Hash}(k_i, p_i)$.
   - *Sets* $\tau_i = (p_1, r_1, \ldots, p_i, r_i)$.
3. *At the end of this process, $\mathcal{P}'$ sends proof* $\tau = (p_1, r_1, \ldots, p_\nu, r_\nu)$.

The verifier $\mathcal{V}'$ obtains input $(\mathsf{CRS}, (a, p), \tau)$ *and it performs the following steps:*

1. *$\mathcal{V}'$ parses* $\tau = (p_1, r_1, \ldots, p_\nu, r_\nu)$.
2. *In Step 1, $\mathcal{V}'$*
   - *Checks that $p_1(\cdot)$ is of degree at most d and that* $a = \sum_{x_1 \in B} p_1(x_1)$ *by invoking* $\mathcal{V}$.
   - *Verifies that* $r_1 = \mathcal{H}.\mathsf{Hash}(k_1, p_1)$.
   - *Sets* $v_2 = p_1(r_1)$.
3. *In Step i for $i \in [\nu]$, $\mathcal{V}'$*
   - *Checks that $p_i(\cdot)$ is of degree at most d and that* $v_i = \sum_{x_i \in B} p_i(x_i)$ *by invoking* $\mathcal{V}$.
   - *Verifies that* $r_i = \mathcal{H}.\mathsf{Hash}(k_i, p_i)$.
   - *Sets* $v_{i+1} = p_i(r_i)$.
4. *If all checks pass, $\mathcal{V}'$ accepts if* $v_\nu = p(r_1, \ldots, r_\nu)$. *Otherwise, $\mathcal{V}'$ rejects.*

Completeness. The completeness of the non-interactive protocol Π' follows from the completeness of the interactive protocol Π.

Soundness. We will prove that selective soundness of our non-interactive protocol Π' follows from the round-by-round soundness of the interactive protocol Π and the lossy SSCI of $\mathcal{H}$.

Assume for the sake of contradiction that there exists $(a^*, p^*) \notin \mathcal{L}$, PPT prover $\mathcal{P}^*$, and polynomial $w'(\cdot)$ such that $\mathcal{P}^*$ breaks selective soundness of our non-interactive protocol Π', that is, for large enough $\lambda \in \mathbb{N}$,

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda) \\ \tau^* \leftarrow \mathcal{P}^*(\mathsf{CRS}, p^*)}} [\mathcal{V}'(\mathsf{CRS}, (a^*, p^*), \tau^*) = 1] \geq \frac{1}{w'(\lambda)}.$$

We parse the proof which $\mathcal{P}^*$ outputs as $\tau^* = (p_1^*, r_1^*, \ldots, p_\nu^*, r_\nu^*)$. By the round-by-round soundness property (Definition 2.4) of the underlying interactive protocol Π, there exists a deterministic public State function such that

$$\mathsf{State}((a^*, p^*), \emptyset) = \mathsf{reject} \text{ and}$$

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda) \\ \tau^* \leftarrow \mathcal{P}^*(\mathsf{CRS}, (a^*, p^*))}} [\mathsf{State}((a^*, p^*), \tau^*) = \mathsf{accept}] \geq \frac{1}{w'(\lambda)}.$$

By definition of the state function State, this means that for $\mathcal{P}^*$, $\exists i = i(\lambda) \in [\nu(\lambda)]$ such that

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Setup}(1^\lambda)\\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*))}}[\mathsf{State}((a^*,p^*),\tau^*_{i-1}) = \mathsf{reject} \wedge \mathsf{State}((a^*,p^*),\tau^*_i) = \mathsf{accept}] \geq \frac{1}{w(\lambda)},$$

where $w(\lambda) = \nu(\lambda)w'(\lambda)$ and we define partial transcript $\tau_j = (p^*_1, r^*_1, \ldots, p^*_j, r^*_j)$ for $j \in [\nu(\lambda)]$. We also note that the State function for the sumcheck protocol at round i, simply performs the verifier's checks and also checks if the latest claim is true, that is, it checks if

$$\sum_{x_{i+1},\ldots,x_\nu \in B} p^*(r^*_1, \ldots, r^*_{i-1}, r^*_i, x_{i+1}, \ldots, x_\nu) = p^*_i(r^*_i).$$

If these checks pass, it accepts, and otherwise rejects. We denote by $\mathbb{E}_i(a^*, p^*, \tau^*)$ the event that

$$\mathsf{State}((a^*,p^*),\tau^*_{i-1}) = \mathsf{reject} \wedge \mathsf{State}((a^*,p^*),\tau^*_i) = \mathsf{accept}.$$

We fix $\mathcal{P}^*$ and $i = i(\lambda)$. Then we construct the following hybrids in which we change the CRS, but leave the rest of the protocol the same. The first hybrid corresponds to the way the CRS is generated in the actual protocol. Both the other hybrids, and our reductions, obtain (non-uniform) input $i = i(\lambda)$.

$$\begin{aligned}
\mathsf{Hybrid}_0: \quad & k_j \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \text{ for } 0 < j \leq i-1\\
& k_i \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)\\
& k_j \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \text{ for } i < j \leq \nu\\
\mathsf{Hybrid}_1: \quad & k_j \leftarrow \mathcal{H}.\mathsf{LossyGen}(1^\lambda) \text{ for } 0 < j \leq i-1\\
& k_i \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)\\
& k_j \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \text{ for } i < j \leq \nu\\
\mathsf{Hybrid}_2: \quad & k_j \leftarrow \mathcal{H}.\mathsf{LossyGen}(1^\lambda) \text{ for } 0 < j \leq i-1\\
& k_i \leftarrow \mathcal{H}.\mathsf{StatGen}(1^\lambda, f_\lambda)\\
& k_j \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \text{ for } i < j \leq \nu
\end{aligned}$$

where for Hybrid_2, $f_\lambda \in \mathcal{F}_\lambda$ will be defined later in Figure 2.

Next, we complete the proof of soundness by analyzing the probability that the verifier accepts a proof in each of these hybrids.

