Efficient NIZKs from LWE via Polynomial Reconstruction and "MPC in the Head"

Riddhi Ghosal* Paul Lou † Amit Sahai ‡

Abstract. All existing works building non-interactive zero-knowledge (NIZK) arguments for NP from the Learning With Errors (LWE) assumption have studied instantiating the Fiat-Shamir paradigm on a *parallel repetition* of an underlying honest-verifier zero knowledge (HVZK) Σ protocol, via an appropriately built correlation-intractable (CI) hash function from LWE. This technique has inherent efficiency losses that arise from parallel repetition.

In this work, we show how to make use of the more efficient "MPC in the Head" technique for building an underlying honest-verifier protocol upon which to apply the Fiat-Shamir paradigm. To make this possible, we provide a new and more efficient construction of CI hash functions from LWE, using efficient algorithms for polynomial reconstruction as the main technical tool.

We stress that our work provides a new and more efficient "base construction" for building LWE-based NIZK arguments for NP. Our protocol can be the building block around which other efficiency-focused bootstrapping techniques can be applied, such as the bootstrapping technique of Gentry et al. (Journal of Cryptology 2015).

1 Introduction

A recent line of work instantiates the Fiat-Shamir heuristic by building correlation-intractable hash functions from the Learning With Errors (LWE) assumption [34,7,29], yielding the first Non-Interactive Zero-Knowledge (NIZK) protocols for NP from LWE. Such protocols are particularly desirable as LWE is believed to be hard even for quantum computers. While this line of work has been exciting in terms of achieving new feasibility based on LWE, our understanding of how to optimize the efficiency of such constructions is still in its infancy.

In particular, before our work, all known papers constructing NIZK arguments for NP from the LWE assumption studied instantiating the Fiat-Shamir paradigm on a *parallel repetition* of an underlying honest-verifier zero knowledge (HVZK) Σ protocol. Unfortunately, parallel repetition entails inherent efficiency loss. Can we do better?

*riddhi@cs.ucla.edu
†pslou@cs.ucla.edu
‡sahai@cs.ucla.edu

Our Work. In this work, we study how to apply the "MPC-in-the-Head" paradigm [30] to the construction of NIZK arguments for NP from the LWE assumption. Moreover, we do so by directly using simple and efficient polynomial reconstruction algorithms [37,27], avoiding the need for more complex coding previously used in [29][1] We note that this paradigm has previously been used to yield practically efficient constructions in other contexts [1,18,11].

The starting point: Zero Knowledge Protocols. A zero knowledge protocol [22] is an interactive protocol which allows a prover to prove to a verifier that an input x is in some NP language L without revealing anything more than the fact that $x \in L$. A classic example of such a protocol was introduced by Goldreich, Micali and Wigderson [21] for *Graph 3-Coloring*. The NP-completeness of Graph 3-Coloring implies that the GMW protocol indeed leads to zero knowledge proofs for all problem in NP. The basic version of this protocol is public coin and has large soundness error, but this error can be made negligible while still preserving *honest-verifier* zero-knowledge by *parallel repetition*. However, such parallel repetition is a source of significant inefficiency, both asymptotically and concretely. This is especially true if the number of parallel repetitions required is large – an issue that we will come back to later!

An alternative to using parallel repetition of such classic protocols is the MPC-in-the-head paradigm introduced by Ishai, Kushilevitz, Ostrovsky and Sahai [30], which allow us to construct highly sound general zero knowledge proof systems for any NP relation $R(x, w)$, where w is a witness to the fact that $x \in L$. Such a protocol makes black box use of an honest-majority MPC protocol Π_f for a functionality f for the circuit for NP relation R. This approach bypasses the computational overhead of a Karp reduction. Moreover, there is a successful line of work on producing highly efficient perfectly-robust MPC with minimal communication [13,14,24,3].

The MPC-in-the-head paradigm avoids the need for parallel repetition entirely. At a high level, the paradigm works by having the prover run the MPC protocol among q virtual servers entirely in the imagination of the prover, and then commit to the views of these virtual servers. The verifier then specifies a small random subset of these servers to the prover. The prover then opens the commitments to the inputs of the chosen servers, and all messages sent and received by those servers. This allows the verifier to check that the prover correctly executed the MPC protocol for almost all servers. It is absolutely crucial that the number of servers that the verifier specifies to open is significantly smaller than the number of servers q, otherwise no security would remain for the prover.

Using the Fiat-Shamir paradigm with Correlation-Intractable Hash Functions to obtain NIZK. A non-interactive zero knowledge protocol (NIZK) [19] lets the

[1] In personal correspondence after the initial posting of our result, Alex Lombardi showed us that it was possible to use the construction in [29] using Parvaresh-Vardy codes over extension fields to achieve parameters compatible with our variant of MPC-in-the-head, albeit at a significant efficiency cost relative to what we achieve here. Refer to Appendix A.1 for a detailed discussion.

prover eliminate the need for interaction by assuming a common random string (CRS[2]) that is given as input to both parties. A beautiful tool for constructing NIZKs is the Fiat Shamir heuristic [15]: it starts with a *public-coin honest-verifier zero knowledge proof* system and transforms it into a NIZK. This works by placing a random hash key in the CRS and replacing each of the verifier's messages in the interactive protocol with the hash of the input and the entire transcript so far. A sequence of works [8,9,7,5,34,28,31] has shown that if this hash function is *correlation-intractable* for certain relations, then the resulting NIZK is sound.

The recent work of [34,29] constructs such a correlation-intractable hash function from the LWE assumption and demonstrates how to apply the Fiat-Shamir transformation to a broad class of public-coin honest-verifier zero knowledge protocols built using parallel repetition. However, it is worth noting that the *number* of parallel repetitions needed for the technique of [29] to apply is actually a rather large polynomial. Specifically, if k is the security parameter for LWE and if the size of the verifier's challenge set is bounded by any polynomial in k, then the number of repetitions required is roughly $O(k^2)$ (though they note this can be optimized to $O(k^{1+\varepsilon})$). One crucial reason for this polynomial expression being $O(k^c)$, for $c > 1$, is that list-recoverable error correcting codes play a starring role in the work of [29], and unfortunately the best-known such codes require large block lengths to achieve the parameters needed for [29] to work[3].

Our New Idea in a Nutshell. Our starting technical observation is that the correlation that needs to be intractable for the hash function is in fact far *more structured* in the case of a variant of the MPC-in-the-head protocol that we consider, than in the case of parallel repetition based protocols. The looser structure of the correlation behind parallel repetition based protocols is what led to the work of [29] requiring general list-recoverable codes. The greater structure present in the case of MPC-in-the-head protocols allows us to significantly relax the requirements, and in particular lets us use an aggregate size analysis when decoding. As a result, we are able to use standard polynomial reconstruction algorithms [37,27] directly to solve our problem. To highlight this structure, we define a new variant of list-recoverability, that we call Recurrent List-Recoverability, over product sets where each term in the product is the same set.

Definition 1 (Recurrent List-Recoverable Codes). *An ensemble of codes* $\{\mathcal{C}_\lambda \colon \mathcal{M}_\lambda \to \mathbb{Z}_{q_\lambda}^{n_\lambda}\}$ *is said to be a* $(\ell(\cdot), L(\cdot))$-recurrent list recoverable *(for* $\ell, L \colon \mathbb{Z}^+ \to \mathbb{Z}^+$*) if there is a polynomial-time algorithm* $\mathsf{Recover}$ *that:*

- *Takes as input* $\lambda \in \mathbb{Z}^+$ *and explicit descriptions of "constraint" sets* $S \subseteq \mathbb{Z}_q^n$ *where* $|S| \leq \ell(\lambda)$.

[2]More generally, CRS can also refer to a common reference string, but our work will achieve NIZKs with a common random string.

[3]In particular, the alternative method pointed out to us by Lombardi using Parvaresh-Vardy codes over extension fields would also incur this $O(k^{1+\varepsilon})$, $\varepsilon > 0$ overhead. We show a more detailed computation in Section A.1

– *Produces as output a list of at most $L(\lambda)$ messages, containing all $m \in \mathcal{M}$ for which $\mathcal{C}(m)_i \in S$ for all $i \in [n]$.*

We show that this aggregate size analysis and polynomial reconstruction algorithms implies the existence of recurrent list-recoverable codes with the desired parameters, resulting in the following theorem.

