SNARGs for Bounded Depth Computations and PPAD Hardness from Sub-Exponential LWE*

Ruta Jawale† Yael Tauman Kalai‡ Dakshita Khurana§ Rachel Zhang¶

Abstract

We construct a succinct non-interactive publicly-verifiable delegation scheme for any log-space uniform circuit under the sub-exponential Learning With Errors (LWE) assumption. For a circuit $C : \{0,1\}^N \to \{0,1\}$ of size S and depth D, the prover runs in time $\mathsf{poly}(S)$, the communication complexity is $D \cdot \mathsf{polylog}(S)$, and the verifier runs in time $(D+N) \cdot \mathsf{polylog}(S)$.

To obtain this result, we introduce a new cryptographic primitive: *lossy correlation-intractable hash functions*. We use this primitive to soundly instantiate the Fiat-Shamir transform for a large class of interactive proofs, including the interactive sum-check protocol and the GKR protocol, assuming the sub-exponential hardness of LWE.

By relying on the result of Choudhuri et al. (STOC 2019), we also establish the sub-exponential average-case hardness of PPAD, assuming the sub-exponential hardness of LWE.

*This manuscript is a merge of the two independent and concurrent works [KZ20] and [JK20].

†UIUC. Email: jawale2@illinois.edu.

‡MSR and MIT. Email: yael@microsoft.com.

§UIUC. Email: dakshita@illinois.edu.

¶MIT. Email: rachelyz44@gmail.com.

1 Introduction

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

This task of constructing succinct proofs for long computations is believed to be impossible information theoretically. Indeed, all works on succinct proofs rely on some computational assumption and argue soundness of the scheme only against cheating provers who cannot break the underlying assumption. Such computationally sound proofs are referred to as *arguments* [BCC88].

In this work we focus on constructing (publicly verifiable) *succinct non-interactive arguments* (SNARGs).[1] As in almost all prior work, we consider the CRS model, which assumes the existence of a common reference string (CRS) known to all parties.[2] Indeed, without a CRS it is impossible to guarantee soundness against non-uniform cheating provers for schemes that are only computationally sound. This is the case since (information theoretically) there exists a proof for an incorrect statement, and thus without a CRS, a non-uniform adversary can simply hardwire such a cheating proof. Our goal is to construct SNARGs under a standard cryptographic assumption. Despite extensive work in this area (which we elaborate on in Section 1.2), there were previously no known constructions under standard well-studied assumptions.

We construct a SNARG for bounded depth deterministic computations by provably instantiating the Fiat-Shamir paradigm [FS86] applied to an interactive delegation scheme, in particular, to the GKR protocol [GKR15] for delegating bounded depth computations. The Fiat-Shamir paradigm is a general transformation for converting any public-coin interactive proof (or argument) to a non-interactive argument in the CRS model. It is used extensively in practice for constructing signature schemes [FS86], SNARGs [Mic94, BCS16], and non-interactive zero knowledge (NIZK) proofs [WTS+18].

Loosely speaking, the Fiat-Shamir transform converts an interactive proof (P, V) for a language L to a non-interactive argument (P', V') for L in the CRS model. The CRS consists of randomly chosen hash functions $h_1, \ldots, h_\ell$ from a hash family $\mathcal{H}$, where ℓ is the number of rounds in (P, V). To compute a non-interactive proof for $x \in L$, the prover $P'(x)$ generates a transcript corresponding to $(P, V)(x)$, denoted by $(\alpha_1, \beta_1, \ldots, \alpha_\ell, \beta_\ell)$, by emulating $P(x)$ and replacing each random verifier message β_i by $\beta_i = h_i(\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1}, \alpha_i)$.[3] The verifier $V'(x)$ accepts if and only if $V(x)$ accepts this transcript and $\beta_i = h_i(\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1}, \alpha_i)$ for every $i \in [\ell]$.

This paradigm is extremely elegant and simple, and despite its abundant use in practice, its soundness is poorly understood. The following is known for constant round public-coin protocols: The Fiat-Shamir paradigm is sound in the Random Oracle Model (ROM) [BR93, PS96], yet at the same time there are counterexamples that demonstrate its insecurity when applied to interactive arguments [Bar01, GK03, BBH+19]. There have been several works that prove its soundness when applied to interactive proofs [KRR17, CCRR18, HL18, CCH+19], albeit under extremely strong

[1] Typically in the literature, the term SNARG refers to NP computations, though the focus of this work is on deterministic computations.

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

[3] Throughout this work, we assume w.l.o.g. that the first message is always sent by the prover, and the last one is sent by the verifier (though this last verifier message is moot).

assumptions. Recently, the work of Canetti *et al.* [CCH+19] and the followup work of Peikert and Shiehian [PS19] prove the soundness of the Fiat-Shamir paradigm, assuming standard hardness of the Learning With Errors (LWE) problem, when applied to a *specific* protocol: namely, (a variant of) the three round zero-knowledge proof of graph Hamiltonicity. This result gave the first NIZK argument from LWE.

This Work. The key focus of our work is on constructing SNARGs. In particular, our goal is to prove soundness (under a standard cryptographic assumption) of the Fiat-Shamir paradigm when applied to *succinct* interactive proofs. The work of Canetti *et al.* [CCH+19] proves the soundness of the Fiat-Shamir paradigm when applied to the (succinct) GKR protocol, albeit under a very strong assumption: the existence of a fully homomorphic encryption (FHE) scheme that has *perfect circular security*, that is, every poly-size adversary, given $\mathsf{Enc}(\mathsf{sk})$, outputs sk with probability at most the probability of guessing (i.e., probability at most $\mathsf{poly}(\kappa) \cdot 2^{-\kappa}$, where κ is the security parameter). By contrast, in our work we prove the soundness of the Fiat-Shamir paradigm when applied to the GKR protocol assuming the sub-exponential hardness of LWE.

More generally, we prove that the Fiat-Shamir heuristic is sound when applied to a large class of interactive proofs that we call FS-*compatible*. We elaborate on this class below, but mention that it includes the (succinct) GKR protocol and the celebrated sum-check protocol [LFKN92, Sha92]. Specifically, we define the notion of a *lossy correlation intractable hash family* (which we elaborate on below), and construct it based on the hardness of the LWE assumption. We then show that the Fiat-Shamir transform, applied to any FS-compatible interactive proof, is sound when using any sub-exponentially secure lossy correlation intractable hash family. In particular, this establishes the soundness of the non-interactive GKR and sum-check protocols (obtained by applying the Fiat-Shamir transform), assuming the sub-exponential hardness of LWE.

PPAD Hardness. Our results mentioned above have an important implication to the hardness of the complexity class PPAD, as defined by Papadimitriou [Pap94]. The importance of this class, as well as the motivation for studying its hardness, stems from the fact that the problem of finding a Nash equilibrium is known to be PPAD-complete [DGP09, CDT09].

There has been significant interest in reducing the hardness of PPAD to that of various cryptographic assumptions [AKV04, BPR15, GPS16, HY17, CHK+19a, CHK+19b, EFKP20, KPY20, LV20], which we discuss in Section 1.2.2. The most relevant to us is the recent work of Choudhuri *et al.* [CHK+19a], which shows average-case PPAD hardness assuming average-case #SAT hardness and assuming adaptive soundness of the Fiat-Shamir transform applied to the sum-check protocol. Combining our result on the soundness of the Fiat-Shamir transformation applied to the sum-check protocol, with the result of Choudhuri *et al.* [CHK+19a], we conclude that the sub-exponential hardness of LWE implies the sub-exponential average-case hardness of PPAD.[4] Moreover, since LWE is believed to be sub-exponentially hard even for quantum computers, and our proof (i.e., our reduction) only interacts with the adversary via black-box, straight-line access, we obtain sub-exponential average-case *quantum-hardness* of the complexity class PPAD, assuming sub-exponential quantum hardness of LWE.

[4] To obtain this conclusion we rely on the fact that sub-exponential hardness of LWE implies sub-exponential average-case hardness of #SAT.

1.1 Our Results

We next describe our results in more detail. Our first contribution is defining the notion of a lossy correlation intractable hash family, and constructing it based on the LWE assumption. A lossy correlation intractable hash family is a combination of a correlation intractable hash family and a lossy trapdoor function family, introduced by the influential works of Canetti, Goldreich and Halevi [CGH04], and Peikert and Waters [PW08], respectively.

At a high level, a hash family H is *correlation intractable* (CI) for a function family F, if for every $f \in F$ it is computationally hard, given a random hash key k, to find any input x such that $H(k, x) = f(x)$. Recently, the work of Canetti *et. al* [CCH+19], and the followup work of Peikert and Shiehian [PS19], construct a CI hash family for the family of all functions computable by circuits of an a priori fixed polynomial size (and the run-time of their CI hash functions grow with this bound). As mentioned above, these works led to the first construction of NIZK proofs for NP from LWE.

Lossy Correlation Intractable Hash Functions. We define and construct a *lossy* CI hash family H that, in addition to being a CI hash family, has the following property: The hash keys can be generated using an alternative mode, called the *lossy mode*, such that keys generated in the lossy mode are indistinguishable from random hash keys.[5] In addition, in the lossy mode, the output of a CI hash function lies within a (sub-exponentially) small space of outcomes. We construct a lossy CI hash family by combining a CI hash family with a lossy trapdoor function family [PW08], both of which can be constructed from the LWE assumption.

Theorem 1.1 (Informal)**.** *Under the* LWE *assumption, there exists a lossy correlation intractable hash family.*

The formal definition and construction of a lossy CI hash family (from a CI hash family and a lossy trapdoor function family) can be found in Section 3, and the high level overview can be found in Section 2. We note that in the formal definition, a lossy CI hash family is associated with parameters (T, T', ω), where T relates to the CI-security,[6], T' relates to the lossy security,[7] and ω relates to the amount of information lost in the lossy mode.

We use a lossy CI hash family (with appropriate parameters) to soundly instantiate the Fiat-Shamir heuristic for a class of interactive proofs, which we call FS-compatible, which as mentioned above includes the sum-check protocol and the GKR protocol.

FS-Compatible Proofs. Loosely speaking, a public-coin interactive proof (for a language L) is said to be FS-compatible if it satisfies the following two properties: (1) It is *round-by-round sound* as defined by [CCH+19]. Loosely speaking, this means that every possible transcript prefix defines a statement which is either true or false. If the statement is false, then a cheating prover cannot expand it to an accepting transcript (except with negligible probability). If it is true, then it can be expanded to an accepting transcript (with probability 1). This is formalized by requiring the existence of a (not necessarily efficient) function State that takes as input an instance x and a

[5] In general, hash keys may not be random, rather they may be generated according to some key generation algorithm. We neglect this in this high-level overview, but do consider this general case in the technical sections.

[6] T-CI security means that no $\mathsf{poly}(T)$ adversary can break the CI requirement.

[7] T'-lossy security means that no $\mathsf{poly}(T')$ adversary can distinguish a key generated by $\mathcal{H}.\mathsf{Gen}$ from a key generated by $\mathcal{H}.\mathsf{LossyGen}$.

transcript prefix $\tau = (\alpha_1, \beta_1, \ldots, \alpha_i, \beta_i)$ (for any round i), where $\alpha_1, \ldots, \alpha_i$ are the prover messages and $\beta_1, \ldots, \beta_i$ are the verifier messages, and labels it as being accepting or rejecting, such that if $\mathsf{State}(x, \tau) = \mathsf{reject}$, then for every α, with overwhelming probability over the next verifier message β, $\mathsf{State}(x, \tau|\alpha|\beta) = \mathsf{reject}$. In other words, "bad" verifier messages that go from a rejecting transcript prefix to an accepting one, are extremely sparse. (2) The other property is that for any instance x and a transcript prefix τ such that $\mathsf{State}(x, \tau) = \mathsf{reject}$, there is a non-uniform advice (which depends only on the verifier's messages in τ) such that there is an efficient (non-uniform) function BAD that is given this advice and, on input a prover's next message α, efficiently computes a verifier's next message β such that $\mathsf{State}(x, \tau|\alpha|\beta) = \mathsf{reject}$.

The formal definition of FS-comaptible proofs can be found in Section 4.1. We mention that the formal definition is associated with parameters (T', d, ρ), where T' is the time it takes to compute the function State, d is related to the probability of a random verifier message converting a transcript prefix from being rejecting to being accepting (specifically, this probability is required to be at most $d/2^\lambda$, assuming each verifier message is in $\{0,1\}^\lambda$), and ρ is the (polynomial) time it takes to compute the function BAD (given the non-uniform advice).

We prove that when we apply the Fiat-Shamir transform to any FS-compatible interactive proof Π, w.r.t. a lossy CI hash family, we obtain a sound non-interactive argument, assuming the parameters (T', d, ρ) of the FS-compatible proof Π and the parameters (T, T', ω) of the lossy CI hash family satisfy a specific relationship.[8] We refer the reader to Section 2 for a more detailed overview, and to Section 4 for the formal treatment.

Theorem 1.2 (Informal)**.** *Applying the Fiat-Shamir transform to any* FS*-compatible interactive proof with (arbitrary) parameters (T', d, ρ), w.r.t. a lossy* CI *hash family with parameters (T, T', ω), results in a sound non-interactive argument, as long as T is large enough (depending on T', d, ω and on the protocol Π).*

We also prove that the sum-check protocol and the GKR protocol are both FS-compatible with appropriate parameters. This is done in Sections 5 and 6, respectively.

As a result we obtain the following two corollaries:

Corollary 1.3 (Informal)**.** *Under the sub-exponential hardness of* LWE[9]*, there exists a hash family H and a polynomial p such that the Fiat-Shamir transform w.r.t. H is sound when applied to the sum-check protocol, assuming the sum-check instance*

$$\sum_{b_1,\ldots,b_\ell \in B} g(b_1, \ldots, b_\ell) = v$$

is over a field $\mathbb{F}$ such that $\log|\mathbb{F}| \geq p(d, \ell, \log|B|)$, where ℓ is the number of variables, d is the univariate degree of g, and where B^ℓ is the set we are summing over.

Corollary 1.4 (Informal)**.** *Under the sub-exponential hardness of* LWE*, there exists a hash family H such that the Fiat-Shamir transform w.r.t. H is sound when applied to the* GKR *protocol. For a log-space uniform circuit $C : \{0,1\}^N \to \{0,1\}$ of size S and depth D, the resulting non-interactive argument has the following efficiency guarantees: the prover runs in time $\mathsf{poly}(S)$, the verifier runs in time $(D + N) \cdot \mathsf{polylog}(S)$, and the communication complexity is $D \cdot \mathsf{polylog}(S)$.*

[8] In particular, the parameter T' of the FS-compatible proof is equal to the parameter T' of the lossy CI hash family, and therefore the same notation.

[9] By sub-exponential hardness of LWE, we mean that there exists a constant $\epsilon > 0$ such that for $T(\lambda) = 2^{\lambda^\epsilon}$, no $\mathsf{poly}(T(\lambda))$-size adversary can distinguish LWE samples of size λ from uniform, with advantage better than $\mathsf{negl}(T(\lambda))$.

We also show how to use Theorem 1.3 to obtain PPAD hardness (and more specifically CLS hardness, for the class CLS $\subseteq$ PPAD), following the blueprint of Choudhuri *et al.* [CHK+19a].

Corollary 1.5 (Informal)**.** CLS *is sub-exponentially hard on average,*[10] *assuming the sub-exponential hardness of* LWE*.*

The formal theorem and proof can be found in Section 7, and a high-level overview can be found in Section 2.

1.2 Related Work

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

1.2.1 Related Work on Delegating Computation

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

SNARGs. Extensive work, starting from the seminal work of Micali [Mic94] and continuing with [Gro10, Lip12, DFH12, GGPR13, BCI+13, BCCT13, BCC+14], construct SNARGs for non-deterministic computations. However, the soundness of these schemes is proved either in the Random Oracle Model [BR93] or based on non-standard hardness assumptions known as "knowledge assumptions."[11] Such assumptions have been criticized for being non-falsifiable (as in [Nao03]) and for yielding non-explicit security reductions. We mention that some of these works form the basis of several efficient implementations which are used in practice. Other schemes (for deterministic computations) are known based on non-standard assumptions related to obfuscation [CHJV15, KLW15, BGL+15, CH16, ACC+16, CCC+16] or multilinear maps [PR17].

Very recently, [KPY19] constructed a SNARG (for deterministic computations) based on an efficiently falsifiable decisional assumption on groups with bilinear maps. While this assumption seems reasonable, it is new to their work and has not been studied before. Moreover, it is known to be broken with quantum attacks. Independently, [CCH+19] constructed a SNARG for all bounded depth computations, assuming the existence of an FHE scheme with optimal circular security – which appears to be an extremely strong assumption.

Designated verifier schemes. A line of works starting from [KRR13, KRR14] and continuing with [KP16, BHK17, BKK+18, BK20] designed delegation schemes for deterministic computations and a sub-class of non-deterministic computations, based on standard assumptions (such as the hardness of LWE). These schemes, however, are not publicly verifiable. Rather, the CRS is generated together with a secret key, which is needed in order to verify the proofs.

Interactive schemes. In the interactive setting, we can achieve publicly verifiable schemes, even for non-deterministic computations, under standard assumptions. Kilian [Kil92] constructed

[10] By sub-exponential average-case hardness of CLS we mean that there is a problem in CLS and an efficiently sampleable distribution over instances of this problem that is sub-exponentially hard on average.

[11] For example, the Knowledge-of-Exponent assumption [Dam92] asserts that any efficient adversary that is given two random generators (g, h) and outputs (g^z, h^z) must also "know" the exponent z.

a four message protocol for any NP language with polylog communication, assuming the existence of a hash family that is collision-resistant w.r.t. sub-exponential time adversaries. It has been shown that this scheme can be converted into a three message protocol assuming a multi-collision resistant hash function [BKP18]. The work of [PRV12] constructs a two-message delegation scheme in addition to a (hard to compute) CRS for low-depth circuits, assuming an attribute-based encryption scheme.

Finally, we mention that in the interactive setting, we can achieve publicly verifiable schemes even unconditionally. For example, [GKR15] and [RRR16] give interactive delegation schemes for bounded depth and bounded space computations with unconditional soundness. As mentioned above, in this work we build on the GKR protocol from [GKR15] to obtain our SNARG.

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

1.2.2 Related Work on PPAD Hardness

Recently, there have been a proliferation of results proving the hardness of the class PPAD, which was defined by Papadimitriou [Pap94]. In his original paper, Papadimitriou suggested proving the hardness of PPAD under cryptographic assumptions. After two decades of little progress on the question, a recent sequence of works [BPR15, GPS16, HY17, CHK+19a, Pie19, CHK+19b, EFKP20] established the hardness of PPAD (and even that of a subclass known as CLS $\subseteq$ PPAD) under strong cryptographic assumptions. The first line of works, starting with that of Bitansky, Paneth and Rosen [BPR15], assumes sub-exponentially secure indistinguishability obfuscation, or functional encryption [GPS16, HY17].

The second line of works, which is more relevant to us, started with the work of Choudhuri *et al.* [CHK+19a] and relies on unambiguous incrementally updateable proofs. The work of [CHK+19a] assumes the adaptive soundness of the Fiat-Shamir transformation when applied to the sum-check protocol (and assuming that #SAT is hard on average). They then use the work of Canetti *et al.* [CCH+19], which proves adaptive soundness of the Fiat-Shamir transformation applied to the sum-check protocol, to obtain PPAD hardness assuming the existence of a perfectly secure FHE (and assuming that #SAT with polylog variables is hard on average). The work of [Pie19] relies on the soundness of the Fiat-Shamir transformation when applied to Pietrzak's interactive proof for repeated squaring [Pie19] (and assuming the hardness of repeated squaring modulo a composite).

Very recently, Lombardi and Vaikuntantanathan [LV20] proved soundness of the Fiat-Shamir transform applied to Pietrzak's interactive proof for repeated squaring, assuming LWE is $2^{\lambda^{1-\epsilon}}$-hard (w.r.t. a hash function that takes time $T(\lambda) = 2^{\lambda^\epsilon}$ time to compute), thus obtaining average-case CLS hardness under this assumption (and assuming that repeated squaring is 2^{λ^ϵ}-hard). Independently, Kalai, Paneth, and Yang [KPY20] obtained average-case CLS hardness under a quasi-polynomial-time assumption on groups with bilinear maps (and assuming SAT with $\log(n)^{1+\epsilon}$ variables is hard on average). Independently, Bitansky and Gerichter [BG20] prove average case hardness of the class PLS (which is not known to be contained in PPAD) under the same assumption.

Despite these works, obtaining PPAD hardness under a standard cryptographic assumption has remained an important open problem.

2 Technical Overview

We now outline our technical approach. We build on ideas from [CCH+19, PS19] to argue that the Fiat-Shamir paradigm is sound when applied to a rich class of public-coin interactive proofs (and in particular, when applied to the sum-check and the GKR protocols). In what follows, we first discuss the hash functions constructed by [CCH+19, PS19] and explain the difficulties with directly applying these hash functions to the sum-check (and to the GKR) protocol. We then explain our key idea of using a lossy trapdoor function family to get around these difficulties.

Background: Correlation Intractable Hash Functions [CGH04]. At a high level, a hash family H is correlation intractable (CI) for a relation $\mathcal{R}(x, y)$ if it is computationally hard, given a random hash key k, to find any input x such that $(x, H(k, x)) \in \mathcal{R}$. It was observed in [CGH04] that the Fiat-Shamir transform is sound if the initial protocol is statistically sound and the hash family used to reduce interaction is a CI hash family for all sparse relations.

Very recently, the beautiful works of [CCH+19, PS19] constructed a CI hash family for a restricted class of relations, assuming circular-secure LWE, and subsequently, plain LWE. The relations considered in these works are functions (i.e., for every x there is a single y such that $\mathcal{R}(x, y) = 1$) of a priori bounded size. Specifically, for any polynomial ρ they consider the class F of all functions computable by a ρ-size circuit. They construct a CI hash family H such that for any function $f \in F$ and any poly-size $\mathcal{A}$:

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

The works of [CCH+19, PS19] used the Fiat-Shamir paradigm, together with this hash family, to obtain a NIZK proof for NP based on LWE. Their main conceptual observation, which is the starting point of our work, is the following: There are several interesting interactive proofs for which the "bad" verifier challenge, which allows a prover to cheat, can be computed by an *efficient* (non-uniform) function.[12] Therefore, replacing the verifier message by the output of a CI hash function results in a verifier message that does not allow a prover to cheat, except with negligible probability.

In this work, we focus on applying the Fiat-Shamir transformation to succinct protocols, such as the sum-check and GKR protocols. In what follows, we first focus on the sum-check protocol, and explain why the approach of [CCH+19, PS19] fails when applied to the sum-check protocol. We then define a special CI hash family, which we call a *lossy* CI hash family. We show how this overcomes the failure point above, and argue that the resulting non-interactive sum-check protocol, obtained by applying the Fiat-Shamir transformation w.r.t. a lossy CI hash family, is sound. Then we show how to construct such a lossy CI hash famiy from a CI hash family and a lossy trapdoor family. Finally, we show that a lossy CI hash family can be used to securely instantiate the Fiat-Shamir paradigm for a broader class of interactive proofs, which we refer to as FS-*compatible* protocols (which includes the GKR protocol as well as the sum-check protocol). We end this overview with a brief explanation of how we obtain our PPAD hardness result.

We start with a brief description of the sum-check protocol.

[12] We note that this non-uniform advice is not efficiently computable, which is what makes these interactive proofs non-trivial.

The Interactive Sum-Check Protocol. In the sum-check protocol, the prover convinces the verifier that

$$\sum_{b_1,\ldots,b_\ell \in B} g(b_1,\ldots,b_\ell) = v,$$

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

$$g_1(\cdot) = \sum_{b_2,\ldots,b_\ell \in B} g(\cdot, b_2,\ldots,b_\ell).$$

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

$$\sum_{b_2,\ldots,b_\ell \in B} g(t_1, b_2,\ldots,b_\ell) = g_1(t_1).$$

In the next round, the prover sends

$$g_2(\cdot) = \sum_{b_3,\ldots,b_\ell \in B} g(t_1, \cdot, b_3,\ldots,b_\ell).$$

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

$$\sum_{b_3,\ldots,b_\ell \in B} g(t_1, t_2, b_3,\ldots,b_\ell) = g_2(t_2).$$

This continues for ℓ rounds, at the end of which the verifer checks on its own that $g_\ell(t_\ell) = g(t_1,\ldots,t_\ell)$ for a random $t_\ell \leftarrow \mathbb{F}$.

Non-Interactive Sum-Check via Fiat-Shamir: First Attempt. We consider applying the Fiat-Shamir transform [FS86] instantiated with a CI hash family (for all functions computable by bounded size circuits) to the sum-check protocol, in order to get a non-interactive argument in the CRS model. Specifically, the CRS contains ℓ hash keys, $k_1,\ldots,k_\ell$, one for each verifier message, and the prover non-interactively computes the i'th verifier message as $H(k_i, \tau_i)$, which is the outcome of the i'th hash function $H(k_i,\cdot)$ applied to the transcript so far.