Claim 4.1. *There exists polynomial $w(\cdot)$ such that for large enough $\lambda \in \mathbb{N}$,*

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_1\\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*))}}[\mathbb{E}_i(a^*,p^*,\tau^*)] \geq \frac{1}{w(\lambda)}.$$

Proof. We will prove that the claim follows from the KI property of $\mathcal{H}$. We can consider sub-hybrids $\mathsf{Hybrid}_{0,j}$ for $j \in [i]$. Let $\mathsf{Hybrid}_{0,0} = \mathsf{Hybrid}_0$ and $\mathsf{Hybrid}_{0,i} = \mathsf{Hybrid}_1$ where we move from $\mathsf{Hybrid}_{0,j-1}$ to $\mathsf{Hybrid}_{0,j}$ by switching the jth key from Gen to LossyGen for $j \in [i]$.

If the claim is not true, then there exists polynomial $w'(\lambda) = 2\nu(\lambda)w(\lambda)$ and some $j \in [i-1]$, such that

$$\left| \Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_{0,j-1} \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*))}} [\mathbb{E}_i(a^*,p^*,\tau^*)] - \Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_{0,j} \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*))}} [\mathbb{E}_i(a^*,p^*,\tau^*)] \right| \geq \frac{1}{w'(\lambda)},$$

We define adversary $\mathcal{B}$ in Figure 1 against the lossy key indistinguishability property of $\mathcal{H}$.

$\mathcal{B}$ interacts with a challenger Ch and with oracle access to $\mathcal{P}^*$, does the following:

- Obtain challenge key k from Ch.
- Sample $k_\ell \leftarrow \mathcal{H}.\mathsf{LossyGen}(1^\lambda)$ for $\ell \in [j-1]$.
- Sample $k_\ell \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$ for $\ell \in [j+1, \nu]$.
- Set $\mathsf{CRS} = (k_1, \ldots, k_{j-1}, k, k_{j+1}, \ldots, k_\nu)$.
- Forward $(\mathsf{CRS}, (p^*, a^*))$ to $\mathcal{P}^*$, and obtain τ^* from $\mathcal{P}^*$.
- Compute $b = (\mathbb{E}_i(a^*, p^*, \tau^*))$.
- Forward b to Ch.

Figure 1: Reduction $\mathcal{B}$ that interacts with challenger Ch and breaks (lossy) KI of jth $\mathcal{H}$

Next, note that $\mathcal{B}$ runs in time $\max(\mathsf{poly}(\lambda), \widetilde{T})$ where $\widetilde{T}$ denotes the time required to check event $\mathbb{E}_i(a^*, p^*, \tau^*)$, which is dominated by the time required to compute the polynomials $q_i^*(\cdot) = \Sigma_{x_{i+1},\ldots x_\nu} p^*(r_1^*, \ldots, r_i^*, x_{i+1}, \ldots x_\nu)$ and $q_{i-1}^*(\cdot) = \Sigma_{x_i,\ldots x_\nu} p^*(r_1^*, \ldots, r_{i-1}^*, x_i, \ldots x_\nu)$, which is at most $\mathsf{poly}(|B|^\nu, d^\nu)$, which (by assumption in the theorem) is at most $\mathsf{poly}(T')$. $\mathcal{B}$ will succeed in distinguishing $k \leftarrow \mathsf{LossyGen}(1^\lambda)$ from $k \leftarrow \mathsf{Gen}(1^\lambda)$ with the same advantage as $\mathcal{P}^*$. Formally,

$$\begin{aligned} &\left| \Pr_{k\leftarrow\mathsf{Gen}(1^\lambda)} [\mathcal{B}(k) = 1] - \Pr_{k\leftarrow\mathsf{LossyGen}(1^\lambda)} [\mathcal{B}(k) = 1] \right| \\ &\geq \left| \Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_{0,j-1} \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*))}} [\mathbb{E}_i(a^*,p^*,\tau^*)] - \Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_{0,j} \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*))}} [\mathbb{E}_i(a^*,p^*,\tau^*)] \right| \geq \frac{1}{w'(\lambda)}. \end{aligned}$$

We have reached a contradiction, as desired. □

Claim 4.2. *There exists a polynomial $w(\cdot)$ such that for large enough $\lambda \in \mathbb{N}$,*

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_1 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [(\mathbb{E}_i(a^*,p^*,\tau^*)) \wedge \ (\forall j \in [i-1], r_j^* = r'_j)] \geq \frac{1}{2^{\nu\lambda^\epsilon}} \cdot \frac{1}{w(\lambda)},$$

where we parse τ^* as $\tau^* = (p_1^*, r_1^*, \ldots, p_\nu^*, r_\nu^*)$, and $\mathbb{S}_j := \{\mathcal{H}.\mathsf{Hash}(k_j, x)\}_{x\in\{0,1\}^{n(\lambda)}}$ for $j \in [i-1]$.

Proof. This claim will follow from the previous claim, and the lossiness of $\mathcal{H}$. Since the guesses r'_j for $j \in [i-1]$ do not interact with the prover $\mathcal{P}^*$ or $\mathcal{V}'$,

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_1 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [(\mathbb{E}_i(a^*,p^*,\tau^*)) \wedge \ \ (\forall j \in [i-1], r^*_j = r'_j)]$$
$$= \Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_1 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*))}} [(\mathbb{E}_i(a^*,p^*,\tau^*))] \times \Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_1 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [(\forall j \in [i-1], r^*_j = r'_j)].$$

By the previous claim, there exists a polynomial $w(\cdot)$ such that for large enough $\lambda \in \mathbb{N}$,

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_1 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*))}} [(\mathbb{E}_i(a^*,p^*,\tau^*))] \geq \frac{1}{w(\lambda)}$$

Since $\{r'_j\}_{j\in[i-1]}$ are sampled uniformly at random over the space of lossy outcomes of $\mathcal{H}$, and since this space has size $|\mathbb{S}_j| \leq 2^{\lambda^\epsilon}$, we have

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_1 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [(\forall j \in [i-1], r^*_j = r'_j)] \geq \left(\frac{1}{|\mathbb{S}_j|}\right)^{(i-1)} \geq \frac{1}{2^{(i-1)\lambda^\epsilon}} \geq \frac{1}{2^{\nu\lambda^\epsilon}}.$$

which completes the proof of the claim. □

Claim 4.3. *For* Hybrid_2 *defined with respect to* f_λ *in Figure 2, there exists a polynomial* $w(\cdot)$ *such that for large enough* $\lambda \in \mathbb{N}$*,*

