Electronic Colloquium on Computational Complexity, Report No. 32 (2021)

Fiat-Shamir via List-Recoverable Codes (or: Parallel Repetition of GMW is not Zero-Knowledge)

Justin Holmgren[*] Alex Lombardi[†] Ron D. Rothblum[‡]

March 5, 2021

Abstract

Shortly after the introduction of zero-knowledge proofs, Goldreich, Micali and Wigderson (CRYPTO '86) demonstrated their wide applicability by constructing zero-knowledge proofs for the NP-complete problem of graph 3-coloring. A long-standing open question has been whether parallel repetition of their protocol preserves zero knowledge. In this work, we answer this question in the negative, assuming a a standard cryptographic assumption (i.e., the hardness of learning with errors (LWE)).

Leveraging a connection observed by Dwork, Naor, Reingold, and Stockmeyer (FOCS '99), our negative result is obtained by making *positive* progress on a related fundamental problem in cryptography: securely instantiating the Fiat-Shamir heuristic for eliminating interaction in public-coin interactive protocols. A recent line of works has shown how to instantiate the heuristic securely, albeit only for a limited class of protocols.

Our main result shows how to instantiate Fiat-Shamir for parallel repetitions of much more general interactive proofs. In particular, we construct hash functions that, assuming LWE, securely realize the Fiat-Shamir transform for the following rich classes of protocols:

1. The parallel repetition of any "commit-and-open" protocol (such as the GMW protocol mentioned above), when a specific (natural) commitment scheme is used. Commit-and-open protocols are a ubiquitous paradigm for constructing general purpose public-coin zero knowledge proofs.
2. The parallel repetition of any base protocol that (1) satisfies a stronger notion of soundness called round-by-round soundness, and (2) has an efficient procedure, using a suitable trapdoor, for recognizing "bad verifier randomness" that would allow the prover to cheat.

Our results are obtained by establishing a new connection between the Fiat-Shamir transform and *list-recoverable codes*. In contrast to the usual focus in coding theory, we focus on a parameter regime in which the input lists are extremely large, but the rate can be small. We give a (probabilistic) construction based on Parvaresh-Vardy codes (FOCS '05) that suffices for our applications.

[*]NTT Research. Email: justin.holmgren@ntt-research.com.

[†]MIT. Email: alexjl@mit.edu. Research conducted in part while the author was an intern at NTT Research. Research supported in part by an NDSEG fellowship. Research supported in part by NSF Grants CNS-1350619 and CNS-1414119, and by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236.

[‡]Technion. Email: rothblum@cs.technion.ac.il. Supported in part by a Milgrom family grant, by the Israeli Science Foundation (Grants No. 1262/18 and 2137/19), and grants from the Technion Hiroshi Fujiwara cyber security research center and Israel cyber directorate.

ISSN 1433-8092

Contents

1 Introduction

Zero-knowledge proofs, introduced by Goldwasser, Micali and Rackoff [GMR85], are a beautifully paradoxical construct. Such proofs allow a prover to convince a verifier that an assertion is true without revealing anything beyond that to the verifier. Soon after the introduction of zero-knowledge proofs, Goldreich, Micali and Wigderson [GMW86] constructed a zero-knowledge proof system (henceforth referred to as the GMW protocol) for the 3-coloring problem. This result is a cornerstone in the development of zero-knowledge proofs, since 3-coloring is **NP**-complete, and so the GMW protocol actually yields zero-knowledge proofs for *any* problem in **NP**.

Roughly speaking, the idea underlying the GMW protocol is for the prover to commit (via a cryptographic commitment scheme) to a random 3-coloring of the graph. The verifier chooses a random edge and the prover decommits to the colors of the two endpoints. Intuitively, the protocol is zero-knowledge since the verifier (even if acting maliciously) knows what to expect: two random different colors. An important point however is that this base protocol has poor soundness. For example, suppose that the input graph $G = (V, E)$ is not 3-colorable, but has a coloring that miscolors only one edge. In such a case, the verifier's probability of detecting the monochromatic edge is only $1/|E|$.

Thankfully, the soundness of the GMW protocol (or any other interactive proof) can be amplified by repetition. That is, in order to reduce the soundness error, one can repeat the base GMW protocol multiple times, either sequentially or in parallel, using independent coin tosses in each repetition (for both parties). At the end of the interaction the verifier accepts if and only if the base verifier accepted in all of the repetitions.

Repetition indeed reduces the soundness error, but does it preserve zero-knowledge? While it is relatively straightforward to argue that *sequential* repetition indeed preserves zero-knowledge [GO94], this yields a protocol with a prohibitively large number of rounds. Thus, a major question in the field is whether *parallel* repetition also preserves zero-knowledge.[1]

Curiously, it has long been known that parallel repetition does not preserve zero-knowledge for some (contrived) protocols [GK96]. However, for "naturally occurring" protocols, the question remained open for decades. A sequence of recent works [KRR17, CCRR18, HL18, CCH+18] showed that zero-knowledge is not preserved by repetition in very high generality (in fact, general 3-message zero-knowledge proofs can be ruled out [FGJ18]), but these works relied on extremely strong, non-falsifiable, and/or poorly understood cryptographic assumptions. The first progress on this question *based on standard assumptions* was due to Canetti *et al.* [CCH+19] and Peikert and Shiehian [PS19], who showed that some *classical* ZK protocols [GMR85, Blu86] fail to remain ZK under parallel repetition. However, their results conspicuously fail to capture the GMW protocol (and indeed fail to capture "most" protocols). Thus, an answer to the following basic question has remained elusive for over 30 years [DNRS99, BLV03]:

Does parallel repetition of the GMW *protocol preserve zero-knowledge*
(under standard cryptographic assumptions)?

As one of our main results, we answer this question in the negative, assuming the hardness of learning with errors (LWE) [Reg03].

[1] In particular, a positive resolution of this question would yield 3-message zero-knowledge proofs for all of **NP** (assuming also non-interactive commitments), thereby settling the long-standing open problem of the round complexity of zero-knowledge proofs.

Theorem 1.1 (Informally Stated, see Theorem 5.13)**.** *Assume that* LWE *holds. Then, there exists a commitment scheme C (in the common random string model) and a polynomial t such that t-fold parallel repetition of the* GMW *protocol (using C as its commitment scheme) is not zero-knowledge.*

We briefly make two remarks on Theorem 1.1:

- The commitment scheme C used in order to prove Theorem 1.1 is a natural one.[2] The common random string consists of a public-key of an encryption scheme (which if using a suitable encryption scheme can simply be a uniformly random string). One commits by simply encrypting messages and decommits by revealing the randomness used in the encryption.

  Still, we point out that Theorem 1.1 leaves open the possibility that parallel repetition of GMW is zero-knowledge when instantiated with a specially tailored commitment scheme.

- The number of repetitions t for which we can show that the t-fold parallel repetition of GMW has $\mathsf{negl}(\lambda)$ soundness error, but is not zero knowledge, is $|E(G)| \cdot \lambda$, where $|E(G)|$ denotes the number of edges in the graph and λ is a security parameter. However, we still leave open a (very) small window of possible values for t so that the t-fold repetition of GMW is both sound and zero-knowledge (see Remark 5.14 for further discussion).

  The number of repetitions t for which we can show that the t-fold parallel repetition of GMW has $\mathsf{negl}(n)$ soundness error, but is not zero knowledge, is $|E(G)| \cdot n^{\epsilon}$ for any $\epsilon > 0$, where $|E(G)|$ denotes the number of edges in the graph. Under the subexponential LWE assumption, the n^{ϵ} factor can be reduced to $\log^{c} n$ for some $c > 1$. This still leaves open a (very) small window of possible values for t so that the t-fold repetition of GMW is both sound and zero-knowledge (see Remark 5.14 for further discussion).

We prove Theorem 1.1 through a more general result showing that parallel repetition does not preserve zero-knowledge for a large class of protocols. This class includes all general-purpose public-coin[3] zero-knowledge proofs for **NP** that we are aware of (when instantiated with a specific commitment scheme). In particular, this includes protocols based on the influential MPC-in-the-head paradigm [IKOS07] and more generally based on zero-knowledge PCPs (see, e.g., a recent survey [Ish20]).

All of the above negative results are shown by making *positive* progress on the closely related question of soundly instantiating the prolific Fiat-Shamir heuristic, which is our main focus, and is discussed next.

1.1 Securely Instantiating Fiat-Shamir

The Fiat-Shamir heuristic [FS86] is a generic technique for eliminating interaction in *public-coin* interactive proofs.[4] This technique has been extremely influential both in practice and in theory.

Consider for example a 3-message public-coin interactive proof that $x \in L$. In such a protocol first the prover sends a message α, the verifier responds with random coins β and finally the prover sends the last message γ. The basic idea underlying the Fiat-Shamir heuristic is to replace the

[2] In fact, this instantiation dates back to the original [GMW86] paper.

[3] Recall that an interactive proof is *public-coin* if all the verifier does throughout the interaction is simply toss random coins and immediately reveal them to the prover.

[4] The original goal in [FS86] was to efficiently compile (interactive) identification schemes into signature schemes, but the technique is applicable to more general protocols.

random coin tosses β of the verifier by applying a hash function to the the transcript thus far, i.e., by setting $\beta = h(x, \alpha)$. Since the prover can now compute the verifier's coin tosses, the entire interaction consists of having the prover send the message (α, β, γ) in one shot.

It has been long known that the Fiat-Shamir heuristic is sound when the hash function is modeled as a *random oracle* [BR94, PS96, BCS16]. In reality however, we need to realize the hash function with a concrete cryptographic hash function. Following [CCH$^+$19], we say that a hash function family $\mathcal{H}$ is FS-$\mathsf{compatible}$[5] with a (public-coin) interactive protocol Π, if applying the Fiat-Shamir transform to Π, with a random choice of $h \in \mathcal{H}$, yields a computationally sound argument system. A central problem in cryptography is to construct FS-compatible hash functions for a variety of interactive protocols of interest, thereby making them non-interactive.

While designing FS-compatible hash function families is an extremely important goal in its own right, Dwork, Naor, Reingold, and Stockmeyer [DNRS99] also showed that the existence of an FS-compatible hash function family for a (public-coin) interactive proof Π for a language $L \notin \mathbf{BPP}$, is *equivalent* to Π *not* being zero-knowledge.[6] This means, in particular, that in order to prove Theorem 1.1, it suffices to construct an FS-compatible hash function for the GMW protocol.

For a long time almost all results on instantiating Fiat-Shamir were negative [CGH98, Bar01, GK03, BDG$^+$13]. However, a recent line of work [KRR17, CCRR18, HL18, CCH$^+$19, PS19, BKM20, LV20a, JKKZ20] has made substantial *positive* progress, culminating in secure realizations of Fiat-Shamir in certain (important) cases, based on standard cryptographic assumptions.

In particular, a combination of the results of [CCH$^+$19, PS19] implies the existence of hash functions, based on LWE, that are FS-compatible for a certain class of interactive proofs. More specifically (and restricting our attention to three message protocols), this class contains interactive proofs, in the CRS model, in which for every $x \notin L$ and first prover message α, the number of random coins β that could lead the verifier to accept is polynomially bounded, and moreover, there is an efficient algorithm that finds these "bad" β's (given x, α and possibly a trapdoor associated with the CRS).

Fortunately, a natural variant of Blum's [Blu86] zero-knowledge protocol for Hamiltonicity has the above property. This is due to the fact that Blum's protocol is obtained by applying parallel repetition to a base protocol which has only a *single* choice of bad randomness. Since $1^t = 1$, the number of bad random choices when the base protocol is repeated is still 1 (and this unique bad randomness can be efficiently found). Since Hamiltonicity is $\mathbf{NP}$-complete, the works of [CCH$^+$19, PS19] yielded *non-interactive* zero-knowledge[7] proof-systems for all of $\mathbf{NP}$.

While the base GMW protocol has a polynomial number of bad random strings (after all, even the *total* number of verifier random strings is polynomial), in contrast to Blum's protocol, when the protocol is repeated, this number becomes *exponential*. This means that the approach of [CCH$^+$19, PS19] no longer applies. A similar problem occurs for the parallel repetition of any base protocol with more than a single bad random choice for the verifier, which is extremely common.

We emphasize that the interest in these additional zero-knowledge protocols is not purely theo-

[5]We remark that the term "FS-compatible" has a different meaning in a recent work of [JKKZ20]. More specifically, [JKKZ20] defines "FS-compatibilty" to be a property of a *protocol* Π; their property consists of technical conditions that suffice for their specific hash family to instantiate FS for Π.

[6]Roughly speaking, [DNRS99] consider a malicious verifier that answers according to the Fiat-Shamir hash function. They show that a successful simulation of such a verifier can be used to decide the language.

[7]In contrast to the discussion in the beginning of the introduction, in the context of applying Fiat-Shamir positively in order to construct *non-interactive zero-knowledge proofs*, it suffices that the base interactive proof be *honest-verifier* zero-knowledge. Honest-verifier is indeed known to be preserved under parallel repetition.

retical. In particular, some of the most efficient zero-knowledge proof-systems, such as those based on the MPC-in-the-head paradigm, also do not have a polynomial set of bad randomnesses and consequently the techniques of [CCH+19, PS19] are not applicable to them.

Fiat-Shamir for Commit-and-Open Protocols. Our second main result shows how to securely realize the Fiat-Shamir transformation when applied to a much broader class of interactive proofs than what was known before (including the GMW protocol). More specifically, this class consists of the "parallel repetition of any commit-and-open protocol". By a commit-and-open protocol, we basically refer to protocols that have the following structure:

1. P commits to a string w.
2. V samples random coins r and sends them to P. These random coins, together with the main input x, specify a subset S of indices of w.
3. P decommits to w_S and V accepts or rejects based on some predicate $V(x, r, w_S)$.

Note that the GMW protocol indeed fits into this framework: w is a (random) 3-coloring of the graph, the set S specifies a random edge and V simply checks that the edge is properly colored.

Theorem 1.2 (Informally Stated, see Theorem 5.12)**.** *Assume that* LWE *holds. Then, there exists a commitment scheme C (in the* CRS *model), such that for every commit-and-open protocol Π_C there exists a polynomial t and a hash function family $\mathcal{H}$, such that the hash family $\mathcal{H}$ is* FS*-compatible with the t-fold parallel repetition $(\Pi_C)^t$ of Π_C.*

By the connection established by [DNRS99], Theorem 1.1 follows immediately from Theorem 1.2.

Remark 1.3. *An important example of a commit-and-open protocol is Kilian's [Kil92] celebrated succinct argument-system, as well as its generalizations based on interactive oracle proofs [BCS16]. However, we point out that Theorem 1.2 is not applicable to this protocol since Kilian relies on a particular* succinct *commitment scheme (based on Merkle hashing), whereas the commitment scheme C that we use is inherently non-succinct.*

Indeed, the question of securely applying Fiat-Shamir to Kilian's protocol (as envisioned by Micali [Mic93]), remains a fundamental open problem (see also [GW11, BBH+19]).

Because it applies to parallel repetitions of *all* commit-and-open protocols (rather than just those with a single bad challenge), Theorem 1.2 substantially generalizes the class of protocols that have sound Fiat-Shamir instantiations in the standard model. We believe that Theorem 1.2 (and the techniques underlying its proof) are likely to lead to new feasibility results for non-interactive cryptographic protocols in the standard model.

Fiat-Shamir for Parallel Repetition of Multi-Round Protocols. We next turn to discuss our results for *multi-round* protocols. Let Π be a public-coin multi-round interactive proof system. As above, the application of Fiat-Shamir to such a protocol simply replaces the verifier's random coin tosses in each round with a hash of the entire transcript up to that point.

When considering protocols with a large number of rounds, some care must be taken. For example, if we take the *sequential* repetition of (say) the GMW protocol and try to apply Fiat-Shamir, it is not too difficult to see that the resulting non-interactive protocol is not sound *regardless* *of the Fiat-Shamir hash function* (e.g., even if the hash function is modeled as a random oracle). The issue is that after the compilation, the cheating prover can effectively "rewind" the verifier to a previous state (see [BCS16] for more details).

Thus, following [CCH+19], we restrict our attention to protocols satisfying a stronger soundness condition called *round-by-round soundness.* Loosely speaking, a protocol is round-by-round (RBR) sound, if soundness holds in each round individually. In more detail, RBR soundness dictates the existence of a predicate State (which need not be efficiently computable) mapping partial transcripts to the set $\{\mathsf{accept}, \mathsf{reject}\}$ such that:

1. If $x \notin L$ then the State of the empty transcript is rejecting.

2. Given a rejecting partial transcript τ and any prover message α, with all but negligible probability over the verifier's next coin tosses β, the partial transcript $(\tau|\alpha|\beta)$ is also rejecting (where '$|$' denotes concatenation).

3. The verifier always rejects *full* rejecting transcripts.

Note that round-by-round soundness implies standard soundness: the protocol starts off in a rejecting state and, with high probability, will remain so until the very end in which case the verifier is required to reject. Prototypical examples of protocols satisfying round-by-round soundness include the *sumcheck protocol* [LFKN90] and the related [GKR08] protocol (see [CCH+19, JKKZ20] for details).

We say that a protocol with RBR soundness has efficiently recognizable bad randomness if given a *rejecting* partial transcript $\tau|\alpha$, ending with a prover message α, the set of verifier coins β that make $(\tau|\alpha|\beta)$ turn into an *accepting* partial transcript is efficiently recognizable (potentially also given access to a trapdoor of a CRS, if such exists).

The works [CCH+19, PS19] imply LWE-based FS-compatible hash functions for interactive proofs with *negligible* RBR soundness error in which the bad randomness is not just efficiently recognizable, but moreover the set is efficiently *enumerable* (i.e., the set of bad randomness is polynomially bounded and can be explicitly generated in polynomial time). We extend their result to protocols obtained by taking parallel repetition of an r-round base protocol with RBR soundness error *close to* $1/r$, and without any constraint on the number of choices of bad randomness.

Theorem 1.4 (Informally Stated, see Theorem 6.13)**.** *Let* Π *be a* $2r+1$*-message interactive proof with round-by-round soundness error* $\frac{1-\epsilon}{r}$ *with efficiently reconizeable bad randomness. Then, there exists a polynomial* $t = t(n, \lambda, \epsilon)$*, and a hash family* $\mathcal{H}$*, such that* $\mathcal{H}$ *is* FS*-compatible with* Π^t*.*

Remark 1.5. *Theorem 1.2 actually follows from Theorem 1.4 since constant-round protocols with negligible soundness are automatically round-by-round sound, and the specific type of commitment scheme makes the bad randomnesses efficiently computable.*

However, we set apart these two results for two reasons. First, the proof of Theorem 1.2 is simpler than that of Theorem 1.4 and suffices for many protcols of interest. Second, we are unable to achieve a tight result with respect to the number of repetitions in Theorem 1.4 as we did for Theorem 1.2.