Theorem 1. *(Restatement of Theorem 6). For arbitrary constants $0 < \eta, \alpha < 1$ and $0 < \delta \leq \varepsilon < 1$, there exists a probabilistic constructible ensemble for codes*

$$\left\{\mathcal{C}_k : \mathbb{Z}_{q^2}^{k+1} \to \mathbb{Z}_q^{\eta q}\right\}$$

such that $\mathcal{C}_k$ is $(\alpha q, T^2)$-Recurrent List Recoverable with probability at least $1 - e^{-\omega(k \log k)}$, where $q = k \log^{1+\varepsilon+\frac{\delta}{2}} k$ and $T = O(k \log^{2\varepsilon-\frac{\delta}{2}} k)$.

Main Technical Milestone: Quasi-linear blocklength. As noted above, the (ordinary) list-recoverable codes constucted in [29] have block length $O(k^{1+\varepsilon})$, for $\varepsilon > 0$, in the number of input symbols k above. In contrast, in our theorem above, we achieve quasi-linear blocklength $\tilde{O}(k)$. This improvement is despite using a qualitatively *weaker* algebraic component (polynomial reconstruction) in our codes compared to the one used previously (Parvaresh-Vardy codes over extension fields). We discuss why this is possible in our technical overview below.

Composing this recurrent list-recoverable code with the Peikert-Shiehian correlation intractable hash function allows us to instantiate the Fiat-Shamir technique with the MPC-in-the-head technique.

Theorem 2. *(Restatement of Theorem 8). Assuming that $\mathsf{LWE}_{\frac{m}{2\log q},m,q,\chi}$ holds for the particular parameter settings where χ is a B-bounded distribution for $B = q^{\Omega(1)}$, $q = \mathsf{poly}(k)$, k is the security parameter, and a MPC protocol with perfect αn-robustness and perfect, statistical, or computational security exists, where $\alpha \in (0, 1/2)$ is a constant and n is the size of the challenge set in the interactive protocol, there exists NIZKs with computational soundness for all of NP whose proof size is*

$$\mathcal{O}(|C| + q \cdot \mathrm{depth}(C)) + \mathsf{poly}(k)$$

where C is an arithmetic circuit for the NP verification function and $q = k \log^{1+\epsilon} k$ for any $\epsilon > 0$.

Bootstrapping. A NIZK with proof size $|w| + \mathsf{poly}(\lambda)$ for witness w and security parameter λ can be constructed using Fully Homomorphic Encryption [17] to bootstrap an underlying NIZK. Their construction uses this NIZK to prove that the fully homomorphic encryption key generation and evaluation is performed correctly by the Prover. Our construction provides an efficient base NIZK construction and can be used in conjunction with the construction of [17] to yield a more efficient form of this bootstrapping. Similarly, other (future) methods of bootstrapping for efficiency can potentially make use of our NIZK as a base construction.

1.1 Technical Overview

MPC-in-the-head An MPC protocol [4,12,20,38] allows us to compute a q-party functionality (a function of their inputs) while maintaining privacy of the inputs and correctness of the output. In a n-private MPC protocol, any adversary that corrupts at most n players is unable to learn any information about the non-corrupted players' private inputs beyond that obtainable from learning the output of the function. Zero-knowledge protocols can be viewed as a special case of secure two-party computation, where the function verifies the validity of a witness held by the prover.

Modifying the IKOS protocol Recall that we will be using the Fiat-Shamir paradigm (more on this below) to convert a public coin honest-verifier zero knowledge (HVZK) proof into a NIZK argument. All previous work studied using parallel repetition of a HVZK protocol. We aim to avoid this by starting with an HVZK protocol based on the MPC-in-the-head paradigm [30], as we explain next. The HVZK protocol we use slightly modifies the original protocol presented in [30] by asking the Prover to commit to a single copy of the transcript rather than commit to several (possibly overlapping) views. For any party the Verifier specifies to the Prover, the Prover opens up the relevant commitments in the transcript. The modification, not only simplifies the soundness proof, but ensures that each party's view can be independently verified rather than cross checking different party views for consistency of the views, as was the case in the original protocol. In this way, each party that the Verifier specifies constitutes an independently verifiable challenge. This property of independently verifiable challenges is necessary to cleanly define a single fixed bad challenge set S for the correlation-intractable hash function (the bad challenge space is $S \times S \times \ldots \times S$).

Let R_L be a relation corresponding to a NP language L. In other words, $R_L(x, w) = 1$ if and only if $x \in L$ and w is a witness for x. Define a functionality f_L such that $f_L(x, w_1, w_2, ..., w_q) = R_L(x, w_1 \oplus w_2 \oplus \cdots \oplus w_q)$. Thus, f_L can be viewed as a function computed by q parties where x is the public input and w_i is the private input for Player i. The HVZK protocol Π_{ZK} begins with the Prover carrying out all the steps of a q-party MPC protocol Π_{f_L} in her head. First, she secret shares w into $w_1, \ldots, w_q$ and executes the q-party MPC protocol to produce the protocol transcript of inputs, initial randomness, and messages sent. The Prover sends commitments to the transcript of the execution to the Verifier. Now the Verifier picks a random set S of $n < q$ parties, challenging the Prover to open the commitments to the private inputs, their randomness, and all messages sent or received by parties in S. The Verifier accepts if the openings form a consistent MPC protocol (that is, every message sent matches what the MPC's next message function would output given the previous messages received) and every party in the set S outputs 1.

The HVZK property follows from the privacy guarantee of the MPC. Assuming that the underlying MPC protocol Π_{f_L} is perfectly robust, violating the soundness requires a cheating prover to commit to many messages that are not

consistent with the rest of the transcript and we show in Lemma 2 that such a cheating prover gets caught with overwhelming probability.

Fiat-Shamir Heuristic We begin by reviewing the Fiat-Shamir Heuristic, a generic technique that compresses public-coin interactive arguments into non-interactive arguments in the CRS model. The Fiat-Shamir Heuristic is defined with respect to a public hash function family $\mathcal{H}$. Let us consider the following three-round interactive proof between a prover P and verifier V, in which P's goal is to convince V that $x \in \mathcal{L}$, for some language $\mathcal{L} \in \mathsf{NP}$:

1. P sends a first message α.
2. V responds with a uniform randomly chosen string β.
3. P finally sends a message γ to V.

Note that V accepts the proof (α, β, γ) if and only if $x \in \mathcal{L}$. In order to convert this to a non-interactive proof, the CRS consists of a randomly chosen hash function $h \leftarrow \mathcal{H}$. P computes $\beta = h(x, \alpha)$ and uses this compute γ. Finally, V can recompute β using the publicly known h and checks if the transcript $(x, \alpha, \beta, \gamma)$ is accepting.

This technique requires a careful analysis of soundness, because V no longer has the capability to generate uniformly random strings β. One way to ensure that the Fiat-Shamir transform is indeed sound is to instantiate the hash function with one that is *Correlation Intractable* (CI), which we now define.

Suppose $x \notin \mathcal{L}$. Let us define the set of "bad" βs as:

$$\mathsf{Bad}_\alpha = \{\beta \mid \exists \gamma \text{ such that } V(x, \alpha, \beta, \gamma) = 1\},$$

A CI hash requires that it is computationally infeasible for an efficient cheating prover to come up with an α such that $h(x, \alpha) \in \mathsf{Bad}_\alpha$ when given $h \leftarrow \mathcal{H}$ as input, where $\mathcal{H}$ is a Correlation Intractable hash family with respect to Bad_α. Formally, we say that $\mathcal{H}$ is a correlation intractable hash function family for Bad_α if for all PPT adversaries $\mathcal{A}$,

$$\Pr_{h \leftarrow \mathcal{H}}[h(x, \alpha) \in \mathsf{Bad}_\alpha \mid \mathcal{A}(h, x) = \alpha] \leq \mathsf{negl}(\lambda).$$

Peikert and Shiehian [34] constructed a CI hash family when $|\mathsf{Bad}_\alpha| = 1$ from the LWE assumption. In fact, Canetti et. al. [7] have shown that this construction can be extended to settings when $|\mathsf{Bad}_\alpha|$ is polynomially bounded.