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_2 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [(\mathbb{E}_i(a^*,p^*,\tau^*)) \wedge \ \ (\forall j \in [i-1], r^*_j = r'_j)] \geq \frac{1}{2^{\nu\lambda^\epsilon}} \cdot \frac{1}{w(\lambda)},$$

where we parse τ^* *as* $\tau^* = (p^*_1, r^*_1, \ldots, p^*_\nu, r^*_\nu)$ *and* $\mathbb{S}_j := \{\mathcal{H}.\mathsf{Hash}(k_j, x)\}_{x\in\{0,1\}^{n(\lambda)}}$ *for* $j \in [i-1]$*.*

Proof. This claim follows from the $T(\lambda)$ key indistinguishability of $\mathcal{H}$ and the previous claim. Assume that the statement of the claim is not true, then there exists polynomial $w'(\cdot)$ such that

$$\Bigg| \Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_1 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [(\mathbb{E}_i(a^*,p^*,\tau^*)) \wedge \ \ (\forall j \in [i-1], r^*_j = r'_j)]$$
$$- \Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_2 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [(\mathbb{E}_i(a^*,p^*,\tau^*)) \wedge \ \ (\forall j \in [i-1], r^*_j = r'_j)] \Bigg|$$
$$\geq \frac{1}{2^{\nu\lambda^\epsilon}} \cdot \frac{1}{w(\lambda)} \geq \frac{1}{w'(T(\lambda))}$$

The function f_λ has hardwired:

- $i, p^*, (r'_1, \ldots r'_{i-1})$ such that r'_j is sampled uniformly in $\mathbb{S}_j$ for $j \in [i-1]$, and
- $q'_i(\cdot) = \sum_{x_{i+1},\ldots,x_\nu \in B} p^*(r'_1, \ldots, r'_{i-1}, \cdot, x_{i+1}, \ldots, x_\nu)$.

On input p^*_i, f_λ computes output as follows:

- Sample r'_i uniformly from the d roots of $p^*_i(\cdot) - q'_i(\cdot)$[a].
- Output r'_i.

[a] This can be done with overwhelming probability via the Cantor-Zassenhaus [CZ81] algorithm, which can be used to generate a specific root in any arbitrary ordering of the roots.

Figure 2: Function $f_\lambda \leftarrow \mathcal{F}_\lambda$

$\mathcal{B}$ interacts with a challenger Ch and with oracle access to $\mathcal{P}^*$, does the following:

- Sample $\{r'_j \leftarrow \mathbb{S}_j\}_{j\in[i-1]}$, use them to define f_λ as in Figure 2, and send f_λ to Ch.
- Obtain challenge key k from Ch.
- Samples $k_j \leftarrow \mathcal{H}.\mathsf{LossyGen}(1^\lambda)$ for $j \in [i-1]$.
- Samples $k_j \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$ for $i+1 \le j \le \nu$.
- Sets $\mathsf{CRS} = (k_1, \ldots, k_{i-1}, k, k_{i+1}, \ldots, k_\nu)$.
- Forwards $(\mathsf{CRS}, (a^*, p^*))$ to $\mathcal{P}^*$, and obtains proof τ^* from $\mathcal{P}^*$.
- Compute $b = (\mathbb{E}_i(a^*, p^*, \tau^*)) \wedge (\forall j \in [i-1], r^*_j = r'_j)$.
- Forward b to Ch.

Figure 3: Reduction $\mathcal{B}$ that interacts with challenger Ch and breaks (statgen) KI of ith hash $\mathcal{H}$

where the last inequality follows by our setting of $\mathsf{poly}(T(\lambda)) \ge 2^{\nu\lambda^\epsilon}$. Then we define adversary $\mathcal{B}$ in Figure 3 against the $T(\lambda)$ key indistinguishability of $\mathcal{H}$. As defined $\mathsf{poly}(T)$-sized $\mathcal{B}$ will succeed in distinguishing $k \leftarrow \mathsf{Gen}(1^\lambda)$ from $k \leftarrow \mathsf{StatGen}(1^\lambda, f_\lambda)$ with the same advantage as $\mathcal{P}^*$. Formally,

$$\left| \Pr_{k\leftarrow\mathsf{Gen}(1^\lambda)}[\mathcal{B}(k)=1] - \Pr_{k\leftarrow\mathsf{StatGen}(1^\lambda)}[\mathcal{B}(k)=1] \right|$$

$$= \left| \Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_1 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [(\mathbb{E}_i(a^*, p^*, \tau^*)) \wedge \ \ (\forall j \in [i-1], r^*_j = r'_j)] \right.$$

$$
\left. - \Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_2 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [(\mathbb{E}_i(a^*,p^*,\tau^*)) \wedge \ \ (\forall j\in[i-1], r^*_j = r'_j)] \right|
$$
$$
\geq \frac{1}{w'(T(\lambda))}.
$$

We have reached a contradiction, proving our claim. □

Claim 4.4. *For* Hybrid_2 *defined with respect to* f_λ *in Figure 2, there exists a negligible function* $\mu(\cdot)$ *such that for large enough* $\lambda \in \mathbb{N}$,

$$
\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_2 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [(\mathbb{E}_i(a^*,p^*,\tau^*)) \wedge \ \ (\forall j\in[i-1], r^*_j = r'_j)] \leq d\cdot\mu(T(\lambda)),
$$

where we parse τ^* *as* $\tau^* = (p^*_1, r^*_1, \ldots, p^*_\nu, r^*_\nu)$ *and* $\mathbb{S}_j := \{\mathcal{H}.\mathsf{Hash}(k_j, x)\}_{x\in\{0,1\}^{n(\lambda)}}$ *for* $j \in [i-1]$.