Recall that in order to use the ideas from [CCH+19, PS19], we need to argue that there is an efficient (non-uniform) function that computes the "bad" challenge, where a bad challenge is one that allows the prover to cheat. Let us first understand what a bad challenge is in the context of the sum-check protocol. Denote by $g_1,\ldots,g_\ell$ the messages of the honest prover in the sum-check protocol, and denote by $g_1^*,\ldots,g_\ell^*$ the messages of a malicious prover. Note that if a malicious prover (successfully) cheats, then it must be that $g_1^* \neq g_1$. The verifier sends a challenge $t_1 \leftarrow \mathbb{F}$, which reduces the task of proving the original sum-check to the task of proving that

$$\sum_{b_2,\ldots,b_\ell \in B} g(t_1, b_2,\ldots,b_\ell) = g_1^*(t_1).$$

We say that t_1 is a bad challenge if $g_1^*(t_1) = g_1(t_1)$, since it converts a false statement to a true statement, and thus allows the prover to cheat.

The first issue we encounter is the following: The polynomial $g_1^* - g_1$ is of degree d and therefore may have as many as d roots, which implies that there can be as many as d bad challenges. Recall that the CI hash family defined and constructed in [CCH+19, PS19] avoids only a *single, efficiently computable* bad challenge, whereas here we have d possible bad challenges that we must all avoid.

Fortunately, *any* CI hash family, as described above, that avoids only a single bad challenge readily extends to avoid d possible bad challenges (as observed in [CCH+19]). Namely, if we let f denote an efficiently computable function that outputs one of the d bad challenges at random, then for any poly-size adversary $\mathcal{A}$,

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

This means that if $f_1(x), \ldots, f_d(x)$ are the d bad challenges, then for any poly-size $\mathcal{A}$,

$$\Pr_{k \leftarrow \mathsf{Gen}(1^\lambda)}[\mathcal{A}(k) = x \ : \ H(k,x) \in \{f_1(x), \ldots, f_d(x)\}] = d \cdot \mathsf{negl}(\lambda).$$

The next main obstacle is the following: in order to apply the CI hash family of [CCH+19, PS19], we need the bad challenge function (that outputs a random bad challenge) to be efficiently computable. It is quite straightforward to argue that the bad challenge function corresponding to the *first* verifier challenge is efficiently computable: The non-uniform advice is the (true) function g_1 and the function on input g_1^* outputs a random root of $g_1^* - g_1$, which can be computed efficiently (for example) via the Cantor-Zassenhaus algorithm [CZ81].

The problem kicks in after the first message. For a general round $i \in [\ell]$, the bad challenges are the roots of the polynomial $g_i^* - g_i$, for the same reason as for the first round: the prover can transition from a false statement in round i to a true statement in round $i+1$ if and only if $g_i(t_i) = g_i^*(t_i)$. It is tempting to simply hardwire g_i as the non-uniform advice. However, recall that

$$g_i(\cdot) = \sum_{b_{i+1}, \ldots, b_\ell \in B} g(t_1, \ldots, t_{i-1}, \cdot, b_{i+1}, \ldots, b_\ell),$$

and thus, it depends on the first $i-1$ challenges of the verifier $t_1, \ldots, t_{i-1}$. In the non-interactive setting these challenges depend on the previous prover messages, and thus cannot be a priori fixed and hardwired as non-uniform advice. Moreover, computing this function g_i (as opposed to taking it as non-uniform advice) takes time $O(|B|^{\ell-i})$, which may super polynomial in general.

As a first attempt, we consider guessing the first $i-1$ values $t_1, \ldots, t_{i-1}$, using these guesses to (non-uniformly) compute g_i, and hardwiring the resulting g_i into the bad challenge function. As before, on input g_i^*, the bad challenge function f efficiently computes a random root of the polynomial $g_i^* - g_i$ via the Cantor-Zassenhaus algorithm [CZ81]. Now if the challenges $t_1, \ldots, t_{i-1}$ were guessed correctly, the function f indeed outputs a random bad challenge.

Note that for every $i \in [\ell]$, $t_1, \ldots, t_{i-1}$ are guessed correctly with probability $1/|\mathbb{F}|^{i-1}$. Unfortunately, we cannot afford to have such a small probability of guessing correctly. We cannot even afford our guess to be correct with probability $1/|\mathbb{F}|$, since our hash functions output hash values in $\{0,1\}^{\log|\mathbb{F}|}$ (they output elements $t \in \mathbb{F}$). Thus we cannot hope to argue that a poly-size adversary outputs a bad challenge with probability smaller than $1/|\mathbb{F}|$ (since a random guess will be a bad challenge with probability $\geq 1/|\mathbb{F}|$). However, we could afford guessing correctly with probability $2^{-(\log|\mathbb{F}|)^\epsilon}$ (for a small enough constant $\epsilon > 0$), and then rely on sub-exponential security of their CI hash family (which in turn corresponds to relying on the sub-exponential hardness of the LWE assumption).

Lossy Correlation Intractable Hashing to the Rescue. In order to achieve the goal that the correct polynomial g_i is guessed and hardwired to the bad challenge function with probability at least $2^{-(\log|\mathbb{F}|)^\epsilon}$, we will make it possible to *artificially* decrease the output space of the hash function family, so that $t_1, \ldots, t_{i-1}$ are guessed correctly with probability at least $2^{-(\log|\mathbb{F}|)^\epsilon}$. To this end, we define and construct a *lossy correlation-intractable hash family*, which in addition to satisfying the CI property discussed above, has an extra *lossy mode*. In this mode, the space of outcomes (or image) of the hash function is restricted to being of size at most $2^{(\log|\mathbb{F}|)^\epsilon}$ for a small enough constant $\epsilon > 0$.

To see why this is helpful, consider the non-interactive sum-check protocol obtained by applying the Fiat-Shamir transform, where the CRS contains ℓ hash keys $k_1, \ldots, k_\ell$ corresponding to the *lossy* CI hash family. Now suppose there exists a cheating prover P^* that successfully cheats in the resulting non-interactive sum-check protocol with non-negligible probability. Recall that this means there must be a round $i \in [\ell]$ such that $g_i^* \neq g_i$ and yet $g_i^*(t_i) = g_i(t_i)$ (with non-negligible probability).

In the analysis, one can consider an alternative distribution of hash keys $k_1, \ldots, k_\ell$, where the first $i-1$ hash keys $k_1, \ldots, k_{i-1}$ are generated using the lossy mode, and the rest of the keys are generated as before (using the standard mode). Assuming that keys generated via the lossy mode are (sufficiently) indistinguishable from keys generated using the standard mode, we conclude that it will still be the case that $g_i^* \neq g_i$ and yet $g_i^*(t_i) = g_i(t_i)$ (with non-negligible probability). To be more precise, we will assume that keys generated in the lossy mode are indistinguishable even for $\mathsf{poly}(|B|^\ell)$-size adversaries – because g_i can be computed in time $\mathsf{poly}(|B|^\ell)$. At this point, we can guess $t_1, \ldots, t_{i-1}$ with probability $(2^{-(\log|\mathbb{F}|)^\epsilon})^{i-1} \geq 2^{-(\log|\mathbb{F}|)^{2\epsilon}}$ assuming $\log|\mathbb{F}| \geq \ell^{1/\epsilon}$, and in turn contradict the sub-exponential CI property of the underlying CI hash family.

Constructing Lossy CI Hash Functions. Our construction of a lossy CI hash family (for all bounded-size circuits) combines a lossy trapdoor family [PW08] with a CI hash family (for all bounded-size circuits) [PS19], both of which can be constructed based on the LWE assumption. As mentioned above, we will need our lossy CI hash family to be sub-exponentially secure, and as a result we rely on the sub-exponential hardness of LWE.

A lossy trapdoor family, defined and constructed by Peikert and Waters [PW08], is 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 efficiently invert the function. The lossy mode, in contrast, information theoretically loses information about its input. More specifically, it is associated with a lossy parameter ω and the guarantee is that the size of the function's output space is bounded by $2^{n-\omega}$, assuming the domain is $\{0,1\}^n$.

We construct lossy CI hash functions by concatenating CI hash functions with lossy trapdoor functions. Each lossy CI hash key consists of a pair of keys (k, k) where k is a key for a (underlying) CI hash function and k is a key for the lossy trapdoor function. In the normal mode of operation, k is generated using the injective mode of the underlying lossy trapdoor family. In the lossy mode of operation of our lossy CI hash function, k is generated using the lossy mode of the underlying lossy trapdoor family. To evaluate a lossy CI hash function with keys (k, k) on input x, we first compute the lossy trapdoor function with key k on input x to obtain a value y, and then evaluate the CI hash function with key k on input y. Denoting the underlying CI hash family by H, the underlying lossy family by G, and our resulting lossy CI hash family by H', we have that

$$H'((k, \mathsf{k}), x) \triangleq H(k, G(\mathsf{k}, x)).$$

We denote the standard key generation algorithm (which generates (k, k) where k is an injective key) by Gen and the lossy one (where k is a lossy key) by $\mathsf{LossyGen}$.

The indistinguishability of keys generated by Gen and keys generated by $\mathsf{LossyGen}$, follows from the security of the trapdoor hash family, which asserts that injective keys are indistinguishable from lossy ones. Importantly, as we argue below, the Gen mode continues to satisfy correlation intractability (for all functions computable by bounded-size circuits, though the bound here is smaller than the bound for the underlying CI hash family H). Roughly speaking, this is due to the trapdoor inversion algorithm, together with the fact that H is a CI hash family (for all functions computable by bounded-size circuits). More specifically, fix any function f that is computable by a bounded-size circuit. To argue that for every poly-size $\mathcal{A}$

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

it suffices to argue that for every poly-size $\mathcal{A}$

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

where $\mathsf{Inv}(\mathsf{td},\cdot)$ is the inversion algorithm that on input $G(\mathsf{k},x)$ outputs x. The function $f_{\mathsf{td}}(\cdot) \triangleq f(\mathsf{Inv}(\mathsf{td},\cdot))$ is computed by a bounded size circuit, and thus the equation above holds by the CI property of H, assuming H is CI for the family of all functions computable by circuits of size ρ, where ρ is such that $f \circ f_{\mathsf{td}}$ can be computed by a circuit of size ρ.

It remains to notice that in $\mathsf{LossyGen}$ mode, the underlying lossy function is first applied to the input x, which restricts the space of possible outcomes to $2^{n-\omega}$, as desired. The formal definition, construction and analysis of a lossy CI hash family can be found in Section 3.

FS-Compatible Interactive Proofs. So far we argued the soundness of the non-interactive sum-check protocol, obtained by applying the Fiat-Shamir transformation w.r.t. a lossy CI hash family. It may appear that we relied on many specific properties of the interactive sum-check protocol. However, as we argue below, our techniques are quite general. Specifically, using a lossy CI hash family and the template described above, we can eliminate interaction in any public-coin interactive proof (for some language L) as long as it has the following two key properties.

The first property is round-by-round-soundness as defined by [CCH+19]. Loosely speaking, this means that every possible transcript prefix defines a statement which is either true or false. If the statement is false then a cheating prover cannot expand it to an accepting transcript (except with negligible probability). If it is true then it can be expanded to an accepting transcript (with probability 1). The other property is that for any rejecting transcript prefix $\tau = (\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1})$, where $\alpha_1, \ldots, \alpha_{i-1}$ are the prover messages and $\beta_1, \ldots, \beta_{i-1}$ are the verifier messages, there is a non-uniform advice (which depends only on the verifier's messages) such that there is an efficient (non-uniform) function BAD that is given this advice, and on input a prover's next message α_i, efficiently computes a verifier's next message β_i that converts the rejecting transcript prefix τ into an accepting prefix $\tau|\alpha_i|\beta_i$. We describe these properties more formally below.

- **Round-by-round soundness [CCH+19].** There is a (not necessarily efficient) algorithm State that takes as input an instance x and a transcript prefix $\tau = (\alpha_1, \beta_1, \ldots, \alpha_i, \beta_i)$, and outputs accept or reject, such that for every (x, τ), if $\mathsf{State}(x, \tau) = \mathsf{reject}$ then for any next message of the prover α, with overwhelming probability over the next verifier message β, it holds that

$\mathsf{State}(x, \tau|\alpha|\beta) = \mathsf{reject}$. Moreover, for every $x \notin L$, $\mathsf{State}(x, \emptyset) = \mathsf{reject}$, and for every $x \in L$ and honestly generated prefix τ, $\mathsf{State}(x, \tau) = \mathsf{accept}$

- **Efficient** BAD **function.** For every $x \notin L$ and every $i \in [\ell]$ (where ℓ denotes the number of rounds), and every fixing of the first $i-1$ verifier messages, denoted by $\beta_1, \ldots, \beta_{i-1}$, there exists a (non uniform) efficient randomized function BAD that takes the first i messages of the prover, denoted by $\alpha_1, \ldots, \alpha_i$, and outputs an element β_i, such that if $\mathsf{State}(x, \tau) = \mathsf{reject}$, for $\tau = (\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1}, \alpha_i)$, then β_i is a random element in the (small) set

$$\{\beta \ : \ \mathsf{State}(x, \tau|\beta) = \mathsf{accept}\}. \tag{1}$$

Any interactive proof that has these two properties is said to be FS-*compatible*. More specifically, we associate with each FS-compatible protocol parameters (T', d, ρ), where T' is the time it takes to compute the function State,[13] d is a bound on the number of bad challenges, i.e., a bound on the size of the set defined in Equation (1),[14] and ρ is a polynomial bound on the size of the circuit computing the function BAD.

We prove that both the sum-check protocol and the GKR protocol are FS-compatible (w.r.t. some parameters (T', d, ρ)). This is proven formally in Sections 5 and 6, respectively. Moreover, we prove that applying the Fiat-Shamir transform w.r.t. a lossy CI hash family (for all functions computable by a bounded-size circuit) to any FS-compatible interactive proof results in a sound non-interactive argument. More specifically, if the interactive proof is (T', d, ρ) FS-comptible, then to prove soundness, the lossy CI hash family needs to be (T, T', ω) secure for all functions computable by ρ-size circuits (for a large enough T), which means that the following three conditions hold:

- T'**-Key Indistinguishability.** A $\mathsf{poly}(T')$-size adversary cannot distinguish between lossy keys (generated by LossyGen) and standard ones (generated by Gen).
- T**-Correlation Intractability.** For any $\mathsf{poly}(T)$-size adversary $\mathcal{A}$, and any function f computed by a ρ-size circuit,
$$\Pr[\mathcal{A}(k) = x \ : \ H(k, x) = f(x)] = \mathsf{negl}(T).$$
- ω**-Lossiness.** For every key k generated by LossyGen, the number of elements in the co-domain of this hash family is $2^{n-\omega}$, assuming the domain is $\{0,1\}^n$.

Moreover, T should be larger than the inverse probability of guessing all the verifier's messages in a protocol transcript, assuming the hash functions are generated in lossy mode.

Roughly speaking, the analysis of the non-interactive protocol is very similar to the analysis for the sum-check protocol, and proceeds as follows: Assume that there exists a cheating prover P^* who successfully cheats w.r.t. some instance $x \notin L$. Denote the transcript by τ and denote the transcript prefix corresponding to the first i rounds by τ_i. There must be a round $i \in [\ell]$ such that with non-negligible probability

$$\mathsf{State}(x, \tau_{i-1}) = \mathsf{reject} \ \ \wedge \ \ \mathsf{State}(x, \tau_i) = \mathsf{accept}.$$

In the analysis, we consider an alternative distribution of hash keys $k_1, \ldots, k_\ell$, where the first $i-1$ hash keys $k_1, \ldots, k_{i-1}$ are generated by LossyGen, and the rest of the keys are generated by

[13] This function is usually inefficient for the verifier to compute on his own.

[14] The parameter d can be super-polynomial, though for the sum-check and GKR protocols it is polynomial.

Gen. The T'-key indistinguishability property and the fact that State is computable in time T' imply that the above equation still holds with non-negligible probability. By the ω-lossy property, we can guess $\beta_1, \ldots, \beta_{i-1}$ with probability $2^{-(i-1)(n-\omega)}$. By the FS-compatibility property, there exists a ρ-size function BAD such that $\mathsf{BAD}(\alpha_1, \ldots, \alpha_i)$ outputs an element β_i so that for $\tau = (\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1}, \alpha_i)$, if $\mathsf{State}(x, \tau) = \mathsf{reject}$, then β_i is a random element in the set defined in Equation (1). Finally, we rely on the T-CI to argue that for every $\mathsf{poly}(T)$-size adversary,

$$\Pr[\mathcal{A}(k) = \alpha \ : \ H(k, \alpha_1, \ldots \alpha_i) = \mathsf{BAD}(\alpha_1, \ldots, \alpha_i)] = \mathsf{negl}(T)$$

This contradicts our assumption, as long as $T \geq 2^{\ell(n-\omega)}$. This proof sketch is oversimplified, and we refer the reader to Section 4 for the precise theorem and proof.

Applying this to the GKR protocol implies that the resulting non-interactive GKR is sound. However, the fact that T must be exponential in the number of rounds, which in the GKR protocol corresponds to the depth of the computation being delegated, implies that the communication complexity (as well as the verifier running time) becomes polynomial in the depth (as opposed to linear in the depth). To obtain verification time that is linear in the depth of computation, we consider a refined version of the above analysis, where we consider *history-independent* protocols, which are protocols where the prover and verifier can forget the initial transcript, and only need to memorize the last ν rounds (for some parameter ν). For such ν-history-independent protocols, we are able to get better parameters, and in particular, need $T \geq 2^{\nu(n-\omega)}$, where in the GKR protocol ν does not depend on the depth, but rather depends only poly-logarithmically on the size of the circuit being delegated. We refer the reader to Section 4 for details.

PPAD Hardness. To establish the sub-exponential 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 sum-check* can be used to reduce #SAT instances to rSVL instances, where rSVL is a problem in the class $\mathsf{CLS} \subseteq \mathsf{PPAD}$. Roughly, unambiguity requires that it is (computationally) hard to find *two different accepting proofs*, even for a true statement. While our non-interactive sum-check protocol does satisfy unambiguity due to the special structure of the sum-check protocol, we unfortunately fall short of proving full-fledged adaptive soundness. Loosely speaking, this is due to the fact that the multi-variate polynomial g needs to be known in advance in order to precompute the true claims g_i for each round $i \in [\ell]$.

However, we observe that the proof in [CHK+19a] does not require full adaptivity over the choice of g. Specifically, they require a weaker form of adaptive unambiguous soundness, which we call *prefix adaptive* unambiguous soundness, where g is chosen non-adaptively but a prefix $\sigma_1, \ldots, \sigma_j$ can be chosen adaptively, and they require that the non-interactive sum-check proof of the (partially adaptive) statement

$$\sum_{b_{j+1}, \ldots, b_\ell \in B} g(\sigma_1, \ldots, \sigma_j, b_{j+1}, \ldots, b_\ell)$$

satisfies soundness and unambiguity. We also observe that their proof only requires unambiguous soundness to hold for prefixes $\sigma_1, \ldots, \sigma_j$ for which each element σ_i in the prefix is either in the support of the hash function or an element in $[0, d]$.[15]

[15] We identify the set $[0, d]$ with a set of $d + 1$ field elements.

The proof techniques we developed allow us to prove such prefix adaptive unambiguous soundness for the non-interactive sum-check protocol. We mention that the guarantee that the prefix $\sigma_1, \ldots, \sigma_j$ is in the image of the hash function (or is in $[0, d]$) is crucial for us, since otherwise the adaptivity in σ would be too large for us to handle. Since we have the guarantee that these elements are in the image of the hash function, in the analysis we can modify the hash keys to be lossy and hence obtain the guarantee that $\sigma_1, \ldots, \sigma_j$ each belong to a small set of size $2^{n-\omega}$, which is small enough for us to guess these values in order to prove unambiguous soundness of the non-interactive sum-check protocol. We refer the reader to Section 7 for details.

Notation. Throughout this paper, for any probabilistic algorithm $\mathcal{A}$ and any input x, we use the terminology "for every $y \in \mathcal{A}(x)$" as a shorthand to say "for every y in the support of $\mathcal{A}(x)$". Throughout this work, we consider interactive proofs $\Pi = (P, V)$, and make the following simplifying assumptions:

1. Denoting by ℓ the number of rounds in Π, and we always assume for simplicity and w.l.o.g. that each round consists of two messages, the first sent by the prover and the second sent by the verifier. Thus a transcript of an ℓ-round protocol is of the form $\tau = (\alpha_1, \beta_1, \ldots, \alpha_\ell, \beta_\ell)$, where $(\alpha_1, \ldots, \alpha_\ell)$ are the prover messages and $(\beta_1, \ldots, \beta_\ell)$ are the verifier messages. For every $i \in [\ell]$, we denote by τ_i the first i rounds of τ. Namely, $\tau_i \triangleq (\alpha_1, \beta_1, \ldots, \alpha_i, \beta_i)$. For every $1 \leq i < j \leq \ell$, we denote by $\tau_{[i,j]} \triangleq (\alpha_i, \beta_i, \ldots, \alpha_j, \beta_j)$.
2. We assume w.l.o.g. that all of the verifier messages $\beta_1, \ldots, \beta_\ell$ are of the same length, and we denote this length by $\lambda = \lambda(|x|)$, where x is the input. Intuitively, this length determines the soundness of the interactive proofs that we are interested in (FS-compatible ones).

We think of all the parameters (except $|x|$) as a function of λ, unless explicitly stated otherwise. For example, we think of ℓ as a function of λ, as opposed to a function of $|x|$. This is useful since we use a Fiat-Shamir hash function with security parameter $\lambda(|x|)$, and thus bounding all the parameters in terms of λ will be helpful when proving soundness of the Fiat-Shamir transformation.

3 Lossy Correlation Intractable Hash Functions

In this section we define and construct the lossy correlation intractable (lossy CI) hash family that we use for instantiating the Fiat-Shamir paradigm. This is obtained by combining any lossy trapdoor function family with any correlation intractable (CI) hash family for bounded size circuits. In Section 3.1, we recall the notion of a lossy trapdoor function family and the notion of a CI hash family. Then, in Section 3.2, we define our notion of a lossy CI hash family, and provide a construction.

3.1 Preliminaries

3.1.1 Lossy Trapdoor Functions

Lossy trapdoor functions were first defined and constructed (based on LWE) in an influential work of Peikert and Waters [PW08]. Loosely speaking, a lossy trapdor function family contains two types of functions: injective ones and lossy ones, such that one cannot distinguish between a random

injective function in the family and a random lossy function in the family. An injective function can be generated together with a trapdoor, which allows one to efficiently invert the function, whereas a lossy function "loses" most information about its preimage, since its image is much smaller than its domain.

Definition 3.1 ((T, ω)-Lossy Trapdoor Family)**.** A quadruple $(\mathsf{InjGen}, \mathsf{LossyGen}, \mathsf{Eval}, \mathsf{Inv})$ of PPT algorithms is said to be a (T, ω)-lossy trapdoor function family if there exist polynomials $n = n(\lambda)$, $n' = n'(\lambda)$, $s = s(\lambda)$ and $t = t(\lambda)$ for which the following syntax and properties are satisfied:

- **Syntax.**
  - $\mathsf{InjGen}(1^\lambda)$ takes as input a security parameter 1^λ and outputs a pair $(\mathsf{k}, \mathsf{td})$, where $\mathsf{k} \in \{0,1\}^s$ is a key corresponding to an injective function and $\mathsf{td} \in \{0,1\}^t$ is a corresponding trapdoor.
  - $\mathsf{LossyGen}(1^\lambda)$ takes as input a security parameter 1^λ and outputs a key $\mathsf{k} \in \{0,1\}^s$ corresponding to a lossy function.
  - $\mathsf{Eval}(\mathsf{k}, x)$ takes as input a key $\mathsf{k} \in \{0,1\}^s$ and an element $x \in \{0,1\}^n$ and outputs an element $y \in \{0,1\}^{n'}$.
  - $\mathsf{Inv}(\mathsf{k}, \mathsf{td}, y)$ takes as input a key $\mathsf{k} \in \{0,1\}^s$, a trapdoor $\mathsf{td} \in \{0,1\}^t$, and an element $y \in \{0,1\}^{n'}$, and outputs an element $x \in \{0,1\}^n \cup \{\bot\}$.
- **Properties.** The following properties hold:
  - **Injective Mode.** For every $\lambda \in \mathbb{N}$ and every $\mathsf{k} \in \mathsf{InjGen}(1^\lambda)$ the function $\mathsf{Eval}(\mathsf{k}, \cdot)$ is injective. Furthermore, for every $x \in \{0,1\}^{n(\lambda)}$, $\Pr[\mathsf{Inv}(\mathsf{k}, \mathsf{td}, \mathsf{Eval}(\mathsf{k}, x)) = x] = 1$.[16]
  - **ω-Lossiness.** For every $\lambda \in \mathbb{N}$ and every $\mathsf{k} \in \mathsf{LossyGen}(1^\lambda)$ the function $\mathsf{Eval}(\mathsf{k}, \cdot)$ has an image of size $2^{n(\lambda)-\omega(\lambda)}$.
  - **T-Security.** There exists a negligible function μ such that for every $\mathsf{poly}(T)$-size adversary $\mathcal{A}$ and for every $\lambda \in \mathbb{N}$,

$$\left| \Pr_{\mathsf{k} \leftarrow \mathcal{G}.\mathsf{LossyGen}(1^\lambda)} [\mathcal{A}(\mathsf{k}) = 1] - \Pr_{\mathsf{k} \leftarrow \mathcal{G}.\mathsf{InjGen}(1^\lambda)} [\mathcal{A}(\mathsf{k}) = 1] \right| = \mu(T(\lambda))$$

Theorem 3.2. *[PW08, BDGM19, DGI+19]. Assuming the sub-exponential hardness of* LWE*, for every constant $0 < \delta < 1$ and every polynomial $n(\lambda)$, there exists a constant $0 < \epsilon < 1$ for which there exists a (T, ω)-lossy function family for $\omega(\lambda) = n(\lambda) - \lambda^\delta$ and $T = 2^{\lambda^\epsilon}$.*

3.1.2 CI Hash Family

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

Definition 3.3. A hash family $\mathcal{H}$ is associated with two algorithms $(\mathcal{H}.\mathsf{Gen}, \mathcal{H}.\mathsf{Hash})$, and a parameter $n = n(\lambda)$, such that:

[16] Typically, this requirement is only required to hold with overwhelming probability. We require it to hold with probability 1, only for the sake of simplifying the proof. This stronger requirement can be obtained assuming a leveled FHE with perfect correctness, following the construction in [HO12, BDGM19].