Correlation Intractable Hash Functions from List Recoverable Codes In their recent work, [29] propose a correlation intractable hash function family for any *three round public coin commit and open protocol*. The classical GMW protocol for 3-coloring with parallel repetition falls in the category of the protocols that [29] dealt with. To illustrate the techniques from [29], we briefly review them in the context of parallel repetition of the basic GMW protocol.

In the GMW protocol, the Prover who knows a 3-coloring of a graph G first commits to a randomly chosen permutation on the 3-coloring. The Verifier then

randomly picks an edge of G and asks the Prover to open the vertex colors incident to that edge. If the colors differ, the Verifier accepts; otherwise, the verifier rejects. Repeating the interactive protocol in parallel achieves negligible soundness error while keeping the round complexity low. In any iteration of the interactive protocol there are at most $|E| - 1$ edges which can allow the prover to cheat (referred to as the "bad" challenge set). We define S_i to be the bad challenge set in the ith iteration of the interactive protocol. In a parallel repetition of the protocol n times, these bad challenge sets form a product of sets $S_1 \times \cdots \times S_n$, where $\forall\ i \in [n], |S_i| \leq |E| - 1$. For $G \notin$ 3-COL, a malicious Prover is able to convince the Verifier to accept if for all iterations $i \in [n]$ the challenge edges selected by the Verifier in the ith iteration belong to S_i. This product of sets defines a product relation $\mathcal{R} = S_1 \times \cdots \times S_n$.

The usefulness of CI hash families prior to the work of [29], such as those in [34,7], were limited to functions and polynomially bounded relations. Our relation $\mathcal{R}$ does not fall in this category as there may be exponentially many bad challenges on which an adversary can find the desired correlation. The work of [29] addresses this concern by constructing new correlation intractable hash functions for such product relations that are *efficiently verifiable* (defined in Section 6). In order to do so, they use *list recoverable codes* to construct another relation $\mathcal{R}'$ which is "efficiently enumerable" and therefore amenable to the techniques of [34,7].

To build this relation $\mathcal{R}'$, they use a derandomization approach based on list-recoverable error correcting codes. Informally, an error correcting code is a function $\mathcal{C} : \mathcal{M} \to \mathbb{Z}_q^n$. Here, n is called the block length of the code. We say that an error correcting code $\mathcal{C}$ is (ℓ, L)-list recoverable if for all sets $S_1, S_2, \ldots, S_n \subseteq \mathbb{Z}_q$ each of size at most ℓ, the number of messages v in $\mathcal{M}$ such that $\mathcal{C}(v) \in S_1 \times \cdots \times S_n$ is less than $L + 1$. Moreover, there must exist an efficient algorithm Recover which extracts all such v. This notion was introduced in [26]. The parameters of the codes can be interpreted as follows in the context of the GMW protocol:

- The size of the alphabet q is the maximum size of the Verifier's challenge set, i.e. $q = |E|$.
- The input list size ℓ is $|E| - 1$ which corresponds to the maximum size of a bad challenge set for a single execution of the GMW protocol.
- The block length n is the number of parallel repetitions.
- The output list size L must be polynomially bounded.

The new CI Hash function they construct is given by $\mathcal{H}' \coloneqq \mathcal{C}(\mathcal{H}(\cdot))$ where $\mathcal{C}$ is the list recoverable error correcting code as defined above and $\mathcal{H}$ is the previous CI hash function from [34].

Our recurrent list-recoverable codes achieve a quasi-linear block size of $O(k \log^{1+\epsilon} k)$ for arbitrary $\epsilon > 0$. We emphasize that this block size is not known to be achievable by any previous framework.

Exploiting the MPC-in-the-head Product Relation We first highlight the structure of the bad challenge set when using MPC-in-the-head to build a zero-

knowledge protocol. Consider a cheating Prover that simulates a q-party MPC protocol and corrupts an α fraction of them in an attempt to fool the Verifier. The Prover commits to a transcript of the execution (denoted by com). The Verifier then specifies n parties to the Prover. The Prover must decommit to the corresponding commitments to inputs and the randomness of the specified parties as well as the messages incident (sent or received) to these parties. Let $S_{\mathsf{com}} \subseteq [q]$ be the set of the parties for which the messages sent are consistent with the input, the randomness, and the previous messages received and where the final output of the party is 1. The *bad* challenge set (equivalently the bad challenge relation) that convinces a Verifier to accept, denoted by $\mathcal{R}_{\mathsf{MPC}} \subseteq [q]^n$, is therefore seen to be the product $\underbrace{S_{\mathsf{com}} \times \cdots \times S_{\mathsf{com}}}_{n \text{ times}}$. Observe that this product relation is a specific product relation where each component is the same set S_{com}. The special structure of the bad challenge set in the MPC-in-the-head setting opens up a new avenue for us to exploit in order to construct a CI hash for $\mathcal{R}_{\mathsf{MPC}}$.

Revisiting Random Codes A common technique in coding theory introduced by Forney in 1966 [16] is that of code concatenation. Code concatenation involves two codes, an inner code $\mathcal{C}_{in}$ and an outer code $\mathcal{C}_{out}$. The code concatenation encoding scheme first encodes a message m with the outer code $\mathcal{C}_{out}$ to produce $e = \mathcal{C}_{out}(m)$. Then it encodes each symbol in e with the inner code $\mathcal{C}_{in}$. We denote the resulting code as $\mathcal{C}_{out} \circ \mathcal{C}_{in}$ [4].

This technique was used by [29] to obtain list-recoverable codes. In particular, their list-recoverable codes result from concatenating an inner code, given by a family of random codes, with an outer code, given by an algebraic code instantiated by the Parvaresh-Vardy code [33]. The inner code reduces the size of the lists to be fed as input to the outer code, achieving an overall smaller block length. The question before us is: Can we use the inner code to help us reduce the size of the lists to be fed as input to the outer code, thereby helping us achieve an overall block length that is smaller than the input list size to the outer code?

Suppose we have a random code $\mathcal{C}_{\mathsf{rand}} : \mathbb{Z}_Q \to \mathbb{Z}_q^m$, where the parameters Q, q, m are all polynomial in the security parameter. Then a list recovery algorithm is trivial to implement by enumerating every codeword and checking to see if the components of the codeword lie in the input lists. If one analyzes the list recoverability of such a code, one immediately encounters a fundamental barrier: If ℓ is the input list size to the list recovery algorithm, then the output list size must also sometimes be at least ℓ. This is simply because the input lists can correspond to the union of ℓ different codewords in $\mathcal{C}_{\mathsf{rand}}$. Indeed, the work of [29] analyzed the list recoverability of a single random code further to show

[4] The standard notation for code concatenation $\mathcal{C}_{out} \circ \mathcal{C}_{in}$ differs in two ways from the standard function composition notation in which $f \circ g(x) = f(g(x))$. Firstly, $\mathcal{C}_{out}$ is used first to encode the message m. Secondly, $\mathcal{C}_{in}$ is applied index-by-index to each symbol in the $\mathcal{C}_{out}(m)$

that this worst case is close to tight, but as we noted above, their analysis is not good enough for us.

Can we exploit the fact that the inputs lists must all be equal, and equal to S_{com} in particular? Unfortunately the output list size of the random code must be at least ℓ/m, as the worst case S_{com} could be equal to the union of all the symbols found in ℓ/m codewords. This seems to present a fundamental barrier to us regarding the applicability of random codes as "inner" codes in concatenated codes, since the random code blows up the overall blocklength of the concatenated code by a factor of m, while only shrinking the list size by at most a factor of m. In other words, we seem to have made no progress.

Many random codes are better than one. The key insight behind our work is that while the barrier above applies to a *single* random code, a much different picture emerges if we consider the *sum* of the list sizes output by the recover algorithm of *many* random codes.