Proof. This claim follows by the statistical correlation intractability of $\mathcal{H}$. For sake of contradiction assume there exists polynomial $w(\cdot)$ such that for large enough $\lambda \in \mathbb{N}$,

$$
\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_2 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [(\mathbb{E}_i(a^*,p^*,\tau^*)) \wedge \ \ (\forall j\in[i-1], r^*_j = r'_j)] \geq \frac{d}{w(T(\lambda))}. \tag{7}
$$

Fix $(\mathsf{CRS}, \tau^*, \{r'_j\}_{j\in[i-1]})$ such that $\mathbb{E}_i(a^*,p^*,\tau^*)$ and $r^*_j = r'_j$ for $j \in [i-1]$ where $\{r'_j \in \mathbb{S}_j\}_{j\in[i-1]}$, $\mathsf{CRS} \in \mathsf{Support}(\mathsf{Hybrid}_2)$, and $\tau^* \in \mathsf{Support}(\mathcal{P}^*(\mathsf{CRS}, (a^*, p^*)))$.

Let us define correctly evaluated polynomials for $j \in [\nu]$ as

$$
q^*_j(\cdot) := \sum_{x_{j+1},\ldots,x_\nu\in B} p^*(r^*_1, \ldots, r^*_{j-1}, \cdot, x_{j+1}, \ldots, x_\nu).
$$

From event $\mathbb{E}_i(a^*,p^*,\tau^*)$ we have that $\mathsf{State}((a^*,p^*),\tau^*_{i-1}) = \mathsf{reject}$ and $\mathsf{State}((a^*,p^*),\tau^*_i) = \mathsf{accept}$. The sumcheck protocol's State function on input $((a^*,p^*),\tau^*_k)$ for any $k \in [\nu]$ runs all of the following checks:

1. In Step 1, State
   - Checks that $p^*_1(\cdot)$ is of degree at most d and that $a = \sum_{x_1\in B} p^*_1(x_1)$ by invoking $\mathcal{V}$.
   - Sets $v^*_2 = p^*_1(r^*_1)$.
2. In Step j for $j \in [k]$, State
   - Checks that $p^*_j(\cdot)$ is of degree at most d and that $v^*_j = \sum_{x_j\in B} p^*_j(x_j)$ by invoking $\mathcal{V}$.
   - Sets $v^*_{j+1} = p^*_j(r^*_j)$.
3. State checks that the last claim is correct, that is
$$
v^*_k = \sum_{x_{k+1},\ldots,x_\nu\in B} p^*(r^*_1, \ldots, r^*_k, x_{k+1}, \ldots, x_\nu).
$$

   If all checks pass, State outputs accept. Otherwise, outputs reject.

Therefore, if $\mathsf{State}((a^*, p^*), \tau^*_{i-1}) = \mathsf{reject}$, and $\mathsf{State}((a^*, p^*), \tau^*_i) = \mathsf{accept}$, this means that $v^*_i = \sum_{x_{i+1},\ldots,x_\nu \in B} p^*(r^*_1, \ldots, r^*_i, x_{i+1}, \ldots, x_\nu)$, but $v^*_{i-1} \neq \sum_{x_i,\ldots,x_\nu \in B} p^*(r^*_1, \ldots, r^*_{i-1}, x_i, \ldots, x_\nu)$. This implies that r^*_i is such that $p^*_i(r^*_i) = q^*_i(r^*_i)$. In other words, r^*_i is a root of the polynomial $p^*_i(\cdot) - q^*_i(\cdot)$. This implies that

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_2 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [(r^*_i \text{ is a root of } p^*_i(\cdot) - q^*_i(\cdot)) \wedge (\forall j \in [i-1], r^*_j = r'_j)] \geq \frac{d}{w(T(\lambda))} \tag{8}$$

We now turn to the function f_λ which, conditioned on $r^*_j = r'_j$ for $j \in [i-1]$, has hardwired

$$q'_i(\cdot) = \sum_{x_{i+1},\ldots,x_\nu \in B} p^*(r'_1, \ldots, r'_{i-1}, \cdot, x_{i+1}, \ldots, x_\nu) = q^*_i(\cdot).$$

This means that $f_\lambda(p^*_i)$, by definition, also outputs a root of the polynomial $p^*_i(\cdot) - q^*_i(\cdot)$. Let us denote this root by s^*. By the statistical CI property of $\mathcal{H}$, we have that

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_2 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [r^*_i = s^* \wedge (\forall j \in [i-1], r^*_j = r'_j)] = \mathsf{negl}(T(\lambda)) \tag{9}$$

By Equations (8) and (9), we have that

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_2 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [(r^*_i \text{ is a root of } p^*_i(\cdot) - q^*_i(\cdot), r^*_i \neq s^*) \wedge (\forall j \in [i-1], r^*_j = r'_j)] \geq \frac{d}{w(T(\lambda))} - \mathsf{negl}(T(\lambda)) \tag{10}$$

Roughly, the previous equation implies that $\mathcal{P}^*$ is "guessing" where the function's chosen root s^* is and avoiding it, with better probability than it should be able to (due to key indistinguishability of $\mathcal{H}$). Therefore, we now build a reduction $\mathcal{B}$ in Figure 4 that contradicts key indistinguishability of $\mathcal{H}$. Then, we have that $\mathcal{B}$ runs in time $|B|^\nu \leq \mathsf{poly}(T)$ and is such that by Equation (10),

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_2 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [\mathcal{B} \text{ outputs } 0 \wedge (\forall j \in [i-1], r^*_j = r'_j) | \mathsf{Ch} \text{ used } s^*_1] \geq \frac{1}{w(T(\lambda))} - \mathsf{negl}(T(\lambda)) \tag{11}$$

and

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Hybrid}_2 \\ \tau^*\leftarrow\mathcal{P}^*(\mathsf{CRS},(a^*,p^*)) \\ \{r'_j\leftarrow\mathbb{S}_j\}_{j\in[i-1]}}} [\mathcal{B} \text{ outputs } 0 \wedge (\forall j \in [i-1], r^*_j = r'_j) | \mathsf{Ch} \text{ used } s^*_0] = \mathsf{negl}(T(\lambda)) \tag{12}$$

which contradicts $T(\lambda)$-key-indistinguishability of $\mathcal{H}$. □

Finally, we note that the last two claims yield a contradiction, therefore Theorem 4.1 must be sound, as desired.