Finally, we note that Theorem 1.4 can be combined with the main insight of [JKKZ20] (which is orthogonal to our work) to *further* generalize the class of protocols Π that have sound Fiat-Shamir instantiations. Informally, the [JKKZ20] technique of *lossy* correlation intractability allows us to additionally handle protocols where bad challenges for the i-th round can only be efficiently recognized given non-uniform advice about the *previous* rounds' challenges. For example, this allows us to instantiate Fiat-Shamir for parallel repetitions of the [GKR08] protocol, even when the field size of the base protocol is *poly-logarithmic*. In contrast, [JKKZ20] can only handle variants of [GKR08] with an exponential field size.[8]

1.2 Technical Overview

We now describe our techniques for proving Theorem 1.2, with a particular focus on the GMW protocol for ease of understanding. Our starting point is the work of [CCH+19], which gave the first instantiation of Fiat-Shamir in the standard model based on standard cryptographic assumptions. As in prior work [KRR17, CCRR18, HL18], their Fiat-Shamir instantiation makes use of the framework of *correlation intractability* [CGH98], which we recall here.[9]

A hash family $\mathcal{H}$ is said to be (single input) correlation-intractable for a binary relation R if it is computationally hard, given a hash key $h \leftarrow \mathcal{H}$, to find a "correlation", i.e., an input x such that $(x, h(x)) \in R$. Such a security property is plausibly instantiable, and is satisfied by a random oracle, whenever the relation R is *sparse*, meaning that for any input x, the fraction of outputs y for which $(x, y) \in R$ is negligible.

Despite this plausibility argument, and despite the intriguing connection to Fiat-Shamir in the standard model (which we will see in a moment), there were essentially no instantiations of correlation intractability (beyond very simple relations such as those for which $(x, y) \in R$ if and only if $y = c$ for a constant c) before 2016. However, a flurry of recent works (including [CCR16, KRR17, CCRR18, HL18, CCH+19, PS19, LVW19, BFJ+20, GJJM20, LNPT19, BKM20, LV20a, JKKZ20, LNPY20, LV20b]) have (1) instantiated various flavors of correlation-intractable hash functions based on plausible cryptographic assumptions and (2) applied these hash functions to achieve independently useful cryptographic goals.

We discuss this line of work in detail in Section 1.4, but for now, we recall the following result from [PS19], which is most relevant for our purposes. It is a construction of correlation intractability for *functions*: we say that $\mathcal{H}$ is CI for a function f if it is CI for the relation $R_f = \{(x, f(x))\}$.

Theorem 1.6 ([PS19], informal)**.** *Under the* LWE *assumption, there exists a hash family* $\mathcal{H}$ *that is correlation intractable for all functions that are computable in (a priori bounded) polynomial time.*

As described in the theorem statement, Theorem 1.6 has the following two limitations (which are also present in the predecessor work [CCH+19][10]).

- They only achieve security for relations $R \subseteq X \times Y$ that represent *functions*. That is, for every $x \in X$ there is (at most) a single $y \in Y$ such that $(x, y) \in R$.
- They require that the functions are *efficiently computable*.

[8] [JKKZ20] use a large field in order to avoid parallel repetition. For example, this precludes applications in which one needs to materialize entire truth tables of polynomials over the field.

[9] In fact, [DNRS99] cites personal communication with Chaum and Impagliazzo for an early variant of this connection. Full formalizations of this paradigm appear in [CCRR18, CCH+19].

[10] More specifically, this limitation is present in the subset of results in [CCH+19] that are based on quantitatively standard cryptographic assumptions

Both of these drawbacks turn out to be relevant for Fiat-Shamir instantiations. To see this, we first discuss how CI relates to the instantiation of Fiat-Shamir for interactive proofs. For simplicity, we focus on the task of compiling 3-message public coin interactive proofs. Such protocols have the following syntax.

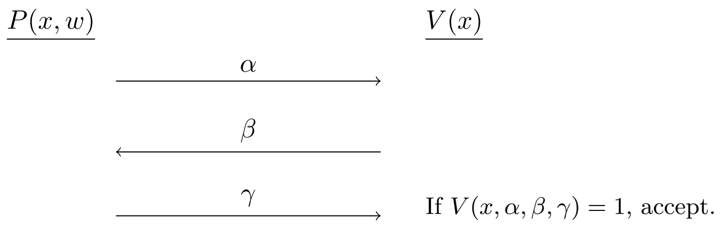

Figure 1: A 3-message public coin interactive proof Π.

After applying the Fiat-Shamir transform using hash family $\mathcal{H}$, we obtain the protocol $\Pi_{\mathrm{FS},\mathcal{H}}$ below.

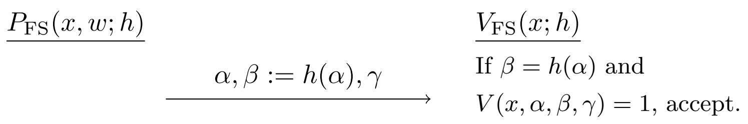
$P_{\mathrm{FS}}(x, w; h)$
$V_{\mathrm{FS}}(x; h)$
$\alpha, \beta := h(\alpha), \gamma$
If $\beta = h(\alpha)$ and
$V(x, \alpha, \beta, \gamma) = 1$, accept.

Figure 2: The Protocol $\Pi_{\mathrm{FS},\mathcal{H}}$.

In this situation, consider the following relation $R^{(0)} = R^{(0)}_{x,\Pi}$ for a false statement x, which we call the (naive) bad-challenge relation for Π:

$$R^{(0)}_{x,\Pi} = \{(\alpha, \beta) : \exists \gamma \text{ s.t. } V(x, \alpha, \beta, \gamma) = 1\}.$$

It follows almost syntactically that if $\mathcal{H}$ is CI for $R^{(0)}_{x,\Pi}$ (for all false statements x), then $\mathcal{H}$ soundly instantiates Fiat-Shamir for Π. Thus, the problem of instantiating Fiat-Shamir is reduced to constructing sufficiently general-purpose correlation intractable hash functions. Bearing in mind the two drawbacks of Theorem 1.6, it is worth noting that $R_{x,\Pi}$ is (in general) not even a function, let alone an efficiently computable one.

Fiat-Shamir for GMW**.** With the above background in mind, we turn to the task at hand: finding a Fiat-Shamir instantiation for the parallel repeated GMW protocol. Abstractly, a t-wise parallel repetition of a protocol Π has the following syntax.

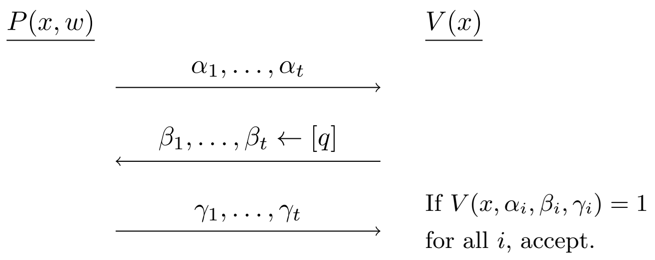
$P(x, w)$
$V(x)$
$\alpha_1, \ldots, \alpha_t$
$\beta_1, \ldots, \beta_t \leftarrow [q]$
$\gamma_1, \ldots, \gamma_t$
If $V(x, \alpha_i, \beta_i, \gamma_i) = 1$
for all i, accept.

Figure 3: A parallel-repeated protocol Π^t.

In the case of GMW, the input x is a graph $G = (V, E)$, the witness w is a 3-coloring of G, the messages α_i are commitments to (a random shuffling of the colors of) w, each $\beta_i = (u_i, v_i) \in E(G)$ specifies a randomly selected edge, and the γ_i are decommitments[11] (z_i, r_i) to the colors $z_i = (w(u_i), w(v_i))$. The verification procedure checks that the decommitments are all valid and that each (revealed) colored edge is not monochromatic. Note that the "alphabet size" q denotes the size of the the verifier's challenge space, which in this case is $q = |E|$.[12]

Recall that by Theorem 1.6, we would be done if (1) the relation $R^{(0)} = R^{(0)}_{x,\Pi^t}$ above represented a function f, and (2) the function f were efficiently computable. As a first step, we show (following [HL18, CCH⁺19]) how to replace the relation $R^{(0)}$ with a relation R that is *efficiently verifiable*, i.e., there is an efficient algorithm that *recognizes* bad challenges.

In a nutshell, the "commit-and-open" structure of the GMW protocol allows us to replace the "naive bad-challenge relation" $R^{(0)}_{x,\Pi^t}$ with the relation

$$R_{x,\Pi^t} := \Big\{ ((\alpha_1, \ldots, \alpha_t), (\beta_1, \ldots, \beta_t)) : \text{ each } z_i := \mathsf{Extract}(\alpha_i[\beta_i]) \text{ has two distinct colors} \Big\},$$

where $\mathsf{Extract}$ denotes a function that extracts a committed bit b from a commitment com. In other words, the relation $R_{x,\Pi}(\alpha, \beta)$ can be verified by extracting from $\alpha[\beta]$ the appropriate committed string z and then checking whether the two colors defined by z are distinct. If the commitment scheme is efficiently extractable (given a trapdoor; e.g., this holds if Com is the encryption algorithm of a public-key encryption scheme), then R_{x,Π^t} can be efficiently verified. Thus, to instantiate Fiat-Shamir for this (natural) instantiation of the GMW protocol, it suffices to construct a hash family $\mathcal{H}$ that is CI for this particular (efficiently verifiable) relation R_{x,Π^t}.

The Problem: Too Many Bad Challenges. The main barrier to instantiating Fiat-Shamir for GMW is due to the *first* drawback of the [CCH⁺19, PS19] results, namely, that R is *not* a function. We quantify the extent to which R is not a function with the following terminology.

Definition 1.7 (d-Bounded Relation)**.** *We say that a relation $R \subseteq \{0,1\}^n \times \{0,1\}^m$ is $d = d(n)$-* bounded *if $|R(x)| \leq d$, for all $x \in \{0,1\}^n$, where $R(x) = \{y \in \{0,1\}^m : (x, y) \in R\}$.*

We focus on *absolute* rather than *relative* boundedness (aka density) due to the limitations of prior work on instantiating correlation intractability. In particular, the CI hash families of [CCH⁺19, PS19] were shown to satisfy correlation intractability for (efficiently computable) *functions*, i.e., 1-bounded relations. In prior work [CCH⁺19, JKKZ20], CI for relations that are *not* functions was only achieved in a very limited sense: for d-bounded relations R, it is noted that a hash family $\mathcal{H}$ that is CI for efficiently computable functions *with $\frac{1}{d}$ quantitative security* is also CI for d-bounded relations that are "efficiently enumerable".[13] This is proved via a trivial "guessing" reduction from CI for functions with a security loss of $\frac{1}{d}$. In prior works, only polynomial (or slightly superpolynomial) values of d were considered for this reason.

[11] A decommitment (m, r) of a string com is a message m and choice of commitment randomness r such that $\mathsf{com} = \mathsf{Com}(m; r)$.

[12] Our results in this overview may appear to require that q is polynomial in n, but we show in Section 3.2 how to reduce from general q to polynomial-size q via *subsampling*. This allows us to handle Fiat-Shamir for parallel repetitions of arbitrary commit-and-open protocols.

[13] A d-bounded relation R is *efficiently enumerable* if there is an efficient algorithm that, on input x, explicitly generates the set of all y such that $(x, y) \in R$.

However, in the case of parallel repeated GMW, the relation $R = R_{x,\Pi^t}$ may be only $(|E(G)| - 1)^t$-bounded. In other words, for every $\alpha = (\alpha_1, \dots, \alpha_t)$, there may be $(|E(G)| - 1)^t$ challenges β such that $(\alpha, \beta) \in R_{x,\Pi^t}$. As a result, the "guessing reduction" above incurs a security loss that is exponential in the security parameter, resulting in a useless reduction. Achieving CI for d-bounded relations for *large* values of d – and instantiating Fiat-Shamir for protocols with *many* bad challenges – was an unsolved problem.

Main Idea: Derandomization. Our high-level idea for resolving this problem is using *derandomization* to reduce the *effective* d-boundedness of the relation R. Namely, we employ a two-step process.

1. Devise a randomness-efficient procedure for sampling challenges $(\beta_1, \dots, \beta_t) \leftarrow \mathsf{Samp}(r)$ such that only *polynomially* many bad choices of r lead to bad challenges (for any given pair (x, α)). Note that we need to do so while maintaining *negligible* soundness error. That is, we want the set of bad challenges to have *absolute* size that is polynomial, while its *relative* size (or density) is negligible.

2. Compose the sampling procedure with a hash family $\mathcal{H}_{\text{inner}}$ that is CI for polynomially-bounded relations. In particular, $\mathcal{H}_{\text{inner}}$ must satisfy CI for a new relation $\tilde{R} := \tilde{R}_{x,\Pi,\mathsf{Samp}}$ that depends on the procedure Samp as well as Π.

This process yields a correlation-intractable hash family for R by a natural composition. Namely, our hash family will consist of hash functions h' defined as

$$h'(x) = \mathsf{Samp}(h(x))$$

where $h \leftarrow \mathcal{H}_{\text{inner}}$ comes from a previously constructed CI hash family (namely, the families from [CCH+19, PS19]).

Another interpretation of our approach is that we instantiate Fiat-Shamir for a (parallel repeated) protocol Π^t by implicitly working with a *derandomized parallel repetition*[14] of Π.

Still, several crucial details remain unclear from this outline:

- How should we instantiate the sampling procedure Samp?
- How do we prove that the resulting hash family $\mathcal{H}'$ is FS-compatible for Π^t?

Indeed, standard derandomization techniques such as expander walks and pseudorandom generators turn out *not* to suffice for our application, as we elaborate below. Instead, we need a *new derandomization technique*: our main technical contribution is a special-purpose instantiation of Samp and proof of security for $\mathcal{H}'$.

[14]The type of derandomization that we require is related to, but different from, the "sampler-based" [Gol11, Vad12] derandomized parallel repetition of Bellare, Goldreich and Goldwasser [BGG90]. The exact approach of [BGG90] does not work for us for reasons similar to the "naive" PRG approach below.

Naive Idea: Use a PRG. As a first (flawed) attempt to solve our problem, one might consider setting $\mathsf{Samp}(r) = G(r)$ for some pseudorandom generator G (either cryptographic [BM82] or "Nisan-Wigderson style" [NW88, IW97, AK97]; indeed, the PRG would only have to fool a specific test related to Π). We briefly describe why this approach fails:

- **The new relation $\tilde{R}$ is *still* not bounded enough**. To understand this point, we need to specify what tests the PRG G has to fool. By staring at the problem, we see that G should have the property that for every statement x and first messages $\alpha_1, \ldots, \alpha_t$, the probability that $G(r) = (\beta_1, \ldots, \beta_t)$ has the property that $(\alpha, G(r)) \in R_{x,\Pi}$ is close to the sparsity of R. Unfortunately, known PRG constructions still have the property that the *absolute* number of such "bad r" is exponential in the seed length,[15] while we need this number to be polynomial in the seed length.

- **The new relation $\tilde{R}$ is not efficiently enumerable**. On top of parameter issues, the relation $\tilde{R}$ constructed in step (2) above seems hard to compute, because it syntactically requires computing preimages (of exponential-size sets!) under the map G. Indeed, the relation $\tilde{R}$ has the form:

$$\tilde{R}_{x,\Pi,G} = \left\{(\alpha, r) : (\alpha, G(r)) \in R_{x,\Pi^t}\right\},$$

  so the set of all r such that $(\alpha, r) \in \tilde{R}_x$ is $G^{-1}(\{\beta : (\alpha, \beta) \in R_x\})$. Since $\tilde{R}$ does not seem to be efficiently enumerable, we do not know how to construct a CI hash family for it.

Our Code-Based Derandomization. Since the naive idea of using a PRG for derandomization fails, we now study our special-purpose derandomization problem in more detail. In particular, we crucially take advantage of the *parallel repetition structure* of the relation R_{x,Π^t} to reframe the problem.

As above, our plan is to use some function $\mathsf{Samp}(r) \to (\beta_1, \ldots, \beta_t)$ along with a hash family $\mathcal{H}$ that is correlation intractable for the relation $\tilde{R}$, which can be expressed as

$$\tilde{R}_{x,\Pi^t,\mathsf{Samp}} = \Big\{(\alpha, r) : (\alpha_i, \mathsf{Samp}(r)_i) \in R_{x,\Pi} \text{ for all } i\Big\}.$$

Moreover, for each fixed pair (x, α_i), we know that the collection S_i of all β_i such that $(\alpha_i, \beta_i) \in R_{x,\Pi}$ is *not too large*: if the protocol Π has soundness error $1 - \epsilon$ (meaning that cheating provers are caught with probability ϵ; in the case of GMW, we have $\epsilon = \frac{1}{|E(G)|}$), then $|S_i| \le (1 - \epsilon)q$ for all i (recall that q denotes the verifier's challenge space in the base protocol).

More abstractly, we are interested in relations of the form

$$\tilde{R}_{x,\Pi^t,\mathsf{Samp}} = \Big\{(\alpha, r) : \mathsf{Samp}(r)_i \in S_i \text{ for all } i\Big\},$$

where:

- Each set $S_i \subseteq [q]$ is promised to have some bounded size $|S_i| \le (1 - \epsilon)q$,

[15]This boils down to the suboptimal ϵ-dependence of the seed length of known PRGs that are ϵ-pseudorandom. In order for the number of "bad r" to be polynomial, we would need a PRG with seed length $O(\log m) + \log(1/\epsilon)$ – that is, we cannot afford any constant $c > 1$ in front of the $\log(1/\epsilon)$ term).

- Each set S_i can be efficiently computed from (x, α). (This property is guaranteed by the efficient verifiability of R).

Since our hope is to use $\mathcal{H}$ from [CCH+19, PS19] – which is only CI for *efficiently enumerable* relations – we have two strong demands of the procedure $(\beta_1, \ldots, \beta_t) \leftarrow \mathsf{Samp}(r)$:

- For all x and all α, the number of r such that $\mathsf{Samp}(r) \in S_1 \times \ldots \times S_t$ should be *polynomial* in the length of r.
- Moreover, the (polynomial-size) set of all such r should be be efficiently computable given (x, α) (or, essentially equivalently, the sets $S_1, \ldots S_t$).

Almost miraculously, if we think of our sampler Samp as the encoding procedure Encode of an error-correcting code, this set of requirements *exactly corresponds* to an important notion in coding theory: (errorless) list recovery [GI01]!

We now (informally) recall the definition of an (error-free) list-recoverable code. Let $\mathsf{Encode} : \{0,1\}^\lambda \to [q]^t$ denote an efficient encoding procedure. We say that $(\mathsf{Encode}, \mathsf{Recover})$ is a (ℓ, L)-list recoverable code if

- For all sets (called $\mathsf{input\ lists}$) $S_1, \ldots, S_t$ of size at most ℓ, the number of messages $m \in \{0,1\}^\lambda$ such that $\mathsf{Encode}(m) \in S_1 \times \ldots \times S_t$ is at most L, and
- The algorithm $\mathsf{Recover}(S_1, \ldots, S_t)$, given descriptions of the input lists $S_1, \ldots, S_t$, efficiently returns the $\le L$ corresponding messages (called the $\mathsf{output\ list}$).