Indeed, suppose we have t completely independently chosen random codes $\mathcal{C}^{(i)}_{\mathsf{rand}} : \mathbb{Z}_Q \to \mathbb{Z}_q^m$ for $i \in [t]$. While it is true that for *each* code there exist input sets S_{com} that would lead to an output list of size ℓ/m, with overwhelming probability, these input sets would have tiny intersections because of the independence of the choice of each code. For $i \in [t]$, let L_i be the list obtained as output of the list recovery algorithm of $\mathcal{C}^{(i)}_{\mathsf{rand}}$ on input lists all equal to S_{com}. It is hopeless to get a better bound on $\max_i \{|L_i|\}$. So instead we aim to bound $\sum_i |L_i|$.

In our work, we give a new analysis of this quantity for t independently chosen random codes. We formulate a new variant of Chernoff's Bound (see Lemma 1), and use this to give our analysis in Theorem 5. This shows that with suitably chosen parameters, with overwhelming probability. for every input list S_{com}, $\sum_i |L_i|$ will be bounded by roughly $\tilde{O}(t + \ell/m)$. In other words, we get t output lists roughly for the "price" of a single output list!

Using Polynomial Reconstruction to leverage the aggregate list bound. Now that we have this bound, how can we take advantage of it to build a CI Hash function? We do so by departing from the language of list recoverability of error correcting codes, and instead adopting the more basic algebraic tool of polynomial reconstruction.

In the polynomial reconstruction problem, we are given as input a prime Q, a degree bound k, and n distinct pairs $\{(\alpha_i, y_i)\}_{i\in[n]}$ where each $\alpha_i, y_i \in \mathbb{Z}_Q$. The algorithm of Guruswami and Sudan [27] outputs a list of every polynomial f over $\mathbb{Z}_Q$ of degree at most k, such that $f(\alpha_i) = y_i$ for at least $\sqrt{kn}$ indices $i \in [n]$. Furthermore, this output list has size at most n^2. Combining polynomial reconstruction to leverage the aggregate list bound results in a recurrent list-recoverable code with the desired parameter settings.

The existence of this code and the Peikert-Shiehan correlation-intractable hash function gives rise to our final construction of a CI hash function as follows: Let $\mathcal{H}$ be the Peikert-Shiehan correlation-intractable hash and let α be the first message of the protocol (including the instance x being proven). Interpret $\mathcal{H}(\alpha)$

as coefficients for a degree k polynomial over field $\mathbb{Z}_Q$. Then use the evaluation map on this polynomial at t fixed distinct elements in $\mathbb{Z}_Q$ to yield the code $\mathcal{C}_{\mathsf{alg}} : \mathbb{Z}_Q^{k+1} \to \mathbb{Z}_Q^t$ to obtain t field elements in $\mathbb{Z}_Q$. We assume that we have already sampled t independent random codes $\mathcal{C}_{\mathsf{rand}}^{(i)} : \mathbb{Z}_Q \to \mathbb{Z}_q^m$ for $i \in [t]$ at setup time (this is part of the description of the hash function). Then we apply the ith random code $\mathcal{C}_{\mathsf{rand}}^{(i)}$ on the ith element of $\mathcal{C}_{\mathsf{alg}}(\mathcal{H}(\alpha))$. If $\mathscr{C}_{\mathsf{rand}} = \{\mathcal{C}_{\mathsf{rand}}^{(i)}\}_{i\in[t]}$, we denote this operation by $\mathcal{C}_k(\mathcal{H}(\alpha))$ where $C_k = (\mathcal{C}_{\mathsf{alg}} \circ \mathscr{C}_{\mathsf{rand}})$. This operation, $(\mathcal{C}_{\mathsf{alg}} \circ \mathscr{C}_{\mathsf{rand}})(\mathcal{H}(\cdot))$, defines our final construction of a CI hash function.

This construction indeed satisfies correlation-intractability by observing an efficient recovery algorithm for $(\mathcal{C}_{\mathsf{alg}} \circ \mathscr{C}_{\mathsf{rand}})(\mathcal{H}(\cdot))$. Namely a brute force enumeration of the codewords for the random codes in $\mathscr{C}_{\mathsf{rand}}$ gives an output list of size $\tilde{O}(t+\ell/m)$ that consists of pairs $\{(\alpha_i, y_i)\}_i$. Of these, at most t pairs can be consistent with a degree-k polynomial. The polynomial reconstruction algorithm of [27] will succeed as long as $t > \sqrt{k \cdot \tilde{O}(t+\ell/m)}$. This provides us with ample room to set parameters, and indeed we have significant freedom when choosing values of k, t, ℓ, m to make this work. Then the polynomial reconstruction algorithm outputs at most $\tilde{O}(t^2 + \ell^2/m^2)$ many polynomials. Therefore this efficient recovery algorithm produces a polynomial-size set so the Peikert-Shiehian CI hash function can now be applied, yielding a CI hash function for the MPC-in-the-head setting, achieving our goal. In the remainder of the paper, we show how to instantiate parameters precisely and provide all details regarding our analysis.

2 Preliminaries

2.1 Proof Systems

Zero Knowledge: We define the standard notion of zero knowledge as well known in prior work [23,21,30].

An NP Relation $R(x, w)$ is an efficiently decidable binary relation which can be viewed as a boolean function that outputs 0 or 1. Any NP relation defines a language $L = \{x : \exists w, R(x, w) = 1\}$. A zero knowledge proof consists of two PPT algorithms, namely, a prover P and verifier V. The prover is given access to instance x and witness w, whereas the verifier only has the instance w.

Definition 2 (Interactive Honest Verifier Zero Knowledge Proof). *The protocol (P, V) for a language L defined above consists of an interactive P and V with the following requirement:*

- *Completeness: If $x \in L$, and both P, V are honest, then V must always accept.*
- *Statistical Soundness: If $x \notin L$, then for any malicious and computationally unbounded prover P^*, V accepts with a negligible probability only.*
- *Zero Knowledge: If $x \in L$, then for any non-malicious PPT verifier V^*, there exists a PPT simulator M such that the view of V^* upon interaction with P is computationally indistinguishable from the output distribution of*

$M(x)$. *Here, view of* V^* *consists of its input* x*, its random coins and all incoming messages.*

Definition 3 (Public Coin). *An interactive proof system is said to be public coin if for every* $x \in \{0,1\}^n$*, and some* $l(n)$*, the messages sent by an honest verifier* V *are i.i.d uniform* $l(n)$ *bit strings. Moreover, the final output of* V *must be efficiently computable in polynomial time given* x *and the transcript upon interaction with* P*.*

Definition 4 (Non-Interactive Zero Knowledge(NIZK) Arguments in the CRS model). *A non interactive zero knowledge argument for a language* L *in the Common Reference String (CRS) model is defined three PPT algorithms:*

- $\mathsf{Setup}(1^n, 1^\lambda)$ *outputs a uniform random string* crs *given a statement of length* n *and security parameter* λ*.*
- *Prover* $P(\mathsf{crs}, x, w)$ *outputs a proof* π *given a statement witness pair* (x, w) *in the NP relation* R*.*
- *Verifier* $V(\mathsf{crs}, x, \pi)$ *either accepts or rejects.*

The following properties must be satisfied:

- *Completeness:* $V(\mathsf{crs}, x, \pi)$ *must always accept if* $x \in L$ *and* $\pi \leftarrow P(\mathsf{crs}, x, w)$*.*
- *Computational Soundness: for every non-uniform* poly *time prover* P^**, there exists a negligible function* $\epsilon(\lambda)$ *such that for any* $n \in \mathbb{N}$ *and* $x \notin L$*,*

$$\Pr[\mathsf{crs} \leftarrow \mathsf{Setup}(1^n, 1^\lambda), \pi^* \leftarrow P(\mathsf{crs}, x), V(\mathsf{crs}, x, \pi^*) \text{ accepts}] \leq \epsilon(\lambda).$$

- *Non Interactive Zero Knowledge: There exists a PPT simulator* M *such that for every* $x \in L$ *such that the distribution of the transcript output by* Setup *and* P*, i.e.,* $(\mathsf{crs}, P(\mathsf{crs}, x, w)) : \mathsf{crs} \leftarrow \mathsf{Setup}(1^n, 1^\lambda)$ *is statistically indistinguishable from the output of* $\mathsf{M}(x)$*. Note that* M *is allowed to generate its own CRS.*