Runtimes. We first calculate the online computation needed to compute a function in $f_\lambda \in \mathcal{F}_\lambda$, defined in Figure 2. In particular we need to calculate the runtime for function f_λ to sample roots

$\mathcal{B}$ interacts with a challenger Ch and with oracle access to $\mathcal{P}^*$, does the following:

- Sample $\{r'_j \leftarrow \mathbb{S}_j\}_{j\in[i-1]}$ and define $f_{0,\lambda}$ with root s_0^* and $f_{1,\lambda}$ with root s_1^* as in Figure 2, where s_0^* and s_1^* are two independent, uniformly sampled roots of $p_i^*(\cdot) - q'_i(\cdot)$.
- Forward $(f_{0,\lambda}, f_{1,\lambda})$ to Ch, and obtain challenge key k from Ch.
- Sample $k_j \leftarrow \mathcal{H}.\mathsf{LossyGen}(1^\lambda)$ for $j \in [i-1]$.
- Sample $k_j \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$ for $i < j \leq \nu$.
- Set $\mathsf{CRS} = (k_1, \ldots, k_{i-1}, k, k_{i+1}, k_\nu)$.
- Forward $(\mathsf{CRS}, (a^*, p^*))$ to $\mathcal{P}^*$, and obtain proof τ^* from $\mathcal{P}^*$.
- Parse τ^* as $\tau^* = (p_1^*, r_1^*, \ldots, p_\nu^*, r_\nu^*)$.
- If $\exists j \in [i-1]$ such that $r_j^* \neq r'_j$, output 0. If $r_i^* \notin \{s_0^*, s_0^*\}$, output 0.
- If $r_i^* = s_0^*$, output 0. Else, output 1.

Figure 4: Reduction $\mathcal{B}$ that interacts with challenger Ch and breaks (statgen) KI of ith hash $\mathcal{H}$

of $p_i^*(\cdot) - q'_i(\cdot)$ for some $i \in [\nu]$. For $\mathbb{F}$ and polynomial $p(\cdot)$ of individual degree d, the root-finding algorithm will take time $\mathsf{poly}(d, \log|\mathbb{F}|)$ [CZ81]. Therefore, the time needed to run the $\mathcal{H}.\mathsf{Hash}$ algorithm will be $\mathsf{poly}(d, \log|\mathbb{F}|)$. The overall verifier runtime equals the time needed to compute $\mathcal{H}.\mathsf{Hash}$, ν times, in addition to the time required to run verification for the underlying sumcheck protocol, which is $\nu \cdot \mathsf{poly}(|B|, d, \log|\mathbb{F}|)$. The prover's runtime is $\mathsf{poly}(|B|^\nu, d^\nu, \log|\mathbb{F}|)$. □

For simplicity, in all our reductions, we assumed that the polynomial (p^*, a^*) was chosen selectively by the prover, and therefore is known to the challenger before generating the CRS. Since we assume at least $T'(\lambda)$ hardness of the underlying primitives, we observe that all the same proofs go through as long for a fully adaptive choice of a^* (since bad challenges do not depend on a^*), and as long as the challenger can *guess* p^* with probability $\frac{1}{w(T'(\lambda))}$ for some polynomial $w(\cdot)$, giving us the following (strengthened) theorem.

Definition 4.2 (U-Somewhat Adaptive Soundness)**.** Let $\Pi = (\mathcal{P}, \mathcal{V})$ denote a non-interactive sumcheck protocol in the CRS model for language $\mathcal{L}_{B,\mathbb{F},\nu,d}$ over field $\mathbb{F}$ with subset B, ν variables and individual degree d. We say that Π is U-somewhat adaptively sound if for every set $\mathbb{U}$ of size at most U, and all $\mathsf{poly}(\lambda)$-sized cheating provers $\mathcal{P}^*$ we have

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Setup}(1^\lambda)\\(\tau^*,p^*,a^*)\leftarrow\mathcal{P}(\mathsf{CRS})}}[\mathcal{V}(\mathsf{CRS}, (a^*, p^*), \tau^*) = 1 | \, p^* \in \mathbb{U}] = \mathsf{negl}(\lambda),$$

where $p^*(\cdot)$ denotes a ν-variate polynomial of individual degree d in each variable.

Theorem 4.3. *Let $B \subset \mathbb{F}$, and $\nu = \nu(\lambda)$. Let $\Pi = (\mathcal{P}, \mathcal{V})$ be the interactive, statistically sound sumcheck protocol for $\mathcal{L}_{B,\mathbb{F},\nu,d}$ from Construction 4.1. There exists a polynomial $\rho(\cdot)$, such that if there exists*

an $(\epsilon, T(\lambda), T'(\lambda))$*-secure lossy SSCI hash function family* $\mathcal{H} : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$ *where* $n(\lambda) = d\log(|\mathbb{F}|)$ *and* $m(\lambda) = \log(|\mathbb{F}|)$ *for functions* $\mathcal{F}$ *computable in time* $\rho(d,\lambda)$ *according to Definition 3.1, then the Fiat-Shamir paradigm will yield a non-interactive, computationally* U*-somewhat adaptively sound sound sumcheck protocol satisfying Definition 4.2, where the verifier runs in time* $\nu \cdot \mathsf{poly}(|B|, d, \log|\mathbb{F}|)$ *and the prover runs in time* $\mathsf{poly}(|B|^\nu, d^\nu, \log|\mathbb{F}|)$*, as long as: there exists a polynomial* $\mathsf{poly}(\cdot)$ *such that* $\mathsf{poly}(T) \geq \max(2^{\nu\lambda^\epsilon}, |B|^\nu, d^\nu, U)$*, and* $\mathsf{poly}(T') \geq (|B|^\nu, d^\nu, U)$*.*

5 #SAT Reduces to EOL under Sub-exponential LWE

In this section, we establish the hardness of finding a Nash equilibrium by proving (in Theorem 5.1) that our non-interactive sumcheck protocol satisfies certain additional (stronger) soundness properties, that we denote by partial adaptive soundness and partial adaptive unambiguity. We then combine this with the work of Choudhuri et. al. [CHK+19a], who show that non-interactive sumcheck with these properties suffices to establish a reduction between instances of #SAT and the PPAD complete problem end-of-metered line (EOL). We note that while the formal theorem in [CHK+19a] required full adaptive soundness of non-interactive sumcheck, the proof of this theorem goes through even when the underlying sumcheck satisfies the weaker soundness properties that we discuss below.