List-recoverable codes were introduced by [GI01] as a tool for constructing more efficient list-decodable codes. For our application, we define $\mathsf{Samp}(r) := \mathsf{Encode}(r) \in [q]^t$, so that

- The *alphabet* q of the code is exactly the challenge space for the base protocol Π.
- The *block-length* t of the code is the *number of repetitions* of the protocol Π,
- The *input list* size $\ell = (1-\epsilon)q$ corresponds to the *boundedness* of the relation R_Π, and
- The *output list* size L is a bound on the number of seeds r that are mapped to bad challenges, and so should be some polynomial in the security parameter λ.[16]

We emphasize that the parameter regime we are interested in is *qualitatively different* than is typical in coding theory. In the coding theory literature (see [HW15, Figure 1] as well as [RW18] for examples), the input list size ℓ is typically very small[17] compared to the alphabet size q, while the parameters they want to optimize are the block-length t (ideally $t = O(\lambda)$), as well as the output list size L (which is important for efficient decoding when the list-recoverable code is used as a component in a larger construction).

On the other hand, our setting has a very large value of ℓ (potentially as high as $(1-\epsilon)q$); we then want to optimize for the block-length t, which is ideally not much larger than $1/\epsilon$, but

[16]The dependence is actually allowed to be $\mathsf{poly}(\lambda, q, 1/\epsilon)$

[17]For example, degree k Reed-Solomon codes over $\mathbf{F}_q$ can handle $\ell \le \frac{q}{k}$, while known higher rate constructions can only tolerate much smaller values of ℓ.

multiplicative factors of $\mathrm{poly}(\lambda)$ do not really bother us (in particular, the code can have rate $o(1)$). Meanwhile, the output list size L is not too important for us (as long as it is polynomial), but it is crucial that list-recovery is computationally efficient (rather than information-theoretic), which differs from many prior works.

As described above, there is a tight connection between list-recoverable codes and correlation-intractable hash families through the construction $h'(x) = \mathsf{Encode}(h(x))$:

Theorem 1.8 (Informally stated, see Theorem 3.7)**.** *Suppose that*

- *$\mathcal{H}$ is a hash family that is CI for efficient functions,*
- *$R = R_{x,\Pi}$ is an* efficiently verifiable *relation with output space $[q]$ and sparsity $1-\epsilon$, and*
- *$(\mathsf{Encode}, \mathsf{Recover})$ is a $((1-\epsilon)q, L)$-list recoverable code mapping $\{0,1\}^\lambda \to [q]^t$.*

Then, the hash family defined by $h'(x) = \mathsf{Encode}(h(x))$ is CI for the relation R_{x,Π^t}, and is therefore FS*-compatible with the protocol Π^t.*

In Section 3.1, we rephrase Theorem 1.8 fully in the language of correlation intractability (without reference to any protocol Π) by defining a natural notion of "product relation". We then show that list-recoverable codes can be used to generically construct CI for product relations from CI for functions. Then, in Sections 5 and 6, we show how this form of CI allows us to prove our general FS results: Theorem 1.2 and Theorem 1.4. For the generalization to many-round protocols, we in fact make use of *error-tolerant* (rather than error-free) list-recoverable codes.

Final Step: Constructing the Codes. However, an important question remains: do there actually exist codes satisfying all of the properties that we need? To summarize (for the case of 3-message protocols), we want the following conditions to hold for a code defined by $\mathsf{Encode} : \{0,1\}^\lambda \to [q]^t$.

1. The code should be (ℓ, L)-list recoverable for $\ell = (1-\epsilon)q$ and $L = \mathrm{poly}(q/\epsilon)$.
2. Both encoding and list recovery should be *computationally efficient* rather than information-theoretic.
3. Subject to (1) and (2), the block-length t should be as small as possible.

Conditions (1) and (2) are necessary to obtain any valid Fiat-Shamir instantiation for some sufficiently large number of (parallel) repetitions of a protocol Π, while condition (3) seeks to minimize the number of repetitions (hopefully to a number not much larger than what is required in the interactive setting).

It is not difficult to argue that a random code $f : \{0,1\}^\lambda \to [q]^t$ satisfies condition (1) with high probability, with t indeed on the order of $1/\epsilon$ (see Theorem 4.2); however, it (of course) does not satisfy condition (2). On the other hand, known list-recoverable codes with *efficient* list-recovery are only designed to handle small input list sizes. This includes algebraic codes [GS98, PV05, GR08], expander codes [SS94, HW15], and codes built by a combination of these tools [GI01, GI02, GI03, GI04]. As mentioned before, prior work did not primarily optimize for the *input list sizes*. In fact, aside from some of the works on algebraic codes, the parameter settings in prior work require

$\ell = q^{o(1)}$;[18] these prior works were instead mostly focused on achieving high rate and very efficient algorithmic encoding/recovery.

In this work, we give a randomized construction of a code satisfying our demands via *code concatenation* [For66] combining an *algebraic code* with a *random code* (in a parameter regime where brute force decoding is polynomial-time). This is similar to the approach of [GI01] (although they use random "pseudolinear" codes rather than truly random codes for reasons of efficiency), but the parameters of our code concatenation (i.e. the relationship between the algebraic code's parameters and the random code's parameters) are quite different from [GI01].

Code concatenation is a technique based on the following simple idea: given two codes $C_{\text{out}}, C_{\text{in}}$ such that *alphabet symbols* of C_{out} can be interpreted as messages for C_{in}, it is possible to encode a message m by first computing $y = \mathsf{Encode}_{\text{out}}(m)$ and then encoding each symbol y_i using C_{in}. Code concatenation admits simple composition theorems for list-recovery, so the main question is whether there are parameter settings for $C_{\text{out}}, C_{\text{in}}$ that meet our demands.

It turns out that by setting the alphabet size q' of the outer code to be polynomially larger than the alphabet size of the inner code (which is q), the concatenation $C_{\text{out}} \circ C_{\text{in}}$ can be shown to be list-recoverable for large input list sizes as long as the outer code is list-recoverable for *moderately large* input list sizes. Moreover, list-recovery is efficient even if the *inner* code must be list-recovered by brute force; this allows for the input list size for $C_{\text{out}} \circ C_{\text{in}}$ to be very large (as this parameter is inherited from C_{in}). In the end, our choice of C_{out} is a Parvaresh-Vardy code with carefully chosen parameters to optimize for the block-length t of the final construction:

Theorem 1.9 (Informal, see Lemma 5.4). *For all $\ell < q = \mathrm{poly}(\lambda)$, there exists a probabilistically constructable family of codes*

$$\left\{ C : \{0,1\}^\lambda \to [q]^{\lambda^2 \cdot \frac{\log(\lambda)}{\log(q/\ell)}} \right\}$$

that is $(\ell, \mathrm{poly}(\lambda))$-list recoverable with all but $2^{-\lambda}$ probability.

In particular, for $\ell = (1-\epsilon)q$, we obtain block-length $t = \tilde{O}(\lambda^2/\epsilon)$. We refer the reader to Sections 4, 5.1 and 6.1 for more details.

1.3 Reflections: Fiat-Shamir via Coding Theory

In summary, our main technique relates correlation intractability for *relations* to correlation intractability for *functions* in two high-level steps.

1. **List Recoverable Codes**. Given a protocol Π whose bad challenges are (approximate) product sets $S = S_1 \times \ldots S_t \subseteq [q]^t$ (such as those arising from parallel repetition), we construct a code $C : \{0,1\}^\lambda \to [q]^t$ that *avoids* all such S: namely, every product set S contains only polynomially many codewords $C(m)$.

2. **Composition**. We prove that such codes *compose* with a hash family $\mathcal{H}$ that is CI for functions to obtain a hash family $C \circ \mathcal{H}$ that is CI for product relations.

One can view this as a special case of a more general paradigm: given the results of [CCH$^+$19, PS19], we can reduce the problem of instantiating Fiat-Shamir for *any* public-coin interactive proof

[18]An interesting concurrent and independent work [DW20] uses expander code-based techniques to construct a variant of list-recoverable codes with constant rate and $\ell = q^{\Omega(1)}$, but this is still far from the parameter regime that we care about.

to a coding-theoretic problem. For example, given a constant-round (or more generally, round-by-round sound) interactive proof Π for a language $\mathcal{L}$, soundness guarantees that for every transcript prefix τ of Π on an input $x \notin \mathcal{L}$ there is a sparse set S_τ of "bad" verifier messages. We would like to construct a code $C : \{0,1\}^\lambda \to [q]$ such that C "evades" S_τ in the sense that there are at most polynomially many messages m for which $C(m) \in S_\tau$, and furthermore there is a polynomial-time algorithm that enumerates all such m. Given such a code C, the composition of the [PS19] hash function with C instantiates Fiat-Shamir for Π (assuming LWE).

For general interactive proofs, the sets S_τ may be extremely complex and decoding seems intractable. In our results above, we took advantage of the following structure of Π that makes decoding feasible:

- Π is a *parallel repetition*, which ensures that each set S_τ is a product set;
- Moreover, the base protocol has *efficiently recognizable* bad challenges.

We were then able to leverage highly non-trivial existing algorithms [GS98, PV05] to solve the resulting coding problem.

An interesting direction for future work is whether other forms of efficient decoding can be used to instantiate Fiat-Shamir for other natural protocols.

1.4 Related Work

1.4.1 Correlation Intractability and Fiat-Shamir.

We survey the recent constructions of correlation intractable (CI) hash families [CCR16, KRR17, CCRR18, HL18, CCH$^+$19, PS19, BKM20] for comparison with our work. These constructions roughly fall into two categories:

CI for Large Classes of Relations based on Non-Standard Assumptions. The initial works [CCR16, KRR17, CCRR18, HL18, CCH$^+$18] constructed hash families that achieve correlation intractability for very broad classes of relations, but they can only prove security based on strong and non-standard cryptographic assumptions. In more detail,

- [CCR16] constructs a hash family that is CI for all *efficiently verifiable* relations (i.e., relations R such that it is efficiently decidable whether $(x, y) \in R$) assuming (sub-exponentially secure) indistinguishability obfuscation (iO) as well as input-hiding obfuscation for evasive circuits [BBC$^+$14].
- [KRR17, CCRR18] construct hash families that are CI for *all* (even hard-to-decide) sparse relations. To do so, they make assumptions that are both extremely quantitatively strong and non-falsifiable [Nao03, GW11]. For example, [CCRR18] assumes the existence of an encryption scheme such that key-recovery attacks, given (even inefficiently generated) key-dependent-message (KDM) ciphertexts, cannot succeed with probability significantly better than random guessing. [KRR17] makes a simiar assumption, and additionally assumes (subexponentially secure) iO.
- [HL18] constructs a hash family that is CI for all "efficiently sampleable relations" (similar in spirit but technically incomparable to "efficiently verifiable relations" as in [CCR16]) assuming

(subexponentially secure) iO and optimally secure one-way functions—that is, a one-way function f with no inversion attacks that are significantly better than random guessing. [CCH+19] (see [CCH+18]) also gives constructions of such a hash family under "optimally secure" variants of the learning with errors (LWE) assumption (without iO).

To summarize, these hash families achieve strong notions of CI (which suffice to instantiate Fiat-Shamir for broad classes of interactive proofs) at the cost of highly non-standard assumptions.

CI for Efficient Functions based on Standard Assumptions Beginning with the work of [CCH+19] (see [CLW18]), a sequence of works [CCH+19, PS19, BKM20] gave constructions of restricted forms of correlation intractability based on widely accepted assumptions. In more detail,

- [CCH+19, PS19] construct hash families that are CI for all *efficiently computable functions*, that is, for relations R such that $(x, y) \in R \iff y = f(x)$ for some efficiently computable function f. [CCH+19] constructs such a hash family under circular-secure fully homomorphic encryption, while [PS19] relies on the plain LWE assumption.
- [BKM20] constructs hash families that are CI for *low-degree polynomial functions* based on any one of various assumptions including LWE, the decisional Diffie-Hellman (DDH) assumption, and the Quadratic Residuosity (QR) assumption. In fact, their hash families are CI for *relations* R that are "efficiently approximable" by low-degree polynomials over $\mathbf{F}_2$, i.e., relations R such that $(x, y) \in R \iff y$ is close to $p(x)$ in Hamming distance.

To summarize, these works construct hash families that are CI for (classes of) *efficient functions* (rather than relations), possibly up to some error tolerance on bits of the output.[19] To emphasize even further, there are two main drawbacks to these CI constructions:

1. They only achieve security for relations $R \subseteq X \times Y$ that represent *functions* (possibly tolerating some error).
2. They require that the functions (or, equivalently, the relations) are *efficiently computable*.

In the context of FS-compatibility, what this means is that prior work has successfully constructed hash families that are FS-compatible with interactive proofs Π whose bad-challenge relations $R_{x,\Pi}$ can be interpreted as *efficient functions*.[20] The 3-message protocols whose bad-challenge relations are (possibly inefficient) functions are those satisfying "special soundness": for every false statement x and every prover message α, there is *at most one* choice of challenge β such that an accepting proof of the form (α, β, γ) exists. Proof systems satisfying this notion include important protocols such as [GMR85, Blu86, FLS90], but a "typical" protocol Π will be extremely far from satisfying this notion. By a "random guessing" reduction, is it not hard to handle protocols Π that have only *polynomially many* bad challenges β for any fixed α, but again, this captures only a small class of protocols.

Finally, we note that while drawback (2) has been circumvented to a small extent in later works [LV20a, JKKZ20], some form of efficiency requirement has been necessary for all bad-challenge

[19]Indeed, the constructions of [CCH+19, PS19] also support a kind of error tolerance, although this was irrelevant for their purposes.

[20]For 3-message protocols, these are abstracted as "trapdoor Σ-protocols" in [CCH+19].

functions of protocols Π with Fiat-Shamir instantiations under standard assumptions. As in prior work [CCH$^+$19, PS19, BKM20], we instead work with protocols Π such that (a relaxation of) the relation $R_{x,\Pi}$ can be efficiently verified *given a trapdoor* td. In the case of [GMW86], this is achieved by using a commitment scheme with a *trapdoor* that can extract committed bits (i.e., a public-key encryption scheme).

One might wonder whether it is possible to directly show that the CI hash families of [CCH$^+$19, PS19] are also CI for relations such as $R_{x,\Pi_{\mathrm{GMW}}}$. The intuitive reason this appears to be hard is as follows: to show that the [CCH$^+$19, PS19] hash families $\mathcal{H}$ are CI for a function f, they show that a hash function $h \leftarrow \mathcal{H}$ is computationally indistinguishable from a hash function (distribution) h_f that on input x internally (1) computes $f(x)$ and then (2) outputs a value y that specifically avoids $f(x)$. It is possible to extend this proof to make $\mathcal{H}$ "avoid" a polynomial number of evaluations $f_1(x), \ldots, f_k(x)$ (by internally computing *all* of them), but for our relations of interest, the number of $(x, y) \in R$ (for a fixed x) can be close to 2^m (for $m = |y|$)! As a result, proving that the [CCH$^+$19, PS19] hash functions satisfy this form of correlation intractability appears out of reach for current techniques.

CI for Approximable Relations We note that in order to instantiate Fiat-Shamir for round-by-round sound protocols (Section 6), we implicitly rely on (and construct) hash families that are correlation intractable for *approximations* of a relation R in a sense similar to the abstraction introduced in [BKM20]. However, in our setting, we think of hash outputs as elements of $[q]^t$ and our metric of interest is Hamming distance in the space $[q]^t$; correspondingly, our security requirement is stronger, in that we want CI for even extremely poor approximations of R (i.e. distance significantly greater than $\frac{1}{2}$). We achieve this notion of CI when R is any (sufficiently bounded) product relation using error-tolerant list-recoverable codes.

1.4.2 List-Recoverable Codes and Cryptography

List-recoverable codes have previously been used [MT07, DS11, HIOS15, KNY18, BKP18] in cryptography in the context of *domain extension* [Mer88] for hash functions. That is, given a hash function $h : \{0,1\}^n \to \{0,1\}^m$, their goal is to construct another hash function $H : \{0,1\}^* \to \{0,1\}^m$ while preserving security properties such as collision-resistance. In particular we highlight the work of [HIOS15] who use list-recoverable codes to construct hash functions H that are *indifferentiable* from random functions (if h is modeled as a random oracle). In their construction (as well as in [MT07, DS11]), it suffices to use off-the-shelf Parvaresh-Vardy codes [GUV09], albeit in somewhat non-standard parameter regimes. For example, [HIOS15] considers a regime with (1) subexponential (rather than polynomial) time list-recovery and (2) input list sizes of size q^δ for some $0 < \delta < 1$ (and q is the alphabet size).

One notable difference between our use of list-recoverable codes as compared to [MT07, DS11, HIOS15, KNY18, BKP18] is that in the context of domain extension, *precomposition* with a list-recoverable code (i.e. encoding the input x and then hashing it) is the technique used; on the other hand, we *post-compose* a hash function h with a code (i.e. we encode the *output* $h(x)$) in order to facilitate a kind of "output compression" (rather than domain extension).

2 Preliminaries

2.1 Interactive Proofs and Zero-Knowledge

Definition 2.1. *An* interactive proof for a language L with completeness error $c = c(n)$ and soundness error $s = s(n)$ *consists of a probabilistic polynomial-time interactive verifier V such that:*

- *(Completeness:) If $x \in L$ then there is an interactive function P_x such that $V(x)$, when interacting with P_x, accepts with probability at least $1 - c(|x|)$.*
- *(Soundness:) If $x \notin L$ then $V(x)$, when interacting with* any *(even computationally unbounded) interactive function P^*, accepts with probability at most $s(|x|)$.*

If the error parameters c or s are omitted, by default we require them to be negligible functions. If $c = 0$, we say that the proof-system has perfect completeness.

One of the main metrics of a proof system is the number of messages m exchanged between the prover and the verifier before the verifier decides whether or not to accept. Typically, this depends only on the input length n of the verifier's input. We call $m = m(n)$ the message complexity of the proof system.

It is useful to have a notion of efficiency for the prover as well as for the verifier. The appropriate notion turns out to depend on the "type" of the language L for which the interactive proof is designed.

- $L \in \mathbf{NP}$ and a strategy P_x as above can be implemented in polynomial-time given x and an $\mathbf{NP}$ witness[21] for x; or
- if $L \in \mathbf{P}$ and a strategy P_x can be implemented in polynomial-time given only x,

then we say that the proof-system has an efficient prover.