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

In this work we focus on hash families $\mathcal{H}$ such that for every $\lambda \in \mathbb{N}$, every key $k \in \mathcal{H}.\mathsf{Gen}(1^\lambda)$ and every $x \in \{0,1\}^{n(\lambda)}$, the output $y = \mathcal{H}.\mathsf{Hash}(k,x)$ is in $\{0,1\}^\lambda$. Throughout this work, we always assume that this is the case.

In what follows when we refer to a hash family, we usually do not mention the domain parameter n explicitly.

Definition 3.4 (T-Correlation Intractable)**.** [CGH04] A hash family $\mathcal{H} = (\mathcal{H}.\mathsf{Gen}, \mathcal{H}.\mathsf{Hash})$ is said to be *T-correlation intractable* (T-CI) for a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ if the following two properties hold:

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

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

The work of Canetti *et al.* [CCH+19] constructs a CI hash family for any function family $\mathcal{F}$ that consists of functions computable by bounded size circuits (where the runtime of the CI hash functions grows polynomially with this bound), assuming the existence of a sub-exponential circular secure FHE scheme. Following their work, Peikert and Shiehian [PS19] overcome the need to rely on circular security and obtain such a CI hash family from the sub-exponential LWE assumption.

Theorem 3.5. *[PS19] Assuming the sub-exponential hardness of* LWE*, there exists a constant $0 < \epsilon < 1$ such that for any polynomial S and any function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, where $\mathcal{F}_\lambda$ consists of functions that are computable by circuits of size $S(\lambda)$, there exists a T-CI hash family $\mathcal{H} = (\mathcal{H}.\mathsf{Gen}, \mathcal{H}.\mathsf{Hash})$ for $\mathcal{F}$ (Definition 3.4), where $T = 2^{\lambda^\epsilon}$. Moreover, there exists a polynomial p such that for every $\lambda \in \mathbb{N}$ and every $k \in \mathcal{H}.\mathsf{Gen}(1^\lambda)$, the function $\mathcal{H}.\mathsf{Eval}(k,\cdot)$ is computable by a circuit of size $p(S(\lambda))$.*

3.2 Lossy CI Hash Functions

In this section we show how to take any CI hash family (Definition 3.4), together with any family of lossy trapdoor functions (Definition 3.1), and obtain what we call a *lossy* CI hash family.

Definition 3.6 ((T, T', ω)-Lossy CI)**.** A hash family

$$\mathcal{H} = (\mathcal{H}.\mathsf{Gen}, \mathcal{H}.\mathsf{LossyGen}, \mathcal{H}.\mathsf{Hash})$$

is said to be (T, T', ω)-*lossy* CI for a function family $\mathcal{F}$ if the following holds:

- $(\mathcal{H}.\mathsf{Gen}, \mathcal{H}.\mathsf{Hash})$ is a T-CI hash family for $\mathcal{F}$ (Definition 3.4).

- The additional key generation algorithm $\mathcal{H}.\mathsf{LossyGen}$ takes as input a security parameter λ and outputs hash key k, such that the following two properties hold:
  - **T'-Key Indistinguishability.** For every $\mathsf{poly}(T')$-size adversary $\mathcal{A}$, there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$

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

  - **ω-Lossiness.** For every $\lambda \in \mathbb{N}$ and every $k \in \mathcal{H}.\mathsf{LossyGen}(1^\lambda)$, denoting by $n = n(\lambda)$ the length of elements in the domain of $\mathcal{H}.\mathsf{Hash}(k, \cdot)$,

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

Theorem 3.7. *There exists a (T, T', ω)-lossy CI hash family for $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ (Definition 3.4) assuming the existence of the following primitives:*

- *A (T', ω)-lossy trapdoor function family $\mathcal{G}$ (Definition 3.1), such that for every $\lambda \in \mathbb{N}$, $f \in \mathcal{F}_\lambda$, and $\mathsf{k} \in \mathcal{G}.\mathsf{Gen}(1^\lambda)$, the domain of $\mathcal{G}.\mathsf{Eval}(\mathsf{k}, \cdot)$ is equal to the domain of f.*
- *A T-CI hash family $\mathcal{H}$ (Definition 3.4) for the function family $\mathcal{F}'$, where the family $\mathcal{F}' = \{\mathcal{F}'_\lambda\}_{\lambda \in \mathbb{N}}$ is defined as follows: for each $\lambda \in \mathbb{N}$, $f' \in \mathcal{F}'_\lambda$ if and only if there exists $f \in \mathcal{F}_\lambda$, and $(\mathsf{k}, \mathsf{td}) \in \mathcal{G}.\mathsf{Gen}(1^\lambda)$ such that $f'_\lambda(\cdot) = f_\lambda(\mathcal{G}.\mathsf{Inv}(\mathsf{k}, \mathsf{td}, \cdot))$.*

Corollary 3.8. *Assuming the sub-exponential hardness of LWE, there exists a constant $0 < \epsilon_1 < 1$ and, for every constant $0 < \delta < 1$, a constant $0 < \epsilon_2 < 1$ such that for any polynomial S and any function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ where $\mathcal{F}_\lambda$ consists of functions that are computable by circuits of size $S(\lambda)$, there exists a (T, T', ω)-lossy correlation intractable hash function family $\mathcal{H}$ for $\mathcal{F}$ (Definition 3.6). Here, $T = 2^{\lambda^{\epsilon_1}}$, $T' = 2^{\lambda^{\epsilon_2}}$, and $\omega = n - \lambda^\delta$, where n is the domain parameter associated with $\mathcal{H}$ (Definition 3.3).*

Proof. Assume the sub-exponential hardness of LWE. Then by Theorem 3.5 there exists a constant $0 < \epsilon_1 < 1$ such that for any polynomial S and any function family $\mathcal{F}' = \{\mathcal{F}'_\lambda\}_{\lambda \in \mathbb{N}}$, where $\mathcal{F}'_\lambda$ consists of functions that are computable by circuits of size $S(\lambda)$, there exists a T-CI hash family for $\mathcal{F}'$, where $T = 2^{\lambda^{\epsilon_1}}$. Moreover, by Theorem 3.2 for every constant $\delta > 0$, there exists $0 < \epsilon_2 < 1$ such that there exists a (T', ω)-lossy trapdoor family for $T'(\lambda) = 2^{\lambda^{\epsilon_2}}$ and $\omega = n - \lambda^\delta$. The corollary now follows from Theorem 3.7. □

Proof of Theorem 3.7. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be a function family, and let

$$\mathcal{G} = (\mathcal{G}.\mathsf{InjGen}, \mathcal{G}.\mathsf{LossyGen}, \mathcal{G}.\mathsf{Eval}, \mathcal{G}.\mathsf{Inv})$$

be a (T', ω)-lossy trapdoor family, such that for every $\lambda \in \mathbb{N}$, $f \in \mathcal{F}_\lambda$, and $\mathsf{k} \in \mathcal{G}.\mathsf{Gen}(1^\lambda)$, the domain of $\mathcal{G}.\mathsf{Eval}(\mathsf{k}, \cdot)$ is equal to the domain of f. Let

$$\mathcal{H} = (\mathcal{H}.\mathsf{Gen}, \mathcal{H}.\mathsf{Hash})$$

be a T-CI hash function family for function family $\mathcal{F}'$ as defined in the theorem statement.

Construction 3.9. *We construct our (T, T', ω)-secure lossy* CI *hash function family*

$$\mathcal{H}' = (\mathcal{H}'.\mathsf{Gen}, \mathcal{H}'.\mathsf{LossyGen}, \mathcal{H}'.\mathsf{Hash})$$

as follows:

- $\mathcal{H}'.\mathsf{Gen}(1^\lambda)$*:*
  - *Sample* $k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$.
  - *Sample* $(\mathsf{k}, \mathsf{td}) \leftarrow \mathcal{G}.\mathsf{InjGen}(1^\lambda)$.
  - *Output* $k' = (k, \mathsf{k})$.
- $\mathcal{H}'.\mathsf{LossyGen}(1^\lambda)$*:*
  - *Sample* $k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$.
  - *Sample* $\mathsf{k} \leftarrow \mathcal{G}.\mathsf{LossyGen}(1^\lambda)$.
  - *Output* $k' = (k, \mathsf{k})$.
- $\mathcal{H}'.\mathsf{Hash}(k', x)$*:*
  - *Parse* $k' = (k, \mathsf{k})$.
  - *Output* $\mathcal{H}.\mathsf{Hash}(k, \mathcal{G}.\mathsf{Eval}(\mathsf{k}, x))$.

$\mathcal{H}'$ is a T-CI hash family for $\mathcal{F}$. We first argue that $\mathcal{H}'$ is indeed a T-CI hash family for $\mathcal{F}$. To this end, note that for every $\lambda \in \mathbb{N}$, $f \in \mathcal{F}_\lambda$ and $(k, \mathsf{k}) \in \mathcal{H}'.\mathsf{Gen}(1^\lambda)$, the functions f and $\mathcal{H}'.\mathsf{Hash}((k, \mathsf{k}), \cdot) \triangleq \mathcal{H}.\mathsf{Hash}(k, \mathcal{G}.\mathsf{Eval}(\mathsf{k}, \cdot))$ have the same domain and co-domain. The equality of the domains follow from our assumption that the domains of $\mathcal{G}.\mathsf{Eval}(\mathsf{k}, \cdot)$ and f are equal. The equalities of the co-domains follow from our assumption that $\mathcal{H}$ is a T-CI hash function for $\mathcal{F}'$, which implies in particular that the co-domain of $\mathcal{H}.\mathsf{Hash}(k, \cdot)$ is equal to the co-domain of f', which in turn is equal to the co-domain of f, as desired.

Next we argue the T-CI property of $\mathcal{H}'$, from the T-CI property of $\mathcal{H}$ and the injectivity of the functions in $\mathcal{G}$ generated via the injective mode.

Suppose for the sake of contradiction that there exists a $\mathsf{poly}(T)$-size adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ such that for every $\lambda \in \mathbb{N}$, there is a function $f \in \mathcal{F}_\lambda$ and a non-negligible function ϵ such that

$$\Pr_{\substack{(k,\mathsf{k}) \leftarrow \mathcal{H}'.\mathsf{Gen}(1^\lambda) \\ x \leftarrow \mathcal{A}_\lambda(k,\mathsf{k})}} [\mathcal{H}'.\mathsf{Hash}((k, \mathsf{k}), x) = f(x)] \geq \epsilon(T(\lambda)).$$

This, together with a simple averaging argument, implies that for every $\lambda \in \mathbb{N}$ there exists $(\mathsf{k}, \mathsf{td}) \in \mathcal{G}.\mathsf{InjGen}(1^\lambda)$ such that

$$\Pr_{\substack{k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \\ x \leftarrow \mathcal{A}_\lambda(k,\mathsf{k})}} [\mathcal{H}'.\mathsf{Hash}((k, \mathsf{k}), x) = f(x)] \geq \epsilon(T(\lambda)). \tag{2}$$

Recall that

$$\mathcal{H}'.\mathsf{Hash}((k, \mathsf{k}), x) \triangleq \mathcal{H}.\mathsf{Hash}(k, \mathcal{G}.\mathsf{Eval}(\mathsf{k}, x)).$$

We use $\mathcal{A}_\lambda$ to define an adversary $\mathcal{B}_\lambda$ that breaks the T-CI property of $\mathcal{H}$. $\mathcal{B}_\lambda$ depends (non-uniformly) on $(\mathsf{k}, \mathsf{td})$ that satisfies equation (2). Given a key $k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$, $\mathcal{B}_\lambda$ runs $\mathcal{A}_\lambda$ on input (k, k) to obtain x, and outputs $y \triangleq \mathcal{G}.\mathsf{Eval}(\mathsf{k}, x)$. By relying on the injective mode property of the lossy trapdoor family (Definition 3.1), we conclude that

$$\Pr_{\substack{k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \\ y \leftarrow \mathcal{B}_\lambda(k)}} [\mathcal{H}.\mathsf{Hash}(k, y) = f(\mathcal{G}.\mathsf{Inv}(\mathsf{k}, \mathsf{td}, y)] \geq \epsilon(T(\lambda)).$$

Setting $f'_\lambda(\cdot) \triangleq f_\lambda(\mathcal{G}.\mathsf{Inv}(\mathsf{k}, \mathsf{td}, \cdot))$, this implies that

$$\Pr_{\substack{k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \\ y \leftarrow \mathcal{B}_\lambda(k)}} [\mathcal{H}.\mathsf{Hash}(k, y) = f'(y)] \geq \epsilon(T(\lambda)).$$

This, together with the fact that $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda \in \mathbb{N}}$ is of $\mathsf{poly}(T)$-size, which in turn follows from the fact that $\mathcal{A}$ is of $\mathsf{poly}(T)$-size, contradicts the T-CI property of $\mathcal{H}$ for $\mathcal{F}'$.

Properties of $\mathcal{H}'.\mathsf{LossyGen}$. We will now argue that $\mathcal{H}'$ has the following properties:

- **T'-Key Indistinguishability.** Suppose for the sake of contradiction that there exists a $\mathsf{poly}(T')$-size adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ and a non-negligible function ϵ such that for every $\lambda \in \mathbb{N}$,

$$\left| \Pr_{(k,\mathsf{k}) \leftarrow \mathcal{H}'.\mathsf{LossyGen}(1^\lambda)} [\mathcal{A}_\lambda(k, \mathsf{k}) = 1] - \Pr_{(k,\mathsf{k}) \leftarrow \mathcal{H}'.\mathsf{Gen}(1^\lambda)} [\mathcal{A}_\lambda(k, \mathsf{k}) = 1] \right| \geq \epsilon(T'(\lambda)).$$

  By a simple averaging argument this implies that for every $\lambda \in \mathbb{N}$ there exist $k \in \mathcal{H}.\mathsf{Gen}(1^\lambda)$ such that

$$\left| \Pr_{\mathsf{k} \leftarrow \mathcal{G}.\mathsf{LossyGen}(1^\lambda)} [\mathcal{A}_\lambda(k, \mathsf{k}) = 1] - \Pr_{\mathsf{k} \leftarrow \mathcal{G}.\mathsf{InjGen}(1^\lambda)} [\mathcal{A}_\lambda(k, \mathsf{k}) = 1] \right| \geq \epsilon(T'(\lambda)). \tag{3}$$

  We use $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ to construct an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda \in \mathbb{N}}$ that breaks the security of the lossy trapdoor function. For every $\lambda \in \mathbb{N}$, the adversary $\mathcal{B}_\lambda$ hardwires the key k for which Equation (3) holds, and on input k it outputs $\mathcal{A}_\lambda(k, \mathsf{k})$. Equation (3), together with the fact that $\mathcal{B}$ is of size $\mathsf{poly}(T')$, contradicts the T'-security of the lossy trapdoor function family $\mathcal{G}$.

- **ω-Lossiness.** The lossiness of $\mathcal{H}'$ follows from the lossiness of the underlying lossy trapdoor family $\mathcal{G}$, as follows: Fix any $\lambda \in \mathbb{N}$ and any $(k, \mathsf{k}) \in \mathcal{H}'.\mathsf{LossyGen}(1^\lambda)$. By definition, it holds that $\mathsf{k} \in \mathcal{G}.\mathsf{LossyGen}(1^\lambda)$. Denoting by $\{0,1\}^n$ the domain of $\mathcal{G}.\mathsf{Eval}(\mathsf{k}, \cdot)$, the ω-lossiness of the underlying lossy trapdoor family implies that

$$|\{\mathcal{G}.\mathsf{Eval}(\mathsf{k}, x)\}_{x \in \{0,1\}^n}| \leq 2^{n-\omega},$$

  which in turn implies that

$$|\{\mathcal{H}'.\mathsf{Hash}((k, \mathsf{k}), x\}_{x \in \{0,1\}^n}| = |\{\mathcal{H}.\mathsf{Hash}(k, \mathcal{G}.\mathsf{Eval}(\mathsf{k}, x))\}_{x \in \{0,1\}^n}| \leq 2^{n-\omega}.$$

□

4 Soundness of the Fiat Shamir Paradigm

In this section we identify a class of interactive proofs for which the Fiat-Shamir transformation is sound when applied with any lossy CI hash family (as defined in Section 3). We call such interactive proofs FS-compatible, and define them formally in Section 4.1. In Section 4.2, we recall the Fiat-Shamir transformation, and in Section 4.3 we prove its soundness when applied to a FS compatible proof (w.r.t. a lossy CI hash family).

4.1 FS-Compatible Interactive Proofs

Loosely speaking, an interactive proof is said to be FS-compatible if it has two properties: The first is "round-by-round" soundness [CCH+19], and the second is that there exists a non-uniform advice that allows one to efficiently compute a "bad" verifier message that goes from a rejecting transcript prefix to an accepting one.

The first condition, round-by-round soundness, is defined below. We note that our notion of round-by-round soundness differs slightly from the one defined in [CCH+19].

Definition 4.1 (Round-by-Round Soundness)**.** [CCH+19] Let $\Pi = (P, V)$ be a public-coin interactive proof system for a language L. We say that Π is (T', d)-round-by-round sound if there exists a deterministic 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. Let $\emptyset$ denote the empty transcript. Then for every $x \notin L$ it holds that $\mathsf{State}(x, \emptyset) = \mathsf{reject}$, and for every $x \in L$ it holds that $\mathsf{State}(x, \emptyset) = \mathsf{accept}$.

2. For every x and every transcript prefix $\tau = (\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1})$, if $\mathsf{State}(x, \tau) = \mathsf{reject}$, then for any (possibly unbounded) adversary $\mathcal{A}$, it holds that[17]

$$\Pr_{\substack{\alpha \leftarrow \mathcal{A}(x,\tau) \\ \beta \leftarrow \{0,1\}^\lambda}} [\mathsf{State}(x, \tau|\alpha|\beta) = \mathsf{accept}] \leq d(\lambda) \cdot 2^{-\lambda}. \tag{4}$$

3. For every complete transcript τ, if $\mathsf{State}(x, \tau) = \mathsf{reject}$ then $V(x, \tau) = 0$, and if $V(x, \tau) = 1$ then $\mathsf{State}(x, \tau) = \mathsf{accept}$.

4. State is computable in time at most $T'(\lambda)$.[18]

The second condition for FS-compatibility is that the interactive proof admits an efficiently computable randomized function BAD for every round $i \in [\ell]$. This function depends, possibly inefficiently, on the instance x and all the verifier's random challenges $\beta_1, \ldots, \beta_{i-1}$ sent before round i. It obtains as input the prover messages $(\alpha_1, \ldots, \alpha_i)$ and it outputs β_i such that if $\mathsf{State}(x, \tau) = \mathsf{reject}$ for $\tau \triangleq (\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1})$, then β_i is a random element in the set

$$\{\beta \ : \ \mathsf{State}(x, \tau|\alpha_i|\beta) = \mathsf{accept}\}.$$

[17] We point out that we modify the definition in [CCH+19] to replace $\mathsf{negl}(\lambda)$ with $d(\lambda) \cdot 2^{-\lambda}$ on the right hand side of Equation (4).

[18] This requirement did not exist in the definition in [CCH+19].

Definition 4.2 (FS-Compatible Interactive Proofs)**.** Let T' and d be functions (not necessarily polynomial) and let ρ be a polynomial. A public-coin interactive proof (P, V) for a language L that has $\ell(\lambda)$ rounds of interaction is said to be (T', d, ρ)-FS-compatible if the following two properties hold:

1. (P, V) is (T', d)-round-by-round sound w.r.t. a state function denoted by State (Definition 4.1).
2. For every $x \notin L$, $i \in [\ell]$ and $\beta_1, \ldots, \beta_{i-1}$, there exists a (non-uniform[19]) randomized function BAD that takes as input $(\alpha_1, \ldots, \alpha_i)$ and randomness r, runs in time $\rho(\lambda)$, and its output satisfies the following guarantee:

   For every $(\alpha_1, \ldots, \alpha_i)$ such that $\mathsf{State}(x, \tau) = \mathsf{reject}$, for $\tau \triangleq (\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1})$, w.p. $1 - \mathsf{negl}(\lambda)$ (over r), $\mathsf{BAD}(\alpha_1, \ldots, \alpha_i)$ outputs a uniformly random element in the set $\mathcal{B}$, where
   $$\mathcal{B} = \{\beta \ : \ \mathsf{State}(x, \tau|\alpha_i|\beta) = \mathsf{accept}\}.$$
   If $\mathcal{B} = \emptyset$, $\mathsf{BAD}(\alpha_1, \ldots, \alpha_i)$ outputs $\perp$. Note that by Equation (4), $|\mathcal{B}| \leq d(\lambda)$.

In Sections 5 and 6 we prove that the sum-check protocol and the GKR protocol are FS-compatible, respectively (for appropriate parameters). Jumping ahead, we note that generally speaking, the number of rounds $\ell(\lambda)$ in the GKR protocol can be super-polynomial, since typically $\lambda = \mathsf{polylog}(S)$, where S is the size of the computation being delegated, whereas ℓ grows with the depth of the computation, which in general can be much larger than $\mathsf{polylog}(S)$. Thus, it seems that there is no hope to argue that BAD is computable in polynomial time $\rho(\lambda)$ if its input length (which is of order $\ell(\lambda)$) is super-polynomial. Luckily, this is only a syntactic problem, since the function BAD corresponding to the GKR protocol does not depend on all the messages $(\alpha_1, \ldots, \alpha_i)$; rather it depends only on the last $\nu(\lambda) = \mathsf{poly}(\lambda)$ of them. We say that the GKR protocol is *ν-history-independent*, which essentially means that each prover message is only a function of the last ν rounds of communication. We formally define history-independence below.

Definition 4.3. A protocol $\Pi = (P, V)$ is said to be *ν-history-independent* if for every $i > \nu$ and for every prefix transcript $\tau_{i-1} = (\alpha_1, \beta_1 \ldots, \alpha_{i-1}, \beta_{i-1})$, the i'th prover message α_i is computed by $\alpha_i = P(x, i, \tau_{[i-\nu, i-1]})$, where $\tau_{[i-\nu, i-1]} \triangleq (\alpha_{i-\nu}, \beta_{i-\nu}, \ldots, \alpha_{i-1}, \beta_{i-1})$.

In what follows, we define the notion of a ν-history-independent FS-compatible interactive proof. This is done by restricting the functions State and BAD to be ν-history-independent.

Definition 4.4. We say that a (T', d, ρ)-FS-compatible interactive proof is *ν-history-independent* if it is (T', d, ρ)-FS-compatible w.r.t functions State and BAD such that:

- There exists a function State' computable in time $\leq T'(\lambda)$ such that for every x, every $i \in [\nu + 1, \ell]$, every $\tau_{i-1} = (\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1})$, it holds that
  $$\mathsf{State}(x, \tau_{i-1}) = \mathsf{State}'(x, i, \tau_{[i-\nu, i-1]}).$$
- For every $(x, \beta_1, \ldots, \beta_{i-1})$ (the non-uniform advice of) the function BAD depends only on $(x, i, \beta_{i-\nu}, \ldots, \beta_{i-1})$, and it takes as input only $\alpha_{i-\nu}, \ldots, \alpha_i$.

[19] The non-uniformity depends on $x, \beta_1, \ldots, \beta_{i-1}$.