Definition 5.1 (Partial Adaptive Unambiguous Soundness)**.** Let $p : \mathbb{F}^\nu \to \mathbb{F}$ be a ν-variate polynomial of degree at most d in each variable and $\boldsymbol{r} = (r_1, \ldots, r_j) \in \mathbb{F}^j$ be any prefix with $j \leq \nu$. Let $\Pi_{FS} = (\mathcal{P}(\mathsf{CRS}, p), \mathcal{V}(\mathsf{CRS}, a, p, \boldsymbol{r}))$ denote the non-interactive sumcheck protocol with security parameter λ obtained by instantiating Fiat-Shamir for sumcheck according to Construction 4.2, where the verifier on input a value a^*, polynomial p^*, prefix $\boldsymbol{r}^*$ and proof τ^* outputs 1 if all claims *after* the prefix $\boldsymbol{r}^*$ accept. We say that Π_{FS} satisfies partial adaptive unambiguous soundness if in addition to completeness, it satisfies the following properties.

- **Partial Adaptive Soundness.** For every PPT cheating prover $\mathcal{P}^*$, and every $j \in [\nu]$, denoting prefix $r_1^*, \ldots, r_j^*$ for $j \leq \nu$ by $\boldsymbol{r}^*$, we have

$$\Pr\left[\mathcal{V}(\mathsf{CRS}_{[j+1,m]}, (a^*, p^*, \boldsymbol{r}^*), \tau^*) = 1 \,\middle|\, \begin{matrix}((a^*,\boldsymbol{r}^*),\tau^*) \leftarrow \mathcal{P}^*(\mathsf{CRS}, p^*), \\ \forall k \in [j] \;\; r_k^* \in (\mathbb{S}_k \cup [d+1]), \\ \sum_{\mathbf{z} \in B^{n-j}} p^*(\boldsymbol{r}^*, \mathbf{z}) \neq a^*\end{matrix}\right] = \mathsf{negl}(\lambda),$$

  where for $j \in [\nu]$, $\mathbb{S}_j := \{\mathcal{H}.\mathsf{Hash}(k_j, x)\}_{x \in \{0,1\}^*}$ denotes the space of all outcomes of the j^{th} hash function, and $\mathsf{CRS}_{[j+1,m]}$ denotes hash keys $k_{j+1}, \ldots k_m$.

- **Partial Adaptive Unambiguity.** For every PPT cheating prover $\mathcal{P}^*$, and every $j \in [\nu]$, denoting prefix $r_1^*, \ldots, r_j^*$ for $j \leq \nu$ by $\boldsymbol{r}^*$, we have

$$\Pr\left[\begin{matrix}(\mathcal{V}(\mathsf{CRS}_{[j+1,m]}, (a^*, p^*, \boldsymbol{r}^*), \tau^*) = 1) \\ \wedge \\ (\mathcal{V}(\mathsf{CRS}_{[j+1,m]}, (a^*, p^*, \boldsymbol{r}^*), \tau) = 1)\end{matrix} \,\middle|\, \begin{matrix}((a^*,\boldsymbol{r}^*),\tau^*,\tau) \leftarrow \mathcal{P}^*(\mathsf{CRS}, p^*), \\ \forall k \in [j] \;\; r_k^* \in (\mathbb{S}_k \cup [d+1]), \\ \tau^* \neq \tau, \sum_{\mathbf{z} \in B^{n-j}} p^*(\boldsymbol{r}^*, \mathbf{z}) = a^*\end{matrix}\right] = \mathsf{negl}(\lambda)$$

  where for $j \in [\nu]$, $\mathbb{S}_j := \{\mathcal{H}.\mathsf{Hash}(k_j, x)\}_{x \in \{0,1\}^*}$ denotes the space of all outcomes of the j^{th} hash function, and $\mathsf{CRS}_{[j+1,m]}$ denotes hash keys $k_{j+1}, \ldots k_m$.

Intuitively, the partial adaptive soundness requirement states that for *selective choice of* p^*, the prover on input the CRS cannot *adaptively generate* a^*, *prefix* $\boldsymbol{r}^* = r_1^*, \ldots, r_j^*$ *where* $j \in [\nu]$, and steps $(j+1, \ldots, \nu)$ of the transcript of a non-interactive proof with prefix $\boldsymbol{r}^*$ that is accepted by the verifier, unless $\sum_{\{x_i \in B\}_{i \in [j+1,\nu]}} p^*(r_1^*, \ldots, r_j^*, x_{j+1}, \ldots x_\nu) = a^*$. Importantly, we note that the verifier only checks steps $(j+1, \ldots, \nu)$ of the proof.

Meanwhile, the partial adaptive unambiguity requirement considers *selective choice of* p^* and allows the prover to *adaptively generate* a^*, *prefix* $\boldsymbol{r}^* = r_1^*, \ldots, r_j^*$ *where* $j \in [\nu]$, and steps $(j+1, \ldots, \nu)$ of *two different transcripts* of a non-interactive proof with prefix $\boldsymbol{r}^*$. Then, the property requires that a verifier that only checks steps $(j+1, \ldots, \nu)$ rejects at least one of the two transcripts.

Theorem 5.1. *Let* $B \subset \mathbb{F}$, *and* ν. *Let* $\Pi = (\mathcal{P}, \mathcal{V})$ *be the interactive, statistically sound sumcheck protocol for* $\mathcal{L}_{B,\mathbb{F},\nu}$ *from Construction 4.1. If there exists an* $(\epsilon, T(\lambda), T'(\lambda))$*-secure lossy SSCI hash function family* $\mathcal{H} : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$ *where* $n(\lambda) = d \log(|\mathbb{F}|)$ *and* $m(\lambda) = \log(|\mathbb{F}|)$ *for function family* $\mathcal{F}$ *according to Definition 3.1, then the Fiat-Shamir paradigm will yield a non-interactive, partially adaptively unambiguously sound sumcheck protocol according to Definition 5.1, where the verifier runs in time* $\nu \cdot \mathsf{poly}(|B|, d, \log|\mathbb{F}|)$ *and the prover runs in time* $\mathsf{poly}(|B|^\nu, d, \log|\mathbb{F}|)$, *as long as: there exists a polynomial* $\mathsf{poly}(\cdot)$ *such that* $\mathsf{poly}(T) \geq \max(2^{\nu\lambda^\epsilon}, |B|^\nu)$ *and* $\mathsf{poly}(T') \geq |B|^\nu$.