2.2 Cryptographic Assumptions and Commitment Schemes

Definition 5 (Decisional Learning with Errors Problem [36]). *Let* $n \geq 1$ *be a parameter for dimension, and let* $q = q(n) \geq 2$ *be a modulus. Let* $m \geq 1$ *be a parameter for number of samples. Let* $\chi = \chi(n)$ *be an error distribution over* $\mathbb{Z}_q$*. The decisional learning with errors problem* $\mathsf{LWE}_{n,m,q,\chi}$ *is to distinguish between the following two distributions:*

$$\left\{(A, As + e) \mid A \xleftarrow{\$} \mathbb{Z}_q^{m \times n}, s \xleftarrow{\$} \mathbb{Z}_q^n, e \xleftarrow{\$} \chi^m\right\}$$

and

$$\left\{(A, u) \mid A \xleftarrow{\$} \mathbb{Z}_q^{m \times n}, u \xleftarrow{\$} \mathbb{Z}_q^m\right\}$$

Definition 6 (Bounded Error Distributions). *Let* $B = B(\lambda)$ *such that* $B(\lambda) \in \mathbb{N}$*. We say that a family of distributions* $\chi = \{\chi_\lambda\}_{\lambda \in \mathbb{N}}$ *over the integers is* B*-bounded if for all* $\lambda \in \mathbb{N}$*,*

$$\Pr\left[x \leftarrow \chi_\lambda \mid |x| \leq B(\lambda)\right] = 1.$$

Definition 7 (Statistically Binding Commitment Scheme in the CRS model). *A Statistically binding commitment scheme in the CRS model is a pair of efficiently computable functions (*$\mathsf{Setup}, \mathsf{Com}$*), where,*

- $\mathsf{Setup}(1^\lambda)$ *outputs a common reference string* crs*.*
- $\mathsf{Com}(\mathsf{crs}, m; r)$ *takes as input* crs*, a message* m *to be commited, and uses randomness* r *to output a commitment* com*.*

They have the following security properties:

- ***Statistical Binding:*** *With high probability over the choice of* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*, there does not exists* r_0, r_1*, and messages* $m_0 \neq m_1$ *such that* $\mathsf{Com}(\mathsf{crs}, m_0; r_0) = \mathsf{Com}(\mathsf{crs}, m_1; r_1)$*.*
- ***Computational Hiding:*** *For messages* $m_0 \neq m_1$*, and randomness* r_0, r_1 *the distribution of* $(\mathsf{crs}, \mathsf{com}_0)$ *is computationally indistinguishable from* $(\mathsf{crs}, \mathsf{com}_1)$*. Here,* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{com}_0 \leftarrow \mathsf{Com}(\mathsf{crs}, m_0; r_0)$*, and* $\mathsf{com}_1 \leftarrow \mathsf{Com}(\mathsf{crs}, m_1; r_1)$*.*

Given a commitment com *and* crs*, a valid corresponding pair* (m, r) *is known as the* opening *for* com*.*

Remark 1. [Non-interactive Perfectly Binding Commitment Schemes from LWE-based PKEs] Any PKE with perfect decryption correctness gives a non-interactive commitment. As observed previously [32], this perfect decryption correctness implies perfect binding even though the committer is allowed to choose the public key maliciously. Since LWE with polynomial modulus-to-noise ratio under a bounded error distribution gives Regev encryption with perfect decryption error [2], it also gives non-interactive perfectly binding, computationally hiding non-interactive commitments.

2.3 Error Correcting Codes

Definition 8. *A* q-ary code *is a function* $\mathcal{C} \colon \mathcal{M} \to \mathbb{Z}_q^n$*, where* n *is called the* block length*,* $\mathcal{M}$ *is called the* message space*, and* $\mathbb{Z}_q$ *is called the* alphabet *of* $\mathcal{C}$*.*

Definition 9 (List-Recoverable Codes [26,27,25]). *An ensemble of codes* $\{\mathcal{C}_\lambda \colon \mathcal{M}_\lambda \to \mathbb{Z}_{q_\lambda}^{n_\lambda}\}$ *is said to be a* $(\ell(\cdot), L(\cdot))$-list recoverable *(for* $\ell, L \colon \mathbb{Z}^+ \to \mathbb{Z}^+$*) if there is a polynomial-time algorithm* $\mathsf{Recover}$ *that:*

- *Takes as input* $\lambda \in \mathbb{Z}^+$ *and explicit descriptions of "constraint" sets* $S_1, \ldots, S_n \subseteq \mathbb{Z}_q^n$ *with each* $|S_i| \leq \ell(\lambda)$*, and*
- *produces as output a list of at most* $L(\lambda)$ *messages, containing all* $m \in \mathcal{M}$ *for which* $\mathcal{C}(m)_i \in S_i$ *for all* $i \in [n]$*.*

Definition 10 (N-independent Concatenated Code). *Let* $\mathscr{C} = \left\{\mathcal{C}_1^{(2)}, \ldots, \mathcal{C}_N^{(2)}\right\}$ *be a collection of* N *codes where for* $i \in [N]$*,* $\mathcal{C}_i^{(2)} : \mathbb{Z}_Q \to \mathbb{Z}_q^m$*. Let* $\mathcal{C}^{(1)} : \mathcal{M} \to \mathbb{Z}_Q^N$ *be a code. The* N-independent concatenated code $\mathcal{C}^{(1)} \circ \mathscr{C} \colon \mathcal{M} \to \mathbb{Z}_q^{Nm}$ *is defined by*

$$(\mathcal{C}_1 \circ \mathscr{C})(x)_{(i-1)m+j} = \mathcal{C}_i^{(2)}\left(\left(\mathcal{C}^{(1)}(x)\right)_i\right)_j,$$

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

Definition 11 (Reed-Solomon codes [35]). *A Reed-Solomon code* $\mathcal{C}_\lambda \colon \mathbb{Z}_Q^{k+1} \to \mathbb{Z}_Q^t$ *is parameterized by a base field size* $q = q(\lambda)$*, a degree* $d = k(\lambda)$*, a block length* $t = t(\lambda)$*, and a set of values* $Q_\lambda = \{\alpha_1, \ldots, \alpha_t\}$*.* $\mathcal{C}_\lambda$ *takes as input a polynomial* p *of degree* k *over* $\mathbb{Z}_q$*, represented by its* $k+1$ *coefficients, and outputs the vector of evaluations* $(p(\alpha_1), \ldots, p(\alpha_t))$ *of* p *on each of the points* α_i*.*

We look into the problem of list recovery for Reed-Solomon Codes for our desired parameters. Note that as mentioned in section 1.1, the primary challenge for us is to have list recoverability of Reed-Solomon with list sizes larger than what is standard in the error correcting codes world. We point out that the problem of list recovery for Reed-Solomon Codes boils down to the following notion of *polynomial reconstruction* due to Sudan's algorithm [37].

Polynomial Reconstruction

- **INPUT:** Integers k_p and n_p distinct pairs $\{(\alpha_i, y_i)\}_{i \in [n_p]}$, where $\alpha_i, y_i \in \mathbb{Z}_Q$.
- **OUTPUT:** A list of all polynomials $p(X) \in \mathbb{Z}_Q[X]$ of degree at most k_p which satisfy $p(\alpha_i) = y_i, \ \forall \ i \in [n_p]$.

This polynomial reconstruction can be performed efficiently by interpolation. We refer readers to Chapter 4 of [25] for a detailed analysis of the algorithm and how to use it for list recovery. In this work we use the following theorem from Guruswami and Sudan [27] as a black-box.

Definition 12 (Agreement Parameter). *For a Reed-Solomon Code* $\mathcal{C}_{\mathsf{alg}} : \mathbb{Z}_Q^{k+1} \to \mathbb{Z}_Q^t$*, the* L *many reconstructed polynomials* $\{p_j\}_{j \in [L]}$ *are said to have an agreement parameter* $t_A \leq t$ *if* $\forall j \in [L], p_j(\alpha_i) = y_i$ *for at least* t_A *many pairs* $(\alpha_i, y_i), i \in [t]$*.*

Note that $t_A = t$ *denotes the case of perfect polynomial reconstruction which is the setting of interest in this work.*

Theorem 3 (Efficient Polynomial Reconstruction of Reed-Solomon Codes).