Formally, for every x, $i \in [\nu+1, \ell]$ and every $(\beta_{i-\nu}, \ldots, \beta_{i-1})$ there exists a (non-uniform) randomized function BAD' of size ρ, that takes as input only $\alpha_{i-\nu}, \ldots, \alpha_i$, such for every $(\beta_1, \ldots, \beta_{i-\nu-1})$, letting BAD denote the bad function that corresponds to $(x, \beta_1, \ldots, \beta_{i-1})$, for every $(\alpha_1, \ldots, \alpha_{i-\nu-1})$

$$\mathsf{BAD}'(\alpha_{i-\nu}, \ldots, \alpha_i) = \mathsf{BAD}(\alpha_1, \ldots, \alpha_i).$$

In Section 6 we prove that the GKR protocol is ν-history-independent FS compatible, for $\nu(\lambda) \leq \mathsf{poly}(\lambda)$.

Note that if we let $\nu = \ell$, any (T', d, ρ)-FS-compatible interactive proof is also ℓ-history-independent. Hence, any property of ν-history-independent FS-compatible interactive proofs holds also for plain FS-compatible interactive proofs, by setting $\nu = \ell$. In particular, in Section 4.3, we prove the soundness of the Fiat-Shamir paradigm applied to FS-compatible proofs that are ν-history independent. By setting $\nu = \ell$, soundness holds for general FS-compatible proofs as well.

4.2 Preliminaries: The Fiat-Shamir Paradigm

Let $\Pi = (P, V)$ be any public-coin interactive proof for a language L. Let $n = n(\lambda)$ denote the communication complexity of Π. Let $\mathcal{H} = (\mathcal{H}.\mathsf{Gen}, \mathcal{H}.\mathsf{Hash})$ be hash family such that, for every security parameter $\lambda \in \mathbb{N}$ and every $k \in \mathcal{H}.\mathsf{Gen}(1^\lambda)$, $\mathcal{H}.\mathsf{Hash}(k, \cdot)$ is a function with a domain $\{0,1\}^{n(\lambda)}$ and co-domain $\{0,1\}^\lambda$. We will also allow inputs to $\mathcal{H}.\mathsf{Hash}(k, \cdot)$ that are shorter than n, by padding all inputs with 0's until the total length is n. We define the non-interactive protocol $\Pi^{\mathcal{H}}_{\mathsf{FS}} = (P', V')$, obtained by applying the Fiat-Shamir transform to Π w.r.t. the hash family $\mathcal{H}$, in Figure 1.

The Non-Interactive Argument $\Pi^{\mathcal{H}}_{\mathsf{FS}}$

Fix an input length $|x|$ and let $\lambda = \lambda(|x|)$.

- The common reference string CRS consists of ℓ keys $\{k_i \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)\}_{i \in [\ell]}$.
- The prover P' takes as input (CRS, x) and does the following:
  1. Set $i = 1$ and $\tau_0 = \emptyset$.
  2. Compute $\alpha_i \leftarrow P(x, \tau_{i-1})$ and $\beta_i = \mathcal{H}.\mathsf{Hash}(k_i, \tau_{i-1}|\alpha_i)$.
  3. Set $\tau_i = (\tau_{i-1}|\alpha_i|\beta_i)$.
  4. If $i = \ell$ then output τ_i. Otherwise, set $i = i + 1$ and go to Item 2.
- The verifier V' takes as input (CRS, x, τ) and does the following:
  1. Parse $\mathsf{CRS} = (k_1, \ldots, k_\ell)$ and $\tau = (\alpha_1, \beta_1, \ldots, \alpha_\ell, \beta_\ell)$.
  2. Accept if and only if $V(x, \tau) = 1$ and for every $i \in [\ell]$ it holds that $\beta_i = \mathcal{H}.\mathsf{Hash}(k_i, \tau_{i-1}|\alpha_i)$, where $\tau_{i-1} = (\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1})$.

Figure 1: The Non-Interactive Argument $\Pi^{\mathcal{H}}_{\mathsf{FS}}$

4.2.1 The Fiat-Shamir Paradigm for History-Independent Protocols

Suppose that Π is a ν-history-independent interactive proof (Definition 4.3).[20] We consider the following history-independent version of the Fiat-Shamir transformation, where we let n be the communication complexity in $\nu+1$ rounds of the protocol Π. For every $i \in [\ell]$, rather than computing β_i as the hash of the entire transcript so far, it is computed by hashing the most recent ν rounds:

$$\beta_i = \mathcal{H}(k_i, \tau_{[i-\nu,i-1]}|\alpha_i),$$

The formal description is included in Figure 2.

ν-History-Independent Version of $\Pi^{\mathcal{H}}_{\mathsf{FS}}$

Fix an input length $|x|$ and let $\lambda = \lambda(|x|)$.

- The common reference string CRS consists of ℓ keys $\{k_i \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)\}_{i\in[\ell]}$.
- The prover P' takes as input (CRS, x) and does the following:
  1. Set $i = 1$ and $\tau_0 = \emptyset$.
  2. Compute $\alpha_i \leftarrow P(x, \tau_{[i-\nu,i-1]})$ and $\beta_i = \mathcal{H}.\mathsf{Hash}(k_i, \tau_{[i-\nu,i-1]}|\alpha_i)$.
  3. Set $\tau_i = (\tau_{i-1}|\alpha_i|\beta_i)$.
  4. If $i = \ell$ then output τ_i. Otherwise, set $i = i+1$ and go to Item 2.
- The verifier V' takes as input (CRS, x, τ) and does the following:
  1. Parse $\mathsf{CRS} = (k_1, \ldots, k_\ell)$ and $\tau = (\alpha_1, \beta_1, \ldots, \alpha_\ell, \beta_\ell)$.
  2. Accept if and only if $V(x,\tau) = 1$ and for every $i \in [\ell]$ it holds that $\beta_i = \mathcal{H}.\mathsf{Hash}(k_i, \tau_{[i-\nu,i-1]}|\alpha_i)$.

Figure 2: The ν-History-Independent Version of $\Pi^{\mathcal{H}}_{\mathsf{FS}}$

4.3 The Soundness of the Fiat-Shamir Paradigm for FS-Compatible Interactive Proofs

In this section, we prove that the (ν-history-independent) Fiat-Shamir paradigm is sound when applied to (ν-history-independent) FS-compatible interactive proofs.

Theorem 4.5. *Let $\Pi = (P, V)$ be a (T', d, ρ)-FS-compatible interactive proof (Definition 4.2) that is ν-history-independent (Definition 4.4) for some language L, where ρ is a polynomial function. Let ℓ be an arbitrary function such that Π has $\ell = \ell(\lambda)$ rounds. Denote the verifier's messages by $\beta_1, \ldots, \beta_\ell \in \{0,1\}^\lambda$, denote the overall prover runtime by $T_P(\lambda)$ and the verification time by $T_V(\lambda)$.*

Let n be an upper bound on the communication complexity of any $\nu+1$ rounds of Π. Suppose that there exists a (T, T', ω)-lossy CI hash function family $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 3.6) for the family $\mathcal{F}$ consisting of all functions computable by circuits of size $\rho(\lambda)$. Moreover, suppose that the functions in $\mathcal{H}_\lambda$ map inputs in $\{0,1\}^{n(\lambda)}$ to outputs in $\{0,1\}^\lambda$, and that $T'(\lambda) \geq \ell(\lambda)$ and $T \geq \max\{d, 2^{\nu(n-\omega)}, T'\}$. Then

[20] Note that every Π is ν-history-independent for some $\nu \leq \ell - 1$.

the resulting non-interactive protocol $\Pi_{\mathsf{FS}}^{\mathcal{H}}$, obtained by applying the ν-history-independent Fiat-Shamir transform to Π w.r.t. the hash family $\mathcal{H}$ (Figure 2), has the following properties:

- **Completeness.** *If Π has completeness 1, then $\Pi_{\mathsf{FS}}^{\mathcal{H}}$ also has completeness 1.*
- **T'-Soundness.** *For any $\mathsf{poly}(T')$-size cheating prover P^*, there exists a negligible function μ such that for every $x^* \notin L$ and $\lambda = \lambda(|x^*|)$,*

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Setup}(1^\lambda)\\ \tau^*\leftarrow P^*(\mathsf{CRS})}}[(V_{\mathsf{FS}}^{\mathcal{H}}(\mathsf{CRS}, x^*, \tau^*) = 1)] = \mu(T'(\lambda)).$$

- **Efficiency.** *There exists a polynomial p that depends on the lossy CI hash family $\mathcal{H}$ such that the total verifier runtime is $\ell(\lambda) \cdot p(\rho(\lambda)) + T_V(\lambda)$ and the total prover runtime is $\ell(\lambda) \cdot p(\rho(\lambda)) + T_P(\lambda)$.*

The remainder of this section is devoted to the proof of Theorem 4.5.

Proof of Theorem 4.5. Fix an interactive proof $\Pi = (P, V)$ for a language L which is ν-history independent (T', d, ρ)-FS-compatible, and fix a (T, T', ω)-lossy CI hash family $\mathcal{H}$ for the family $\mathcal{F}$ consisting of all functions computable by circuits of size $\rho(\lambda)$. Assume that

$$T'(\lambda) \geq \ell(\lambda) \quad \text{and} \quad T \geq \max\{d, 2^{\nu(n-\omega)}, T'\}.$$

Completeness. Completeness is easy to see and follows directly from the completeness of the original protocol Π.

T'-Soundness. Assume for the sake of contradiction that there exists a $\mathsf{poly}(T')$-size prover P^*, a polynomial p', and an infinite set X of $x^* \notin L$ such that for each $x^* \in X$ and $\lambda = \lambda(|x^*|)$,

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Setup}(1^\lambda)\\ \tau^*\leftarrow P^*(\mathsf{CRS})}}[(V_{\mathsf{FS}}^{\mathcal{H}}(\mathsf{CRS}, x^*, \tau^*) = 1)] \geq \frac{1}{p'(T'(\lambda))}.$$

We assume w.l.o.g. that for every distinct $x_1, x_2 \in X$ it holds that $\lambda(|x_1|) \neq \lambda(|x_2|)$. This gives an infinite set $\Lambda \subseteq \mathbb{N}$ such that for every $\lambda \in \Lambda$, there is $x_\lambda^* \notin L$ such that $\lambda = \lambda(|x_\lambda^*|)$, and

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Setup}(1^\lambda)\\ \tau^*\leftarrow P^*(\mathsf{CRS})}}[(V_{\mathsf{FS}}^{\mathcal{H}}(\mathsf{CRS}, x_\lambda^*, \tau^*) = 1)] \geq \frac{1}{p'(T'(\lambda))}.$$

Parse the proof that P^* outputs as $\tau^* = (\alpha_1^*, \beta_1^*, \ldots, \alpha_\ell^*, \beta_\ell^*)$. By the round-by-round soundness property of the underlying interactive protocol Π, there exists a T'-time computable function State satisfying Definition 4.1. This, together with the equation above implies that for every $\lambda \in \Lambda$,

$$\Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Setup}(1^\lambda)\\ \tau^*\leftarrow P^*(\mathsf{CRS})}}[(\mathsf{State}(x_\lambda^*, \emptyset) = \mathsf{reject}) \ \wedge \ (\mathsf{State}(x_\lambda^*, \tau^*) = \mathsf{accept})] \geq \frac{1}{p'(T'(\lambda))}.$$

In what follows, for every $j \in [\ell(\lambda)]$ we denote $\tau^*_j \triangleq (\alpha^*_1, \beta^*_1, \ldots, \alpha^*_j, \beta^*_j)$. By a standard hybrid argument, for every $\lambda \in \Lambda$, there exists $i = i(\lambda) \in [\ell(\lambda)]$ such that

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda) \\ \tau^* \leftarrow P^*(\mathsf{CRS})}} [(\mathsf{State}(x^*_\lambda, \tau^*_{i-1}) = \mathsf{reject}) \ \wedge \ (\mathsf{State}(x^*_\lambda, \tau^*_i) = \mathsf{accept})] \geq \frac{1}{\ell(\lambda) \cdot p'(T'(\lambda))}.$$

Let $p(T'(\lambda)) = \ell(\lambda) \cdot p'(T'(\lambda))$. Our assumption that $T'(\lambda) \geq \ell(\lambda)$ implies that p is bounded by a polynomial. Fix $i = i(\lambda)$, and let $\mathbb{E}_i(x^*_\lambda, \tau^*)$ denote the event that

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

Thus, for every $\lambda \in \Lambda$,

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda) \\ \tau^* \leftarrow P^*(\mathsf{CRS})}} [\mathbb{E}_i(x^*_\lambda, \tau^*)] \geq \frac{1}{p(T'(\lambda))}. \tag{5}$$

Recall that Setup samples $k_j \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$ for $j \in [\ell]$. We define an alternative (non-uniform) algorithm, $\mathsf{Setup}' = \{\mathsf{Setup}'_\lambda\}_{\lambda \in \mathbb{N}}$ that generates the CRS as follows: For every $\lambda \in \Lambda$, Setup'_λ has $i(\lambda)$ hardwired into it, and it samples $k_j \leftarrow \mathcal{H}.\mathsf{LossyGen}(1^\lambda)$ for $j \in [i-1]$ and $k_j \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$ for $j \in [i, \ell]$. We let $\mathsf{Setup}'_\lambda = \mathsf{Setup}(1^\lambda)$ for all $\lambda \in \mathbb{N} \setminus \Lambda$.

The first thing we show is that we can switch from generating the CRS using Setup to using Setup′, and the probability of event $\mathbb{E}_i(x^*_\lambda, \tau^*)$ occurring is still non-negligible.

Claim 4.6. *For every large enough $\lambda \in \Lambda$,*

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathsf{Setup}'_\lambda \\ \tau^* \leftarrow P^*(\mathsf{CRS})}} [\mathbb{E}_i(x^*_\lambda, \tau^*)] \geq \frac{1}{2p(T'(\lambda))}.$$

Proof. Suppose for contradiction that there is an infinite set $\Lambda_0 \subseteq \Lambda$ such that for every $\lambda \in \Lambda_0$,

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathsf{Setup}'_\lambda \\ \tau^* \leftarrow P^*(\mathsf{CRS})}} [\mathbb{E}_i(x^*_\lambda, \tau^*)] < \frac{1}{2p(T'(\lambda))}.$$

This, together with Equation (5), implies that for every $\lambda \in \Lambda_0$,

$$\Pr_{\substack{\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda) \\ \tau^* \leftarrow P^*(\mathsf{CRS})}} [\mathbb{E}_i(x^*_\lambda, \tau^*)] - \Pr_{\substack{\mathsf{CRS} \leftarrow \mathsf{Setup}'_\lambda \\ \tau^* \leftarrow P^*(\mathsf{CRS})}} [\mathbb{E}_i(x^*_\lambda, \tau^*)] > \frac{1}{2p(T'(\lambda))}. \tag{6}$$

For every $\lambda \in \Lambda_0$ and for $i = i(\lambda)$, consider the following series of alternative setup algorithms $\{\mathsf{Setup}_{\lambda,j}\}_{j=0}^{i}$, where $\mathsf{Setup}_{\lambda,j}$ generates the CRS as follows:

$$\begin{aligned} \mathsf{Setup}_{\lambda,j} : \quad & k_\iota \leftarrow \mathcal{H}.\mathsf{LossyGen}(1^\lambda) \text{ for } \iota \in [j-1] \\ & k_\iota \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \text{ for } \iota \in [j, \ell]. \end{aligned}$$

Note that $\mathsf{Setup}_{\lambda,1} = \mathsf{Setup}(1^\lambda)$ and $\mathsf{Setup}_{\lambda,i} = \mathsf{Setup}'_\lambda$. Thus, Equation (6), together with a standard hybrid argument, implies that for every $\lambda \in \Lambda_0$ there exists $j = j(\lambda)$ such that

$$\left| \Pr_{\substack{\mathsf{CRS} \leftarrow \mathsf{Setup}_{\lambda,j} \\ \tau^* \leftarrow P^*(\mathsf{CRS})}} [\mathbb{E}_i(x^*_\lambda, \tau^*)] - \Pr_{\substack{\mathsf{CRS} \leftarrow \mathsf{Setup}_{\lambda,j+1} \\ \tau^* \leftarrow P^*(\mathsf{CRS})}} [\mathbb{E}_i(x^*_\lambda, \tau^*)] \right| \geq \frac{1}{2\ell(\lambda) \cdot p(T'(\lambda))}.$$

Algorithm $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ is defined as follows: For every $\lambda \in \Lambda$, $\mathcal{A}_\lambda$ has the values $j = j(\lambda)$ and x^*_λ hardwired. $\mathcal{A}_\lambda$ takes as input a key k (generated either by $k \leftarrow \mathcal{H}.\mathsf{LossyGen}(1^\lambda)$ or by $k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$) and does the following:

- Sample $k_\iota \leftarrow \mathcal{H}.\mathsf{LossyGen}(1^\lambda)$ for $\iota \in [j-1]$.
- Sample $k_\iota \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$ for $\iota \in [j+1, \ell]$.
- Set $\mathsf{CRS} = (k_1, \ldots, k_{j-1}, k, k_{j+1}, \ldots, k_\ell)$.
- Compute $\tau^* = P^*(\mathsf{CRS})$.
- Output $b = \mathbb{E}_i(x^*_\lambda, \tau^*)$.

Figure 3: Algorithm $\mathcal{A}$ that breaks the T'-key indistinguishability property of $\mathcal{H}$.

We construct a (non-uniform) adversary $\mathcal{A}$ that uses P^* to break the T'-key indistinguishability property of our lossy CI hash family (Definition 3.6). Algorithm $\mathcal{A}$ is defined in Figure 3.

Note that $\mathcal{A}_\lambda$ runs in time $\mathsf{poly}(T'(\lambda))$, since P^*_λ is of size $\mathsf{poly}(T'(\lambda))$ and $\mathbb{E}_i(x^*_\lambda, \tau^*)$ can be checked in $\mathsf{poly}(T'(\lambda))$ time, since State is computable in time $\mathsf{poly}(T')$ (this follows from the fact that Π is (T', d, ρ)-FS-compatible).

By the definition of $\mathcal{A}$, for every $\lambda \in \Lambda_0$,

$$\left| \Pr_{k\leftarrow\mathsf{Gen}(1^\lambda)}[\mathcal{A}(k) = 1] - \Pr_{k\leftarrow\mathsf{LossyGen}(1^\lambda)}[\mathcal{A}(k) = 1] \right|$$
$$= \left| \Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Setup}_{\lambda,j} \\ \tau^*\leftarrow P^*(\mathsf{CRS})}}[\mathbb{E}_i(x^*_\lambda, \tau^*)] - \Pr_{\substack{\mathsf{CRS}\leftarrow\mathsf{Setup}_{\lambda,j+1} \\ \tau^*\leftarrow P^*(\mathsf{CRS})}}[\mathbb{E}_u(x^*_\lambda, \tau^*)] \right| \geq \frac{1}{2\ell(\lambda)\cdot p(T'(\lambda))}.$$

This, together with our assumption that $T'(\lambda) \geq \ell(\lambda)$, implies that $\mathcal{A}$ breaks the T'-key indistinguishability property of $\mathcal{H}$, thus reaching a contradiction.

□

Next, we show that we can guess the values of $\beta^*_{[i-\nu,i-1]}$ in the transcript such that the probability that these values were guessed correctly and event $\mathbb{E}_i(x^*_\lambda, \tau^*)$ occurs is not too small (non-negligible in $T(\lambda)$).

Claim 4.7. *There exists a polynomial q, and for every $\lambda \in \Lambda$ there exist $k_1, \ldots, k_{i(\lambda)-1} \in \mathcal{H}.\mathsf{LossyGen}(1^\lambda)$ and values $\beta_{[i-\nu,i-1]} = (\beta_{i-\nu}, \ldots, \beta_{i-1})$, where $\beta_j \in \mathcal{H}.\mathsf{LossyGen}(k_j, \cdot)$ for every $j \in [i-\nu, i-1]$, such that for every $\lambda \in \Lambda$ and for $i = i(\lambda)$,*

$$\Pr_{\substack{k_i,\ldots k_\ell\leftarrow\mathcal{H}.\mathsf{Gen}(1^\lambda) \\ \tau^*\leftarrow P^*(k_1,\ldots,k_\ell)}}\left[\mathbb{E}_i(x^*_\lambda, \tau^*) \quad \wedge \quad (\beta^*_{[i-\nu,i-1]} = \beta_{[i-\nu,i-1]})\right] \geq \frac{1}{q(T(\lambda))},$$

where we parse $\tau^* = (\alpha^*_1, \beta^*_1, \ldots, \alpha^*_\ell, \beta^*_\ell)$.

Proof. This claim follows from Claim 4.6 and from the ω-lossiness of $\mathcal{H}$, as follows.

Claim 4.6, together with a standard averaging argument, implies that for every $\lambda \in \Lambda$, there exist $k_1, \ldots, k_{i-1} \in \mathcal{H}.\mathsf{LossyGen}(1^\lambda)$ such that

$$\Pr_{\substack{k_i,\ldots k_\ell \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \\ \tau^* \leftarrow P^*(k_1,\ldots,k_\ell)}} [\mathbb{E}_i(x^*_\lambda, \tau^*)] \geq \frac{1}{2p(T'(\lambda))}.$$

The ω-lossiness of $\mathcal{H}$ implies that for every $j \in [i-1]$, the set $\{\mathcal{H}.\mathsf{LossyGen}(k_j, x)\}_{x \in \{0,1\}^n}$ is of size $\leq 2^{n-\omega}$. This, together with the equation above, implies that there exists $\beta_{[i-\nu,i-1]} = (\beta_{i-\nu}, \ldots, \beta_{i-1})$, where $\beta_j \in \mathcal{H}.\mathsf{LossyGen}(k_j, \cdot)$ for every $j \in [i-\nu, i-1]$, such that

$$\Pr_{\substack{k_i,\ldots k_\ell \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \\ \tau^* \leftarrow P^*(k_1,\ldots,k_\ell)}} \left[\mathbb{E}_i(x^*_\lambda, \tau^*) \ \wedge \ (\beta^*_{[i-\nu,i-1]} = \beta_{[i-\nu,i-1]})\right] \geq \frac{1}{2p(T'(\lambda))} \cdot \frac{1}{2^{\nu(n-\omega)}}.$$

Finally, from our assumption that $T \geq \max\{2^{\nu \cdot (n-\omega)}, T'\}$, there exists a polynomial q such that

$$\frac{1}{2p(T'(\lambda))} \cdot \frac{1}{2^{\nu(n-\omega)}} \geq \frac{1}{q(T(\lambda))}.$$

□

Finally, we use the T-CI property of $\mathcal{H}$ (Definition 3.4) to reach a contradiction.

Claim 4.8. *For every $\lambda \in \Lambda$, fix any values of $k_1, \ldots, k_{i(\lambda)-1} \in \mathcal{H}.\mathsf{LossyGen}(1^\lambda)$ and any $\beta_{[i-\nu,i-1]} = (\beta_{i-\nu}, \ldots, \beta_{i-1})$, where $\beta_j \in \mathcal{H}.\mathsf{LossyGen}(k_j, \cdot)$ for every $j \in [i-\nu, i-1]$. Then there exists a negligible function μ such that for every $\lambda \in \Lambda$ and $i = i(\lambda)$,*

$$\Pr_{\substack{k_i,\ldots k_\ell \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \\ \tau^* \leftarrow P^*(k_1,\ldots,k_\ell)}} \left[\mathbb{E}_i(x^*_\lambda, \tau^*) \ \wedge \ (\beta^*_{[i-\nu,i-1]} = \beta_{[i-\nu,i-1]})\right] \leq \mu(T(\lambda)),$$

where we parse $\tau^* = (\alpha^*_1, \beta^*_1, \ldots, \alpha^*_\ell, \beta^*_\ell)$.