Definition 2.2 (Arguments)**.** *An (s, ϵ)-*computationally sound interactive proof *(or* argument*) for a language $\mathcal{L}$ is an interactive proof for $\mathcal{L}$ in which the soundness condition is weakened to:*

- *((s, ϵ)-Computational Soundness): For all $x \in \{0,1\}^n \setminus \mathcal{L}$ and all size-s prover strategies P^*, it holds that $V(x)$ when interacting with P^* accepts with at most ϵ probability.*

We omit s and ϵ if for all $s = s(n) \leq n^{O(1)}$, the protocol satisfies (s, ϵ)-computational soundness for some $\epsilon = \epsilon(n) \leq \text{negl}(n)$.

Definition 2.3 (Public-Coin)**.** *An interactive proof or argument V is said to be* public-coin *if:*

- *For some $\ell(n) \leq n^{O(1)}$ and every $x \in \{0,1\}^n$, the messages sent by $V(x)$ are i.i.d. uniformly random $\ell(n)$-bit strings.*
- *The final output of $V(x)$ when interacting with a prover P is a fixed polynomial-time computable function of x and the transcript τ of its interaction with P. We denote this output by $V(x, \tau)$.*

[21] The notion of an $\mathbf{NP}$ witness relies on associating a relation R with the language L; such a relation will usually be implicit from context.

For 2-message arguments, we also consider the notion of adaptive soundness, in which a prover may decide what it is trying to prove after seeing the verifier's first message.

Definition 2.4 (Adaptive Soundness)**.** *Let V be a 2-message argument in which the verifier's messages have length $\ell = \ell(n)$ and are* independent *of the statement x.*

*V is said to be (s, ϵ)-*adaptively sound *if for all size-s circuit ensembles P^*, the probability that $V(x, \sigma, \alpha) = 1$ and $x \in \{0,1\}^n \setminus \mathcal{L}$ is at most ϵ when sampling*

$$\begin{aligned}&\sigma \leftarrow \{0,1\}^{\ell(n)}\\&(x, \alpha) := P^*(1^n, \sigma).\end{aligned}$$

If for all $s = s(n) \leq n^{O(1)}$, V is (s, ϵ)-adaptively sound for some $\epsilon = \epsilon(n) \leq \mathrm{negl}(n)$, then we say simply that V is adaptively sound*.*

Zero-Knowledge. We recall here the definition of auxiliary-input computational zero-knowledge, referred to henceforth simply as *zero-knowledge*. Our presentation follows [Gol07].

Definition 2.5. *We say that an interactive proof (P, V) for a language L is* zero-knowledge*, if for every (malicious) probabilistic polynomial-time verifier V^* there exists a probabilistic polynomial-time simulator* Sim*, such that for every (even inefficient) function $z = z(x)$, referred to as the auxiliary input, the following two distribution ensembles are computationally indistinguishable:*

- $\{\mathsf{view}_{P,V^*(z(x))}(x)\}_{x \in L}$*, where $\mathsf{view}_{P,V^*}(x)$ includes the entire view of the verifier V^* in the interaction with P on common input x and auxiliary input z; and*
- $\{\mathsf{Sim}(x, z(x))\}_{x \in L}$.

2.2 Cryptographic Primitives and Assumptions

Definition 2.6 (Non-Interactive Statistically Binding Commitments in the CRS Model)**.** *A* non-interactive bit commitment scheme *in the CRS model is a pair of efficient randomized algorithms* $(\mathsf{Setup}, \mathsf{Com})$*, where:*

- $\mathsf{Setup}(1^\lambda)$ *outputs a string* crs*, which we refer to as a common reference string.*
- $\mathsf{Com}(\mathsf{crs}, m; r)$ *takes as input a common reference string* crs *and a message $m \in \{0,1\}$; then, using randomness r, it outputs a commitment* com*.*

We require the following security properties:

- ***Statistical binding****: With high probability over* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*, there do not exist any two strings r_0, r_1 such that* $\mathsf{Com}(\mathsf{crs}, 0; r_0) = \mathsf{Com}(\mathsf{crs}, 1; r_1)$*.*
- ***Computational hiding****: The distribution of* $(\mathsf{crs}, \mathsf{com})$ *when sampling*

$$\begin{aligned}&\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\\&\mathsf{com} \leftarrow \mathsf{Com}(\mathsf{crs}, 0)\end{aligned}$$

is computationally indistinguishable from the distribution when sampling

$$\begin{aligned}&\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\\&\mathsf{com} \leftarrow \mathsf{Com}(\mathsf{crs}, 1).\end{aligned}$$

Given a commitment string com *and common reference string* crs*, we call a valid message-randomness pair* (m, r) *an* opening *for* com*.*

Remark 2.7. *Any public-key encryption scheme* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with perfect decryption correctness*[22] *implies a non-interactive commitment scheme in the CRS model: The CRS is a public key* pk*, and a commit to a message* m *is an encryption of* m *under* pk*.*

Moreover, this commitment scheme has the following "trapdoor extractability" property: given the secret key sk *corresponding to* pk *and a (potentially malicious) commitment* com*, one can efficiently compute* m *such that the only possible opening of* com *(if any) is to* m*.*

Learning with Errors (LWE)**.** We next define the learning with errors problem [Reg03].

Definition 2.8. *The* (Decisional) Learning With Errors (LWE) assumption *with parameters* $n = n(\lambda)$*,* $m = m(\lambda)$*,* $q = q(\lambda)$*,* $\chi \in \mathcal{D}(\mathbb{Z}_q)$*, denoted by* $\mathsf{LWE}_{n,m,q,\chi}$*, states that the distribution ensembles* $\{(\mathbf{A}, \mathbf{b})\}_\lambda$ *and* $\{(\mathbf{A}, \mathbf{r})\}_\lambda$ *are computationally indistinguishable, where* $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$*,* $\mathbf{b}^T = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \pmod q$*,* $\mathbf{s} \leftarrow \mathbb{Z}_q^n$*,* $\mathbf{e} \leftarrow \chi^m$ *and* $\mathbf{r} \leftarrow \mathbb{Z}_q^m$*.*

The subexponential *variant of the* LWE *assumption states that for some* $\epsilon > 0$*, every size-*2^{n^ϵ} *adversary has advantage at most* 2^{-n^ϵ} *in distinguishing these two distributions.*

The subexponential advantage *variant of the* LWE *assumption states that for some* $\epsilon > 0$*, every poly-size adversary has advantage at most* 2^{-n^ϵ} *in distinguishing these two distributions.*

A typical parameter setting for LWE (which suffices for our purposes) is $q = \mathrm{poly}(n)$, $m = \Theta(n \log q)$ and χ defined to be the uniform distribution on $[-B, B] \subseteq \mathbb{Z}_q$ for $B = \mathrm{poly}(\lambda)$ (but significantly smaller than q).

2.3 Correlation-Intractable Hash Functions

In this section, we recall the notion of correlation intractable hash functions as introduced by Canetti, Goldreich and Halevi [CGH98].[23] We first give a syntactic definition of keyed hash functions.

Definition 2.9. *A* hash family *is a collection* $\mathcal{H} = \{h_\lambda : \mathcal{I}_\lambda \times X_\lambda \to Y_\lambda\}_{\lambda\in\mathbb{Z}^+}$ *of keyed hash functions such that* $\{\mathcal{I}_\lambda\}$ *is uniformly* $\mathrm{poly}(\lambda)$*-time sampleable and* $\{h_\lambda\}$ *is uniformly* $\mathrm{poly}(\lambda)$*-time evaluable.*

We will also write $\mathcal{H}_\lambda$ *to denote the distribution on functions* $h_\lambda(I, \cdot)$ *obtained by sampling* $I \leftarrow \mathcal{I}_\lambda$*.*

Correlation-intractability is defined as follows.

Definition 2.10 (Correlation-Intractability)**.** *For a hash family* $\mathcal{H} = \{h_\lambda : \mathcal{I}_\lambda \times X_\lambda \to Y_\lambda\}_\lambda$ *and a relation ensemble* $R = \{R_\lambda \subseteq X_\lambda \times Y_\lambda\}$*, the* correlation intractability game $\mathcal{G}^{\mathsf{CI}}_{\mathcal{H},R}$ *is the following game, played by any adversary* $\mathcal{A}$ *against a fixed "challenger"* $\mathcal{C}$*:*

1. *On input* 1^λ*,* $\mathcal{C}$ *samples* $I \leftarrow \mathcal{I}_\lambda$ *and sends* I *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *sends* $x \in X_\lambda$ *to* $\mathcal{C}$*, and wins the game if* $\big(x, h_\lambda(I, x)\big) \in R_\lambda$*.*

[22] In fact, it is sufficient if for *almost* all key pairs $(\mathsf{pk}, \mathsf{sk})$, it holds for all messages m and randomnesses r that $\mathsf{Dec}\big(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m; r)\big) = m$.

[23] A related security notion was introduced by Okamoto [Oka93] in the context of applying Fiat-Shamir to a particular identification scheme.

*We say that $\mathcal{H}$ is $(s(\cdot),\epsilon(\cdot))$-*correlation intractable for *R if for every size-$s(\lambda)$ circuit $\mathcal{A}$ and every sufficiently large λ, the adversary $\mathcal{A}$ wins the correlation intractability game with probability at most $\epsilon(\lambda)$.*

If we omit s, we mean (s,ϵ)-security simultaneously for all $s(\lambda) \le \lambda^{O(1)}$. *If we omit ϵ, we mean (s,ϵ)-security simultaneously for all $\epsilon(\lambda) \ge \lambda^{-O(1)}$.*

Theorem 2.11 ([PS19]). *Assume that $\mathsf{LWE}_{\frac{m}{2\log q},m,q,B}$ holds for a particular parameter setting $q = \mathrm{poly}(m), B = q^{\Omega(1)}$. Then, for every triple of polynomials $T = T(\lambda), n = n(\lambda), m = m(\lambda)$, there exists a hash function family $\mathcal{H} : \{0,1\}^n \to \{0,1\}^{m \log q}$ that is correlation-intractable for every function ensemble $f = \{f_\lambda\}_\lambda$ that is computable in time $T(\lambda)$.*

2.4 The Fiat-Shamir Transform

Definition 2.12. *Let $\Pi = (P,V)$ be a public-coin interactive protocol and denote its messages by $\alpha_1,\beta_1,\dots,\alpha_r,\beta_r$, where the α_i's are the prover messages and the β_i's are the verifier messages. Suppose that all verifier messages have length ℓ. For a family $\mathcal{H}$ of hash functions mapping $\{0,1\}^* \to \{0,1\}^\ell$, we define $\mathrm{FS}_{\mathcal{H}}[\Pi]$ to be the non-interactive protocol obtained by sampling as a common reference string $h \leftarrow \mathcal{H}$, and replacing each verifier message β_i by $h(x,\alpha_1,\beta_1,\dots,\alpha_i)$, where x is the main input to the protocol. The verifier for $\mathrm{FS}_{\mathcal{H}}[\Pi]$ accepts if and only if the underlying verifier accepts and all messages β_i were computed correctly.*

In case Π is defined in the CRS *model, with* CRS *σ, then we likewise view $\mathrm{FS}_{\mathcal{H}}[\Pi]$ as a protocol in the* CRS *model, using the* CRS *(σ,h).*

Definition 2.13. *We say that a hash function family $\mathcal{H}$ is* FS-compatible *with an interactive proof Π for a language $\mathcal{L}$, if the non-interactive protocol $\mathrm{FS}_{\mathcal{H}}[\Pi]$ is an adaptively sound argument for $\mathcal{L}$. We say that $\mathcal{H}$ is* non-adaptively FS-compatible *with Π if $\mathrm{FS}_{\mathcal{H}}[\Pi]$ is a (not necessarily adaptively) sound argument for $\mathcal{L}$.*

We say that $\mathcal{H}$ is FS*-compatible (or non-adaptively* FS*-compatible) with* quantitative security $\mathrm{SubExp}(\lambda)$ *(for $\lambda = \lambda(n)$) if in addition there exists $\epsilon > 0$ such that $\mathrm{FS}_{\mathcal{H}}[\Pi]$ is $(2^{\lambda^\epsilon}, 2^{-\lambda^\epsilon})$-computationally sound.*

[DNRS99] established the following negative connection between the existence of FS-compatible hash functions and zero-knowledge.

Theorem 2.14 ([DNRS99]). *Let Π be a public-coin interactive proof for a language L. Suppose that there exists an* FS*-compatible hash function family $\mathcal{H}$ for Π. Then, if Π is zero-knowledge, then $L \in \mathsf{BPP}$.*

The proof of Theorem 2.14 is simple but not exactly in this form in [DNRS99], so we provide a proof for completeness.

Proof Sketch. Suppose that Π is zero-knowledge and consider a cheating verifier V^* that gets as auxiliary input a hash function h and answers each prover message by applying h to the transcript thus far (as in Fiat-Shamir). Since Π is zero-knowledge, there exists a simulator Sim for V^*. Consider a decision procedure D for L that samples a random hash function h, runs $\mathsf{Sim}(x,h)$ (i.e., using h as the auxiliary input) and accepts if any only if (1) the transcript is accepting, and (2) the verifier messages in the transcript are computed correctly (i.e., by applying h).

First observe that if $x \in L$, by the zero-knowledge property the simulated transcript τ is computationally indistinguishible from the real interaction. By completeness, the real interaction produces an accepting transcript and so τ is accepting (and consistent with h) with all but negligible probability. Thus, $D(x)$ accepts with all but negligible probability if $x \in L$.

Next, note that if $x \notin L$, the soundness of $\mathrm{FS}_{\mathcal{H}}(\Pi)$ implies that $D(x)$ accepts with only negligible probability. This is because, given a Fiat-Shamir hash function h, one efficient cheating strategy P^* for $\mathrm{FS}_{\mathcal{H}}(\Pi)$ is to run $\mathsf{Sim}(x, h)$ and send the simulated τ transcript as its message. Therefore, such a transcript can be accepting (and consistent with h) with only negligible probability.

We conclude that D is a BPP algorithm for L. $\square$

2.5 Error Correcting Codes and List Recovery

Definition 2.15. *A q-ary code is a function $C : \mathcal{M} \to [q]^n$, where n is called the block length, $\mathcal{M}$ is called the message space, and $[q]$ is called the alphabet of C. The distance of C is the minimum Hamming distance between $C(m)$ and $C(m')$ for distinct $m, m' \in \mathcal{M}$. If C has distance d, then its relative distance is d/n.*

When discussing the asymptotic performance of codes, it makes sense to consider ensembles of codes $\{C_k : \mathcal{M}_k \to [q_k]^{n_k}\}_{k \in \mathbb{Z}^+}$ with varying parameters. We will only consider constructable codes, which are ensembles for which:

- There is an efficiently computable (and invertible) bijection between $\mathcal{M}_k$ and $[|\mathcal{M}_k|]$, and $|\mathcal{M}_k|$ is computable in time $\mathrm{poly}(k)$.
- q_k, and n_k are computable given 1^k in time $\mathrm{poly}(k)$.
- There is a polynomial-time algorithm E that, given $m \in \mathcal{M}_k$ (represented as an integer in $[|\mathcal{M}_k|]$), outputs $C_k(m)$.

Definition 2.16 (Concatenated Code [For66])**.** *Let $C : \mathcal{M} \to [Q]^N$ and $c : [Q] \to [q]^n$ denote codes. The concatenated code $C \circ c : \mathcal{M} \to [q]^{Nn}$ is defined by*

$$(C \circ c)(m)_{(i-1)n+j} = c\big(C(m)_i\big)_j,$$

for all $m \in \mathcal{M}$, $i \in [N]$, and $j \in [n]$.

Definition 2.17 (List-Recoverable Codes [GI01,GS98])**.** *An ensemble of codes $\{C_k : \mathcal{M}_k \to [q_k]^{n_k}\}$ is said to be $(\alpha(\cdot), \ell(\cdot), L(\cdot))$-list recoverable (for $\alpha : \mathbb{Z}^+ \to (0,1)$ and $\ell, L : \mathbb{Z}^+ \to \mathbb{Z}^+$) if there is a polynomial-time algorithm $\mathsf{Recover}$ that:*

- *Takes as input $k \in \mathbb{Z}^+$ and explicit descriptions of "constraint" sets $S_1, \ldots, S_{n_k} \subseteq [q_k]$ with each $|S_i| \leq \ell(k)$;*
- *Produces as output a list of at most $L(k)$ messages, containing all $m \in \mathcal{M}_k$ for which $(C_k(m))_i \in S_i$ for at least an $\alpha(k)$ fraction of $i \in [n_k]$.*

The code $\{C_k\}$ is said to be combinatorially (α, ℓ, L)-list recoverable if an arbitrarily inefficient algorithm $\mathsf{Recover}$ exists with the above functionality. If $\alpha = 1$, we omit it.

When $\ell = 1$, list recoverability is the same as the more common notion of list decodability.

2.6 Concentration Inequalities

Theorem 2.18 (Multiplicative Chernoff). *If $X_1, \dots, X_n$ are independent $\{0,1\}$-valued random variables with $X \overset{\mathsf{def}}{=} \sum_i X_i$ and $\mu \overset{\mathsf{def}}{=} \mathbb{E}[X]$, then for all $\delta \geq 0$,*

$$\Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}. \tag{1}$$

Corollary 2.19. *There is an absolute constant $c > 1$ such that if X and μ are as above, then for any $\tau \geq 3\mu$, we have*

$$\Pr[X \geq \tau] \leq c^{-\tau}.$$

Proof. Follows from viewing τ as $(1+\delta)\mu$ for $\delta \geq 2$ and rewriting Eq. (1) as

$$\begin{aligned}\Pr[X \geq (1+\delta)\mu] &\leq \left(\frac{e^{\delta/(1+\delta)}}{1+\delta}\right)^{(1+\delta)\mu} \\ &= \left(\frac{1+\delta}{e^{\delta/(1+\delta)}}\right)^{-\tau} \\ &\leq \left(\frac{3}{e}\right)^{-\tau}.\end{aligned}$$

□

Theorem 2.20 (Additive Chernoff). *If $X_1, \dots, X_n$ are independent $\{0,1\}$-valued random variables with $X \overset{\mathsf{def}}{=} \sum_i X_i$ and $\mu \overset{\mathsf{def}}{=} \mathbb{E}[X]$, then for all $\epsilon \geq 0$,*

$$\Pr\left[\frac{1}{n}X \geq \mu + \epsilon\right] \leq e^{-2\epsilon^2 n}. \tag{2}$$

3 Derandomization for Correlation Intractability

In this section, we describe and analyze two derandomization techniques that help achieve correlation intractability for more expressive relation classes. The first technique, described in Section 3.1, gives a reduction from CI for (approximate) product relations to CI for functions, based on list-recoverable codes. Next, in Section 3.2, we show a generic technique for reducing the alphabet size of product relations using subsampling.

3.1 Correlation Intractability via List Recovery

Throughout this section, let $R \subseteq X \times Y^t$ be a binary relation. Our positive result on correlation intractability are for relations with a product structure along with relatively mild sparsity and computational efficiency requirements.