Proof. (Sketch). We now sketch the proofs of partial adaptive soundness and partial adaptive unambiguity of our non-interactive sumcheck protocol. These follow along the lines of our main proof of (selective) soundness (Theorem 4.2), except by noting that when hash keys $\{k_j\}_{j \in [i]}$ are generated in lossy mode, the reduction can guess them with probability $2^{-\nu\lambda^\epsilon}$, which suffices to obtain a contradiction to correlation intractability. We outline the proofs in more detail below.

Partial Adaptive Soundness. This property follows from the round-by-round soundness of the underlying interactive sumcheck protocol and the lossy SSCI of $\mathcal{H}$, the proof of which follows the proof of soundness for Π. Consider a malicious prover $\mathcal{P}^*$ that on input (CRS, p^*), where $\mathsf{CRS} = (k_1, \ldots, k_\nu)$ and $p^*(\cdot)$ is a ν-variate polynomial with individual degree d, outputs $((a^*, \mathbf{r}^*), \tau^*)$, where prefix $\mathbf{r}^* = (r_1^*, \ldots, r_j^*)$ for $j \in [\nu]$ and proof $\tau^* = (\mathbf{r}^*, p_{j+1}^*, r_{j+1}^*, \ldots, p_\nu^*, r_\nu^*)$. Towards a contradiction, suppose these are such that

$$\sum_{x_{j+1}, \ldots, x_\nu \in B} p^*(\mathbf{r}^*, x_{j+1}, \ldots, x_\nu) \neq a^*.$$

We can think of $\mathcal{P}^*$ as providing a non-interactive sumcheck proof for false instance (a^*, g^*) where the polynomial $g^*(\cdot)$ is a $(\nu - j)$-variate polynomial with individual degree d defined as follows

$$g^*(x_{j+1}, \ldots, x_\nu) := p^*(r_1^*, \ldots, r_j^*, x_{j+1}, \ldots, x_\nu).$$

We can use the same hybrids in our previous proof to switch into lossy mode with a slightly modified CI function description. In this setting, the CI functions f_λ from Figure 2 will be hardwired with p^* and be additionally required to guess the prefix $\mathbf{r}^*$, which can be thought of as guessing the polynomial g^* to be proven. This prefix has the property that $r_i^* \in \mathbb{S}_i \cup [d+1]$ where $\mathbb{S}_i = \{\mathcal{H}.\mathsf{Hash}(k_j, x)\}_{x \in \{0,1\}^{n(\lambda)}}$ for all $i \in [j]$. As such, our CI function f_λ will be altered to sample the challenges within the prefix uniformly at random from this space. Since $d \ll 2^{\lambda^\epsilon}$, our guessing probability will still bounded by $2^{-\nu\lambda^\epsilon}$ for some $0 < \epsilon < 1$. With these modifications, partial

adaptive soundness follows from our previous soundness proof.

Partial Adaptive Unambiguity. This property follows from the soundness and unambiguity of the underlying sumcheck protocol. Consider a malicious prover $\mathcal{P}^*$ that on input (CRS, p^*), where CRS is the common reference string and $p^*(\cdot)$ is a ν-variate polynomial with individual degree d, outputs $((a^*, \mathbf{r}^*), \tau^*, \tau)$, where prefix $\mathbf{r}^* = (r_1^*, \ldots, r_j^*)$ for $j \in [\nu]$ and proofs $\tau^* \neq \tau$. In this case, we know that

$$\sum_{x_{j+1},\ldots,x_\nu \in B} p^*(r^*, x_{j+1}, \ldots, x_\nu) = a^*.$$

We first note that the protocol, for any step i where $i \in [\nu]$, specifies deterministic calculations to compute the next univariate polynomial p_i and the challenge r_i. This necessarily means that there is one, unique, way to take a true claim in the $i-1^{th}$ step to another true claim in the i^{th} step. Thus, the only way for $\mathcal{P}^*$ to produce two proofs $\tau^* \neq \tau$, is for $\mathcal{P}^*$ to deviate from the protocol and produce a false claim at some step $j < k \leq \nu$ in at least one of the proofs. Without loss of generality, assume that τ^* is the proof with the false claim at the kth step. Given that $\tau^* = (\mathbf{r}^*, p_{j+1}^*, r_{j+1}^*, \ldots, p_k^*, r_k^*, \ldots, p_\nu^*, r_\nu^*)$, this reduces to the case where

$$\sum_{x_{j+1},\ldots,x_\nu \in B} p^*(\mathbf{r}_k^*, x_{j+1}, \ldots, x_\nu) \neq a^*$$

with new prefix $\mathbf{r}_k^* = (\mathbf{r}^*, r_{j+1}^*, \ldots, r_k^*)$ and new proof $\tau_k^* = (\mathbf{r}_k^*, p_k^*, r_k^*, \ldots, p_\nu^*, r_\nu^*)$. This case is exactly the premise of the partial adaptive soundness property with prefix length k. As such, partial adaptive unambiguity also follows. □

We have the following theorem from [CHK+19a], modified to use our non-interactive sumcheck protocol.