Proof. This claim follows from the T-CI property of $\mathcal{H}$, as follows. For every $\lambda \in \Lambda$, fix any $(k_1, \ldots, k_{i(\lambda)-1})$ and $\beta_{[i-\nu,i-1]}$ as in the claim statement. Suppose for the sake of contradiction that there exists a polynomial q' and an infinite set $\Lambda_0 \subseteq \Lambda$ such that for every $\lambda \in \Lambda_0$ and for $i = i(\lambda)$,

$$\Pr_{\substack{k_i,\ldots k_\ell \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \\ \tau^* \leftarrow P^*(k_1,\ldots,k_\ell)}} \left[\mathbb{E}_i(x^*_\lambda, \tau^*) \ \wedge \ (\beta^*_{[i-\nu,i-1]} = \beta_{[i-\nu,i-1]})\right] \geq \frac{1}{q'(T(\lambda))}. \tag{7}$$

Recall that the event $\mathbb{E}_i(x^*_\lambda, \tau^*)$ occurs when

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

This happens only if $\beta^*_i \in \mathcal{B}_{x^*_\lambda, \tau^*_{i-1} | \alpha^*_i}$, where

$$\mathcal{B}_{x^*_\lambda, \tau^*_{i-1} | \alpha^*_i} = \{\beta'_i \ : \ \mathsf{State}(x^*_\lambda, \tau^*_{i-1} | \alpha^*_i | \beta'_i) = \mathsf{accept}\}.$$

Therefore, we can rewrite equation (7) as

$$\Pr_{\substack{k_i,\ldots k_\ell \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)\\ \tau^* \leftarrow P^*(k_1,\ldots,k_\ell)}} \left[\begin{array}{ll} & (\mathsf{State}(x^*_\lambda, \tau^*_{i-1}) = \mathsf{reject}) \\ \wedge & (\beta^*_i \in \mathcal{B}_{x^*_\lambda, \tau^*_{i-1}|\alpha^*_i}) \\ \wedge & (\beta^*_{[i-\nu,i-1]} = \beta_{[i-\nu,i-1]}) \end{array}\right] \geq \frac{1}{q'(T(\lambda))}. \tag{8}$$

In what follows, we construct a $\mathsf{poly}(T)$-size adversary $\mathcal{A}$ and a function $f = \{f_\lambda\}$ computable by a circuit of size $\rho(\lambda)$, such that for every $\lambda \in \Lambda_0$,

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

contradicting the T-CI property of $\mathcal{H}$. To this end, we use the ν-history-independent properties of BAD and State (Definition 4.4). Recall that $\mathsf{BAD}_{x^*_\lambda, \beta^*_{[i-\nu,i-1]}}$ is a function that has hardwired a non-uniform advice that is a deterministic (possibly inefficient) function of $(x^*_\lambda, \beta^*_{[i-\nu,i-1]})$. It takes as input $(\alpha_{i-\nu}, \ldots, \alpha_i)$ and randomness $r \leftarrow \{0,1\}^*$, and outputs an element β such that if $\mathsf{State}(x^*_\lambda, \tau_{i-1}) = \mathsf{reject}$ for

$$\tau_{i-1} = (\alpha_{i-\nu}, \beta^*_{i-\nu}, \ldots, \alpha_{i-1}, \beta^*_{i-1}),$$

then with probability $1 - \mathsf{negl}(\lambda)$, β is a uniformly random element in the set

$$\mathcal{B}_{x^*_\lambda, \tau_{i-1}|\alpha_i} = \{\beta_i : \mathsf{State}(x^*_\lambda, \tau_{i-1}|\alpha_i|\beta_i) = \mathsf{accept}\}.$$

We define for every randomness $r \in \{0,1\}^*$ the (deterministic) function f_r:

$$f_r(\alpha_{i-\nu}, \ldots, \alpha_i) = \mathsf{BAD}_{x^*_\lambda, \beta^*_{[i-\nu,i-1]}}(\alpha_{i-\nu}, \ldots, \alpha_i; r). \tag{9}$$

By Definition 4.2, for every $\alpha_{i-\nu}, \ldots, \alpha_i$ and every $\beta \in \mathcal{B}_{x^*_\lambda, \tau_{i-1}|\alpha_i}$, and denoting

$$\tau_{i-1} = (\alpha_{i-\nu}, \beta^*_{i-\nu}, \ldots, \alpha_{i-1}, \beta^*_{i-1}),$$

then

$$\Pr_{r \leftarrow \{0,1\}^*} [f_r(\alpha_{i-\nu}, \ldots, \alpha_i) = \beta \mid \mathsf{State}(x^*_\lambda, \tau_{i-1}|\alpha_i|\beta) = \mathsf{accept}] = \frac{1 - \mathsf{negl}(\lambda)}{d(\lambda)}.$$

This, combined with Equation (8), implies that

$$\Pr_{\substack{k_i,\ldots k_\ell \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)\\ \tau^* \leftarrow P^*(k_1,\ldots,k_\ell)\\ r \leftarrow \{0,1\}^*}} \left[\begin{array}{ll} & (\mathsf{State}(x^*_\lambda, \tau^*_{i-1}) = \mathsf{reject}) \\ \wedge & (\beta^*_i = f_r(\alpha^*_{i-\nu}, \ldots, \alpha^*_i)) \\ \wedge & (\beta^*_{[i-\nu,i-1]} = \beta_{[i-\nu,i-1]}) \end{array}\right] \geq \frac{1 - \mathsf{negl}(\lambda)}{d(\lambda)} \cdot \frac{1}{q'(T(\lambda))}.$$

Recall that it is assumed that $d(\lambda) \leq T(\lambda)$, which implies that there exists a polynomial q such that the above probability is $\geq \frac{1}{q(T(\lambda))}$.

This implies that there exists r^* and thus f_{r^*}, which runs in time $\rho(\lambda)$, such that

$$\Pr_{\substack{k_i,\ldots k_\ell \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)\\ \tau^* \leftarrow P^*(k_1,\ldots,k_\ell)}} \left[\begin{array}{ll} & (\mathsf{State}(x^*_\lambda, \tau^*_{i-1}) = \mathsf{reject}) \\ \wedge & (\beta^*_i = f_{r^*}(\alpha^*_{i-\nu}, \ldots, \alpha^*_i)) \\ \wedge & (\beta^*_{[i-\nu,i-1]} = \beta_{[i-\nu,i-1]}) \end{array}\right] \geq \frac{1}{q(T(\lambda))}. \tag{10}$$

Adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ for breaking the T-CI property of $\mathcal{H}$.

The adversary $\mathcal{A}_\lambda$ has all the values $k_1, \ldots, k_{i-1}, \beta_{[i-\nu,i-1]}$ (which were fixed above) hardwired. Upon receiving an input k, $\mathcal{A}_\lambda$ does the following:

1. For every $j \in [i+1, \ell]$ choose at random $k_j \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$.
2. Set $\mathsf{CRS} = (k_1, \ldots, k_{i-1}, k, k_{i+1}, \ldots, k_\ell)$.
3. Compute $\tau^* = P^*(\mathsf{CRS})$.
4. Parse $\tau^* = (\alpha_1^*, \beta_1^*, \ldots, \alpha_\ell^*, \beta_\ell^*)$.
5. If $\beta^*_{[i-\nu,i-1]} \neq \beta_{[i-\nu,i-1]}$, then abort.
6. Else output $(\alpha^*_{i-\nu}, \beta^*_{i-\nu}, \ldots, \alpha^*_{i-1}, \beta^*_{i-1}, \alpha^*_i)$.

Figure 4: Breaking the T-CI property of $\mathcal{H}$

We next use P^* to construct an adversary $\mathcal{A}$ that breaks the T-CI property of $\mathcal{H}$ w.r.t. the function f_{r^*}. To ensure that the domains of $f_{r^*}(\cdot)$ and $\mathcal{H}.\mathsf{Hash}(k_i, \cdot)$ are equal, we think of f_{r^*} as taking as input $(\alpha_{i-\nu}, \beta_{i-\nu}, \ldots, \alpha_{i-1}, \beta_{i-1}, \alpha_i)$, ignoring $(\beta_{i-\nu}, \ldots, \beta_{i-1})$, and outputting $f_{r^*}(\alpha_{i-\nu}, \ldots, \alpha_i)$. The adversary $\mathcal{A}$ is defined in Figure 4.

By Equation (10), with probability $1/\mathsf{poly}(T'(\lambda))$ the adversary $\mathcal{A}_\lambda(k)$ outputs

$$(\alpha^*_{i-\nu}, \beta^*_{i-\nu}, \ldots, \alpha^*_{i-1}, \beta^*_{i-1}, \alpha^*_i)$$

such that

$$\beta^*_i = \mathcal{H}.\mathsf{Hash}(k, \alpha^*_{i-\nu}, \beta^*_{i-\nu}, \ldots, \alpha^*_{i-1}, \beta^*_{i-1}, \alpha^*_i) = f_{r^*}(\alpha^*_{i-\nu}, \beta^*_{i-\nu}, \ldots, \alpha^*_{i-1}, \beta^*_{i-1}, \alpha^*_i),$$

contradicting the T-CI property of $\mathcal{H}$. □

Finally, we obtain our desired contradiction, by noting that Claims 4.7 and 4.8 are contradictory.

Efficiency. We note that the prover and verifier runtime in the protocol $\Pi^{\mathcal{H}}_{\mathsf{FS}}$ is at least as large as their respective runtimes, $T_P(\lambda)$ and $T_V(\lambda)$, in the underlying protocol Π. The main difference is that in $\Pi^{\mathcal{H}}_{\mathsf{FS}}$ both the prover and the verifier are required to compute the hash function $\mathcal{H}.\mathsf{Hash}$ for each round of the protocol. Recall that the evaluation time of the hash function is polynomially related to the size of the functions $f_r \in \mathcal{F}_\lambda$, which is identical to the size of the function BAD (see Equation (9)), which is $\rho(\lambda)$. Therefore, there exists a polynomial poly specified by the hash function $\mathcal{H}$ such that the time needed to run the $\mathcal{H}.\mathsf{Hash}$ algorithm is $\mathsf{poly}(\rho(\lambda))$. We conclude that the overall verifier runtime is $\ell \cdot \mathsf{poly}(\rho(\lambda)) + T_V(\lambda)$ and the prover runtime is $\ell \cdot \mathsf{poly}(\rho(\lambda)) + T_P(\lambda)$.

□

5 The Sum-Check Protocol is FS-Compatible

In this section we prove that the sum-check protocol is FS-compatible (Definition 4.2). We start by recalling the sum-check protocol [LFKN92, Sha92].

The Sum-Check Protocol. In the sum-check protocol, a (not necessarily efficient) prover takes as input an ℓ-variate polynomial $g : \mathbb{F}^\ell \to \mathbb{F}$ of degree $\leq d$ in each variable over a finite field $\mathbb{F}$,[21] and a subset $B \subset \mathbb{F}$. Its goal is to convince a verifier that

$$\sum_{(b_1,\ldots,b_\ell)\in B^\ell} g(b_1,\ldots,b_\ell) = v$$

for some element $v \in \mathbb{F}$. The verifier only has oracle access to g, and is given the constant $v \in \mathbb{F}$. It is required to be efficient in both its running time and its number of oracle queries. In Figure 5, we review the sum-check protocol from [LFKN92, Sha92]. We denote this protocol by $\left(P_{\mathsf{SC}}(g), V_{\mathsf{SC}}^g(v)\right)$. We always assume that the field operations can be performed in time $\mathsf{polylog}(|\mathbb{F}|)$.

Sum-Check Protocol for $\sum_{b_1,\ldots,b_\ell\in B} g(b_1,\ldots,b_\ell) = v$

1. Set $i = 1$.
2. P_{SC} computes the univariate polynomial $g_i : \mathbb{F} \to \mathbb{F}$ defined by

   $$g_i(\cdot) \triangleq \sum_{b_{i+1},\ldots,b_\ell\in B} g(t_1,\ldots,t_{i-1},\cdot,b_{i+1},\ldots,b_\ell),$$

   and sends g_i to V_{SC}.
3. V_{SC} checks that g_i is a univariate polynomial of degree at most d, and that

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

   If not, V_{SC} rejects.
4. If $i < \ell$, V_{SC} chooses a random element $t_i \in_R \mathbb{F}$ and sends t_i to P_{SC}. Set $i = i + 1$ and go back to step 2.
5. If $i = \ell$, V_{SC} chooses a random element $t_\ell \in_R \mathbb{F}$ and checks that

   $$g_\ell(t_\ell) = g(t_1,\ldots,t_\ell)$$

   by querying the oracle at the point $(t_1,\ldots,t_\ell)$.

Figure 5: Sum-check protocol $(P_{\mathsf{SC}}(g), V_{\mathsf{SC}}^g(v))$ [LFKN92, Sha92]

Lemma 5.1. *[LFKN92, Sha92] Let $g : \mathbb{F}^\ell \to \mathbb{F}$ be an ℓ-variate polynomial of degree d in each variable. The sum-check protocol $(P_{\mathsf{SC}}, V_{\mathsf{SC}})$ described in Figure 5 satisfies the following properties.*

- **Completeness.** *If $\sum_{(b_1,\ldots,b_\ell)\in B^\ell} g(b_1,\ldots,b_\ell) = v$ then*

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

[21] Think of d as significantly smaller than $|\mathbb{F}|$.

- **Soundness.** *If* $\sum_{(b_1,\ldots,b_\ell)\in B^\ell} g(b_1,\ldots,b_\ell) \neq v$ *then for every (unbounded) interactive prover* P^**,*

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

- **Efficiency.** V_{SC} *has oracle access to* $g : \mathbb{F}^\ell \to \mathbb{F}$*. The prover* $P_{\mathsf{SC}}(g)$ *runs in time* $\leq \mathsf{poly}(|B|^\ell, T_g)$*, where* T_g *is the time it takes to evaluate* g*.*[22] *The verifier* $V^g_{\mathsf{SC}}(v)$ *runs in time* $\leq \mathsf{poly}(|B|, \log|\mathbb{F}|, \ell, d)$*, and queries the oracle* g *at a single point. The communication complexity is* $\leq \mathsf{poly}(|B|, \log|\mathbb{F}|, \ell, d)$*, and the total number of bits sent from the verifier to the prover is* $O(\ell \cdot \log|\mathbb{F}|)$*. Moreover, this protocol is public-coin; i.e., all the messages sent by the verifier are truly random and consist of the verifier's random coin tosses.*

The Sum-Check Protocol is FS-compatible.

Theorem 5.2. *Fix any* $\epsilon > 0$*, and let* $T'(\lambda) = 2^{\lambda^\epsilon}$*. Then there exists a polynomial* ρ *such that the interactive sum-check protocol (Figure 5) is* (T', d, ρ)*-FS-compatible according to Definition 4.2, assuming the instances are multi-variate polynomials of individual degree* d *over a field* $\mathbb{F}$*, and assuming*

$$\log|\mathbb{F}| \geq \max\{d, (\ell \cdot \log(|B|\cdot d))^{2/\epsilon}\}, \tag{11}$$

where ℓ *is the number of variables in the instance polynomial, and* B *is the set we sum over.*

Proof. We define the functions State and BAD in the definition of FS compatibility.

To this end fix an input $x = (g, v)$, where $g : \mathbb{F}^\ell \to \mathbb{F}$ is an ℓ-variate polynomial of individual degree d, and where $v \in \mathbb{F}$. Recall that each verifier message is a random field element $t_i \leftarrow \mathbb{F}$ and each prover message is a univariate polynomial $g_i : \mathbb{F} \to \mathbb{F}$ of degree d.

The function State**.** For every $x = (g, v)$ and every $\tau = (g_1^*, t_1, \ldots, g_i^*, t_i)$, where $t_1, \ldots, t_i \in \mathbb{F}$, $\mathsf{State}(x, \tau)$ ignores $(g_1^*, \ldots, g_{i-1}^*)$ and does the following:

1. If $\tau = \emptyset$, perform the following check: If $v = \sum_{b_1,\ldots,b_\ell \in B} g(b_1, \ldots, b_\ell)$ then output accept, and otherwise output reject.
2. If g_i^* is not a univariate degree at most d polynomial over $\mathbb{F}$, output reject.
3. Compute
$$g_i(\cdot) = \sum_{b_{i+1},\ldots,b_\ell \in B} g(t_1, \ldots, t_{i-1}, \cdot, b_{i+1}, \ldots, b_\ell). \tag{12}$$
4. If $g_i(t_i) = g_i^*(t_i)$, output accept, and otherwise output reject.

State is computable in time $\leq |B|^\ell \cdot T_g \cdot \mathsf{poly}(\lambda)$, where T_g is the time it takes to evaluate g and $\lambda \triangleq \log|\mathbb{F}|$. Note that $T_g \leq d^\ell \cdot \mathsf{polylog}(|\mathbb{F}|) = d^\ell \cdot \mathsf{poly}(\lambda)$, which implies that State is computable in time

$$T'(\lambda) = (|B|\cdot d)^\ell \cdot \mathsf{poly}(\lambda) \leq 2^{\lambda^\epsilon},$$

as desired, where the latter inequality follows from the assumption that

$$\lambda \triangleq \log|\mathbb{F}| \geq (\ell \cdot \log(|B|\cdot d))^{2/\epsilon}.$$

(T', d)-Round-by-round soundness is thus established by observing that State trivially satisfies the other three conditions (Items (1)-(3) in the definition of round-by-round soundness).

[22] We assume that T_g is larger than d and $\log|\mathbb{F}|$.

The function BAD. Given $(x, t_1, \ldots, t_{i-1})$, where $x = (g, v)$, compute the advice g_i as in Equation (12). The function BAD_{g_i}, on input $(g_1^*, \ldots, g_i^*)$, ignores $(g_1^*, \ldots, g_{i-1}^*)$, and does the following:

1. If g_i^* is not a univariate degree at most d polynomial over $\mathbb{F}$ then output $\perp$.
2. Otherwise, output a random root of the polynomial $g_i - g_i^*$, using the Cantor-Zassenhaus algorithm [CZ81].

Note that BAD_{g_i} can be computed in time $\mathsf{poly}(d, \log|\mathbb{F}|) = \mathsf{poly}(\lambda)$, which follows from the fact that $\lambda \triangleq \log|\mathbb{F}| \geq d$.

It remains to argue that for every $\tau \triangleq (g_1^*, t_1, \ldots, g_{i-1}^*, t_{i-1})$ such that $\mathsf{State}(x, \tau) = \mathsf{reject}$, and every g_i^*, with probability $1 - \mathsf{negl}(\lambda)$ (over the randomness of BAD_{g_i}), it holds that $\mathsf{BAD}_{g_i}(g_1^*, \ldots, g_i^*) = \mathsf{BAD}_{g_i}(g_i^*)$ outputs a uniformly random element in the set

$$\mathcal{B} = \{t_i \ : \ \mathsf{State}(x, \tau | g_i^* | t_i) = \mathsf{accept}\}.$$

To this end, note that by definition of State,

$$\mathcal{B} = \{t \ : \ g_i(t) = g_i^*(t)\}$$

where g_i is defined in Equation (12) above. Indeed by definition, $\mathsf{BAD}_{g_i}(g_i^*)$ outputs a random element in $\mathcal{B}$ with overwhelming probability,[23] as desired.

□

Corollary 5.3. *Assuming the sub-exponential hardness of* LWE*, there exists a hash family $\mathcal{H}$ such that the non-interactive argument obtained by applying the Fiat-Shamir transform to the sum-check protocol, with the hash family $\mathcal{H}$, is sound assuming the field size in the sum-check protocol instance satisfies Equation* (11) *(for a small enough constant $\epsilon > 0$). The resulting non-interactive argument has the following efficiency guarantees: the prover runs in time* $\mathsf{poly}(|B|^\ell, T_g)$,[24] *where T_g is the time it takes to compute the ℓ-variate polynomial g that we are summing over. The verifier runs in time* $\mathsf{poly}(|B|, \log|\mathbb{F}|, \ell, d)$*, and the communication complexity is* $\mathsf{poly}(|B|, \log|\mathbb{F}|, \ell, d)$.

Proof. Assume the sub-exponential hardness of LWE. By Corollary 3.8, there exists a constant $0 < \epsilon_1 < 1$ and for every constant $0 < \delta < 1$ there exists a constant $0 < \epsilon_2 < 1$ such that the following holds: For any polynomial S and any hash family $\mathcal{F}$ consisting of functions computable by circuits of size S, there exists a (T, T', ω)-lossy CI hash family for $\mathcal{F}$, where $T = 2^{\lambda^{\epsilon_1}}$, $T' = 2^{\lambda^{\epsilon_2}}$, and $\omega = n - \lambda^\delta$.

Fix $\delta = \epsilon_1/2$, which fixes ϵ_2 above. Assume w.l.o.g. that $\epsilon_2 \leq \epsilon_1$ (otherwise set $\epsilon_2 = \epsilon_1$). By Theorem 5.2, applied with $\epsilon = \epsilon_2$, there exists a polynomial ρ such that the sum-check protocol is (T', d, ρ)-FS-compatible (assuming the field size satisfies Equation (11) w.r.t $\epsilon = \epsilon_2$). By Theorem 4.5, the non-interactive argument obtained by applying the Fiat-Shamir transform to the sum-check protocol, with a (T, T', ω)-lossy CI hash family, is sound if $T'(\lambda) \geq \ell(\lambda)$ and $T \geq \max(d, 2^{\nu(n-\omega)}, T')$. The former follows trivially from the fact that $\ell \leq \lambda^{\epsilon_2}$, which in turn follows from Equation (11)) together with the fact that $\lambda = \log|\mathbb{F}|$. The latter follows from the following calculations: The fact that $T \geq d$ follows from Equation (11); the fact that $T \geq T'$ follows from our assumption that

[23] The negligible probability of error follows from the Cantor-Zassenhaus algorithm, which is a randomized algorithm that has a negligible probability of error.

[24] We assume that T_g is larger than d and $\log|\mathbb{F}|$.

$\epsilon_1 \geq \epsilon_2$; the fact that $T \geq 2^{\nu(n-\omega)}$ follows from the fact that $\lambda^{\epsilon_1} \geq \nu(n-\omega)$, which follows from the fact that $n - \omega = \lambda^{\delta} = \lambda^{\epsilon_1/2}$ together with the fact that $\nu = \ell \leq \lambda^{\epsilon_2/2} \leq \lambda^{\epsilon_1/2}$ (where the former inequality follows from Equation (11)).

The efficiency conditions follows from combining the efficiency guarantees of the sum-check protocol given in Lemma 5.1 with the efficiency guarantees given in Theorem 4.5. □

6 The GKR Protocol is FS-Compatible

In this section we prove that the GKR protocol is ν-history-independent FS-compatible (Definition 4.4) for a polynomial $\nu(\lambda) = \mathsf{poly}(\lambda)$. We start by recalling the GKR protocol in Section 6.1. We then prove that this protocol is ν-history-independent FS-compatible in Section 6.2.

6.1 The GKR Protocol

The GKR protocol is a publicly verifiable interactive proof for proving the correctness of log-space uniform bounded depth computations. The main ingredient used in the GKR protocol is the sum-check protocol (defined in Figure 5). It relies on the *low-degree extension* encoding, defined below.

Low Degree Extension. Let $\mathbb{F}$ be a field and let $B \subseteq \mathbb{F}$ be a set. As in Section 5, we always assume that field operations can be performed in time $\mathsf{polylog}(|\mathbb{F}|)$. Fix an integer $m \in \mathbb{N}$.

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

$$\widehat{W}(t_1, \ldots, t_m) = \sum_{b_1, \ldots, b_m \in B} \widehat{\mathrm{EQ}}(t_1, \ldots, t_m; b_1, \ldots, b_m) \cdot W(b_1, \ldots, b_m), \tag{13}$$

where $\widehat{\mathrm{EQ}} : \mathbb{F}^m \times \mathbb{F}^m \to \mathbb{F}$ is the low degree extension of the function $\mathrm{EQ} : B^m \times B^m \to \{0,1\}$ that, on input $(t_1, \ldots, t_m; b_1, \ldots, b_m)$, compares $t = (t_1, \ldots, t_m)$ with $b = (b_1, \ldots, b_m)$ and outputs 1 if and only if $t = b$. We can explicitly write $\widehat{\mathrm{EQ}}$ as follows:

$$\widehat{\mathrm{EQ}}(t_1, \ldots, t_m; b_1, \ldots, b_m) = \prod_{i=1}^{m} \sum_{\alpha \in B} \prod_{\beta \in B \setminus \{\alpha\}} \frac{(\beta - t_i)(\beta - b_i)}{(\beta - \alpha)^2}.$$

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

Proposition 6.1. *Given a field $\mathbb{F}$, a subset $B \subseteq \mathbb{F}$, and an integer m, the $2m$-variate polynomial $\widehat{\mathrm{EQ}} : \mathbb{F}^m \times \mathbb{F}^m \to \mathbb{F}$ can be constructed in time $\mathsf{poly}(m, |B|, \log|\mathbb{F}|)$. Furthermore, $\widehat{\mathrm{EQ}}$ can be evaluated in time $\mathsf{poly}(m, |B|, \log|\mathbb{F}|)$.*

Evaluating the low degree extension of a function can be done via Equation (13) by summing over $|B|^m$ values.

Proposition 6.2. *Given a field $\mathbb{F}$, a subset $B \subseteq \mathbb{F}$, an integer m, and a function $W : B^m \to \mathbb{F}$ (given as a truth-table), for any $t \in \mathbb{F}^m$, the value of $\widehat{W}(t)$ can be computed in time $|B|^m \cdot \mathsf{poly}(m, |B|, \log|\mathbb{F}|)$.*

6.1.1 The GKR Protocol.

The GKR protocol is a publicly verifiable interactive proof for proving the correctness of log-space uniform bounded depth computations. Fix any log-space uniform circuit $C : \{0,1\}^N \to \{0,1\}$ of size S and depth D. Let $B \subseteq \mathbb{F}$ and $m \in \mathbb{N}$, where $\mathbb{F}$ is an extension field of $\mathsf{GF}[2]$ for which addition and multiplication in $\mathbb{F}$ can be done in time $\mathsf{polylog}(|\mathbb{F}|)$, and where $S \leq |B|^m \leq \mathsf{poly}(S)$. A typical choice is $|B| = \mathsf{polylog}(S)$ and $m = \frac{\log S}{\log \log(S)}$, or $|B| = 2$ and $m = \log S$.