Definition 3.1 (Product Relation). *We say that R is a* product relation *if for every x, the set $R_x = \{y : (x,y) \in R\} \subseteq Y^t$ has a decomposition*

$$R_x = S_1 \times S_2 \times \dots S_t$$

(where $S_1, \dots, S_t$ may depend on x).

We generalize Definition 3.1 to handle (a large fraction of) errors:

Definition 3.2 (Approximate Product Relation)**.** *We say that R is an α-*approximate product relation *if for every x, the set $R_x = \{y \in Y^t : (x, y) \in R\}$ consists exactly of all those $y \in Y^t$ for which*

$$\big|\{i \in [t] : y_i \in S_i\}\big| \geq \alpha t.$$

for some sets $S_1, \ldots, S_t \subseteq Y$ that may depend on x.

We construct hash functions that are CI for (approximate) product relations satisfying a form of *efficient verifiability.*

Definition 3.3 (Efficient Product Verifiability)**.** *We say that an (α-approximate) relation R is* efficiently product verifiable *if there is a polynomial-size circuit C such that, on every input x with some corresponding sets $(S_1, \ldots, S_t)$ (as in Definition 3.2) corresponding to x, it holds that $C(x, y, i) = 1$ if and only if $y \in S_i$.*

Whenever we consider an approximate product relation R, we assume (and, when necessary, provide) a specific decomposition $\Big\{(S_{1,x}, \ldots, S_{t,x})\Big\}_{x \in X}$ for R; the decomposition is represented by a circuit deciding membership of y in $S_{i,x}$ given (x, y, i).

The notion of *sparsity* that is most relevant for these relations is simply a bound on the (relative) size of the component sets S_i:

Definition 3.4 (Product Sparsity)**.** *We say that a product (resp., α-approximate product) relation R has* product sparsity *ρ if for every input x, the sets $S_1, \ldots, S_t$ as in Definition 3.1 (resp., Definition 3.2) have size at most ρq.*

In order to construct a hash family that is correlation intractable for (approximate) product relations, we simply compose a "base" CI hash function with an appropriate list recoverable code.

Definition 3.5 (Encoded Hash Function)**.** *Let $h : \mathcal{I} \times X \to Z$ be a hash function with index set $\mathcal{I}$, domain X, and codomain Z, and let $\mathcal{C} : Z \to Y^t$ be a (probabilistically) constructable code*[24] *(see Section 4). We write $\mathcal{C} \circ h$ to denote the hash function $\tilde{h} : \tilde{\mathcal{I}} \times X \to Y^t$, where $\tilde{\mathcal{I}} \stackrel{\mathsf{def}}{=} \mathcal{I} \times \mathcal{C}$ and $\tilde{h}((i, C), x) \stackrel{\mathsf{def}}{=} C\Big(h(i, x)\Big)$.*

In order to *analyze* the correlation intractability of encoded hash functions, we introduce a kind of *derandomization* for (approximate) product relations R, which will help us achieve correlation intractability for R.

Specifically, if $R \subseteq X \times Y^t$ is a product (or approximate product) relation and $C : Z \to Y^t$ is a code, we define

$$\tilde{R}_C = \Big\{(x, z) : (x, C(z)) \in R\Big\}.$$

The following lemma then holds syntactically.

Lemma 3.6. *If $\mathcal{C}$ is a (probabilistically) constructable code ensemble, $R = \{R_\lambda\}$ is a relation (ensemble) and if $\mathcal{H}$ is a hash family that is correlation intractable for $\tilde{R}_C$, then $\mathcal{C} \circ \mathcal{H}$ as in Definition 3.5 is a hash family that is correlation intractable for R.*

[24] We omit the parameterization of h (respectively $\mathcal{C}$) by a security parameter (respectively the message length) for simplicity.

Proof. Suppose that $\mathcal{C} \circ \mathcal{H}$ is *not* CI for R; then, there exists an efficient adversary $\mathcal{A}(h)$ that on input $h \leftarrow \mathcal{H}$, outputs an x such that $(x, C(h(x)) \in R$ with non-negligible probability. By the definition of $\tilde{R}_C$, we know that $(x, C(h(x)) \in R$ implies that $(x, h(x)) \in \tilde{R}_C$, so this contradicts the CI of $\mathcal{H}$ with respect to $\tilde{R}_C$. □

Theorem 3.7. *Let T be an arbitrary time bound; then, define $\mathcal{R} = \mathcal{R}_{\alpha,\epsilon,T}$ to be the class of all time-T verifiable α-approximate product relations $R \subseteq X \times Y^t$ with product sparsity $1-\epsilon$. Moreover, suppose that*

- *$C : Z \to Y^t$ is a code that is $(\alpha, (1-\epsilon)q, L)$ list-recoverable in* $\mathrm{poly}(L)$*-time, with* $L = \mathrm{poly}(n, q)$*;* and
- *The hash family $\mathcal{H}$ is (quantitatively $\frac{\mathrm{negl}(\lambda)}{T'}$-) correlation intractable for all functions that are computable within some sufficiently large time bound* $T' = \mathrm{poly}(T, t, |Y|)$.

Then, $C \circ \mathcal{H}$ is correlation intractable for all relations in $\mathcal{R}$. In particular, when $T, |Y|, t$ are all fixed polynomials in a security parameter λ, then if $\mathcal{H}$ is CI for functions *computable in* $\mathrm{poly}(\lambda)$ *time, then $C \circ \mathcal{H}$ is CI for $\mathcal{R}$.*

Proof. Lemma 3.6 tells us that for any time-T verifiable (approximate) product relation R, the hash family $\mathcal{H}'$ is CI for R as long as $\mathcal{H}$ is CI for the derandomized relation $\tilde{R}$ above.

We now claim that subject to the hypotheses above, $\tilde{R}$ is *efficiently enumerable* in the sense of [CCH+19]: there is an efficient (meaning $\mathrm{poly}(T, t, |Y|)$) algorithm that, given x, enumerates all $z \in Z$ such that $(x, z) \in \tilde{R}_C$. Indeed, this is possible via the following procedure:

- First, construct the sets $S_1, \ldots, S_t$ given $x = (x_1, \ldots, x_t)$; this can be done in time $t \cdot T \cdot |Y|$.
- Then, evaluate $\mathsf{Recover}(S_1, \ldots, S_t)$. By the correctness of list-recovery, this produces (with high probability) a poly-size list of all $z \in Z$ for which $(x, z) \in \tilde{R}_C$.

The runtime of this entire enumeration procedure is a fixed polynomial $\mathrm{poly}(T, t, |Y|)$. Finally, we recall that in [CCH+19] (see [CLW18] Section 3.1), it was noted that if $\mathcal{H}$ is ϵ-CI for time-T' computable functions, then it is $\frac{\epsilon}{T'}$-CI for time-T' enumerable relations (and in particular $\tilde{R}$); thus, we conclude that $\mathcal{H}'$ is CI for R with the claimed quantitative parameters. □

3.2 Handling Large Alphabets via Subsampling

While Theorem 3.7 could plausibly apply to product relations in $X \times Y^t$ with $|Y| = \lambda^{\omega(1)}$, our instantiations (Theorems 5.1 and 6.1) can only directly handle alphabets of size $|Y| = \mathrm{poly}(\lambda)$; this is because we employ list-recovery algorithms that take as input (uncompressed) lists of size $|Y|^{\Omega(1)}$ (and we also explicitly assume that $t \geq |Y|^{\Omega(1)}$ in our code constructions).

However, we can achieve correlation intractability even for large values of $|Y|$ — assuming we have sparsity $\rho \leq 1 - \frac{1}{\mathrm{poly}(\lambda)}$. We do so by first *subsampling* a random sub-alphabet $\tilde{Y} \subseteq Y$ and *restricting* the relation R to this sub-alphabet. That is, given a relation $R \subseteq X \times Y^t$ and alphabet $\tilde{Y} \subseteq Y$, we define

$$R_{\tilde{Y}} = R \cap (X \times \tilde{Y}^t) \tag{3}$$

We note that:

- If membership in a set S_i can be verified in time T, then membership in $S_i \cap \tilde{Y}$ can be verified in time $T + |\tilde{Y}| \log |Y|$.
- A hash function that is CI for $R_{\tilde{Y}}$ is *also* CI for R when viewed as a hash function with output space $\tilde{Y}^t \subseteq Y^t$.

Moreover, we note that a sufficiently large (random) subset of Y preserves the sparsity of the S_i under intersection.

Lemma 3.8. *Suppose that $R \subseteq X \times Y^t$ is an α-approximate product relation with product sparsity ρ. For some $\epsilon > 0$ and $\lambda \in \mathbb{Z}^+$, let $\tilde{Y} \subseteq Y$ be a uniformly random subset of size $q \geq \frac{\log |X| + \log t + \lambda}{\epsilon^2}$, i.e. $\tilde{Y}$ is sampled uniformly at random from $\binom{Y}{q}$.*

Then $R_{\tilde{Y}}$ as defined in Eq. (3) is an α-approximate product relation that, with probability $1-2^{-\lambda}$ over the choice of $\tilde{Y} \leftarrow \binom{Y}{q}$, has product sparsity $\leq \rho + \epsilon$.

Proof. This follows from union bounding over $|X| \cdot t$ subsets $S_{ij} \subseteq Y$ (depending on the relation and indexed by $i \in X$, $j \in [t]$), each of size at most $\rho|Y|$. For each such S_{ij}, when $\tilde{Y}$ is sampled as above, it holds with probability at least $1 - (t \cdot |X| \cdot 2^{\lambda})^{-2}$ that the intersection $S_{ij} \cap \tilde{Y}$ has size at most $(\rho + \epsilon) \cdot |\tilde{Y}|$. This follows from a standard Chernoff bound (Theorem 2.20). $\square$

We conclude that sub-sampling gives a reduction from CI over large alphabets to CI over polynomial-size alphabets.

Corollary 3.9. *Let $R \subseteq X \times Y^t$ be an α-approximate product relation with product sparsity ρ, and let $q = q(\lambda)$ be an integer such that $q \geq \frac{\log |X| + \log t + \lambda}{\epsilon^2}$, where λ is a computational security parameter.*

Suppose that for each $\tilde{Y} \in \binom{Y}{q}$, $\mathcal{H}_{\tilde{Y}}$ is a family of hash functions mapping $X \to \tilde{Y}^t$ that is CI for α-approximate product relations with product sparsity $\rho + \epsilon$. Then the hash family $\mathcal{H}$, where a random element of $\mathcal{H}$ is sampled as $h \leftarrow \mathcal{H}_{\tilde{Y}}$ for uniformly random $\tilde{Y} \leftarrow \binom{Y}{q}$, is CI for R.

Corollary 3.9 will be used in Section 5 and Section 6 to obtain CI hash functions with large output alphabets, which in turn yields Fiat-Shamir instantiations for parallel repetitions of interactive proofs with large verifier challenge spaces.

4 Basic List Recovery Bounds

In this section, we recall and rephrase some facts about the list-recoverability of three objects from the coding-theory literature: Parvaresh-Vardy codes [PV05, GR08], random codes (as analyzed by [GI01]), and generic code concatenation [For66]. These bounds will be used in Section 5 and Section 6 to build new codes that combine with Theorem 3.7 in different ways.

We begin with a description of what is achieved by Parvaresh-Vardy codes.

Theorem 4.1 (Parvaresh-Vardy codes [PV05, GR08]). *There is an explicit code*

$$C : [q]^k \to [q^s]^q,$$

parameterized by integers $s, k, q \in \mathbb{Z}^+$ (with q a power of two) such that for every $\alpha \in [0, 1]$, the code is (efficiently) (α, ℓ, L)-list recoverable in time $\text{poly}((2s)^s, q, \ell)$ *as long as*

$$\ell < \left(\frac{\alpha}{s+1}\right)^{s+1} \cdot \frac{q^s}{k^s}$$

and

$$L > c \cdot (2s)^s \cdot \frac{q\ell}{k}$$

for some absolute constant c.

C can be evaluated in time less than the above bound on the time required to list recover.

We also need bounds on the list-recoverability of random codes. List-recovery bounds for random (and flavors of pseudorandom) codes were stated (but not proved) in [GI01]; for completeness we prove here the results that we use. We first give the fully parameterized result and then specialize to parameter regimes of interest.

Theorem 4.2. *There exists a constant $c > 0$ such that for any q, Q, α, ℓ, and L (all of which are functions of n), a random function $f : [Q] \to [q]^n$ is combinatorially (α, ℓ, L)-list recoverable with probability $1 - 2^{-\Omega(L)}$ as long as*

$$L \geq c \cdot \left(Q \cdot \rho + \ell \cdot n \cdot \log\left(\frac{q}{\ell}\right)\right), \tag{4}$$

where the parameter ρ is

$$\rho \overset{\mathsf{def}}{=} \Pr\left[\mathsf{Binom}(n, \ell/q) \geq \alpha n\right].$$

This list recovery can be done (by brute force) in time $O(Q \cdot n \cdot \ell \cdot \log q)$. Evaluation of f can be done in time $O(Q \cdot n \cdot \log q)$.

Theorem 4.2 follows by a straightforward application of the probabilistic method, details follow.

Proof. Let f be a random function mapping $[Q] \to [q]^n$. We want to show that with high probability, for all sets $S_1, \ldots, S_n \subseteq [q]$ of size ℓ, the size of the set $f^{-1}(B)$ is at most L, where by B we denote

$$B \overset{\mathsf{def}}{=} \Big\{ z \in [q]^n : |i \in [n] : z_i \in S_i| \geq \alpha n \Big\}.$$

We analyze this by union bounding over $\binom{q}{\ell}^n \leq \left(\frac{q \cdot e}{\ell}\right)^{\ell n}$ events corresponding to the possible choices of $S_1, \ldots, S_n$.

To analyze an individual one of these events, we note that for fixed sets $S_1, \ldots, S_n$, the random variable $\left|f^{-1}(B)\right|$ follows a binomial distribution $\mathsf{Binom}\left(Q, \tilde{\rho}\right)$, for

$$\begin{aligned}
\tilde{\rho} &\overset{\mathsf{def}}{=} \Pr_{z \leftarrow [q]^n}\Big[z \in B\Big] \\
&= \Pr_{z \leftarrow [q]^n}\Big[\Big|i \in [n] : z_i \in S_i\Big| \geq \alpha n\Big] \\
&\leq \Pr\left[\mathsf{Binom}(n, \ell/q) \geq \alpha n\right] \\
&= \rho.
\end{aligned}$$

A multiplicative Chernoff bound (Corollary 2.19) implies that for some constant $c_0 > 0$,

$$\Pr\Big[\Big|f^{-1}(B)\Big| > L\Big] < c_0^{-L},$$

provided that $L > 3\tilde{\rho} \cdot Q$.

Then, the union bound gives the desired conclusion about f as long as

$$c_0^{-L} \cdot \left(\frac{q \cdot e}{\ell}\right)^{\ell n} \leq 2^{-\Omega(L)},$$

which holds if $L \geq c_1 \cdot \ell n \cdot \log\left(\frac{q}{\ell}\right)$ for some absolute constant $c_1 > 0$. Combining these two conditions on L yields Theorem 4.2. $\square$

We also make use of the (known) fact that concatenated codes inherit list recoverability from their constituent parts.

Lemma 4.3. *Suppose that*

- $C : \mathcal{M} \to [Q]^N$ *is an* $(\frac{\alpha-\beta}{1-\beta}, \ell', L)$*-list recoverable code and*
- $c : [Q] \to [q]^n$ *is a* (β, ℓ, ℓ')*-list recoverable code*

for $1 \geq \alpha > \beta > 0$ *and* $\ell, L \in \mathbb{Z}^+$*. Then* $C \circ c$ *is* (α, ℓ, L)*-list recoverable. Moreover, if list-recovery for* C *can be computed in time* T *and list-recovery for* c *can be computed in time* t*, then list-recovery for* $C \circ c$ *can be computed in time* $T + n \cdot t$*.*

In the special case of errorless list recovery $(\alpha = 1)$*, it suffices for* C *to be* (ℓ', L)*-list recoverable and* c *to be* (ℓ', ℓ)*-list recoverable to imply that* $C \circ c$ *is* (ℓ, L)*-list recoverable.*

Proof. Let $\{S_{i,j}\}_{i \in [N], j \in [n]}$ be subsets of $[q]$ of size at most ℓ. We want to bound the size of the set

$$S \overset{\mathsf{def}}{=} \{m \in \mathcal{M} : (C \circ c)(m)_{i,j} \in S_{i,j} \text{ for at least } \alpha n N \text{ choices of } (i,j)\}.$$

To do this, we note that by Markov's inequality, for any $m \in S$ with $(C \circ c)(m) = z_{1,1}, \ldots, z_{N,n}$, we have

$$\Big|\{i \in [N] : z_{i,j} \in S_{i,j} \text{ for at least } \beta \cdot n \text{ choices of } j \in [n]\}\Big| \geq \frac{\alpha - \beta}{1 - \beta} N.$$

Therefore, the (β, ℓ', ℓ)-list recoverability of c implies that there exist lists $L^{(1)}, \ldots, L^{(N)} \subseteq [Q]$ of size at most ℓ' such that for all such m, $C(m)_i \in L^{(i)}$ for at least $\frac{\alpha-\beta}{1-\beta} N$ choices of i. By the $(\frac{\alpha-\beta}{1-\beta}, \ell, L)$-list recoverability of C_1, there are at most L such messages m.

Moreover, the collection of such messages can be recovered in time $T + N \cdot t$ via the following algorithm:

- Given the collection of sets $\{S_{i,j}\}$, compute the lists $L^{(1)}, \ldots, L^{(N)}$ defined by these sets (with respect to c) in (total) time $N \cdot t$.
- Then, run the list recovery algorithm for C on $L^{(1)}, \ldots, L^{(N)}$ to obtain the final list.

Finally, in the case of errorless list recovery, we have by assumption that *all* symbols $z_{i,j}$ of each block have to lie in the appropriate $S_{i,j}$, so the claim follows. $\square$

5 Fiat-Shamir for Commit-And-Open Protocols

In this section, we obtain our positive results for Fiat-Shamir by applying Theorem 3.7.

In Section 5.1, give a CI instantiation for product relations (Theorem 5.1). To prove this theorem, we give a randomized construction of codes that are (ℓ, L)-list recoverable for large values of ℓ; the codes are obtained by concatenating Parvaresh-Vardy codes [PV05] with a random code. We carefully choose the parameters of the two codes to optimize the block-length of the concatenation. We augment Theorem 5.1 with alphabet reduction (Corollary 3.9) to handle larger alphabets.

In Sections 5.2 and 5.3, we state and prove our Fiat-Shamir instantiations. We give a general result for (parallel repetitions of) 3-message protocols with efficiently verifiable bad challenges, and then focus on commit-and-open protocols (Definition 5.10) as a special case. Finally, in Section 5.4, we state our negative results on parallel repeated zero knowledge, which are obtained by invoking [DNRS99].