Theorem 5.2. *For a parameter ν, fix a finite field $\mathbb{F}$ of sufficiently large size p (say $O(2^{(\nu^{1/\epsilon^2})})$, where ϵ comes from the assumed sub-exponential security of LWE). Let f be a ν-variate polynomial over $\mathbb{F}$ of individual degree at most d. Obtain instances of the rSVL family in [CHK+19a] using the non-interactive sumcheck protocol in Construction 4.2. Then given an adversary $\mathcal{A}$ that solves these instances in polynomial time and with non-negligible probability $\epsilon = \epsilon(\nu)$, it is possible to either*

- *count the number of satisfying assignments to f in polynomial time with probability ϵ, or*
- *break partial adaptive unambiguous soundness of the non-interactive sumcheck protocol as described in Definition 5.1 with probability at least $\epsilon/(d+1) \cdot \nu$.*

Proof. (Sketch). This proof follows along the lines of Theorem 21 in [CHK+19a]. The only difference is that [CHK+19a] build a reduction that breaks *adaptive unambiguous soundness of the non-interactive sumcheck protocol*, whereas in our case, we need to work with the weaker notion of *partial* adaptive unambiguous soundness, since our non-interactive sumcheck may not be fully adaptively sound. In more detail, given an adversary that solves an rSVL instance corresponding to

$$\sum_{x_1,\ldots x_\nu} p(x_1, \ldots, x_\nu)$$

[CHK+19a] show that one can either solve #SAT, or break *adaptive unambiguous soundness of the underlying non-interactive sumcheck protocol*. The rSVL instance itself, roughly, consists of a pair (v, i) which is of the form $(t, \{(\widetilde{y}_1, \widetilde{\pi_1}), \ldots (\widetilde{y}_\ell, \widetilde{\pi}_\ell)\})$. Here $t \in [d+1]^{\leq \nu}$ is an index, and $(\widetilde{y}_1, \widetilde{\pi_1}), \ldots (\widetilde{y}_\ell, \widetilde{\pi}_\ell)$ are (separate) partial sums and proofs that will each be verified with respect to certain specific prefixes. These prefixes can be represented as $\boldsymbol{r}^1, \ldots \boldsymbol{r}^\ell$, where for $k \in [\ell]$ each $\boldsymbol{r}^k$, by design in [CHK+19a], will be of the form $r_1^k, \ldots r_j^k$. Moreover, by design, every $k \in [\ell], \iota \in [j]$, r_j^k will either be chosen in $[d+1]$ or computed as the output of the hash function on some transcript. In addition, the polynomial $p(\cdot)$ will be chosen selectively. As a result, it is sufficient to rely on partial adaptive unambiguous soundness based on selective choice of $p(\cdot)$ and partial adaptive choice of each r_k^j as either in $\mathbb{S}_j$, or chosen from a relatively small space ($[d+1]$). Specifically, we observe that the reduction in the proof of Theorem 21 [CHK+19a] as is, contradicts partial adaptive unambiguous soundness of the underlying non-interactive sumcheck protocol. □

Combining the two theorems, setting $B = O(1), d = O(1)$, $|\mathbb{F}| = 2^\lambda$, $T = 2^{\lambda^\epsilon}$, $T' = 2^{\lambda^{\epsilon^2}}$, and $\nu = \lambda^{\epsilon^2}$, we see that for any $\nu = \kappa$ number of variables, there exists an efficient instantiation of the hash function family with $\log |\mathbb{F}| = \mathsf{poly}(\kappa)$, under sub-exponential LWE. As a result of this and the work of [CHK+19a], we have the following corollary.

Corollary 5.1. *Assuming sub-exponential LWE,* #SAT *reduces to* EOL.

References

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

[AKV04] Tim Abbot, Daniel Kane, and Paul Valiant. On algorithms for nash equilibria. *Unpublished manuscript*, 2004. https://web.mit.edu/tabbott/Public/final.pdf.

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, pages 407–437, 2019.

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

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1480–1498. IEEE Computer Society, 2015.

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptol. ePrint Arch.*, 2018:46, 2018.

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

[CCH+18] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, and Ron D. Rothblum. Fiat-shamir from simpler assumptions. *IACR Cryptol. ePrint Arch.*, 2018:1004, 2018.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1082–1090. ACM, 2019.

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

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-shamir and correlation intractability from strong kdm-secure encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018*, volume 10820 of *Lecture Notes in Computer Science*, pages 91–122. Springer, 2018.

[CDT09] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player nash equilibria. *J. ACM*, 56(3):14:1–14:57, 2009.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

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

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

[CHK+19a] Arka Rai Choudhuri, Pavel Hubácek, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. Finding a nash equilibrium is no easier than breaking fiat-shamir. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1103–1114. ACM, 2019.

[CHK+19b] Arka Rai Choudhuri, Pavel Hubácek, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. Ppad-hardness via iterated squaring modulo a composite. *IACR Cryptol. ePrint Arch.*, 2019:667, 2019.

[CLW18] Ran Canetti, Alex Lombardi, and Daniel Wichs. Non-interactive zero knowledge and correlation intractability from circular-secure FHE. *IACR Cryptol. ePrint Arch.*, 2018:1248, 2018.

[CZ81] David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, pages 587–592, 1981.

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, pages 3–32, 2019.

[DGP09] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. *Commun. ACM*, 52(2):89–97, 2009.

[EFKP20] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Continuous verifiable delay functions. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, pages 125–154, 2020.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4):27:1–27:64, 2015.

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 579–604. Springer, 2016.

[HL18] Justin Holmgren and Alex Lombardi. Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 850–858. IEEE Computer Society, 2018.

[HY17] Pavel Hubácek and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1352–1371. SIAM, 2017.

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

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

[KPY19] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1115–1124, 2019.

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of fiat-shamir for proofs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 224–251. Springer, 2017.

[KS17] Ilan Komargodski and Gil Segev. From minicrypt to obfustopia via private-key functional encryption. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 122–151, 2017.

[KZ20] Yael Tauman Kalai and Rachel Zhang. Snargs for bounded depth computations from sub-exponential lwe. Cryptology ePrint Archive, Report 2020/860, 2020. https://eprint.iacr.org/2020/860.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[LV20] Alex Lombardi and Vinod Vaikuntanathan. Fiat-shamir for repeated squaring with applications to ppad-hardness and vdfs. Cryptology ePrint Archive, Report 2020/772, 2020. https://eprint.iacr.org/2020/772.

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

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

[Pap94] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994.

[Pie19] Krzysztof Pietrzak. Simple verifiable delay functions. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 60:1–60:15, 2019.

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

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 89–114. Springer, 2019.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of*

Computing, Victoria, British Columbia, Canada, May 17-20, 2008, pages 187–196. ACM, 2008.

[PW10] Rafael Pass and Hoeteck Wee. Constant-round non-malleable commitments from sub-exponential one-way functions. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, pages 638–655, 2010.

[SCG+14] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, 2014.

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.