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

1. C is an arithmetic circuit over $\mathbb{F}$.
2. C is a layered circuit of fan-in 2. Namely, the gates in C can be (uniquely) partitioned into layers such that the inputs to a gate in layer i are the outputs of gates in layer $i-1$. Layer 0 is the input layer, and layer D is the output layer.
3. Each layer has exactly S gates (we pad all the layers which consist of less than S gates, including the input and output layers).

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

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

where, for $\omega_1, \omega_2, \omega_3 \in B^m$, $\mathsf{add}_i(\omega_1, \omega_2, \omega_3) = 1$ if and only if ω_1 is an addition gate in layer i applied to gates ω_2 and ω_3, which are in layer $i-1$. Similarly, $\mathsf{mult}_i(\omega_1, \omega_2, \omega_3) = 1$ if and only if ω_1 is a multiplication gate in layer i applied to gates ω_2 and ω_3 in layer $i-1$.

It will be convenient to work with circuits for which there are (not necessarily lowest degree) extensions $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{mult}}_i : \mathbb{F}^{3m} \to \mathbb{F}$ that agree with add_i and mult_i on B^{3m}, and which can be evaluated quickly (in time $\mathsf{poly}(\log S, \log|\mathbb{F}|)$). It is not known if such extensions of add_i and mult_i exist for any given log-space uniform circuit C. But it was shown by Goldreich [Gol18] that any log-space uniform circuit C can be converted into a slightly larger circuit C' computing the same function, for which such extensions do exist.

Proposition 6.3. *[Gol18] For any log-space uniform circuit $C : \{0,1\}^N \to \{0,1\}$ of size S and depth D, there is a circuit C' of size $S' = \mathsf{poly}(S)$ and depth $D' = D \cdot \mathsf{polylog}(S)$, such that the following holds:*

1. *For every $x \in \{0,1\}^N$, $C(x) = C'(x)$.*
2. *For any B' and m' such that $S' \leq {B'}^{m'} \leq \mathsf{poly}(S')$, denote by*

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

the functions corresponding to C', B', and m', as defined above. Then there exist degree $\leq \mathsf{poly}(m', |B'|)$ polynomials

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

such that $\widetilde{\mathsf{add}'}_i|_{B'^{3m'}} \equiv \mathsf{add}'_i$ and $\widetilde{\mathsf{mult}'}_i|_{B'^{3m'}} \equiv \mathsf{mult}'_i$, and both these polynomials can be evaluated by arithmetic circuits that are constructible and evaluable in time $\mathsf{poly}(m', |B'|, \log|\mathbb{F}|)$.

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

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

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

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

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

be the low-degree extension of V_i.

Overview of the GKR protocol. Suppose the prover wishes to prove that $C(w) = 0$, or equivalently, that $\widehat{V}_D(\omega_{\text{out}}) = 0$. This is done in D phases (where D is the depth of C). In the GKR protocol (using the simplification from [Gol18]), in the i'th phase ($1 \le i \le D$) the prover reduces the task of proving that $\widehat{V}_i(z) = v$ to the task of proving two equations: $\widehat{V}_{i-1}(z_1) = v_1$ and $\widehat{V}_{i-1}(z_2) = v_2$, where z_1, z_2 are random elements in $\mathbb{F}^m$ determined by the random coin tosses of the verifier. This is done by running the sum-check protocol.

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

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

Thus, by Equation (13), for every $z \in \mathbb{F}^m$,

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

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

$$f_{i,z}(p,\omega_1,\omega_2) \triangleq \widehat{\mathrm{EQ}}(z;p) \cdot \begin{pmatrix} \widetilde{\mathsf{add}}_i(p,\omega_1,\omega_2) \cdot \left(\widehat{V}_{i-1}(\omega_1) + \widehat{V}_{i-1}(\omega_2)\right) \\ + \widetilde{\mathsf{mult}}_i(p,\omega_1,\omega_2) \cdot \left(\widehat{V}_{i-1}(\omega_1) \cdot \widehat{V}_{i-1}(\omega_2)\right) \end{pmatrix}. \tag{14}$$

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

$$\widehat{V}_i(z) = \sum_{p,\omega_1,\omega_2 \in B^m} f_{i,z}(p,\omega_1,\omega_2).$$

Note that $f_{i,z}$ is a $3m$-variate polynomial of size $\le \mathsf{poly}(S)$ and degree $\le \mathsf{poly}(m, |B|)$. In the GKR protocol, the prover and verifier run the sum-check protocol, and thus reduce the task of verifying the statement $\widehat{V}_i(z) = v$ to verifying a statement of the form $f_{i,z}(p,\omega_1,\omega_2) = v'$ for $p, \omega_1, \omega_2 \in \mathbb{F}^m$ which are determined by the random coin tosses of the verifier.

Note that the verifier cannot compute the function $f_{i,z}$ on its own. However, by Proposition 6.1 and Proposition 6.3, he can compute efficiently the functions $\widehat{\mathrm{EQ}}$, $\widetilde{\mathsf{add}}_i$, and $\widetilde{\mathsf{mult}}_i$. Thus, to compute

the value of $f_{i,z}(p, \omega_1, \omega_2)$, he only needs help in computing the values of $\widehat{V}_{i-1}(\omega_1)$ and $\widehat{V}_{i-1}(\omega_2)$. Indeed, as mentioned above, the sum-check protocol reduces the task of verifying one claim of the form $\widehat{V}_i(z) = v$ to verifying *two* claims of the form $\widehat{V}_{i-1}(w_1) = v_1$ and $\widehat{V}_{i-1}(w_2) = v_2$.

To avoid an exponential blowup in the number of equations that need to be checked, the task of verifying that both $\widehat{V}_{i-1}(w_1) = v_1$ and $\widehat{V}_{i-1}(w_2) = v_2$ is reduced to the task of verifying a single statement of the form $\widehat{V}_{i-1}(z_{i-1}) = v_{i-1}$ by running a *two-to-one* protocol.

However, it was shown in [KPY18] that this two-to-one protocol is unnecessary. Rather, the exponential blowup can be avoided by running two sum-check protocols in *parallel*, where the verifier uses the *same randomness* in both sum-check protocols. More specifically, they show that the task of proving two claims of the form $\widehat{V}_i(z_i^{(1)}) = v_i^{(1)}$ and $\widehat{V}_i(z_i^{(2)}) = v_i^{(2)}$ can be reduced to the task of proving *two* claims of the form $\widehat{V}_{i-1}(z_{i-1}^{(1)}) = v_{i-1}^{(1)}$ and $\widehat{V}_{i-1}(z_{i-1}^{(2)}) = v_{i-1}^{(2)}$, where $z_{i-1}^{(1)}$ and $z_{i-1}^{(2)}$ are determined by the verifier's random coin tosses. Finally, the verifier checks on its own the value of $\widehat{V}_0$ on two points $z_0^{(1)}$ and $z_0^{(2)}$ that are determined by the verifier's randomness in the sum-check protocols corresponding to the final layer, which can be done efficiently since it amounts to computing two points on the low-degree extension of x.

We provide a formal description of this (slightly modified) GKR protocol in Figure 6.

Theorem 6.4. *[KPY18] The* GKR *protocol described in Figure 6 has the following properties for any circuit* $C : \{0,1\}^N \to \{0,1\}$ *of size* S *and depth* D*, for which* $\widetilde{\mathsf{add}}_i$ *and* $\widetilde{\mathsf{mult}}_i$ *are both degree* $\mathsf{poly}(m, |B|)$ *polynomials that are evaluable in* $\mathsf{poly}(m, |B|, \log|\mathbb{F}|)$ *time.*

- **Completeness.** *For every* $w \in \{0,1\}^N$ *such that* $C(w) = 0$*,*
$$\Pr[(P_{\mathsf{GKR}}, V_{\mathsf{GKR}})(C, w) = 1] = 1.$$
- **Soundness.** *For every* $w \in \{0,1\}^N$ *such that* $C(w) \neq 0$*, and every* P^**,*
$$\Pr[(P^*, V_{\mathsf{GKR}})(C, w) = 1] \leq \frac{\mathsf{poly}(m \cdot |B|) \cdot D}{|\mathbb{F}|}.$$
- **Efficiency.** *The prover runs in time* $\mathsf{poly}(S)$*, the verifier runs in time* $(D + n) \cdot \mathsf{polylog}(S)$*, and the communication complexity is* $D \cdot \mathsf{polylog}(S)$*, assuming* $\log|\mathbb{F}| = \mathsf{polylog}(S)$.

Recall that Proposition 6.3 says that any log-space uniform circuit C can be converted into the format necessarily for the above theorem statement (say, a to a circuit C'), with a slight sacrifice in circuit size and depth ($S' = \mathsf{poly}(S)$ and $D' = D \cdot \mathsf{polylog}(S)$). Thus, the GKR protocol gives a delegation scheme for *log-space uniform* circuits as well, by running the GKR protocol on C'.

6.2 The GKR Protocol is FS-Compatible

Theorem 6.5. *Fix any* $\epsilon > 0$*, and let* $T'(\lambda) = 2^{\lambda^\epsilon}$*. Then there exists a polynomial* ρ *such that the interactive* GKR *protocol (Figure 6) is* ν*-history-independent* (T', d, ρ)*-*FS*-compatible according to Definition 4.2, for* $d \leq \mathsf{poly}(|B|, m)$ *and* $\nu = 6m$*, and assuming the extension field* $\mathbb{F}$ *satisfies that*
$$\log|\mathbb{F}| \geq \max\{d, (\log S)^{2/\epsilon}\}, \tag{15}$$
where S *is the size of the circuit being delegated. We assume that this circuit satisfies the property that it has degree* $\leq \mathsf{poly}(m, |B|)$ *polynomials* $\widetilde{\mathsf{add}}_i$ *and* $\widetilde{\mathsf{mult}}_i$ *that can be evaluated in* $\mathsf{poly}(m, |B|, \log|\mathbb{F}|)$ *time.*[25]

[25] Recall that by Proposition 6.3 one can convert any log-space uniform circuit to one which has this property with minimal blowup in parameters, and thus this assumption is practically without loss of generality.

The GKR Protocol $(P_{\mathsf{GKR}}, V_{\mathsf{GKR}})(C, w)$

Let $\mathbb{F}$ be an extension field of $\mathsf{GF}[2]$, let $B \subseteq \mathbb{F}$ and $m \in \mathbb{N}$ such that $S \leq |B|^m = \mathsf{poly}(S)$.
In this protocol, P_{GKR} proves to V_{GKR} that $\widehat{V}_D(\omega_{\text{out}}) = 0$, where $\omega_{\text{out}} \in B^m$ corresponds to the output wire. This is done in D phases. For every $i \in [D]$, in the i'th phase, the task of proving that $\widehat{V}_i(z_i^{(b)}) = v_i^{(b)}$ for $b \in [2]$ is reduced to the task of proving that $\widehat{V}_{i-1}(z_{i-1}^{(b)}) = v_{i-1}^{(b)}$ for $b \in [2]$. This is done as follows:

1. Set $z_D^{(1)} = z_D^{(2)} = \omega_{\text{out}}$ and $v_D^{(1)} = v_D^{(2)} = 0$, and set $i = D$.

2. P_{GKR} and V_{GKR} run 2 sum-check protocols in parallel: $(P_{\mathsf{SC}}(f_{i,z_i^{(b)}}), V_{\mathsf{SC}}^{f_{i,z_i^{(b)}}}(v_i^{(b)}))$ for $b \in [2]$, where the verifier uses the same randomness in both sum-checks, and where for all $i \in [D]$ and $z \in \mathbb{F}^m$, $f_{i,z} : \mathbb{F}^{3m} \to \mathbb{F}$ is defined as follows:

$$f_{i,z}(p, \omega_1, \omega_2) = \widehat{\mathrm{EQ}}(z, p) \cdot \begin{pmatrix} \widetilde{\mathsf{add}}_i(p, \omega_1, \omega_2) \cdot \left(\widehat{V}_{i-1}(\omega_1) + \widehat{V}_{i-1}(\omega_2)\right) \\ + \widetilde{\mathsf{mult}}_i(p, \omega_1, \omega_2) \cdot \left(\widehat{V}_{i-1}(\omega_1) \cdot \widehat{V}_{i-1}(\omega_2)\right) \end{pmatrix},$$

where $p, \omega_1, \omega_2 \in \mathbb{F}^m$. Namely, the prover proves that for every $b \in [2]$,

$$\widehat{V}_i(z_i^{(b)}) = \sum_{p, \omega_1, \omega_2 \in B^m} f_{i,z_i^{(b)}}(p, \omega_1, \omega_2) = v_i^{(b)}.$$

At the end of this sum-check protocol, V_{GKR} needs to check that for every $b \in [2]$,

$$f_{i,z_i^{(b)}}(p, \omega_1, \omega_2) = \rho_i^{(b)},$$

where $p, \omega_1, \omega_2 \in \mathbb{F}^m$ are random elements chosen by V_{SC} in the sum-check protocol, and $\{\rho_i^{(b)}\}_{b \in [2]}$ are values determined by the sum-check protocols.

3. V_{GKR} asks the prover for the values of $\widehat{V}_{i-1}(\omega_1)$ and $\widehat{V}_{i-1}(\omega_2)$, and obtains values r_1, r_2.

4. V_{GKR} checks that indeed for every $b \in [2]$ it holds that

$$\rho_i^{(b)} = \widehat{\mathrm{EQ}}(z_i^{(b)}, p) \cdot \left(\widetilde{\mathsf{add}}_i(p, \omega_1, \omega_2) \cdot (r_1 + r_2) + \widetilde{\mathsf{mult}}_i(p, \omega_1, \omega_2) \cdot (r_1 \cdot r_2)\right),$$

by computing $\widehat{\mathrm{EQ}}(z_i^{(b)}, p)$, $\widetilde{\mathsf{add}}_i(p, \omega_1, \omega_2)$ and $\widetilde{\mathsf{mult}}_i(p, \omega_1, \omega_2)$ on its own. If this is not the case it rejects. Otherwise, if $i \geq 1$, then go back to Item 2 with $i = i - 1$, with $z_i^{(b)} = \omega_b$ and $v_i^{(b)} = r_b$ for $b \in \{1, 2\}$.

5. If $i = 0$, V_{GKR} checks on its own that indeed for every $b \in \{1, 2\}$ it holds that $\widehat{V}_0(\omega_b) = r_b$, by evaluating $\widehat{V}_0(\omega_b) = \sum_{p \in [N]} \widehat{\mathrm{EQ}}(\omega_b, p) \cdot x_p$, where $[N]$ is the set of input wires.

Figure 6: The (slightly modified) GKR protocol $(P_{\mathsf{GKR}}, V_{\mathsf{GKR}})$ [GKR15, KPY18]

Proof. Fix any ϵ. Let d be the univariate degree of the polynomial $f_{i,z}$ as defined in Equation (14). By the definition of $f_{i,z}$, together with Proposition 6.3 and the definition of low-degree extension, $d \leq \mathsf{poly}(|B|, m)$. We prove that the GKR protocol is ν-history-independent (T', d, ρ)-FS-compatible,

for $\nu = 6m$, $T'(\lambda) = 2^{\lambda^\epsilon}$, d as above, and for ρ that will be determine below.[26] To this end, we define the functions State and BAD corresponding to the GKR protocol.

Fix an input $x = (C, w)$, where $w \in \{0,1\}^N$ and C is a circuit $C : \{0,1\}^N \to \{0,1\}$ of size S and depth D, where the polynomials $\widetilde{\mathsf{add}}_i$ and $\widetilde{\mathsf{mult}}_i$ are degree $\leq \mathsf{poly}(m, |B|)$ polynomials that can be evaluated in $\mathsf{poly}(m, |B|, \log|\mathbb{F}|)$ time.

Recall that the GKR protocol is associated with parameters $\mathbb{F}, B, m$, where $\mathbb{F}$ is an extension field of GF[2], and $B \subseteq \mathbb{F}$ and $m \in \mathbb{N}$ are such that $S \leq |B|^m \leq \mathsf{poly}(S)$. The GKR protocol consists of D sequential phases, where in the i'th phase, the prover and verifier run two sum-check protocols in parallel, where the verifier uses the same randomness in both. Thus, each verifier message is simply a random field element $t \leftarrow \mathbb{F}$. Each of the sum-check protocols consist of $3m$ rounds (since each polynomial $f_{i,z}$ is a $3m$-variate polynomial), so the total number of rounds in the GKR protocol is $3m \cdot D$.[27] For every $u \in [3m \cdot D]$, parsing $u = i \cdot (3m) + j$ with $i \in \{0, 1, \ldots, D\}$ and $j \in \{0, 1, \ldots, 3m-1\}$, denote by

$$\tau = (\bar{\tau}_1, \ldots, \bar{\tau}_{i-1}, (g^{(b)}_{i,1})_{b\in[2]}, t_{i,1}, \ldots, (g^{(b)}_{i,j})_{b\in[2]}, t_{i,j}), \tag{16}$$

where for every $\alpha \in [i-1]$

$$\bar{\tau}_\alpha = ((g^{(b)}_{\alpha,1})_{b\in[2]}, t_{\alpha,1}, \ldots, (g^{(b)}_{\alpha,3m})_{b\in[2]}, t_{\alpha,3m}, \{r^{(b)}_{D-\alpha}\}_{b\in[2]}),$$

and for every $j' \in [3m]$, $(g^{(b)}_{\alpha,j'})_{b\in[2]}$ are the two univariate polynomials that the prover sends in the j'th round of the sum-checks in layer α of the GKR protocol, and $t_{\alpha,j'}$ is the random message sent by the verifier in the j' round of the same sum-checks. The elements $\{r^{(b)}_{D-\alpha}\}_{b\in[2]}$ are the message sent by the prover at the end of these sum-checks.

The function State**.** For any $x = (C, w)$ and any transcript prefix τ as in Equation (16),

$$\mathsf{State}(x, \tau) = \mathsf{State}(x, i, \tau_{i-1}, (g^{(b)}_{i,1})_{b\in[2]}, t_{i,1}, \ldots, (g^{(b)}_{i,j})_{b\in[2]}, t_{i,j})$$

does the following:

1. If $i > 1$ then let

   $$z^{(1)}_{D-(i-1)} = (t_{i-1,m+1}, \ldots, t_{i-1,2m}) \in \mathbb{F}^m \quad \text{and} \quad z^{(2)}_{D-(i-1)} = (t_{i-1,2m+1}, \ldots, t_{i-1,3m}) \in \mathbb{F}^m$$

   and let $\{r^{(b)}_{D-(i-1)}\}_{b\in[2]}$ be the prover's last message in τ_{i-1}.

   If $i = 1$ then let

   $$z^{(0)}_D = z^{(1)}_D = \omega_{\mathsf{out}} \quad \text{and} \quad r^{(0)}_D = r^{(1)}_D = 0.$$

2. If $\tau = (i, \tau_{i-1}, \emptyset)$ then output accept if and only if for every $b \in [2]$

   $$r^{(b)}_{D-(i-1)} = \hat{V}_{D-(i-1)}(z^{(b)}_{D-(i-1)}),$$

   Otherwise output reject.

[26] The functions State and BAD, as we will describe below, will actually take in the most recent $\leq \nu = 6m$ messages. They can be viewed as functions of the last $\nu = 6m$ messages by disregarding the extra, unused messages.

[27] At the end of each sum-check, the prover will additionally send over a couple of values $r^{(b)}_{D-\alpha}$. We can view this additional prover message as being part of the first prover message for the next sum-check, so there are $3m$ rounds per layer of the circuit C, rather than $3m + 1$.

3. Recall that the inputs to the i'th phase (parallel) sum-checks are, for $b \in [2]$,

$$\hat{V}_{D-(i-1)}(z^{(b)}_{D-(i-1)}) = \sum_{p,\omega_1,\omega_2 \in B^m} f_{D-(i-1),z^{(b)}_{D-(i-1)}}(p,\omega_1,\omega_2) = r^{(b)}_{D-(i-1)}.$$

Otherwise, if $j > 0$, for every $b \in [2]$, compute

$$g_b(\cdot) = \sum_{b_{j+1},\ldots,b_{3m} \in B} f_b(t_{i,1},\ldots,t_{i,j-1},\cdot,b_{j+1},\ldots,b_{3m}). \tag{17}$$

where

$$f_b \triangleq f_{D-(i-1),z^{(b)}_{D-(i-1)}}. \tag{18}$$

Accept if and only if for every $b \in [2]$ it holds that $g_b(t_{i,j}) = g^{(b)}_{i,j}(t_{i,j})$ and $g^{(b)}_{i,j}$ is a univariate polynomial of degree at most d.

Note that State only depends on the communication in the last two sum-checks, and thus is ν-history-independent for $\nu = 6m$. Moreover, note that it is computable in time $2|B|^{3m} \cdot T_f$, where T_f is the time it takes to compute f_1 or f_2. Recall that by definition $\lambda = \log|\mathbb{F}|$. Note that f_b is computable in time $S \cdot \mathsf{polylog}(|\mathbb{F}|)$ and $|B|^m \leq \mathsf{poly}(S)$, which imply that State is computable in time $\mathsf{poly}(S)$.[28] Together with the fact that $\log|\mathbb{F}| \geq (\log S)^{2/\epsilon}$, this implies that $T'(\lambda) \leq 2^{\lambda^\epsilon}$, as desired.

Round-by-round soundness is established by observing that State trivially satisfies the other three conditions (Items (1)-(3) in the definition of round-by-round soundness).

The function BAD**.** Fix any $(x, \bar{t})$, where

$$\bar{t} = (\bar{t}_1,\ldots,\bar{t}_{i-1},t_{i,1},\ldots,t_{i,j})$$

for $i \in \{0,1,\ldots,D\}$ and $j \in \{0,1\ldots,3m-1\}$. For every $\alpha \in [i-1]$,

$$\bar{t}_\alpha = (t_{\alpha,1},\ldots,t_{\alpha,3m}),$$

denotes all of the verifier messages in the α'th phase of the GKR protocol, and $t_{i,1},\ldots,t_{i,j}$ are the first j verifier messages in the i'th layer of the GKR protocol. The function BAD depends only on $(x,i,\bar{t}_{i-1},t_{i,1},\ldots,t_{i,j})$, via advice (g_1,g_2), computed as follows:

1. If $i > 1$ then let

$$z^{(1)}_{D-(i-1)} = (t_{i-1,m+1},\ldots,t_{i-1,2m}) \in \mathbb{F}^m \quad \text{and} \quad z^{(2)}_{D-(i-1)} = (t_{i-1,2m+1},\ldots,t_{i-1,3m}) \in \mathbb{F}^m.$$

If $i = 1$ then let

$$z^{(1)}_{D-(i-1)} = z^{(2)}_{D-(i-1)} = \omega_{\mathsf{out}}.$$

2. Output the advice (g_1,g_2), computed as in Equation (17).

The function $\mathsf{BAD} = \mathsf{BAD}_{g_1,g_2}$, takes as input $(\bar{g}_{i-1},(g^{(b)}_{i,1})_{b\in[2]},\ldots,(g^{(b)}_{i,j})_{b\in[2]})$, it ignores all its inputs except for the last two polynomials $(g^*_1,g^*_2) = (g^{(1)}_{i,j},g^{(2)}_{i,j})$, and does the following:

[28] This follows from the fact that we always choose a field $\mathbb{F}$ such that $\log|\mathbb{F}| < S$.

1. If g_1^* or g_2^* is not a univariate degree d polynomial over $\mathbb{F}$ then output $\perp$.
2. Otherwise, for every $b \in [2]$ let R_b denote the set of all roots of the polynomial $g_b - g_b^*$, and output a random element in $R_1 \cap R_2$, using the Cantor-Zassenhaus algorithm [CZ81].

Note that BAD can be computed in time $\mathsf{poly}(d, \log \mathbb{F}) = \mathsf{poly}(\lambda)$, where the latter equation follows from the fact that $\lambda \triangleq \log|\mathbb{F}| \geq d$.

It remains to argue that for every partial transcript τ as in Equation (16) such that $\mathsf{State}(x, \tau) = \mathsf{reject}$, and for every (g_1^*, g_2^*) with probability $1 - \mathsf{negl}(\lambda)$ (over the randomness of BAD_{g_1,g_2}), $\mathsf{BAD}_{g_1,g_2}(g_1^*, g_2^*)$ outputs a uniformly random element in the set

$$\mathcal{B} = \{t_i \ : \ \mathsf{State}(x, \tau|(g_1^*, g_2^*)|t_i) = \mathsf{accept}\}.$$

To this end, note that by definition of State,

$$\mathcal{B} = \{t \ : \ ((g_1(t), g_2(t)) = (g_1^*(t), g_2^*(t))\}$$

where g_1 and g_2 are defined in Equation (17). Indeed by definition, $\mathsf{BAD}_{g_1,g_2}(g_1^*, g_2^*)$ outputs a random element in $\mathcal{B}$ with overwhelming probability,[29] as desired. □

Corollary 6.6. *Assuming the sub-exponential hardness of* LWE*, there exists a hash family $\mathcal{H}$ such that the non-interactive argument, obtained by applying the Fiat-Shamir transform to the* GKR *protocol w.r.t. the hash family $\mathcal{H}$, is sound assuming the field size in the* GKR *protocol satisfies Equation* (15) *(for a small enough constant $\epsilon > 0$).*

For a circuit $C : \{0,1\}^N \to \{0,1\}$ of size S and depth D, for which $\widetilde{\mathsf{add}}_i$ and $\widetilde{\mathsf{mult}}_i$ are both degree $\mathsf{poly}(m, |B|)$ polynomials that are evaluable in $\mathsf{poly}(m, |B|, \log|\mathbb{F}|)$ time, the resulting non-interactive argument has the following efficiency guarantees: the prover runs in time $\mathsf{poly}(S)$, the verifier runs in time $(D + N) \cdot \mathsf{polylog}(S)$, and the communication complexity is $D \cdot \mathsf{polylog}(S)$.