The polynomial reconstruction problem with n_p *input pairs, degree* k_p*, and* agreement parameter t_A *can be solved in polynomial time as long as* t_A *is at least* $\sqrt{k_p \cdot n_p}$*. Furthermore, at most* n_p^2 *polynomials will be output by the algorithm.*

2.4 Correlation Intractable Hash Function Family and the Fiat-Shamir Transform

We present this section by following the same flavor as [29].

Definition 13 (Hash Family). *A hash family is a collection* $\mathcal{H} = \{h_\lambda \colon I_\lambda \times X_\lambda \to Y_\lambda\}_\lambda$ *of keyed hash functions such that* $\{I_\lambda\}$ *is uniformly* $\mathsf{poly}(\lambda)$*-time sampleable and* $\{h_\lambda\}$ *is uniformly* $\mathsf{poly}(\lambda)$*-time evaluable. We will also write* $\mathcal{H}_\lambda$ *to denote the distribution on functions* $h_\lambda(i, \cdot)$ *obtained by sampling* $i \in I_\lambda$*.*

Definition 14 (Correlation-Intractability [10]). *For a hash family $\mathcal{H} = \{h_\lambda \colon I_\lambda \times X_\lambda \to Y_\lambda\}_\lambda$ and a relation ensemble $R = \{R_\lambda \subseteq X_\lambda \times Y_\lambda\}$, the correlation intractability game is the following game, played by any adversary $\mathcal{A}$ against a fixed challenger $\mathcal{C}$:*

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

We say that $\mathcal{H}$ is correlation intractable for R if every nonuniform poly-time $\mathcal{A}$ wins the correlation-intractability game only with probability negligible in the security parameter λ.

Definition 15. *Let Π be a public coin interactive protocol where the messages exchanged between P and V are denoted by $(\alpha_1, \beta_1, \ldots, \alpha_r, \beta_r)$ for r rounds of interaction. Here α_i and β_i denote messages sent by P and V respectively. If the verifier's messages are l bits long, then for a hash function family $\mathcal{H} : \{0,1\}^* \to \{0,1\}^l$, we define $FS_\mathcal{H}[\Pi]$ to be the non interactive protocol by sampling a common reference string $h \leftarrow \mathcal{H}$ and computing the message β_i if V as $h(x, \alpha_1, \beta_1, \ldots, \alpha_i)$. The verifier for $FS_\mathcal{H}(\Pi)$ accepts iff the verifier for the interactive protocol accepts and all β_i are correctly computed.*

Definition 16 (FS Compatible). *We say that a hash function family $\mathcal{H}$ is FS- compatible for an interactive proof Π for language L if the non interactive protocol $FS_{\mathcal{H}(\Pi)}$ defined above is a non interactive argument.*

2.5 Secure Multiparty Computation (MPC)

We define the standard notion of a Multiparty Computation along with some of the necessary properties of a MPC protocol necessary in our work. All the definitions are standard in literature [6,30,19].

Definition 17 (q-Party Protocol). *Let $P_1, \ldots, P_q$ be q parties, and let each P_i each have a shared public input x, a private input w_i, and private randomness r_i. Let $m_j^{(i)}$ be the messages received by party P_i in the j^{th} round. We specify a q-party protocol by its next message function NEXT which on input $(1^\lambda, i, x, w_i, r_i, (m_1^{(i)}, \ldots, m_j^{(i)}))$ where λ is the security parameter, outputs all messages sent or output by P_i in round $j+1$ given inputs x, w_i, r_i and round messages $(m_1^{(i)}, \ldots, m_j^{(i)})$.*

Definition 18 (View of a Party). *The view V_i of a party P_i during protocol Π contains common input x, private input w_i, randomness r_i, its received messages $\{m_j^{(i)}\}$, and all messages sent or output by P_i.*

Definition 19 (Transcript of an Execution). *The transcript Ξ of an execution of a q-party protocol Π is a set containing the public input, every party's randomness r_i, every party's private input w_i, every message sent in each round.*

Definition 20 (Correctness).

Let f be a deterministic functionality that on inputs $(x, w_1, \ldots, w_q)$ outputs $(f(x, w_1, \ldots, w_q))_{i \in q}$. We say that a q-party protocol Π_f realizes f with perfect (respectively statistical) correctness if for all inputs $(x, w_1, \ldots, w_q)$, the probability that there exists an $i \in [q]$ such that the output of party P_i is not equal to $f(x, w_1, \ldots, w_q)$ is 0 (respectively $\mathsf{negl}(\lambda)$).

Definition 21 (n-Privacy). *Let $1 \leq n < q$. We say that Π_f realizes f with perfect (respectively statistical) n-privacy if there is a PPT simulator Sim such that for all inputs $x, w_1, \ldots, w_q$ and every set of corrupted players $T \subseteq [q]$ where $|T| \leq n$, the joint views $\{V_i\}_{i \in T}$ of players in T is distributed identically (respectively statistically close) to $\mathsf{Sim}(T, x, (w_i)_{i \in T}, (f_i(x, w_1, \ldots, w_q))_{i \in T})$.*

Definition 22 (n-Robustness (imported from [30]). *We say that Π_f realizes f with perfect (resp., statistical) n-robustness if it is perfectly (resp., statistically) correct in the presence of a semi-honest adversary as in Definition 20, and furthermore for any computationally unbounded malicious adversary corrupting a set T of at most n players, and for any inputs $(x, w_1, \ldots, w_q)$, the following robustness property holds. If there is no $(w'_1, \ldots, w'_q)$ such that $f(x, w'_1, \ldots, w'_q) = 1$, then the probability that some uncorrupted player outputs 1 in an execution of Π_f in which the inputs of the honest players are consistent with $(x, w_1, \ldots, w_n)$ is 0 (resp., is negligible in λ).*

Efficiently Instantiable Perfectly Robust MPC Protocol

Remark 2. Several previous works give perfectly robust communication-efficient MPC protocols [14,3,24].

Theorem 4 (Theorem 7 from [24]). *In the client-server model, let c denote the number of clients, and $n = 2s + 1$ denote the number of parties (servers). Let k be the security parameter and let $\mathbb{F}$ denote a finite field. For an arithmetic circuit C over $\mathbb{F}$ and for all $1 \leq o \leq s$, there exists an information-theoretic MPC protocol which securely computes the arithmetic circuit C in the presence of a semi-honest adversary controlling up to c clients and $s - o + 1$ parties. The communication complexity of this protocol is $\mathcal{O}(|C| \cdot n/k + n \cdot (c + \mathrm{depth}(C)) + n^5 \cdot k)$ elements in $\mathbb{F}$.*

Remark 3. The client-server generalizes the standard MPC model of parties. To translate this communication complexity into the standard MPC model, every party has a single client and single server so if there are q parties there are q clients and q servers. Choose $o = s$, then in the standard MPC model, the communication complexity is given by,

$$\mathcal{O}(|C| + q \cdot \mathrm{depth}(C)) + \mathsf{poly}(k).$$

where $o, k, |C|$ are as defined in the previous theorem.

Remark 4. The protocol defined above was proved to have perfect security in the Universal Composability (UC) Model [6].