5.1 Correlation Intractability for Efficiently Verifiable Product Relations

Our main result in this section is a construction of correlation intractable hash families for product relations over polynomial-size alphabets.

Theorem 5.1. *Let $R = R_\lambda \subseteq X_\lambda \times Y_\lambda^{t_\lambda}$ be an ensemble of product relations that are time-$T(\lambda)$ product-verifiable as in Definition 3.3 with product sparsity at most ρ, where $|Y_\lambda|$, $\log|X_\lambda|$, $T(\lambda)$, and t_λ are all upper bounded by $\lambda^{O(1)}$ and $t_\lambda \geq \lambda/\log(1/\rho)$.*

Then there exists a hash family $\mathcal{H} = \{\mathcal{H}_\lambda : X_\lambda \to Y_\lambda^{t_\lambda}\}_{\lambda\in\mathbb{Z}^+}$ that is correlation intractable for R under the LWE *assumption. Moreover, $\mathcal{H}$ depends only on (X, Y, ρ, t, T) (and is otherwise independent of R) and can be evaluated in time* $\mathrm{poly}(\log|X|, |Y|, t, T)$.

Several remarks follow on the efficiency properties of Theorem 5.1:

- The dependence of the evaluation time on $\mathcal{H}$ on $|Y|$ can be reduced if R's product decomposition can be computed explicitly in time $\ll |Y| \cdot T$ (which is the generic bound for a time T-product verifiable relation). This can apply in situations where ρ is very small. Alternatively, all dependencies on $|Y|$ can be generically reduced via alphabet reduction (see Theorem 5.5).

- If we write $\rho = 1 - \epsilon$, it suffices to have $t_\lambda \geq \lambda/\epsilon$ (which approaches $\lambda/\log(1/\rho)$ for small ϵ).

- With our usual "polynomial hardness" notions of security—that is, requiring that any $\mathrm{poly}(\lambda)$-size adversary cannot win correlation intractability games with probability $\lambda^{-\Omega(1)}$—it is equivalent (by a standard scaling argument) to replace this requirement by the seemingly weaker requirement that $t_\lambda \geq \lambda^\delta/\log(1/\rho)$ for any arbitrarily small constant $\delta > 0$.

- Under a sub-exponential variant of LWE, the requirement that $t_\lambda \geq \lambda/\log(1/\rho)$ can be weakened to $t_\lambda \geq \log^c \lambda/\log(1/\rho)$ for a large enough constant c, while still retaining standard polynomial security in the correlation intractability of the resulting hash family.

- On the other hand, correlation intractability against larger adversaries (or smaller success probabilities) is also achievable by increasing t_λ. For example, assuming sub-exponential LWE, it is possible to achieve security against size-2^λ adversaries by requiring $t_\lambda \geq \lambda^c/\log(1/\rho)$ for a sufficiently large constant c.

To prove Theorem 5.1, we first construct a family of list-recoverable codes for our parameter regime of interest. We start with the following proposition, which follows immediately from Theorem 4.1 and gives list recoverable codes in the errorless case (i.e., $\alpha = 1$), with polynomial input list sizes, output list size, alphabet and block length.

Proposition 5.2. *For all constants c and all $\ell = \ell(k) \leq k^c$, there exists a constructable ensemble of codes*

$$C = \left\{ C_k : \{0,1\}^k \to [Q_k]^{O(k^2)} \right\}_{k \in \mathbb{Z}^+}$$

for some $\ell \leq Q_k \leq O(\ell^4)$ such that C_k is $(\ell, O(k\ell))$-list recoverable in time $\mathrm{poly}(k, \ell, c^c)$.

Proof. Suppose we are given a polynomially bounded function $\ell(\cdot)$. We obtain such an ensemble from Theorem 4.1 by setting:

- $\alpha = 1$;
- $s = 2\log_k(\ell)$, which is bounded by a constant depending on $\ell(\cdot)$; and
- q is the smallest power of two that is at least k^2,

which results in $Q_k \leq (2k^2)^s \leq O(\ell^4)$. □

We remark that it is also possible to set $q = O(k^{1+\epsilon})$ for an arbitrarily small constant $\epsilon > 0$, which would result in a slightly better bound for the block length of C, but no qualitatively new applications for Fiat-Shamir.

We additionally use the following bound on the list recoverability of random functions (without errors), which follows from Theorem 4.2

Proposition 5.3. *For all $\ell = \ell(k)$, $q = q(k)$, $L = L(k)$, and $Q = Q(k)$ satisfying $\ell < q$ and $L \geq \ell \cdot \omega(\log Q)$, setting $n = n(k) = \left\lceil \frac{\log Q}{\log(q/\ell)} \right\rceil$, a random function $f : [Q] \to [q]^n$ is combinatorially (ℓ, L)-list recoverable with probability $1 - 2^{-\Omega(L)}$.*

Proof. We apply Theorem 4.2 with $\alpha = 1$ and $n = \left\lceil \frac{\log Q}{\log(q/\ell)} \right\rceil$, which ensures that $\rho \overset{\mathsf{def}}{=} (\ell/q)^n$ is at most $1/Q$. Because $n \cdot \log(q/\ell)$ is $O(\log Q)$, we have that $L \geq \omega(Q \cdot \rho + \ell \cdot n \cdot \log \frac{q}{\ell})$, from which it follows by Theorem 4.2 that f is combinatorially (ℓ, L)-list recoverable with probability $1 - 2^{-\Omega(L)}$. □

Concatenating the codes of Propositions 5.2 and 5.3 yields codes with list recoverability parameters that are useful for our applications.

Lemma 5.4. *For all $\ell = \ell(k)$, $q = q(k)$ with $\ell < q \leq k^{O(1)}$, there exists a probabilistically constructable ensemble of codes*

$$\left\{ C_k : \{0,1\}^k \to [q]^{O\left(\frac{k^2 \log(k)}{\log(q/\ell)}\right)} \right\}$$

such that each C_k is $(\ell, O(k^2\ell))$-list recoverable in time $k^{O(1)}$ with all but $2^{-\Omega(k)}$ probability.

More precisely, the running time is a fixed polynomial in k, ℓ, $\log_k(\ell)^{\log_k(\ell)}$, and $\frac{1}{\log(q/\ell)}$.

Proof. Consider any choice of $\ell = \ell(k)$ and $q = q(k)$ as above. Then by Proposition 5.2, there is some $Q = Q_k \leq (k\ell)^4$ and a constructable ensemble of codes $C' = \{C'_k : \{0,1\}^k \to [Q]^{O(k^2)}\}$ that is $(k\ell, O(k^2\ell))$-list recoverable in time $k^{O(1)}$. For this Q, Proposition 5.3 guarantees that with $n = n(k) = \left\lceil \frac{\log Q}{\log(q/\ell)} \right\rceil$, a random function $f = f_k : [Q] \to [q]^n$ is combinatorially $(\ell, k\ell)$-list recoverable with probability $1 - 2^{-\Omega(k\ell)}$. Such an f is sampleable in time $(k\ell)^4 \cdot n \log(q)$ because $Q_k \leq (k\ell)^4$. Similarly, the brute-force $(\ell, k\ell)$-list recovery algorithm for f runs in time $(k\ell)^4 \cdot \ell n \log(q)$. Concatenating C' with f yields the desired ensemble:

$$\{(C'_k \circ f_k) : \{0,1\}^k \to [q]^{k^2 \cdot n}\}_{k \in \mathbb{Z}^+}$$

is $(\ell(k), O(k^2\ell))$-list recoverable in $k^{O(1)}$ time by Lemma 4.3. □

Proof of Theorem 5.1. We are finally ready to prove our main theorem. We compose (a random instance of) the code from Lemma 5.4 with the CI hash family of Theorem 2.11. Theorem 3.7 then implies that the composition yields a good correlation-intractable hash family for the claimed relations.

The Large Alphabet Case. Finally, we combine Theorem 5.1 with Corollary 3.9 to obtain the following result on CI for product relations over *large alphabets*. We specialize this result to the case $\rho = 1 - \epsilon$ for convenience.[25]

Theorem 5.5. *Let $R \subseteq X_\lambda \times Y_\lambda^{t_\lambda}$ be a product relation that is time-T product-verifiable (where $\log|X|, T, t = \mathrm{poly}(\lambda)$) with product sparsity at most $1 - \epsilon$ for $\epsilon \geq \lambda^{-O(1)}$.*

Then, if $t \geq \lambda/\epsilon$, there exists a hash family $\mathcal{H} = \{\mathcal{H}_\lambda : X_\lambda \to Y_\lambda^{t_\lambda}\}_{\lambda \in \mathbb{Z}^+}$ that is correlation intractable for R under the LWE *assumption. Moreover, $\mathcal{H}$ depends only on $(X_\lambda, Y_\lambda, T_\lambda, t_\lambda, \epsilon)$ and can be evaluated in time* $\mathrm{poly}(\log|X|, t, T)$.

5.2 Fiat-Shamir for Trapdoor 3-Message Protocols

We now describe a general Fiat-Shamir instantiation for 3-message public coin interactive proofs with *trapdoor decidable bad challenges*, defined below. This notion is a generalization of (instance-dependent) trapdoor Σ-protocols as defined in [CCH+19].

Definition 5.6 (Bad-Challenge Relation). *Let Π denote a 3-message public coin interactive proof system for a language $\mathcal{L}$ in the (possibly empty) CRS model. We define the* bad challenge relation *$R^{(\Pi,\mathsf{crs})}$ for Π (with a fixed CRS* crs*) to be*

$$R^{(\Pi,\mathsf{crs})} = \Big\{(x|\alpha, \beta) : x \notin \mathcal{L} \text{ and } \exists\gamma : V(\mathsf{crs}, x, \alpha, \beta, \gamma) = 1\Big\}.$$

For an instance $x \notin \mathcal{L}$, we define the non-adaptive bad challenge relation *$R^{(\Pi,\mathsf{crs},x)}$ to be*

$$R^{(\Pi,\mathsf{crs},x)} = \Big\{(\alpha, \beta) : \exists\gamma : V(\mathsf{crs}, x, \alpha, \beta, \gamma) = 1\Big\}.$$

[25] The approximations incurred by this specialization cost at most a factor of $O(\log(\lambda))$ in the number of repetitions our techniques can achieve for *exact* product relations.

Definition 5.7 (Trapdoor Decidable Bad Challenges)**.** *We say that a 3-message public-coin proof system* Π *for a language* $\mathcal{L}$ *in the CRS model has* (time-T) trapdoor decidable bad challenges *if there exist*

- *An efficient algorithm* $\mathsf{TrapGen}(1^\lambda)$ *that outputs a pair* $(\mathsf{crs}, \mathsf{td})$*;*
- *A sparse binary relation* $R^{(\mathsf{td})}$*; and*
- *An algorithm* $\mathsf{BadChallengeTest}(\mathsf{td}, x, \alpha, \beta)$ *that takes as input the trapdoor* td*, the instance* x*, a first message* α*, and a second message (or challenge)* β*,*

satisfying the following properties:

- *When sampling* $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda)$*, the distribution of* crs *is statistically indistinguishable from that of an honestly generated CRS.*
- $R^{(\mathsf{td})}$ *contains the bad-challenge relation* $R^{(\Pi,\mathsf{crs})}$ *(Definition 5.6).*
- $\mathsf{BadChallengeTest}(\mathsf{td}, x, \alpha, \beta)$ *runs in time* T *and outputs* 1 *if and only if* $(x|\alpha, \beta) \in R^{(\mathsf{crs})}$*.*

Definition 5.8. *We say that* Π *has* (time-T) *instance-dependent* trapdoor decidable bad challenges *if it satisfies Definition 5.7 with the following modifications:*

- $\mathsf{TrapGen}(1^\lambda, w)$ *also takes as input non-uniform advice* w *about the instance* x*; and*
- $\mathsf{BadChallengeTest}$ *and* $R^{(\mathsf{td},x)}$ *are defined with respect to the* non-adaptive *bad challenge relation* $R^{(\Pi,\mathsf{crs},x)}$ *instead of with respect to* $R^{(\Pi,\mathsf{crs})}$*.*
- *CRS indistinguishability is only required to be computational.*

By applying Theorems 5.1 and 5.5, we obtain Fiat-Shamir instantiations for 3-message proof systems Π with (instance-dependent) trapdoor decidable bad challenges.

Theorem 5.9. *Suppose that* Π *is a 3-message public coin proof system that has time-*T *trapdoor decidable bad challenges, such that the relation* $R^{(\mathsf{td})}$ *in Definition 5.7 has sparsity at most* $1 - \epsilon$ *for* $\epsilon \geq \lambda^{-O(1)}$*. Then, for any* $t \geq \lambda/\epsilon$*, there exists a hash family* $\mathcal{H}$ *that is FS-compatible with* $\Pi^{(t)}$ *(guaranteeing* adaptive soundness*).*

Similarly, if Π *has time-*T *instance-dependent trapdoor decidable bad challenges and each* $R^{(\mathsf{td},x)}$ *has sparsity at most* $1 - \epsilon$*, then there exists an* $\mathcal{H}$ *that is FS-compatible with* $\Pi^{(t)}$ *(guaranteeing selective soundness).*

In both cases, $\mathcal{H}$ *can be evaluated in time* $\mathrm{poly}(T, t, \lambda)$*.*

Proof Sketch. By (statistical) CRS indistinguishability, we may assume that crs is sampled as $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda)$ in the (adaptive) soundness security game. If Π is repeated t times in parallel, then following [CCH+19], we know that $\mathcal{H}$ is FS-compatible with $\Pi^{(t)}$ if it is CI for the relation

$$R = \Big\{(x|\alpha_1|\dots|\alpha_t,\ \beta_1|\dots|\beta_t) : \exists\gamma : V\left(x, \alpha_i, \beta_i, \gamma_i\right) = 1 \text{ for all } i.\Big\}$$

By the definition of $\mathsf{BadChallengeTest}$, R is *contained* in the relation

$$R' = \Big\{(x|\alpha_1|\dots|\alpha_t,\ \beta_1|\dots|\beta_t) : \mathsf{BadChallengeTest}(\mathsf{td}, x, \alpha_i, \beta_i) = 1 \text{ for all } i\Big\},$$

By assumption on $\mathsf{TrapGen}$, R' is a time T verifiable product relation with product sparsity at most $1 - \epsilon$. Thus, the theorem follows from Theorem 5.1. The proof for the non-adaptive case is analogous. $\square$

5.3 Commit and Open Protocols

A commit-and-open protocol is a ubiquitous type of protocol that always has trapdoor-decidable bad challenges when the commitment scheme is instantiated using public-key encryption.

Definition 5.10 (3-Message Commit-and-Open Protocol)**.** *An interactive proof system for a language $\mathcal{L}$ is said to be* commit-and-open *if it is defined relative to a statistically binding commitment oracle* Com *and has the following structure.*

1. *The verifier takes as an input a string $x \in \{0,1\}^n$.*
2. *The prover sends a message α consisting of a string of commitments $(\mathsf{com}_i)_{i\in[M]}$ for some $M = M(n) \leq n^{O(1)}$.*
3. *The verifier V sends a random challenge $\beta \leftarrow [q]$ for some $q = q(n)$.*
4. *The prover sends a message γ containing openings of the commitments $(\mathsf{com}_i)_{i\in S_\beta}$, where $S_\beta \subseteq [M]$ is a set that is efficiently computable from β. We denote the i^{th} such opening by (b_i, d_i).*
5. *The verifier checks that for each $i \in S_\beta$, (b_i, d_i) is a valid opening of com_i (and otherwise rejects). If so, the verifier accepts if some predicate $V(x, \alpha, \beta, (b_i)_{i=1}^{|S_\beta|}) = 1$. In particular, this predicate ignores the d_i's.*

The important attributes of Definition 5.10 (that distinguish commit-and-open protocols from arbitrary 3-message protocols) are that the third message *only* consists of openings, and that the verifier rejects incorrect openings and otherwise ignores the decommitments d_i. Therefore, by instantiating Com using a public-key encryption scheme, the PKE decryption key sk allows for efficient verification of whether a challenge β is "bad" for a pair (x, α), because the bits (b_i) for a valid decommitment can be *extracted* from α using sk.

Lemma 5.11. *Let Π denote a 3-message commit-and-open protocol. Then, if* Com *is instantiated using a public-key encryption scheme as in Remark 2.7, then Π has time-T trapdoor decidable bad challenges, where T is equal to the runtime of $V(\cdot)$ plus a fixed polynomial in the security parameter.*

Proof. We give Π the syntax of a protocol with trapdoor decidable bad challenges as follows:

- $\mathsf{TrapGen}(1^\lambda)$ is defined to sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and output $(\mathsf{crs} = \mathsf{pk}, \mathsf{td} = \mathsf{sk})$.
- The relation $R^{(\mathsf{td})}$ is defined as

$$R^{(\mathsf{td})} = \Big\{(x|\alpha,\, \beta) : V(x, \alpha, (b_i)_{i=1}^{|S_\beta|}) = 1,\ \text{for } b_i = \mathsf{Dec}(\mathsf{sk}, \mathsf{com}_i)\Big\}.$$

- The algorithm $\mathsf{BadChallengeTest}(\mathsf{td}, x, \alpha, \beta)$ parses $\alpha = (\mathsf{com}_i)$, computes $b_i = \mathsf{Dec}(\mathsf{sk}, \mathsf{com}_i)$ for all $i \in S_\beta$, and computes $V(x, \alpha, (b_i)_{i=1}^{|S_\beta|})$.

Correctness follows immediately from the decryption correctness property of PKE. □

Given Lemma 5.11, Theorem 5.5 implies that all commit-and-open protocols have sound Fiat-Shamir instantiations when sufficiently repeated in parallel.

Theorem 5.12. *Assume that* LWE *holds, and let* Π *be any 3-message commit-and-open interactive proof with soundness error* $1-\epsilon$*, where* $\epsilon \geq \lambda^{-O(1)}$ *for a computational security parameter* λ*. Let the commitment scheme in* Π *be instantiated using public-key encryption as in Remark 2.7.*

Then for any $t = t(\lambda) \geq \lambda/\epsilon$*, there is a hash family* $\mathcal{H}$ *that is Fiat-Shamir compatible with* Π^t *as in Definition 2.13. The hash functions in* $\mathcal{H}$ *are evaluable in time* $T \cdot \mathrm{poly}(\lambda)$*, where* $T = T(n)$ *is the running time of the verifier for* Π*, and* n *is the length of an input for* Π*.*

Proof. This follows immediately from Theorem 5.9 and Lemma 5.11. □

5.4 Zero Knowledge is Not Preserved by Parallel Repetition

Finally, we invoke [DNRS99] to conclude that parallel repetition of commit-and-open protocols (such as GMW) does not preserve zero-knowledge.