Proof. Assume the sub-exponential hardness of LWE. By Corollary 3.8, there exists a constant $0 < \epsilon_1 < 1$, such that for every constant $0 < \delta < 1$, there exists a constant $0 < \epsilon_2 < 1$ such that the following holds: For any polynomial S and any hash family $\mathcal{F}$ consisting of functions computable by circuits of size S, there exists a (T, T', ω)-lossy CI hash family for $\mathcal{F}$, where $T(\lambda) = 2^{\lambda^{\epsilon_1}}, T'(\lambda) = 2^{\lambda^{\epsilon_2}}$, and $\omega = n - \lambda^\delta$.

Fix $\delta = \epsilon_1/2$, which fixes ϵ_2 above. Assume w.l.o.g. that $\epsilon_2 \leq \epsilon_1$ (otherwise set $\epsilon_1 = \epsilon_2$). By Theorem 6.5, applied with $\epsilon = \epsilon_2$, there exists a polynomial ρ such that the GKR protocol is ν-history independent (T', d, ρ)-FS-compatible, for $d = \mathsf{poly}(|B|, m)$ and $\nu = 6m$ (assuming the field size satisfies Equation (15) w.r.t. $\epsilon = \epsilon_2$). By Theorem 4.5, the non-interactive argument obtained by applying the Fiat-Shamir transform to the GKR protocol, with a (T, T', ω)-lossy CI hash family, is sound if $T' \geq 3mD$ and $T \geq \max(d, 2^{\nu(n-\omega)}, T')$. The former inequality holds by Equation (15) together with the fact that $\lambda = \log|\mathbb{F}|$. The latter inequality holds via the following calculations: the fact that $T \geq d$ follows from Equation (15); the fact that $T \geq T'$ follows from our assumption that $\epsilon_1 \geq \epsilon_2$; the fact that $T \geq 2^{\nu(n-\omega)}$ follows from the fact that $\lambda^{\epsilon_1} \geq \nu(n - \omega)$, which follows from the fact that $n - \omega = \lambda^\delta = \lambda^{\epsilon_1/2}$ together with the fact that $\nu = 6m \leq \lambda^{\epsilon_2/2} \leq \lambda^{\epsilon_1/2}$ (where the former inequality follows from Equation (15)). □

[29] The negligible probability of error follows from the Cantor-Zassenhaus algorithm, which is a randomized algorithm that has a negligible probability of error.

In light of Proposition 6.3, this gives a non-interactive argument for any log-space uniform circuit $C : \{0,1\}^N \to \{0,1\}$ of size S and depth D, assuming the sub-exponential hardness of LWE. This non-interactive argument is generated by computing a non-interactive argument as above for the circuit $C' : \{0,1\}^N \to \{0,1\}$ of size $S' = S \cdot \mathsf{polylog}(S)$ and depth $D' = D \cdot \mathsf{polylog}(S)$.

Corollary 6.7. *Assuming the sub-exponential hardness of* LWE*, for any log-space uniform $C : \{0,1\}^N \to \{0,1\}$ of size S and depth D, there is a non-interactive argument for the language $\{x : C(x) = 0\}$. This non-interactive argument has the following efficiency guarantees: the prover runs in time $\mathsf{poly}(S)$, the verifier runs in time $(D+N) \cdot \mathsf{polylog}(S)$, and the communication complexity is $D \cdot \mathsf{polylog}(S)$.*

7 PPAD Hardness

The main result of [CHK+19a] is that the class CLS $\subseteq$ PPAD is hard, assuming that the Fiat-Shamir transformation applied to the sum-check protocol is adaptively and unambiguously sound. While our non-interactive sum-check protocol is not known to be adaptively and unambiguously sound, we show that it satisfies weaker conditions, which we call *prefix-adaptive soundness* and *prefix-adaptive unambiguity*, that are sufficient for PPAD hardness.

In what follows, we consider a non-interactive sum-check protocol over the set $B = \{0,1\}$. Furthermore, we denote by $[0,d]$ a subset of $\mathbb{F}$ of size $d+1$. This can be chosen arbitrarily: we can take $[0,d]$ to be the first $d+1$ elements of $\mathbb{F}$ lexicographically.

The main theorem of [CHK+19a], restated with the prefix-adaptive versions of soundness and unambiguity, is stated below.

Theorem 7.1. *[CHK+19a] The complexity class* CLS *is sub-exponentially hard on average assuming* #SAT *is sub-exponentially hard on average, and assuming that there exists a constant $\epsilon > 0$, and for every $\ell \in \mathbb{N}$ there exists a field $\mathbb{F} = \mathbb{F}_\ell$ with $\log|\mathbb{F}| \leq \mathsf{poly}(\ell)$ and a non-interactive sum-check protocol $(\mathsf{niSC.Setup}, P_{\mathsf{niSC}}, V_{\mathsf{niSC}})$ for ℓ-variate polynomials over $\mathbb{F}$ of individual degree $d \leq \mathsf{poly}(\ell)$, which satisfies the following guarantees:*

- **Prefix-Adaptive Soundness:** *For any $\mathsf{poly}(2^{\ell^\epsilon})$-size cheating prover $P^* = \{P^*_\ell\}_{\ell \in \mathbb{N}}$, there is a negligible function μ such that for every $\ell \in \mathbb{N}$ and any ℓ-variate polynomial $g : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree d, and for $\lambda = \log|\mathbb{F}|$ and $\mathsf{CRS} = (k_1, \ldots, k_\ell) \leftarrow \mathsf{niSC.Setup}(1^\lambda)$,*

$$\Pr\left[P^*(\mathsf{CRS}) = (j, \sigma_1, \ldots, \sigma_j, I, \{\xi_i\}_{i \in I}, v, \tau) : \begin{cases} & 0 \leq j \leq \ell \\ \wedge & I \subseteq [j] \\ \wedge & \sigma_i = \mathcal{H}.\mathsf{Hash}(k_i, \xi_i) \ \ \forall i \in I \\ \wedge & \sigma_i \in [0,d] \ \ \forall i \in [j] \setminus I \\ \wedge & \displaystyle\sum_{b_1, \ldots, b_{\ell-j} \in \{0,1\}} g^{(j)}(b_1, \ldots, b_{\ell-j}) \neq v \\ \wedge & V^{g^{(j)}}_{\mathsf{niSC}}(v, \mathsf{CRS}^{(j)}, \tau) = 1 \end{cases} \right] = \mu(2^{\ell^\epsilon}).$$

Here, $g^{(j)}$ is defined by

$$g^{(j)}(b_1, \ldots, b_{\ell-j}) \triangleq g(\sigma_1, \ldots, \sigma_j, b_1, \ldots, b_{\ell-j}),$$

and $\mathsf{CRS}^{(j)} = (k_{j+1}, \ldots, k_\ell)$.

- **Prefix-Adaptive Unambiguity:** *For any* $\mathsf{poly}(2^{\ell^\epsilon})$*-size cheating prover* $P^* = \{P^*_\ell\}_{\ell\in\mathbb{N}}$*, there is a negligible function* μ *such that for every* $\ell \in \mathbb{N}$ *and any* ℓ*-variate polynomial* $g : \mathbb{F}^\ell \to \mathbb{F}$ *of individual degree* d*, and for* $\lambda = \log|\mathbb{F}|$ *and* $\mathsf{CRS} = (k_1, \ldots, k_\ell) \leftarrow \mathsf{niSC.Setup}(1^\lambda)$*,*

$$\Pr\left[\begin{array}{l} P^*(\mathsf{CRS}) = (j, \sigma_1, \ldots, \sigma_j, I, \{\xi_i\}_{i\in I}, v, \tau_1, \tau_2) : \\ \end{array} \left\{\begin{array}{ll} & 0 \le j \le \ell \\ \wedge & I \subseteq [j] \\ \wedge & \sigma_i = \mathcal{H}.\mathsf{Hash}(k_i, \xi_i) \ \ \forall i \in I \\ \wedge & \sigma_i \in [0, d] \ \ \forall i \in [j] \setminus I \\ \wedge & V^{g^{(j)}}_{\mathsf{niSC}}(v, \mathsf{CRS}^{(j)}, \tau_1) = 1 \\ \wedge & V^{g^{(j)}}_{\mathsf{niSC}}(v, \mathsf{CRS}^{(j)}, \tau_2) = 1 \\ \wedge & \tau_1 \neq \tau_2 \end{array}\right\}\right] = \mu(2^{\ell^\epsilon}),$$

  where $g^{(j)}$ *and* $\mathsf{CRS}^{(j)}$ *are defined as above.*

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

We note that Theorem 7.1, as stated, differs from the corresponding theorem in [CHK+19a]. Specifically, in [CHK+19a] they require *full adaptivity* of soundness and unambiguity, and in particular, they allow the cheating prover to choose the multivariate polynomial $g^{(j)}$ (on which it outputs a non-interactive sum-check proof) completely adaptively. While we do not guarantee that our non-interactive sum-check protocol has full adaptivity, upon inspecting their proof, we notice that they do not need the full adaptivity of $g^{(j)}$ to obtain their PPAD hardness result (and probably require it only for the sake of simplicity); rather, they only need the adaptivity in choosing the prefix $\sigma_1, \ldots, \sigma_j$, which is chosen either in the support of the hash function or in the set $[0, d]$.

The rest of this section is partitioned into two parts. In Section 7.1, we give a sketch of the proof of Theorem 7.1. Then, in Section 7.2, we show that our non-interactive sum-check protocol satisfies prefix-adaptive soundness and prefix-adaptive unambiguity. As for the efficiency condition of Theorem 7.1, the fact that V_{niSC} runs in $\mathsf{poly}(\ell)$ time follows straightforwardly from the efficiency of the non-interactive sum-check verifier given in Corollary 5.3. Therefore, using the fact that the sub-exponential hardness of LWE implies the sub-exponential average-case hardness of #SAT, we obtain the following corollary:

Corollary 7.2. CLS *is sub-exponentially hard on average assuming the sub-exponential hardness of* LWE*.*

7.1 Sketch of Proof of Theorem 7.1

The proof of Theorem 7.1 in [CHK+19a] proceeds as follows. They first define the *Relaxed-Sink-of-Verifiable-Line* (rSVL) problem, and prove that it is in the complexity class CLS. Then they show how to reduce #SAT instances to rSVL instances. We start by defining the rSVL problem.

Definition 7.3. The *Relaxed-Sink-of-Verifiable-Line* (rSVL) problem $(\mathsf{S}, \mathsf{V}, T^*, s_0)$ consists of a time bound $T^* \in \mathbb{N}$, a starting state $s_0 \in \{0,1\}^L$, and $\mathsf{poly}(\ell)$-sized functions $\mathsf{S} : \{0,1\}^L \to \{0,1\}^L$ and $\mathsf{V} : [0, T^*] \times \{0,1\}^L \to \{0,1\}$ with the guarantee that for all $t \in [0, T^*]$ and $s \in \{0,1\}^L$, if $s = \mathsf{S}^t(s_0)$ then $\mathsf{V}(t, s) = 1$. The goal is to find one of the following:

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

Proof sketch of Theorem 7.1. Beginning with a Boolean formula ϕ in ℓ variables $x_1, \ldots, x_\ell$, the task of the #SAT problem is to find the number of satisfying solutions to ϕ, or to evaluate

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

We arithmetize ϕ to get an ℓ-variate polynomial g of individual degree $d \leq \mathsf{poly}(\ell)$. Then the number of satisfying solutions to ϕ can be written also as

$$\sum_{b_1,\ldots,b_\ell \in \{0,1\}} g(b_1, \ldots, b_\ell).$$

This computation can be delegated via the sum-check protocol to a $\mathsf{poly}(2^\ell)$-size prover who gives a verifiable and unambiguous "proof"[30] of correctness.

The main technical contribution of [CHK+19a] is that this proof can be computed in an incremental and efficiently updatable manner. Namely, the summation above can be performed via a sequence of $\mathsf{poly}(2^\ell)$ steps, where going from step t to step $t+1$ can be done in $\mathsf{poly}(\log|\mathbb{F}|) \leq \mathsf{poly}(\ell)$ time. More specifically, let s_t denote the state after the t'th computation step. Incremental verifiability means that each intermediate state s_t includes a proof of its correctness, and the proof can be updated and verified in time $\mathsf{poly}(\log|\mathbb{F}|)$. This incrementally verifiable counting procedure is used to construct, given a #SAT instance, an instance of the rSVL problem.

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

Suppose that there exists a sub-exponential size adversary that solves the rSVL instance corresponding to the computation

$$\sum_{b_1,\ldots,b_\ell \in \{0,1\}} g(b_1, \ldots, b_\ell).$$

It solves the instance by either finding the sink, which corresponds to successfully computing v such that

$$\sum_{b_1,\ldots,b_\ell \in \{0,1\}} g(b_1, \ldots, b_\ell) = v$$

(i.e., solving #SAT), or by finding false positives, which corresponds to breaking the unambiguous soundness of the underlying non-interactive sum-check argument.

We observe that the latter breaks, in fact, the *prefix-adaptive* soundness or *prefix-adaptive* unambiguity. In a nutshell, the polynomial g is fixed by the chosen #SAT instance, and the adaptivity is in choosing (t, s) such that $\mathsf{V}(t, s) = 1$. As we argue below, every state s is associated with a set of prefixes $(\sigma_1, \ldots, \sigma_j)$, such that for every $i \in [j]$, σ_i is either in the image of the i'th Fiat-Shamir hash function or in the set $[0, d]$, and for each such prefix it contains a non-interactive sum-check proof for the computation $\sum_{b_{j+1},\ldots,b_\ell} g(\sigma_1, \ldots, \sigma_j, b_{j+1}, \ldots, b_\ell)$.

[30] We really mean "argument" here, generated according to our non-interactive sum-check protocol. We use the terminology "proof" to be consistent with the terminology in [CHK+19a].

Specifically, each intermediate state s is a small (poly-size) step towards the computation of the sum

$$v_0 = \sum_{b_1,\ldots,b_\ell \in \{0,1\}} g(b_1, \ldots, b_\ell),$$

along with a proof $\tau = (\alpha_1, \beta_1, \ldots, \alpha_\ell, \beta_\ell)$ (which is precisely the non-interactive sum-check proof) for this computation. This is done, broadly, in the same fashion as the computation done by the honest non-interactive sum-check prover. First, the univariate polynomial

$$\alpha_1(\cdot) = \sum_{b_2,\ldots,b_\ell \in \{0,1\}} g(\cdot, b_2, \ldots, b_\ell)$$

is computed together with a corresponding certificate π_1,[31] and β_1 is set to $\mathcal{H}.\mathsf{Hash}(k_1, \alpha_1)$. Next, this same procedure is repeated for the $(\ell - 1)$-variate sum

$$v_1 = \sum_{b_2,\ldots,b_\ell \in \{0,1\}} g(\beta_1, b_2, \ldots, b_\ell).$$

A polynomial

$$\alpha_2(\cdot) = \sum_{b_3,\ldots,b_\ell \in \{0,1\}} g(\beta_1, \cdot, b_3, \ldots, b_\ell)$$

is computed together with a corresponding certificate π_2, and β_2 is set to $\mathcal{H}.\mathsf{Hash}(k_2, (\alpha_1, \beta_1, \alpha_2))$. The same procedure is repeated for the $(\ell - 2)$-variate sum

$$v_2 = \sum_{b_3,\ldots,b_\ell \in \{0,1\}} g(\beta_1, \beta_2, b_3, \ldots, b_\ell),$$

and so on. At the end of ℓ such procedures, we will have the sequence of values

$$(\alpha_1, \pi_1, \beta_1, \ldots, \alpha_\ell, \pi_\ell, \beta_\ell).$$

We now discard all the certificates π_i, and keep only

$$\tau = (\alpha_1, \beta_1, \ldots, \alpha_\ell, \beta_\ell)$$

as the proof for the entire calculation $\sum_{b_1,\ldots,b_\ell} g(b_1, \ldots, b_\ell)$. The final state s_{T^*} thus consists of the sum v_0 as well as the sum-check proof τ.

Note that the bulk of the calculative cost is in calculating (α_i, π_i), which takes time $2^{O(\ell - i)} >> \mathsf{poly}(\ell)$. The calculation of (α_i, π_i) is thus broken up into recursive subprocesses. Specifically, to calculate (α_i, π_i) (given the already calculated $(\alpha_1, \pi_1, \beta_1, \ldots, \alpha_{i-1}, \pi_{i-1}, \beta_{i-1})$), we recursively compute for each $\gamma \in [0, d]$ the value

$$v_{i,\gamma} = \sum_{b_{i+1},\ldots,b_\ell \in \{0,1\}} g(\beta_1, \ldots, \beta_{i-1}, \gamma, b_{i+1}, \ldots, b_\ell)$$

along with a sum-check proof $\tau_{i,\gamma}$ of the sum's correctness. Then, we interpolate the $d + 1$ points $\{(\gamma, v_{i,\gamma})\}_{\gamma \in [0,d]}$ to get the polynomial α_i, and we set the certificate $\pi_i = \{\tau_{i,\gamma}\}_{\gamma \in [0,d]}$.

[31] As we clarify below, this certificate is not a sum-check proof, but rather consists of a set of $d + 1$ sum-check proofs.

Any arbitrary state s is thus a partial computation of (v_0, τ) and is associated with some

$$(\alpha_1, \pi_1, \beta_1, \ldots, \alpha_i, \pi_i, \beta_i)$$

(which has already been computed) and some intermediate computation of $(\alpha_{i+1}, \pi_{i+1})$. This intermediate computation consists of a set $\{v_{i+1,\gamma}, \tau_{i+1,\gamma}\}_{\gamma \in [0,d'-1]}$ for some $d' \leq d$, where

$$v_{i+1,\gamma} = \sum_{b_{i+2},\ldots,b_\ell} g(\beta_1, \ldots, \beta_i, \gamma, b_{i+2}, \ldots, b_\ell)$$

and $\tau_{i+1,\gamma}$ is its proof, and a partial computation of $(v_{i+1,d'}, \tau_{i+1,d'})$. This partial computation of $(v_{i+1,d'}, \tau_{i+1,d'})$ again consists of some

$$(\alpha_{i+1,d',1}, \pi_{i+1,d',1}, \beta_{i+1,d',1}, \ldots, \alpha_{i+1,d',i'}, \pi_{i+1,d',i'}, \beta_{i+1,d',i'})$$

and some intermediate computation of $(\alpha_{i+1,d',i'+1}, \pi_{i+1,d',i'+1})$, which again is some set of finished sum-check proofs and a partial computation of another, and so on. The algorithm V, on input (t, s), checks that s is in the correct stage of computation (that corresponds to time t), that each of the sum-check proofs in s verifies correctly, that the prover messages are interpolated correctly and the corresponding β's are the correct hash value.

Note that in the above procedure, we perform a recursive computation to compute the $(\ell - j)$-variate sum

$$\sum_{b_{j+1},\ldots,b_\ell \in \{0,1\}} g(\sigma_1, \ldots, \sigma_j, b_{j+1}, \ldots, b_\ell)$$

together with a corresponding proof for many different prefixes $(\sigma_1, \ldots, \sigma_j)$. In fact, the entire procedure is a recursive calculation for the empty prefix, which is done by calling the recursive procedure for each prefix (γ) (for $\gamma \in [0, d]$), which allows us to compute $(\alpha_1, \pi_1, \beta_1)$. Once β_1 has been computed, a recursive computation is done for each of the prefixes (β_1, γ) for $\gamma \in [0, d]$ to compute (α_2, π_2) and thus β_2. Once β_2 is computed, a recursive computation is done for each of the prefixes $(\beta_1, \beta_2, \gamma)$ to compute (α_3, π_3), and so on.

On the other hand, inside the computation for the prefix (γ_1) where $\gamma_1 \in [0, d]$, we recursively compute the sum

$$\sum_{b_2,\ldots,b_\ell \in \{0,1\}} g(\gamma_1, b_2, \ldots, b_\ell)$$

along with a sum-check proof $(\alpha_{\gamma_1,1}, \beta_{\gamma_1,1}, \ldots, \alpha_{\gamma_1,\ell}, \beta_{\gamma_1,\ell})$. We compute $\alpha_{\gamma_1,1}$ recursively, by computing for every $\gamma_2 \in [0, d]$, the sum

$$\sum_{b_3,\ldots,b_\ell \in \{0,1\}} g(\gamma_1, \gamma_2, b_3, \ldots, b_\ell)$$

together with a sum-check proof. From all these sums and proofs one can efficiently compute $(\alpha_{\gamma_1,1}, \pi_{\gamma_1,1})$, and thus also efficiently compute $\beta_{\gamma_1,1}$. In order to compute $(\alpha_{\gamma_1,2}, \pi_{\gamma_1,2}, \beta_{\gamma_1,2})$ we have to perform the recursive computation for prefixes of the form $(\gamma_1, \beta_{\gamma_1,1}, \gamma)$, for every $\gamma \in [0, d]$, and so on. Observe that every prefix $(\sigma_1, \ldots, \sigma_j)$ that we encounter must satisfy the property that each σ_ι for $\iota \in [j-1]$ is either in the image of $\mathcal{H}.\mathsf{Hash}(k_\iota, \cdot)$ or in $[0, d]$, and furthermore, the last prefix item σ_j must be in $[0, d]$.

We want to be clear about what part of the transcript is hashed to compute a verifier message $\beta_{\sigma_1,\ldots,\sigma_j,i}$. By $\beta_{\sigma_1,\ldots,\sigma_j,i}$, we mean the i'th verifier message in the sum-check proof for the sum

$$v_{\sigma_1,\ldots,\sigma_j} = \sum_{b_{j+1},\ldots,b_\ell \in \{0,1\}} g(\sigma_1,\ldots,\sigma_j,b_{j+1},\ldots,b_\ell)$$

corresponding to the prefix $(\sigma_1,\ldots,\sigma_j)$. We assume we have already computed the first length $i-1$ transcript

$$(\alpha_{\sigma_1,\ldots,\sigma_j,1},\beta_{\sigma_1,\ldots,\sigma_j,1},\ldots,\alpha_{\sigma_1,\ldots,\sigma_j,i-1},\beta_{\sigma_1,\ldots,\sigma_j,i-1}).$$

To compute $\alpha_{\sigma_1,\ldots,\sigma_j,i}$ and $\beta_{\sigma_1,\ldots,\sigma_j,i}$, we first recursively compute the $d+1$ sums

$$v_{\sigma_1,\ldots,\sigma_j,i,\gamma} = \sum_{b_{j+i+1},\ldots,b_\ell \in \{0,1\}} g(\sigma_1,\ldots,\sigma_j,\beta_{\sigma_1,\ldots,\sigma_j,1},\ldots,\beta_{\sigma_1,\ldots,\sigma_j,i-1},\gamma,b_{j+i+1},\ldots,b_\ell)$$

for every $\gamma \in [0,d]$ along with proofs $\tau_{\sigma_1,\ldots,\sigma_j,i,\gamma}$, then interpolate these values $\{(\gamma, v_{\sigma_1,\ldots,\sigma_j,i,\gamma})\}_{\gamma\in[0,d]}$ to get a polynomial $\alpha_{\sigma_1,\ldots,\sigma_j,i}$. The corresponding certificate $\pi_{\sigma_1,\ldots,\sigma_j,i}$ is set to $\{\tau_{\sigma_1,\ldots,\sigma_j,i,\gamma}\}_{\gamma\in[0,d]}$. Finally, to compute $\beta_{\sigma_1,\ldots,\sigma_j,i}$, we hash the entire transcript so far for the prefix $(\sigma_1,\ldots,\sigma_j)$,[32] that is,

$$\beta_{\sigma_1,\ldots,\sigma_j,i} = \mathcal{H}.\mathsf{Hash}(k_{j+i},(\alpha_{\sigma_1,\ldots,\sigma_j,1},\beta_{\sigma_1,\ldots,\sigma_j,1},\ldots,\alpha_{\sigma_1,\ldots,\sigma_j,i-1},\beta_{\sigma_1,\ldots,\sigma_j,i-1},\alpha_{\sigma_1,\ldots,\sigma_j,i})).$$

This leads us to our conditions of *prefix-adaptive soundness and unambiguity*. Finding a false positive for rSVL requires finding a partial computation state for which at least one of its calculations, which consist of some prefix $(\sigma_1,\ldots,\sigma_j)$ followed by a sum-check proof $\tau_{\sigma_1,\ldots,\sigma_j}$ for the corresponding sum, is different from the honest computation. This happens precisely when $\tau_{\sigma_1,\ldots,\sigma_j}$ is either not unique (i.e., a second, different $\tau_{\sigma_1,\ldots,\sigma_j}$ is efficiently computable) or $\tau_{\sigma_1,\ldots,\sigma_j}$ is an accepting proof for an incorrect statement. But our conditions of prefix-adaptive unambiguity and soundness, respectively, prevent precisely these possibilities.[33]

For further details about this reduction, we refer the reader to the original paper [CHK+19a], which contains this proof in much more detail.