3 A Chernoff bound

In our work, we will analyze the sum of n Bernoulli random variables X_i where the probability p that $X_i = 1$ is much smaller than $1/n$. We derive a "custom" Chernoff bound that is useful for this case:

Lemma 1 (Chernoff for Bernoulli distributions $\mathsf{Ber}(p)$ with small p). *For $i \in [n]$ let $X_i \sim \mathsf{Ber}(p)$ be independent identically distributed Bernoulli random variables for $p = p(n) \in (0, 1]$. Let $X \triangleq \sum_{i=1}^{n} X_i$. Then for $t \geq 0$, we have:*

$$\Pr[X - np \geq t] \leq \left(\frac{1}{e} + \frac{t}{enp}\right)^{-t}$$

Proof. Let $\tau = np + t$. For tidiness, we use the notation $\exp(a)$ to denote e^a for any $a \in \mathbb{R}$. For all $\lambda \geq 0$, by Markov's inequality,

$$\begin{aligned}
\Pr[X \geq \tau] &\leq \frac{\mathbb{E}\left[e^{\lambda X}\right]}{e^{\lambda\tau}} \\
&= \frac{\left(pe^{\lambda} + (1-p)\right)^n}{e^{\lambda\tau}} \\
&= \frac{\left(1 + p\left(e^{\lambda} - 1\right)\right)^n}{e^{\lambda\tau}} \\
&\leq \frac{\exp\left(np(e^{\lambda} - 1)\right)}{\exp(\lambda\tau)} \\
&= \exp\left(np\left(e^{\lambda} - 1\right) - \lambda(np + t)\right).
\end{aligned}$$

Minimizing for $\lambda \geq 0$, we choose $\lambda = \ln(1 + t/np)$. Plugging in for λ gives,

$$\begin{aligned}
\exp\left(np\left(e^{\lambda} - 1\right) - \lambda(np + t)\right) = e^t\left(1 + \frac{t}{np}\right)^{-(t+np)} &\leq e^t\left(1 + \frac{t}{np}\right)^{-t} \\
&= \left(\frac{1}{e} + \frac{t}{enp}\right)^{-t}.
\end{aligned}$$

This immediately yields:

Corollary 1. *For $i \in [n]$ let $X_i \sim \mathsf{Ber}(p)$ be independent identically distributed Bernoulli random variables for $p = p(n) \in (0, 1]$. Let $X \triangleq \sum_{i=1}^{n} X_i$. Then for $t > enp$,*

$$\Pr[X - np \geq t] \leq \left(\frac{t}{enp}\right)^{-t}.$$

4 Recurrent List Recoverable Error Correcting Codes

We present a new notion of *Recurrent List Recoverable* error correcting codes by N-independent concatenating Reed Solomon with random codes. This is a

special case of general list recoverability of concatenated codes which we shall formally define later in the section. First, we introduce *Aggregate List Recovery* for Random Codes where a collection of independent random codes have identical constraint sets which are input to their corresponding Recover algorithm.

4.1 Aggregate List Recoverability of Random Codes

Definition 23 (Aggregate List Recoverability). *Given a collection of t independent codes $\{C_j : \mathbb{Z}_Q \to \mathbb{Z}_q^n\}_{j=1}^t$, we say that they are (t, ℓ, T)-aggregate list recoverable if the constraint sets $S_{j1}, \ldots, S_{jn}$ that the* Recover *algorithm corresponding to the j^{th} code takes as input are such that $\forall i\ \forall j, S_{ji} = S$ and $|S| \leq \ell$. Furthermore the output list for* Recover *of the j^{th} code is of size L_j, where $\sum_{j\in[t]} L_j \leq T$.*

Theorem 5 (Aggregate List Recoverability of t independent random codes). *Let $\{\mathcal{C}_{\mathsf{rand},i} \colon \mathbb{Z}_Q \to \mathbb{Z}_q^m\}_{i\in[t]}$ be a collection of t independent random codes, and assume that there exist $\varepsilon, \delta, \alpha, T$ such that the following hold,*

- $q = k \log^{1+\varepsilon+\frac{\delta}{2}} k$, $\varepsilon > \delta > 0$,
- $t = k \log^{\varepsilon} k$
- $Q = q^2$,
- $l = \alpha q$, *for some constant* $\alpha \in (0, 1)$
- $T \leq \frac{1}{k^2 \log^{2+2\varepsilon+\delta} k} + k \log^{2\varepsilon-\frac{\delta}{2}} k$, *and*
- $\alpha^m \leq \frac{1}{q^4 t}$,

then t of such independent random codes are (t, l, T)-aggregate list recoverable with probability at least $1 - e^{-\omega(k \log k)}$.

Proof. Given a function $\mathcal{C}_{\mathsf{rand},i} \colon \mathbb{Z}_Q \to \mathbb{Z}_q^m$, let $S \subseteq \mathbb{Z}_q$ be a subset of size l. Let $X_{i,x}$ be an indicator variable such that,

$$X_{i,x} = \begin{cases} 1 & \text{if } (\mathcal{C}_{\mathsf{rand},i}(x))_j \in S, \forall, j \in [m], \\ 0 & \text{otherwise} \end{cases}$$

Thus, $T = \sum_{i,x} X_{i,x}$. Now, $\Pr[X_{i,x} = 1] = \frac{|S|}{q} = \alpha^m$, where the probability is taken over the choice of the set S. Thus, $E[T] = Qt\alpha^m$.

A direct application of Corollary 1 immediately gives an upper bound on the size of T. We have,

$$\Pr[T - Qt\alpha^m \geq k_0] \leq \left(\frac{k_0}{eQt\alpha^m}\right)^{-k_0}.$$

Plugging in Q, α^m, t, k_0 as $q^2, \frac{1}{q^4 t}, k \log^{\epsilon} k, k \log^{2\epsilon-\frac{\delta}{2}} k$ respectively, we get,

$$\begin{aligned} \Pr[T \geq \frac{1}{k^2 \log^{2+2\varepsilon+\delta} k} + k \log^{2\varepsilon-\frac{\delta}{2}} k] &\leq \left(\frac{q^2 k_0}{e}\right)^{-k \log^{2\varepsilon-\frac{\delta}{2}} k} \\ &\leq \left(\frac{k^3 \log^{2+4\varepsilon+\frac{\delta}{2}} k}{e}\right)^{-k \log^{2\varepsilon-\frac{\delta}{2}} k}. \end{aligned}$$

Taking a union bound over all choices of S, the probability that there exists a set S for which the size of T is greater than $\frac{1}{k^2 \log^{2+2\varepsilon+\delta} k} + k \log^{2\varepsilon-\frac{\delta}{2}} k$ is upper bounded by,

$$
\begin{aligned}
&\binom{q}{\alpha q} \left(\frac{k^3 \log^{2+4\varepsilon+\frac{\delta}{2}} k}{e} \right)^{-k \log^{2\varepsilon-\frac{\delta}{2}} k} \\
&\leq \left(\frac{e}{\alpha}\right)^{\alpha q} \left(\frac{k^3 \log^{2+4\varepsilon+\frac{\delta}{2}} k}{e} \right)^{-k \log^{2\varepsilon-\frac{\delta}{2}} k} \\
&= \frac{\exp\left\{\alpha q - \alpha q \ln \alpha + k \log^{2\varepsilon-\frac{\delta}{2}} k\right\}}{\left(k^3 \log^{2+4\varepsilon+\frac{\delta}{2}} k\right)^{k \log^{2\varepsilon-\frac{\delta}{2}} k}} \\
&= \frac{\exp\left\{\alpha' q + k \log^{2\varepsilon-\frac{\delta}{2}} k\right\}}{\left(k^3 \log^{2+4\varepsilon+\frac{\delta}{2}} k\right)^{k \log^{2\varepsilon-\frac{\delta}{2}} k}} \qquad \text{where, } \alpha' = \alpha(1 - \ln \alpha) \\
&= \exp\left\{ \alpha' k \log^{1+\varepsilon+\frac{\delta}{2}} k + \tilde{k} - \tilde{k} \ln \tilde{k} - \tilde{k} \ln \left(k^2 \log^{2+2\varepsilon+\delta} k \right) \right\} \\
&\qquad\qquad \text{where, } \tilde{k} = k \log^{2\epsilon-\frac{\delta}{2}} k \\
&= \exp\left\{ \tilde{k} \Big(\alpha' \log^{1-\varepsilon+\delta} k + 1 - \ln \tilde{k} - \ln \left(k^2 \log^{2+2\varepsilon+\delta} k \right) \Big) \right\} \\
&= \exp\left\{ \tilde{k} \left(\alpha' \log^{1-\varepsilon+\delta} k + 1 - 3 \ln k - \ln \left(\log^{4\varepsilon+2+\frac{\delta}{2}} k \right) \right) \right\} \\
&= \exp\left\{ \tilde{k} \left(-\omega(\log k) \right) \right\} \\
&= \exp\left\{ -\omega\left(k \log k \right) \right\}
\end{aligned}
$$

Thus, the probability that $\mathcal{C}_{\mathsf{rand},i}$ are $(\alpha q, L_i)$-list recoverable such that $\sum_i L_i \leq \frac{1}{k^2 \log^{2+2\varepsilon+\delta} k} + k \log^{2\varepsilon-\frac{\delta}{2}} k$ is at least $1 - e^{-\omega(k \log k)}$.