Theorem 5.13. *Assume that* LWE *holds. Then, there exists a commitment scheme* C *such that for every 3-message commit-and-open proof-system* Π *in the commitment oracle model (as per Definition 5.10) for a language* $L \notin \mathbf{BPP}$ *with soundness error* $1-\epsilon$*, it holds that* $(\Pi_C)^t$ *is not zero-knowledge, where* Π_C *denotes the instantiation of the commitment oracle in* Π *by* C *and* $t = \lambda/\epsilon$ *for a security parameter* λ*.*

Proof. Fix C to be the public-key encryption based commitment-scheme of Remark 2.7. Let Π be a 3-message commit-and-open proof-system with soundness error $1-\epsilon$ (in the commitment oracle model). Denote by Π_C the instantiation of Π using C in place of the commitment oracle. By Theorem 5.12, there exists a hash function family $\mathcal{H}$ such that $\mathrm{FS}_{\mathcal{H}}[(\Pi_C)^t]$ is computationally sound, where $t = \lambda/\epsilon$. In other words, $\mathcal{H}$ is FS-compatible with $(\Pi_C)^t$.

Thus, by Theorem 2.14, and using the assumption that $L \notin \mathbf{BPP}$, we have that $(\Pi_C)^t$ is not zero-knowledge. □

Remark 5.14. *Assuming the subexponential variant of LWE, the number of repetitions* t *in Theorem 5.13 can be reduced to* $\frac{\log^c \lambda}{\epsilon}$ *for some constant* $c > 1$ *(in fact, under the strongest plausible LWE assumption* c *can even be* $1+\delta(n)$ *for some* $\delta(n) = o(1)$*). This still leaves open the somewhat bizarre possibility that for some very specific values of* t *(e.g.,* $t = \log(\lambda) \cdot \log^\star(\lambda)$*), the parallel repeated protocol* Π^t *is both sound and zero knowledge.*

In fact, it is known that this sort of gap is difficult to avoid: [BLV03] show that for any HVZK commit-and-open protocol Π *with poly-size challenge space, if Circuit-*SAT *has a* $2^{o(n)}$ *time algorithm, then some* $\omega(1)$*-parallel repetition of* Π is *zero knowledge. Thus, resolving this gap implies an exponential lower bound for Circuit-*SAT*.*

6 Fiat-Shamir for Round-By-Round Sound Protocols

In this section, we extend the results of Section 5 to the setting of Fiat-Shamir for *multi-round* protocols. We achieve this in three steps.

- In Section 6.1, we construct a (probabilistic) code with efficient list recovery *in the presence of errors.*
- We then combine this code with Lemma 3.6 to obtain a CI hash family for efficiently verifiable approximate product relations.
- In Section 6.2, we apply our new CI hash family to instantiate FS for a family of round-by-round sound interactive proofs. There are three variants of this result, depending on the precise efficiency requirement imposed on the interactive proof. In particular, we achieve FS instantiations for a larger class of protocols by making use of *lossy* correlation intractability [JKKZ20].

6.1 CI for Efficiently Verifiable Approximate Product Relations

Our main result in this section is a construction of correlation intractable hash families for *approximate* product relations (see Definition 3.2) over polynomial-size alphabets.

Theorem 6.1. *Let $R \subseteq X \times Y^t$ be a time-T verifiable α-approximate product relation with product sparsity at most $\rho < \alpha$.*

Set $\lambda = t \cdot (\alpha - \rho)^3$. Then, assuming that all $\mathrm{poly}(T, \lambda)$*-time adversaries solve LWE*[26] *with probability at most ϵ, there is a hash family $\mathcal{H} = \mathcal{H}_n$ that is $\big(T + \mathrm{poly}(\lambda), \epsilon \cdot \mathrm{poly}(\lambda)\big)$-correlation intractable for R.*

Moreover, $\mathcal{H}$ depends only on (X, Y, T, t, α) (and not otherwise on R).

Remark 6.2. *By pre-composing our hash family $\mathcal{H}$ with a lossy trapdoor function, we also obtain a hash family $\mathcal{H}'$ satisfying* lossy correlation intractability *[JKKZ20] for the same class $\mathcal{R}$ of relations.*

To prove Theorem 6.1, we first construct a family of list-recoverable codes in the presence of errors. We begin by describing the salient list recovery (with errors) properties of Parvaresh-Vardy codes, which follow as a corollary of Theorem 4.1.

Proposition 6.3. *For every $\alpha = \alpha(k) \geq k^{-O(1)}$ and every $\ell = \ell(k) \leq k^{O(1)}$, there exists $Q(k) \leq k^{O(1)}$ and a constructable ensemble of codes*

$$\{C_k : \{0,1\}^k \to [Q(k)]^{O(k^2/\alpha)}\}_{k \in \mathbb{Z}^+}$$

that is $(\alpha, \ell, O(k\ell/\alpha))$-list recoverable in time $k^{O(1)}$, where the exponent depends on all previous parameters.

More precisely, $Q(k)$ is bounded by $\big(2k^2/\alpha(k)\big)^{2\log_k(\ell/\alpha)}$ and the list recovery algorithm's running time is a fixed polynomial in k, ℓ, $1/\alpha$, and $\log_k(\ell/\alpha)^{\log_k(\ell/\alpha)}$.

Proof. We obtain such an ensemble from Theorem 4.1 by setting:

- q to be the smallest power of two that is at least $k^2/\alpha(k)$ (which is $k^{O(1)}$ because $\alpha(k)$ is $k^{-O(1)}$);

[26]Specifically, we need to assume the hardness of $\mathsf{LWE}_{n,m,q,\chi}$ for $n = O(\frac{\lambda}{\log \lambda})$, $m = 2n \log q$, $q = \lambda^{O(1)}$, and χ the uniform distribution on $[-B, B]$ for some $B = \lambda^{\Omega(1)}$, as in Definition 2.8.

- s to be a large enough constant (depending on α) so that $\left(\frac{\alpha}{s+1}\right)^{s+1} \cdot \frac{q^s}{k^s} > \ell$ for all sufficiently large k. Specifically, one should set s to be the smallest integer that is at least $\log_k(\ell/\alpha)$. □

Next, we describe a corollary of Theorem 4.2, which (similarly to Proposition 5.3) focuses on asymptotics, this time for list recovery with errors.

Proposition 6.4. *For all $q = q(k)$, $\ell = \ell(k)$, $Q = Q(k)$, and $\alpha = \alpha(k)$, $\ell < q$, and $\alpha > \frac{\ell}{q}$, there exists $L = L(k) \le O\left(\frac{\ell \cdot \log(Q) \cdot \log \frac{q}{\ell}}{(\alpha - \ell/q)^2}\right)$ such that a random function*

$$f : [Q] \to [q]^{\frac{\log Q}{2(\alpha-\ell/q)^2}}$$

is combinatorially (α, ℓ, L)-list recoverable with all but $2^{-\Omega(L)}$ probability.

Proof. Given q, ℓ, L, and α as above, define $n = n(k) = \frac{\log Q}{2(\alpha-\ell/q)^2}$. This n is big enough that by the additive Chernoff bound (Theorem 2.20), we have

$$\rho \overset{\mathsf{def}}{=} \Pr[\mathsf{Binom}(n, \ell/q) \ge \alpha n] \le 1/Q.$$

Then setting $L = c \cdot \left(Q \cdot \rho + \ell \cdot n \cdot \log \frac{q}{\ell}\right)$ for a large enough constant c and applying Theorem 4.2 implies the corollary. □

By concatenating the two codes above (with carefully chosen parameters), we obtain codes with list recoverability parameters that are useful for our applications.

Lemma 6.5. *For every $\ell = \ell(k)$, $q = q(k)$, $\alpha = \alpha(k)$ with $\ell < q \le k^{O(1)}$ and $\alpha \ge \ell/q + k^{-O(1)}$, there is a probabilistically constructable ensemble of codes*

$$\left\{C_k : \{0,1\}^k \to [q]^{O\left(\frac{k^2 \log k}{(\alpha-\ell/q)^3}\right)}\right\}_{k\in\mathbb{Z}^+}$$

that is $(\alpha, \ell, k^{O(1)})$-list recoverable in time $k^{O(1)}$ with all but $2^{-\Omega(k)}$ probability.

More precisely,[27] *the running time of the list recovery algorithm is a fixed polynomial in $k, \ell, \frac{1}{\alpha-\ell/q}$, and Q^* for*

$$\log_k(Q^*) \le 2\log_k\left(\frac{8\ell k \log(q/\ell)}{(\alpha - \frac{\ell}{q})^3}\right)\log_k\left(\frac{4k^2}{\alpha - \frac{\ell}{q}}\right) = O(1).$$

Proof. Suppose we are given ℓ, q, and α as above. Let $\beta \overset{\mathsf{def}}{=} \frac{1}{2}(\alpha + \frac{\ell}{q})$. Proposition 6.4 guarantees that for all $Q = Q(k) \le k^{O(1)}$, there is some $L_Q = L_Q(k) \le k^{O(1)}$ such that with $n(k) = \frac{\log Q}{2(\beta-\ell/q)^2}$, a random function $f_Q : [Q] \to [q]^n$ is combinatorially (β, ℓ, L_Q)-list recoverable with all but $2^{-\Omega(L_Q)}$ probability. More precisely, this $L_Q(k)$ satisfies $L_Q(k) \le O\left(\frac{\ell \cdot \log(Q) \cdot \log \frac{q}{\ell}}{(\beta-\ell/q)^2}\right)$, which is $O\left(\frac{\ell \cdot \log k \cdot \log \frac{q}{\ell}}{(\beta-\ell/q)^2}\right)$ because we required that $Q \le k^{O(1)}$. For the same reason, the brute force list recovery algorithm for such an f_Q is efficient (running in time $O(Q \cdot n \cdot \log q) = k^{O(1)}$).

[27] We write down this explicit expression because it determines the runtime of the Fiat-Shamir hash functions in Theorem 6.13.

Let $L^\star = L^\star(k)$ satisfy $\omega\left(\frac{\ell\cdot\log(Q)\cdot\log\frac{q}{\ell}}{(\beta-\ell/q)^2}\right) \le L^\star \le k^{O(1)}$ (for instance, set $L^\star = \frac{\ell\cdot k\cdot\log\frac{q}{\ell}}{(\beta-\ell/q)^2}$). Setting $\tilde{\alpha} = \frac{\alpha-\beta}{1-\beta}$, Proposition 6.3 guarantees the existence of $Q^\star = Q^\star(k) \le k^{O(1)}$ such that there is a constructable ensemble of codes

$$C = \{C_k : \{0,1\}^k \to [Q^*(k)]^{O(k^2/\tilde{\alpha})}\}$$

that is $\left(\tilde{\alpha}, L^\star, O(\frac{kL^\star}{\tilde{\alpha}})\right)$-list recoverable. More precisely, Proposition 6.3 gives $Q^\star(k) \le (2k^2/\tilde{\alpha})^{2\log_k(L^\star/\tilde{\alpha})}$. Also note that

$$\tilde{\alpha} = \frac{\alpha-\beta}{1-\beta} \ge \alpha - \beta = \frac{1}{2}\cdot(\alpha - \ell/q) \ge k^{-O(1)}$$

and

$$\beta - \frac{\ell}{q} = \frac{1}{2}(\alpha - \frac{\ell}{q}).$$

Choosing $f_Q = f_{Q^\star}$ from above, we conclude that the concatenation

$$C \circ f : \{0,1\}^k \to [q]^{O(nk^2/\tilde{\alpha})}.$$

satisfies our desired properties by Lemma 4.3. $\square$

By plugging the (randomized) code from Lemma 6.5 and the hash family of Theorem 2.11 into Theorem 3.7, we obtain Theorem 6.1.

6.2 Applications to Fiat-Shamir for Round-by-Round Sound Protocols

Following [CCH+18, CCH+19], we consider the notion of round-by-round soundness to capture a form of soundness for interactive proofs of greater than 3 messages that is compatible with the notion of correlation intractability.

Definition 6.6 (Round-by-Round Soundness, [CCH+18, CCH+19])**.** *Let $\Pi = (P, V)$ be a $2r+1$-message public coin interactive proof system for a language L. We say that Π has* round-by-round soundness error $\delta(\cdot)$ *(or is δ-*RBR sound*) if there is a deterministic (not necessarily efficiently computable) function* State*, which takes as input an instance x and a transcript prefix τ and outputs either* acc *or* rej *such that the following holds:*

1. *If $x \notin L$, then $\mathsf{State}(x, \emptyset) = \mathsf{rej}$, where $\emptyset$ denotes the empty transcript.*
2. *For every input x and partial transcript $\tau = \tau_i$, if $\mathsf{State}(x, \tau) = \mathsf{rej}$, then for every potential prover message α_{i+1}, it holds that*

$$\Pr_{\beta_{i+1}}\left[\mathsf{State}(x, \tau|\alpha_{i+1}|\beta_{i+1}) = \mathsf{acc}\right] \le \delta(n).$$

3. *For any* full *transcript τ (i.e., consisting of $2r+1$ messages), if $\mathsf{State}(x, \tau) = \mathsf{rej}$ then $V(x, \tau) = 0$.*

We say that Π is round-by-round sound *if it has round-by-round soundness error δ for some $\delta(n) = \mathrm{negl}(n)$.*

By a union bound, a proof system with round-by-round soundness error δ has standard soundness error at most $r \cdot \delta$.

Canetti *et al.* [CCH+18] related the soundness of Fiat-Shamir, when applied to a round-by-round sound protocol, to the correlation intractability of the hash function $\mathcal{H}$.

Theorem 6.7 ([CCH+18, Theorem 5.8])**.** *Suppose that* $\Pi = (P, V)$ *is a* $2r+1$*-message public-coin interactive proof for a language* L *with perfect completeness and round-by-round soundness with state function* State*. Let* X_n *denote the set of partial transcripts (including the input and all messages sent) and let* Y_n *denote the set of verifier messages when* Π *is executed on an input of length* n*.*

Finally, define the relation ensemble $R = R_{\mathsf{State}}$ *as follows:*

$$R^{(n)}_{\mathsf{State}} \stackrel{\mathsf{def}}{=} \left\{ \Big((x, \tau|\alpha), \beta\Big) : \begin{array}{c} x \in \{0,1\}^n, \\ \mathsf{State}(x, \tau) = \mathsf{rej}, \text{ and} \\ \mathsf{State}(x, \tau|\alpha|\beta) = \mathsf{acc} \end{array} \right\}.$$

If a hash family $\mathcal{H} = \{\mathcal{H}_n : X_n \to Y_n\}$ *is correlation intractable for* R*, then the non-interactive protocol* $\Pi_{\mathrm{FS},\mathcal{H}}$ *is an adaptively sound argument system for* L*.*

In this work, we consider protocols Π with round-by-round soundness error $\rho < \frac{1}{r}$. We then consider applying the Fiat-Shamir transform to a parallel repetition Π^t (for sufficiently large t). To analyze this, we must also analyze how parallel repetition works for round-by-round sound protocols.[28]

Definition 6.8 (Threshold State Function)**.** *Let* Π *denote a* $2r+1$*-message public-coin interactive proof system with round-by-round soundness* δ *and corresponding state function* State*. We then define the* threshold state function $\mathsf{State}^{(t)}$ *defined on the* t*-fold parallel repetition* Π^t*: decomposing a (partial) transcript of* $\Pi^{(t)}$ *as a tuple* $(\tau_1, \ldots, \tau_t)$ *(where each* τ_i *is a partial transcript for* Π*), we define*

$$\mathsf{State}^{(t)}(x, \tau_1, \ldots, \tau_t) = \mathsf{rej} \iff \Big| \{ i \in [t] : \mathsf{State}(x, \tau_i) = \mathsf{rej} \} \Big| \geq 1 + \frac{r-j}{r} \cdot (t-1),$$

where j *is the number of verifier messages in each* τ_i*.*

Lemma 6.9. *If* Π *is a protocol as in Definition 6.8, then* $\mathsf{State}^{(t)}$ *gives* $\Pi^{(t)}$ *the structure of a round-by-round sound proof system with RBR soundness error bounded by*

$$\delta^{(t)} := \exp\left(-2 \left(\frac{t-1}{r \cdot t} - \delta \right)^2 t \right),$$

provided that $\frac{t-1}{r \cdot t} > \delta$*.*

Proof. This follows from the fact that for any partial transcript $(x, \tau_1, \ldots, \tau_t)$, if $\mathsf{State}^{(t)}(x, \tau_1, \ldots, \tau_t)$ $= \mathsf{rej}$ but $\mathsf{State}^{(t)}(x, \tau_1|\alpha_{1,j+1}|\beta_{1,j+1}, \ldots, \tau_t|\alpha_{t,j+1}|\beta_{t,j+1}) = \mathsf{acc}$, then at least $\frac{t-1}{r}$ "slots" of τ

[28] In [CCH+18, CCH+19], it is noted that sufficient parallel repetition of *any* public-coin interactive proof results in a round-by-round sound protocol, but this transformation results in a rather complex State function; we want a transformation that roughly *preserves* the State function of the starting protocol (which we assume to satisfy some form of RBR soundness).

changed from rej to acc according to State. Thus, for any τ such that $\mathsf{State}^{(t)}(x,\tau) = \mathsf{rej}$ and any $\alpha = (\alpha_{1,j+1},\dots,\alpha_{t,j+1})$, the probability over β that $\mathsf{State}^{(t)}(x,\tau|\alpha|\beta) = \mathsf{acc}$ is at most the probability that at least $\frac{t-1}{r}$ out of t i.i.d. Bernoulli events with mean δ occur. By a Chernoff bound, this happens with probability at most $\delta^{(t)}$, as desired. □

The proof of Lemma 6.9 in fact shows that the "bad challenge relation" for $(\Pi^{(t)}, \mathsf{State}^{(t)})$ is an *α-approximate product relation* with product sparsity δ, where $\alpha = \frac{t-1}{r\cdot t}$. Therefore, if the relation $R_{\mathsf{State}^{(t)}}$ is efficiently product-verifiable (or, equivalently, the relation R_{State} is efficiently verifiable), we can apply Theorem 6.1 to obtain a sound Fiat-Shamir instantiation for the protocol $\Pi^{(t)}$, provided that t is large enough.

6.2.1 Notions of Bad Challenge Efficient Decidability

In this section, let Π be a $2r+1$-message (public-coin) interactive proof system for a language $\mathcal{L}$.