7.2 Prefix-Adaptive Soundness and Unambiguity of the Non-Interactive Sum-Check

In what follows, we prove that, under the sub-exponential hardness of LWE, there exists a hash family $\mathcal{H}$ such that the non-interactive sum-check protocol, obtained by applying the Fiat-Shamir transform w.r.t. $\mathcal{H}$, satisfies prefix-adaptive soundness and prefix-adaptive unambiguity from Theorem 7.1 (assuming the field size is large enough).

Theorem 7.4. *Assuming the sub-exponential hardness of* LWE*, there exists a constant $\epsilon' > 0$ and a hash family $\mathcal{H}$, such that applying the Fiat-Shamir transform w.r.t. $\mathcal{H}$ to the sum-check protocol results in a*

[32] In the original work [CHK+19a], the hash functions $\mathcal{H}.\mathsf{Hash}(k_i,\cdot)$ took as input only part of the preceding transcript. Due to the way we defined the Fiat-Shamir paradigm in this paper, in particular the fact that the hash functions take as input the entire preceding transcript, this modification to the PPAD reduction is necessary. We could have avoided this change by defining the Fiat-Shamir hash functions to take as input only the preceeding prover message, as opposed to the entire transcript.

[33] The conditions of prefix-adaptive soundness and unambiguity as defined in Theorem 7.1 actually give a slightly stronger guarantee, since they require soundness and unambiguity for prefix $(\sigma_1,\ldots,\sigma_j)$ for which σ_j is either in $[0,d]$ *or* in the image of $\mathcal{H}.\mathsf{Hash}(k_\iota,\cdot)$, while it suffices for the reduction to restrict to σ_j to be in $[0,d]$.

non-interactive argument $(\mathsf{niSC.Setup}, P_{\mathsf{niSC}}, V_{\mathsf{niSC}})$ *with the prefix-adaptive soundness and unambiguity guarantees from Theorem 7.1, assuming the underlying multi-variate polynomial is over a field* $\mathbb{F}$ *of size*

$$\log|\mathbb{F}| \geq \max\{d, (\ell \cdot \log(2 \cdot d))^{2/\epsilon'}\}, \tag{19}$$

where ℓ *is the number of variables in the polynomial and* $d \leq \mathsf{poly}(\ell)$ *is its univariate degree, and assuming that the summation is over* $\{0,1\}^{\ell}$.[34] *Namely, there exists a constant* $\epsilon > 0$ *such that for any* $\mathsf{poly}(2^{\ell^{\epsilon}})$*-size cheating prover* $P^* = \{P^*_{\ell}\}_{\ell \in \mathbb{N}}$*, there is a negligible function* μ *such that for every* $\ell \in \mathbb{N}$ *and every* ℓ*-variate polynomial* $g : \mathbb{F}^{\ell} \to \mathbb{F}$ *of individual degree* d*, and for* $\lambda = \log|\mathbb{F}|$ *and* $\mathsf{CRS} \leftarrow \mathsf{niSC.Setup}(1^{\lambda})$,[35]

$$\Pr\left[P^*(\mathsf{CRS}) = (j, \sigma_1, \ldots, \sigma_j, I, \{\xi_i\}_{i \in I}, v, \tau_1, \tau_2) : \left\{ \begin{array}{ll} & 0 \leq j \leq \ell \\ \wedge & I \subseteq [j] \\ \wedge & \sigma_i = \mathcal{H}.\mathsf{Hash}(k_i, \xi_i) \ \ \forall i \in I \\ \wedge & \sigma_i \in [0, d] \ \ \forall i \in [j] \setminus I \\ \wedge & V^{g^{(j)}}_{\mathsf{niSC}}(v, \mathsf{CRS}^{(j)}, \tau_1) = 1 \\ \wedge & V^{g^{(j)}}_{\mathsf{niSC}}(v, \mathsf{CRS}^{(j)}, \tau_2) = 1 \\ \wedge & \left[\begin{array}{r} (\tau_1 \neq \tau_2) \ \vee \\ (v \neq \sum\limits_{b_1, \ldots, b_{\ell - j}} g^{(j)}(b_1, \ldots, b_{\ell - j})) \end{array} \right] \end{array} \right\} \right] = \mu(2^{\ell^{\epsilon}}).$$

Proof. By Theorem 5.2, for every constant $\epsilon' > 0$ and for $T'(\lambda) = 2^{\lambda^{\epsilon'}}$, there exists a polynomial ρ such that the interactive sum-check protocol (Figure 5) is (T', d, ρ)-FS-compatible (Definition 4.2), assuming the instances are multi-variate polynomials of individual degree d over a field $\mathbb{F} = \mathbb{F}_{\ell}$ that satisfies Equation (19), and assuming the summation is over $\{0,1\}^{\ell}$ (i.e., assuming $B = \{0,1\}$).

By Corollary 3.8 , assuming the sub-exponential hardness of LWE, there exists a constant $0 < \epsilon_1 < 1$ such that for every constant $0 < \delta < 1$, and in particular for $\delta = \epsilon_1/2$, there exists a constant $0 < \epsilon_2 < 1$ such that the following holds: Let $\mathcal{F}$ be the function family of all functions that are computable by circuits of size $\rho(\lambda)$ (where ρ is the function corresponding to $\epsilon' = \epsilon_2$). Then there exists a (T, T', ω)-lossy correlation intractable hash function family $\mathcal{H}$ for $\mathcal{F}$ (Definition 3.6), where $T = 2^{\lambda^{\epsilon_1}}$, $T' = 2^{\lambda^{\epsilon_2}}$, and $\omega = n - \lambda^{\delta}$, where n is the domain parameter associated with $\mathcal{H}$ (Definition 3.3).

We prove Theorem 7.4 (i.e., prefix-adaptive soundness and unambiguity) w.r.t. this family $\mathcal{H}$, with $\epsilon' = \epsilon_2$ and ϵ such that $2^{\ell^{\epsilon}} \leq 2^{\lambda^{\epsilon_2}}$. Note that since $\lambda \leq \mathsf{poly}(\ell)$, we can take ϵ to be a constant. To this end, for every $\ell \in \mathbb{N}$, fix an ℓ-variate polynomial $g : \mathbb{F}^{\ell} \to \mathbb{F}$ of individual degree $d \leq \mathsf{poly}(\ell)$, where $\mathbb{F} = \mathbb{F}_{\ell}$ is a finite field of size at most $\mathsf{poly}(\ell)$ satisfying Equation (19). Fix any $\mathsf{poly}(2^{\ell^{\epsilon}})$-size cheating prover $P^* = \{P^*_{\ell}\}_{\ell \in \mathbb{N}}$, and suppose for the sake of contradiction that

[34] This assumption is only for the sake of simplicity, and to be consistent with Theorem 7.1.

[35] Note that the written condition captures both prefix-adaptive soundness and prefix-adaptive unambiguity.

there is a non-negligible function η such that for every $\ell \in \mathbb{N}$,

$$\Pr\left[P^*(\mathsf{CRS}) = (j, \sigma_1, \ldots, \sigma_j, I, \{\xi_i\}_{i\in I}, v, \tau_1, \tau_2) : \left\{\begin{array}{ll} & 0 \le j \le \ell \\ \wedge & I \subseteq [j] \\ \wedge & \sigma_i = \mathcal{H}.\mathsf{Hash}(k_i, \xi_i) \ \ \forall i \in I \\ \wedge & \sigma_i \in [0, d] \ \ \forall i \in [j] \setminus I \\ \wedge & V^{g^{(j)}}_{\mathsf{niSC}}(v, \mathsf{CRS}^{(j)}, \tau_1) = 1 \\ \wedge & V^{g^{(j)}}_{\mathsf{niSC}}(v, \mathsf{CRS}^{(j)}, \tau_2) = 1 \\ \wedge & \left[\begin{array}{r} (\tau_1 \neq \tau_2) \ \vee \\ (v \neq \sum\limits_{b_1,\ldots,b_{\ell-j}} g^{(j)}(b_1, \ldots, b_{\ell-j})) \end{array}\right] \end{array}\right\}\right] \ge \eta(2^{\ell^\epsilon}), \tag{20}$$

where the probability is over $\mathsf{CRS} = (k_1, \ldots, k_\ell)$ generated by sampling $k_1, \ldots, k_\ell \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$, for $\lambda \triangleq \log|\mathbb{F}|$.

A standard hybrid argument implies that for every $\ell \in \mathbb{N}$ there exists $j = j(\ell)$ such that Equation (20) holds with this fixed $j = j(\ell)$, with probability $\ge \eta(2^{\ell^\epsilon})/\ell$. Denote this fixed j by $j^* = j^*(\ell)$. The fact that τ_1 and τ_2 are either distinct or are a proof of a false statement implies that if all the conditions in Equation (20) hold, then there must exist $j' \in [1, \ell - j^*]$ and $b \in \{0, 1\}$ such that for $x^* = (g^{(j^*)}, v)$ and $\tau^* = \tau_b$,

$$\mathsf{State}(x^*, \tau^*_{j'-1}) = \mathsf{reject} \ \ \wedge \ \ \mathsf{State}(x^*, \tau^*_{j'}) = \mathsf{accept},$$

where τ^*_i denotes the first i rounds in τ^*.

This, together with a standard hybrid argument, implies that for every $\ell \in \mathbb{N}$ and the fixed $j^* \in [\ell]$, there exists $j' \in [1, \ell - j^*]$ and a $\mathsf{poly}(2^{\ell^\epsilon})$-size cheating prover $P^* = \{P^*_\ell\}_{\ell\in\mathbb{N}}$ such that

$$\Pr\left[P^*(\mathsf{CRS}) = (j, \sigma_1, \ldots, \sigma_j, I, \{\xi_i\}_{i\in I}, v, \tau^*) : \left\{\begin{array}{ll} & j = j^* \\ \wedge & I \subseteq [j] \\ \wedge & \sigma_i = \mathcal{H}.\mathsf{Hash}(k_i, \xi_i) \ \ \forall i \in I \\ \wedge & \sigma_i \in [0, d] \ \ \forall i \in [j] \setminus I \\ \wedge & V^{g^{(j)}}_{\mathsf{niSC}}(v, \mathsf{CRS}^{(j)}, \tau^*) = 1 \\ \wedge & \mathsf{State}(x^*, \tau^*_{j'-1}) = \mathsf{reject} \\ \wedge & \mathsf{State}(x^*, \tau^*_{j'}) = \mathsf{accept} \end{array}\right\}\right] \ge \frac{\eta(2^{\ell^\epsilon})}{2\ell^2}, \tag{21}$$

where the probability is over $\mathsf{CRS} = (k_1, \ldots, k_\ell)$ generated by sampling $k_1, \ldots, k_\ell \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$, for $\lambda \triangleq \log|\mathbb{F}|$.

We next argue that there exists a negligible function μ' such that the equation above holds with probability at least $\frac{\eta(2^{\ell^\epsilon})}{2\ell^2} - \mu'(2^{\ell^\epsilon})$, even if we choose $\mathsf{CRS} = (k_1, \ldots, k_\ell)$ as follows:

$$k_i \leftarrow \mathcal{H}.\mathsf{LossyGen}(1^\lambda) \ \ \forall i \in [j^* + j' - 1] \quad \text{and} \quad k_i \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda) \ \ \forall i \in [j^* + j', \ell].$$

To see this, we will argue that the difference in probabilities when we choose CRS in the two ways is negligible in 2^{ℓ^ϵ}. Since $2^{\ell^\epsilon} \leq T'(\lambda)$, it will suffice to argue that the difference in probabilities is negligible in $T'(\lambda)$. This follows from the T'-key indistinguishability property of the lossy CI hash family, which says that a key generated by $\mathcal{H}.\mathsf{LossyGen}(1^\lambda)$ and a key generated by $\mathcal{H}.\mathsf{Gen}(1^\lambda)$ are indistinguishable by $\mathsf{poly}(T'(\lambda))$-size adversaries, except with probability negligible in $T'(\lambda)$. Note that the condition in the equation above can be checked in time $\mathsf{poly}(T'(\lambda))$, since State can be computed in time $\mathsf{poly}(T'(\lambda))$.

By a standard averaging argument, this implies that for every $\ell \in \mathbb{N}$ there exist (fixed) $k_1, \ldots, k_{j^*+j'-1} \in \mathcal{H}.\mathsf{LossyGen}(1^\lambda)$, where $\lambda = \log|\mathbb{F}|$, and there exists a $\mathsf{poly}(2^{\ell^\epsilon})$-size cheating prover $P^* = \{P^*_\ell\}_{\ell \in \mathbb{N}}$ and a non-negligible function η' such that for every $\ell \in \mathbb{N}$,

$$\Pr\left[\begin{array}{l} P^*(\mathsf{CRS}) = (j, \sigma_1, \ldots, \sigma_j, I, \{\xi_i\}_{i\in I}, v, \tau^*) : \left\{\begin{array}{ll} & j = j^* \\ \wedge & I \subseteq [j] \\ \wedge & \sigma_i = \mathcal{H}.\mathsf{Hash}(k_i, \xi_i) \ \ \forall i \in I \\ \wedge & \sigma_i \in [0,d] \ \ \forall i \in [j] \setminus I \\ \wedge & V^{g^{(j)}}_{\mathsf{niSC}}(v, \mathsf{CRS}^{(j)}, \tau^*) = 1 \\ \wedge & \mathsf{State}(x^*, \tau^*_{j'-1}) = \mathsf{reject} \\ \wedge & \mathsf{State}(x^*, \tau^*_{j'}) = \mathsf{accept} \end{array}\right. \end{array}\right] \geq \eta'(2^{\ell^\epsilon}), \tag{22}$$

where the probability is over $k_{j^*+j'}, \ldots, k_\ell \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$, for $\lambda = \log|\mathbb{F}|$. Here we view $\mathsf{CRS} = (k_1, \ldots, k_\ell)$ where $k_1, \ldots, k_{j^*+j'-1}$ are fixed (as outputs of $\mathcal{H}.\mathsf{LossyGen}(1^\lambda)$).

The fact that $k_1, \ldots, k_{j^*}$ are ω-lossy, together with the fact that $\omega = n - \lambda^\delta$, implies that for each $i \in [j^*]$, there are at most $2^{\lambda^\delta+1}$ possibilities for σ_i (there are $d+1$ possibilities in $[0,d]$ and $\leq 2^{\lambda^\delta}$ values in the image of $\mathcal{H}.\mathsf{Hash}(k_i, \cdot)$, and note that $d+1 \leq 2^{\lambda^\delta}$). Let $(\beta^*_1, \ldots, \beta^*_{j'-1})$ denote the first $j'-1$ verifier messages in τ^*. The fact that $k_{j^*+1}, \ldots, k_{j^*-j'-1}$ are ω-lossy implies that if the verifier accepts τ^*, then for every $i \in [j'-1]$, β^*_i is in the image of $\mathcal{H}.\mathsf{Hash}(k_{j^*+i}, \cdot)$), and hence there are at most 2^{λ^δ} possibilities for each β^*_i.

Thus, we conclude that there are at most $2^{(j^*+j'-1)\cdot\lambda^\delta + j^*} \leq 2^{\ell\lambda^\delta}$ possibilities for

$$(\sigma_1, \ldots, \sigma_{j^*}, \beta^*_1, \ldots, \beta^*_{j'-1}).$$

Therefore, by a standard averaging argument, for every $\lambda \in \mathbb{N}$, there exist $(\sigma_1, \ldots, \sigma_{j^*}, \beta_1, \ldots, \beta_{j'-1})$ such that the probability above holds, w.r.t. these fixed $(\sigma_1, \ldots, \sigma_{j^*}, \beta_1, \ldots, \beta_{j'-1})$, with probability at least $\eta'(2^{\ell^\epsilon}) \cdot 2^{-\ell\lambda^\delta}$. By a similar calculation as was done in Section 5 (see the proof of Corollary 5.3), one can argue that this value is a non-negligible function of $T(\lambda)$, which we will denote by $\zeta(T(\lambda))$.[36]

Note that the fixing of $\sigma_1, \ldots, \sigma_{j^*}$ fixes the $(\ell - j^*)$-variate polynomial $g^{(j^*)}$. In what follows, we denote this polynomial by g^*, and denote by $\beta_{[1,j'-1]} \triangleq (\beta_1, \ldots, \beta_{j'-1})$. Moreover, rather than thinking of λ as a function of ℓ, we think of ℓ as a function of λ. Namely, for every $\lambda \in \mathbb{N}$, if there exists ℓ such that $\lambda = \log|\mathbb{F}_\ell|$, then $\ell(\lambda) = \ell$ (and if there are multiple ℓ's then $\ell(\lambda)$ outputs the

[36] To recap, note that in order to argue that $\eta'(2^{\ell^\epsilon}) \cdot 2^{-\ell\lambda^\delta}$ is a non-negligible function of $T(\lambda) = 2^{\lambda^{\epsilon_1}}$, it suffices to argue that $2^{\ell^\epsilon} \leq 2^{\lambda^{\epsilon_2}} \leq 2^{\lambda^{\epsilon_1}}$, which follows from $\epsilon_2 \leq \epsilon_1$, and that $\lambda^{\epsilon_1} \geq \ell \cdot \lambda^\delta$, which follows from the fact that $\delta = \epsilon_1/2$ and from the fact that $\ell \leq \lambda^{\epsilon_2/2} \leq \lambda^{\epsilon_1/2}$, which in turn follows from Equation (19) together with the assumption that $\epsilon_2 \leq \epsilon_1$.

smallest one), and if $\lambda \neq \log|\mathbb{F}_\ell|$ for every $\ell \in \mathbb{N}$ then we let $\ell(\lambda) = \perp$.) We abuse notation and denote by $P^*_\lambda = P^*_{\ell(\lambda)}$, and if $\ell(\lambda) = \perp$ then P^*_λ simply outputs $\perp$. Thus there exists a $\mathsf{poly}(T'(\lambda))$-size adversary P^* such that for every $\lambda \in \mathbb{N}$,

$$\Pr\left[P^*(\mathsf{CRS}) = (v, \tau^*) \ : \begin{cases} & \beta^*_{[1,j'-1]} = \beta_{[1,j'-1]} \\ \wedge & \mathsf{State}(x^*, \tau^*_{j'-1}) = \mathsf{reject} \\ \wedge & \mathsf{State}(x^*, \tau^*_{j'}) = \mathsf{accept} \end{cases}\right] \geq \zeta(T(\lambda)),$$

where the probability is over $k_{j^*+j'}, \ldots, k_\ell \leftarrow \mathcal{H}.\mathsf{Gen}(1^\lambda)$, and we view $\mathsf{CRS} = (k_{j^*+1}, \ldots, k_\ell)$, where $k_{j^*+1}, \ldots, k_{j^*+j'-1}$ are fixed. We also remind the reader that $\beta^*_{[1,j'-1]}$ denotes the first $j'-1$ verifier messages in τ^*.

Next, we would like to argue that such a prover contradicts correlation intractability of $\mathcal{H}$. To this end, we would like to use Claim 4.8 to derive a contradiction. We note however that Claim 4.8 assumes that the instance $x^* = (g^*, v)$ is fixed (independent of the CRS), whereas in our setting g^* is indeed fixed, but v is chosen adaptively as a function of the CRS.

Despite the fact that a priori it is not clear that Claim 4.8 extends to this setting where v can be chosen adaptively, upon examining its proof, we observe that it does (without any change to the proof). The reason is that the fact that x^* and $\beta^*_{[1,j'-1]}$ are fixed is only used to fix the function $\mathsf{BAD}_{x^*,\beta^*_{[1,j'-1]}}$, which in turn is used to fix a function f_{r^*} that is used to break the T-CI property of the hash function $\mathcal{H}$.

In our setting, since x^* is not fixed, it may seem that the function $\mathsf{BAD}_{x^*,\beta^*_{[1,j'-1]}}$ is not fixed, and therefore it is not clear that we can contradict the T-CI property. We recall, however, that the sum-check protocol has the property that for $x^* = (g^*, v)$, $\mathsf{BAD}_{x^*,\beta^*_{[1,j'-1]}}$ does not depend on v, rather it only depends on g^* and $\beta^*_{[1,j'-1]}$ (see the definition of BAD in Section 5). Therefore, $\mathsf{BAD}_{x^*,\beta^*_{[1,j'-1]}}$ is fixed, and thus the proof of Claim 4.8 goes through as is, implying a contradiction. □

Acknowledgements

This material is based on work supported in part by DARPA under Contract Nos. HR001120C0023 and 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. Rachel Zhang is supported by the Paul E. Gray (1954) UROP Fund.

We would like to thank Alex Lombardi for enlightening discussions on correlation intractability.

References

[ACC+16] Prabhanjan Ananth, Yu-Chi Chen, Kai-Min Chung, Huijia Lin, and Wei-Kai Lin. Delegating RAM computations with adaptive soundness and privacy. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 3–30, 2016.

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

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001.

[BBH+19] James Bartusek, Liron Bronfman, Justin Holmgren, Fermi Ma, and Ron D. Rothblum. On the (in)security of kilian-based snargs. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 522–551. Springer, 2019.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BCC+14] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *IACR Cryptology ePrint Archive*, 2014:580, 2014.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 111–120, 2013.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, pages 315–333, 2013.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, 2016.

[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.

[BG20] Nir Bitansky and Idan Gerichter. On the cryptographic hardness of local search. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 6:1–6:29. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[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.

[BHK17] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 474–482, 2017.

[BK20] Zvika Brakerski and Yael Kalai. Witness indistinguishability for any single-round argument with applications to access control. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 97–123. Springer, 2020.

[BKK+18] Saikrishna Badrinarayanan, Yael Tauman Kalai, Dakshita Khurana, Amit Sahai, and Daniel Wichs. Succinct delegation for low-space non-deterministic computation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 709–721, 2018.

[BKP18] Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: a paradigm for keyless hash functions. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 671–684. ACM, 2018.

[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.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[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+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.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-shamir and correlation intractability from strong kdm-secure encryption. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, pages 91–122, 2018.

[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.

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

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *Proceedings of CRYPTOï¿œ91*, pages 445–456, 1992.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 54–74, 2012.

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. 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 III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2019.

[DGP09] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009.

[EFKP20] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Continuous verifiable delay functions. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 125–154. Springer, 2020.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, pages 102–, 2003.

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

[Gol18] Oded Goldreich. On doubly-efficient interactive proof systems. *Foundations and Trends in Theoretical Computer Science*, 13(3):158–246, 2018.

[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.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340. Springer, 2010.

[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.

[HO12] Brett Hemenway and Rafail Ostrovsky. Extended-ddh and lossy trapdoor functions. In Marc Fischlin, Johannes A. Buchmann, and Mark Manulis, editors, *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*, volume 7293 of *Lecture Notes in Computer Science*, pages 627–643. Springer, 2012.

[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.

[JK20] Ruta Jawale and Dakshita Khurana. Lossy correlation intractability and PPAD hardness from sub-exponential LWE. IACR Cryptology ePrint Archive, 2020/911. https://eprint.iacr.org/2020/911/20200728:183423, 2020.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 723–732. ACM, 1992.

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

[KP16] Yael Tauman Kalai and Omer Paneth. Delegating RAM computations. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 91–118, 2016.

[KPY18] Yael Kalai, Omer Paneth, and Lisa Yang. On publicly verifiable delegation from standard assumptions. *IACR Cryptol. ePrint Arch.*, 2018:776, 2018.

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

[KPY20] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. Ppad-hardness and delegation with unambiguous proofs. CRYPTO, 2020.

[KRR13] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 565–574, 2013.

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *STOC*, pages 485–494. ACM, 2014.

[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.

[KZ20] Yael Tauman Kalai and Rachel Zhang. Snargs for bounded depth computations from sub-exponential LWE. IACR Cryptology ePrint Archive, 2020/860. https://eprint.iacr.org/2020/860/20200712:125111, 2020.

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

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC*, pages 169–189, 2012.

[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.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *Proceedings of the 23rd Annual International Cryptology Conference*, pages 96–109, 2003.

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

[PR17] Omer Paneth and Guy N. Rothblum. On zero-testable homomorphic encryption and publicly verifiable non-interactive arguments. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*, pages 283–315, 2017.

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 422–439, 2012.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 1996.

[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.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 49–62, 2016.

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

[WTS+18] Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zksnarks without trusted setup. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 926–943. IEEE Computer Society, 2018.