Definition 6.10 (Trapdoor Decidable Bad Challenges)**.** *We say that public-coin interactive proof Π for a language $\mathcal{L}$ in the CRS model has* round-by-round soundness error δ with time-T trapdoor decidable bad challenges *if there exist*

- *An efficient algorithm* $\mathsf{TrapGen}(1^\lambda)$ *that outputs a pair* $(\mathsf{crs},\mathsf{td})$*;*
- *A δ-sparse binary relation $R^{(\mathsf{td})}$; and*
- *An algorithm* $\mathsf{BadChallengeTest}(\mathsf{td}, x, j, \tau_{j-1}|\alpha_j, \beta_j)$ *that takes as input the trapdoor* td*, the instance x, a transcript prefix $\tau_{j-1}|\alpha_j$ (consisting of j prover messages and $j-1$ verifier messages), and a verifier message β_j,*

satisfying the following properties:

- *When sampling* $(\mathsf{crs},\mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda)$*, the distribution of* crs *is statistically indistinguishable from that of an honestly generated CRS.*
- *$R^{(\mathsf{td})}$ contains the bad-challenge relation R_{State} (Definition 5.6).*
- $\mathsf{BadChallengeTest}(\mathsf{td}, x, j, \tau_{j-1}|\alpha_j, \beta_j)$ *runs in time T and outputs 1 if and only if $(x|\tau_{j-1}|\alpha_j,\ \beta_j) \in R^{(\mathsf{State})}$.*

Definition 6.10 is a strict generalization of Definition 5.7 and captures the multi-round protocols for which we can instantiate Fiat-Shamir based on polynomial hardness of appropriately chosen CI hash families. As in Section 5.3, a similar definition captures *non-adaptively sound* Fiat-Shamir instantiations.

Definition 6.11. *We say that Π has* round-by-round soundness error δ with time-T *instance-dependent* trapdoor decidable bad challenges *if it satisfies Definition 6.10 with the following modifications:*

- $\mathsf{TrapGen}(1^\lambda, w)$ *also takes as input non-uniform advice w about the instance x; and*
- $\mathsf{BadChallengeTest}$ *and $R^{(\mathsf{td},x)}$ are defined with respect to the* non-adaptive *bad challenge relation*

$$R_{\mathsf{State},x} = \left\{ (\tau|\alpha,\ \beta) : \begin{array}{c} \mathsf{State}(x,\tau) = \mathsf{rej},\ \textit{and} \\ \mathsf{State}(x,\tau|\alpha|\beta) = \mathsf{acc} \end{array} \right\}.$$

instead of with respect to R_{State}.

- *CRS indistinguishability is only required to be computational.*

Finally, we give a third definition further generalizing the previous two in a way that captures the Sumcheck and GKR protocols with *succinct* bad challenge testing. However, this variant requires stronger assumptions on the CI hash compiler.

Definition 6.12. *We say that* Π *has* round-by-round soundness error δ with time-T *prefix-dependent* trapdoor decidable bad challenges *if it satisfies Definition 6.10 with the following modifications:*

- $\mathsf{TrapGen}(1^\lambda, z_{\beta^*})$ *also takes as input non-uniform advice* $z = f(x, \beta^*)$ *about the instance* x and *a string* $\beta^* = (\beta_1^*, \ldots, \beta_r^*)$ *consisting of (fixed) verifier messages.*
- $\mathsf{BadChallengeTest}$ *and* $R^{(\mathsf{td},x)}$ *are defined with respect to the* non-adaptive *bad challenge relation* $R^{(\mathsf{State},x)}$ *instead of with respect to* R_{State}. Moreover, *"correctness" is relaxed to the following set containment: for all rounds* j *and strings* β^*, *when sampling* $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda, f(x, \beta^*))$,

$$R^{(\mathsf{td},x)} \supseteq \Big\{ (\tau|\alpha_j,\ \beta_j) \in R^{(\mathsf{State},x)} : (\beta_1, \ldots, \beta_{j-1}) = (\beta_1^*, \ldots, \beta_{j-1}^*) \Big\}.$$

- *CRS indistinguishability is only required to be computational.*

6.2.2 Putting Everything Together

Given our efficient bad challenge notions from Section 6.2.1 and our CI hash family from Section 6.1, we are ready to state our Fiat-Shamir result for round-by-round sound protocols.

Theorem 6.13. *Let* Π *be a* $2r+1$*-message (public-coin) interactive proof system for a language* $\mathcal{L}$ *in which the verifier's messages are uniformly random on* $[q]$ *for some* $q \in \mathbb{Z}^+$ *and prover messages are bit strings of length* $a = a(n)$. *Let* $\delta = \delta(n) \in (0,1)$ *and* $\lambda = \lambda(n) \in \mathbb{Z}^+$ *be functions, and define*

$$t = \frac{\lambda}{(\frac{1}{2r} - \delta)^3}.$$

Then, there exists $T_{\mathsf{Dec}} = T_{\mathsf{Dec}}(n)$ *that is a polynomial in* λ, δq, $\frac{1}{\frac{1}{2r} - \delta}$, *and* $Q^\star$, *where*

$$\log_\lambda Q^\star = \log_\lambda \left(\frac{8\delta q \log(1/\delta)}{\left(\frac{1}{2r} - \delta\right)^3} \right) \log_\lambda \left(\frac{\lambda^2}{\frac{1}{2r} - \delta} \right),$$

such that:

- *If* Π *has round-by-round soundness error* δ *with time-*T *trapdoor decidable bad challenges, then assuming the hardness of LWE there is a hash family* $\mathcal{H}$ *that is adaptively FS-compatible with* Π^t *as in Definition 2.13.*
- *If* Π *has round-by-round soundness error* δ *with time-*T instance-dependent *trapdoor decidable bad challenges, then assuming the hardness of LWE there is a hash family* $\mathcal{H}$ *that is* non-adaptively *FS-compatible with* Π^t.
- *If* Π *has round-by-round soundness error* δ *with time-*T prefix-dependent *trapdoor decidable bad challenges, then under the* subexponential advantage *variant of the LWE assumption, there is a hash family* $\mathcal{H}$ *that is non-adaptively FS-compatible with* Π^t.

Moreover,

- *Assuming subexponential hardness for LWE, the first two results extend to also give FS-compatibility with* SubExp(λ) *quantitative security.*
- *These hash families depend only on* $(a(\cdot), q(\cdot), \delta(\cdot), T(\cdot), \lambda(\cdot), r(\cdot))$ *and otherwise do not depend on* Π.
- *Hash function evaluation can be done in time that is* $O\big((qT + T_{\mathsf{Dec}}) \cdot \mathrm{poly}(\lambda)\big)$. *The* qT *term can also be replaced by the amount of time required to enumerate bad challenges for* Π.

Example Application: Fiat-Shamir for GKR We now sketch how, assuming subexponential LWE, Theorem 6.13 allows us to soundly apply the Fiat-Shamir transform to the doubly-efficient public-coin interactive proof of Goldwasser, Kalai, and Rothblum [GKR08]. This interactive proof, which we refer to as GKR, is applicable to (log-space uniform) bounded-depth computations.

We will fix some family of (log-space uniform) circuits with depth $d = d(n)$ and size $s = s(n)$. GKR is additionally parameterized by a finite field of order $q = q(n)$. The best efficiency (in our case) is achieved for q a power of two, which yields the following parameters:

- The round complexity is $O(d \cdot \log n)$;
- The prover runs in time $\mathrm{poly}(s, \log q)$;
- The verifier runs in time $n \cdot \mathrm{poly}(d, \log s, \log q)$;
- The proof system has round-by-round soundness error $\delta = \delta(n)$ with time-T $(= T(n))$ prefix-dependent trapdoor decidable bad challenges, where $\delta = O(\frac{\log n}{q})$ and $T = \mathrm{poly}(\log n, \log s, \log q)$.

  In particular, the number of bad verifier challenges at any round is $\ell = O(\log n)$

When applying the Fiat-Shamir transform to GKR, we would like to preserve the feature that the verifier's running time is much less than (and ideally polylogarithmic in) the time required to evaluate the circuit. Specifically, we would like the Fiat-Shamir hash functions to be evaluable in time $n \cdot \mathrm{poly}(d, \log q, \log s, \log n)$. This was done by [JKKZ20] for very large q, i.e. $q > (d\ell)^{\kappa^{1/\epsilon}}$ for a computational security parameter κ. Here we focus on the other extreme of parameter settings, where q is small (say $\mathsf{polylog}(n)$), and soundness is amplified by parallel repetition.

To accomplish this, when applying Theorem 6.13 we set $\lambda = (d \log n) \cdot \kappa^{1/\epsilon}$, where ϵ denotes the exponent of our subexponential LWE assumption. Applying Theorem 6.13, we bound the runtime of the verifier as follows. First, note that

$$\frac{1}{2r} - \delta \geq \frac{1}{4r} = \frac{1}{4d\log(n)},$$

and so $\log_\lambda(Q^\star) = O(A \cdot B)$ for

$$A = \log_\lambda(8d\ell \log(\lambda)\log(n)) = O(1)$$

and

$$B = \log_\lambda(\lambda^2 d \log(n))) = O(1).$$

Therefore, $Q^\star = \mathrm{poly}(\lambda) = \mathrm{poly}(d, \kappa)$ and so verification runs in time $n \cdot \mathrm{poly}(d, \kappa, \log s, q, \log n)$, which is $n \cdot \mathrm{poly}(d, \kappa, \log s, \log n)$ by the assumption that q is $\mathsf{polylog}(n)$.

Finally, we note that:

- Because Theorem 6.13 gives us sub-exponential security in κ, if our goal is to achieve $\mathrm{poly}(n)$ security (i.e., $\mathrm{negl}(n)$ soundness error against $\mathrm{poly}(n)$ size provers), we can set $\kappa = \mathsf{polylog}(n)$. Then, the hash function evaluation time (and hence the verifier running time) will be $\tilde{O}(n)$.
- By using a root-finding algorithm (see [CCH+19, JKKZ20]) instead of a root verification algorithm (as used in Theorem 6.13 above) in our CI analysis, we can reduce the verifier runtime dependence on q to $\mathrm{poly}(\ell, \log q)$ (instead of $\mathrm{poly}(q)$), enabling us to handle all field sizes (not just polylogarithmic).
- At the expense of a larger number of repetitions (incurring a multiplicative overhead of q), we could replace our Parvaresh-Vardy based code with a concatenation of a Reed-Solomon code with a random code for a faster running time of the hash function (i.e. some fixed polynomial in (λ, d) for all choices of λ, d instead of explicitly requiring $\lambda \geq d \log(n)$).

Acknowledgements

We thank Vinod Vaikuntanathan for helpful discussions and feedback.

References

[AK97] Vikraman Arvind and Johannes Köbler. On resource-bounded measure and pseudorandomness. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 235–249. Springer, 1997.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd FOCS*, pages 106–115. IEEE Computer Society Press, October 2001.

[BBC+14] Boaz Barak, Nir Bitansky, Ran Canetti, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Obfuscation for evasive functions. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 26–51. Springer, Heidelberg, February 2014.

[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, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 522–551. Springer, Heidelberg, December 2019.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016.

[BDG+13] Nir Bitansky, Dana Dachman-Soled, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, Adriana López-Alt, and Daniel Wichs. Why "Fiat-Shamir for proofs" lacks a proof. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 182–201. Springer, Heidelberg, March 2013.

[BFJ+20] Saikrishna Badrinarayanan, Rex Fernando, Aayush Jain, Dakshita Khurana, and Amit Sahai. Statistical ZAP arguments. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 642–667. Springer, Heidelberg, May 2020.

[BGG90] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Randomness in interactive proofs. In *31st FOCS*, pages 563–572. IEEE Computer Society Press, October 1990.

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 738–767. Springer, Heidelberg, August 2020.

[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, *50th ACM STOC*, pages 671–684. ACM Press, June 2018.

[Blu86] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, volume 1, page 2. Citeseer, 1986.

[BLV03] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. In *44th FOCS*, pages 384–393. IEEE Computer Society Press, October 2003.

[BM82] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd FOCS*, pages 112–117. IEEE Computer Society Press, November 1982.

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, August 1994.

[CCH$^+$18] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, and Ron D. Rothblum. Fiat-Shamir from simpler assumptions. Cryptology ePrint Archive, Report 2018/1004, 2018. https://eprint.iacr.org/2018/1004. Part 1 of [CCH$^+$19].

[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, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.

[CCR16] Ran Canetti, Yilei Chen, and Leonid Reyzin. On the correlation intractability of obfuscated pseudorandom functions. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 389–415. Springer, Heidelberg, January 2016.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 91–122. Springer, Heidelberg, April / May 2018.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

[CLW18] Ran Canetti, Alex Lombardi, and Daniel Wichs. Fiat-Shamir: From practice to theory, part II (NIZK and correlation intractability from circular-secure FHE). Cryptology ePrint Archive, Report 2018/1248, 2018. https://eprint.iacr.org/2018/1248. Part 2 of [CCH+19].

[DNRS99] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. In *40th FOCS*, pages 523–534. IEEE Computer Society Press, October 1999.

[DS11] Yevgeniy Dodis and John P. Steinberger. Domain extension for MACs beyond the birthday barrier. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 323–342. Springer, Heidelberg, May 2011.

[DW20] Dean Doron and Mary Wootters. High-probability list-recovery, and applications to heavy hitters. *ECCC*, 2020. https://eccc.weizmann.ac.il/report/2020/162/.

[FGJ18] Nils Fleischhacker, Vipul Goyal, and Abhishek Jain. On the existence of three round zero-knowledge proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 3–33. Springer, Heidelberg, April / May 2018.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317. IEEE Computer Society Press, October 1990.

[For66] G David Forney. Concatenated codes. 1966.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1986.

[GI01] Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *42nd FOCS*, pages 658–667. IEEE Computer Society Press, October 2001.

[GI02] Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *34th ACM STOC*, pages 812–821. ACM Press, May 2002.

[GI03] Venkatesan Guruswami and Piotr Indyk. Linear time encodable and list decodable codes. In *35th ACM STOC*, pages 126–135. ACM Press, June 2003.

[GI04] Venkatesan Guruswami and Piotr Indyk. Linear-time list decoding in error-free settings: (extended abstract). In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *ICALP 2004*, volume 3142 of *LNCS*, pages 695–707. Springer, Heidelberg, July 2004.

[GJJM20] Vipul Goyal, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Statistical zaps and new oblivious transfer protocols. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 668–699. Springer, Heidelberg, May 2020.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–115. IEEE Computer Society Press, October 2003.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 113–122. ACM Press, May 2008.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 171–185. Springer, Heidelberg, August 1986.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994.

[Gol07] Oded Goldreich. *Foundations of cryptography: volume 1, basic tools.* Cambridge university press, 2007.

[Gol11] Oded Goldreich. A sample of samplers: A computational perspective on sampling. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 302–332. Springer, 2011.

[GR08] Venkatesan Guruswami and Atri Rudra. Soft decoding, dual bch codes, and better list-decodable e-biased codes. In *2008 23rd Annual IEEE Conference on Computational Complexity*, pages 163–174. IEEE, 2008.

[GS98] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. In *39th FOCS*, pages 28–39. IEEE Computer Society Press, November 1998.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *Journal of the ACM (JACM)*, 56(4):1–34, 2009.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

[HIOS15] Iftach Haitner, Yuval Ishai, Eran Omri, and Ronen Shaltiel. Parallel hashing via list recoverability. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 173–190. Springer, Heidelberg, August 2015.

[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 FOCS*, pages 850–858. IEEE Computer Society Press, October 2018.

[HW15] Brett Hemenway and Mary Wootters. Linear-time list recovery of high-rate expander codes. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP 2015, Part I*, volume 9134 of *LNCS*, pages 701–712. Springer, Heidelberg, July 2015.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

[Ish20] Yuval Ishai. Zero-knowledge proofs from information-theoretic proof systems. 2020. https://zkproof.org/2020/08/12/information-theoretic-proof-systems/.

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *29th ACM STOC*, pages 220–229. ACM Press, May 1997.

[JKKZ20] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Zhang. SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE. *IACR Cryptol. ePrint Arch*, 2020:980, 2020. To appear in STOC 2021.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

[KNY18] Ilan Komargodski, Moni Naor, and Eylon Yogev. Collision resistant hashing for paranoids: Dealing with multiple collisions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 162–194. Springer, Heidelberg, April / May 2018.

[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, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 224–251. Springer, Heidelberg, August 2017.

[LFKN90] C Lund, L Fortnow, H Karloff, and N Nisan. The polynomial-time hierarchy has interactive proofs. *Proceedings of STOC 1990*, pages 2–10, 1990.

[LNPT19] Benoît Libert, Khoa Nguyen, Alain Passelègue, and Radu Titiu. Simulation-sound arguments for LWE and applications to KDM-CCA2 security. Cryptology ePrint Archive, Report 2019/908, 2019. https://eprint.iacr.org/2019/908.

[LNPY20] Benoît Libert, Khoa Nguyen, Thomas Peters, and Moti Yung. One-shot fiat-shamir-based nizk arguments of composite residuosity in the standard model. Cryptology ePrint Archive, Report 2020/1334, 2020. https://eprint.iacr.org/2020/1334.

[LV20a] Alex Lombardi and Vinod Vaikuntanathan. Fiat-shamir for repeated squaring with applications to PPAD-hardness and VDFs. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 632–651. Springer, Heidelberg, August 2020.

[LV20b] Alex Lombardi and Vinod Vaikuntanathan. Multi-input correlation intractable hash functions via shift-hiding. *IACR Cryptology ePrint Archive*, Report 2020/1378, 2020. https://eprint.iacr.org/2020/1378.

[LVW19] Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. 2-message publicly verifiable WI from (subexponential) LWE. Cryptology ePrint Archive, Report 2019/808, 2019. https://eprint.iacr.org/2019/808.

[Mer88] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 369–378. Springer, Heidelberg, August 1988.

[Mic93] Silvio Micali. Fair public-key cryptosystems. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 113–138. Springer, Heidelberg, August 1993.

[MT07] Ueli M. Maurer and Stefano Tessaro. Domain extension of public random functions: Beyond the birthday barrier. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 187–204. Springer, Heidelberg, August 2007.

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003.

[NW88] Noam Nisan and Avi Wigderson. Hardness vs. randomness (extended abstract). In *29th FOCS*, pages 2–11. IEEE Computer Society Press, October 1988.

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Heidelberg, August 1993.

[PS96] David Pointcheval and Jacques Stern. Provably secure blind signature schemes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 252–265. Springer, Heidelberg, November 1996.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114. Springer, Heidelberg, August 2019.

[PV05] Farzad Parvaresh and Alexander Vardy. Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In *46th FOCS*, pages 285–294. IEEE Computer Society Press, October 2005.

[Reg03] Oded Regev. New lattice based cryptographic constructions. In *35th ACM STOC*, pages 407–416. ACM Press, June 2003.

[RW18] Atri Rudra and Mary Wootters. Average-radius list-recoverability of random linear codes. In Artur Czumaj, editor, *29th SODA*, pages 644–662. ACM-SIAM, January 2018.

[SS94] Michael Sipser and Daniel A. Spielman. Expander codes. In *35th FOCS*, pages 566–576. IEEE Computer Society Press, November 1994.

[Vad12] Salil P. Vadhan. *Pseudorandomness.* Now Publishers Inc., 2012. https://people.seas.harvard.edu/~salil/pseudorandomness/.