4.2 Recurrent List Recoverability

We first define recurrent list-recoverability as a special case of list-recoverability where the sets are identical, $S_1 = \ldots = S_n$.

Definition 24 (Recurrent List-Recoverable Codes). *An ensemble of codes $\{\mathcal{C}_\lambda \colon \mathcal{M}_\lambda \to \mathbb{Z}_{q_\lambda}^{n_\lambda}\}$ is said to be a $(\ell(\cdot), L(\cdot))$-*recurrent list recoverable *(for $\ell, L \colon \mathbb{Z}^+ \to \mathbb{Z}^+$) if there is a polynomial-time algorithm* Recover *that:*

- *Takes as input $\lambda \in \mathbb{Z}^+$ and explicit descriptions of "constraint" sets $S \subseteq \mathbb{Z}_q^n$ where $|S| \leq \ell(\lambda)$.*

– *Produces as output a list of at most $L(\lambda)$ messages, containing all $m \in \mathcal{M}$ for which $\mathcal{C}(m)_i \in S$ for all $i \in [n]$.*

Theorem 6. *For arbitrary constants $0 < \eta, \alpha < 1$ and $0 < \delta \leq \varepsilon < 1$, there exists a probabilistic constructible ensemble for codes*

$$\left\{\mathcal{C}_k : \mathbb{Z}_{q^2}^{k+1} \to \mathbb{Z}_q^{\eta q}\right\}$$

such that $\mathcal{C}_k$ is $(\alpha q, T^2)$-Recurrent List Recoverable with probability at least $1 - e^{-\omega(k \log k)}$, where $q = k \log^{1+\varepsilon+\frac{\delta}{2}} k$ and $T = O(k \log^{2\varepsilon-\frac{\delta}{2}} k)$

Proof. Let $\mathscr{C}$ be a collection of t independent random codes $\{\mathcal{C}_{\mathsf{rand},i} \colon \mathbb{Z}_Q \to \mathbb{Z}_q^m\}_{i\in[t]}$ with $t = k \log^{\varepsilon} k$, $Q = q^2$ and m such that $\alpha^m \leq \frac{1}{q^4 t}$. Then, Theorem 5 tells us that with parameters set as above, the collection $\mathscr{C}$ is $(t, \alpha q, T)$- aggregate list recoverable with probability at least $1 - e^{-\omega(k \log k)}$, for $T \leq \frac{1}{k^2 \log^{2+2\varepsilon+\delta} k} + k \log^{2\varepsilon-\frac{\delta}{2}} k$.

Let $\mathcal{C}_{\mathsf{alg},k} : \mathbb{Z}_Q^{k+1} \to \mathbb{Z}_Q^t$ be a Reed Solomon Code. Theorem 3 tells us that if $\mathcal{C}_{\mathsf{alg},k}$ is a Reed Solomon Code, then $O(k^2 \log^{4\varepsilon-\delta} k)$ polynomials can be recovered by polynomial reconstruction as long as $t \geq \sqrt{k \cdot T}$, where T is the total number of input pairs. Choose $T = O(k \log^{2\epsilon-\frac{\delta}{2}} k)$ and $t = k \log^{\varepsilon} k$,then the necessary condition is satisfied. Thus, we can feed this list T to the polynomial reconstruction algorithm of $\mathcal{C}_{\mathsf{alg},k}$.

Combining these two results and our choice of parameters which satisfy the list recoverability constraint for Reed-Solomon in Theorem 3, we get that polynomial reconstruction outputs a list Lst of size $O(k^2 \log^{4\varepsilon-\delta} k)$. Moreover, our choice of parameter ensures that there exists a constant $0 < \eta < 1$ such that $mt = \frac{2k \log^{1+\epsilon} k + 23k \log k \log\log k}{\log \frac{1}{\alpha}} \leq \eta k \log^{1+\epsilon} k$.

Thus, our code ensemble $\mathcal{C}_k$ can be constructed by an t-independent concatenation of $\mathcal{C}_{\mathsf{alg},k}$ with $\mathscr{C}$, i.e., $\mathcal{C}_k = \mathcal{C}_{\mathsf{alg},k} \circ \mathscr{C}$. To elaborate further, according to Definition 10, we first apply $\mathcal{C}_{\mathsf{alg},k}$ on a message $m \in \mathbb{Z}_{q^2}^{k+1}$. This produces $\mathcal{C}_{\mathsf{alg},k}(m) \coloneqq (m'_1, \ldots, m'_t) \in \mathbb{Z}_Q^t$. The final code output is then $\mathcal{C}_k = \mathcal{C}_{\mathsf{alg},k} \circ \mathscr{C}(m) \coloneqq (\mathcal{C}_{\mathsf{rand},i}(m'_1), \ldots, \mathcal{C}_{\mathsf{rand},t}(m'_t))$.

5 Zero Knowledge from Secure Computation

Definition 25 (Functionality f_L). *For a language $L \in \mathsf{NP}$ and its corresponding relation R_L, let f_L be the functionality for q players $P_1, \ldots, P_q$. Given a public input x and q shares of the witness $w_1, \ldots, w_q$ received from the Prover, the functionality delivers to all players 1 if $(x, w) \in R_L$ and 0 otherwise.*

Following [30], we slightly modify their zero knowledge protocol which makes "black box" use of an MPC protocol Π_{f_L}. This means that the zero knowledge protocol simply implements the next message function for each party without looking into the details of the circuits that describe these functions. The next

message function NEXT is used by the prover and verifier to interact. NEXT determines the next message to be sent based on the inputs and messages received so far. In particular, we commit to a single transcript of the entire protocol rather than committing to views of a party. We also note that Protocol 1 achieves only honest-verifier zero knowledge. Although, the scheme can be extended to obtain a standard zero knowledge proof, it leads to an increase in the number of rounds (cf. Theorem 4.4 in [30]). Hence, we stick to honest-verifier zero knowledge which suffices for the purpose of producing a NIZK argument.

Protocol 1 (Honest Verifier Zero Knowledge Interactive Protocol Π_{HVZK})

1. *Prover picks at random $w_1, \ldots, w_q$ whose exclusive-or equals the witness w. She simulates the execution of the MPC protocol Π_{f_L} on input $(x, w_1, \ldots, w_q)$. The prover then computes the transcript Ξ at the end and commits to each element of Ξ using a* statistically binding commitment scheme $\mathsf{Com}_{\mathsf{SB}}$. *Finally, she sends the commitments to the Verifier. Such a commitment scheme can be instantiated from Remark 1*
2. *Verifier sends to Prover a challenge set of indices $S_{\mathsf{Ch}} \triangleq \{i_1, \ldots, i_\beta\}$.*
3. *Prover opens all commitments to private inputs w_i, and all messages sent or received by players indexed by $i \in S_{\mathsf{Ch}}$ in Ξ.*
4. *Given the public values x, the Verifier accepts if and only if the Prover successfully opens all the requested commitments, all sent messages are consistent with the application of the next-message function* NEXT *on the appropriate set of received messages, and the output of all parties (computed deterministically by the received messages and their inputs) is* 1.

Fig. 1: HVZK Interactive Protocol using MPC.

Completeness and Honest Verifier Zero Knowledge. The correctness property follow directly from an identical argument to that in [30]. However, we present a sketch here for the sake of completeness. If $(x, w) \in R_L$ and the prover is honest and $w_1 \oplus \ldots \oplus w_q = w$, then the perfect correctness of Π_{f_L} implies that all the messages which were a part of the transcript Ξ will always be consistent with the application of the next-message function NEXT, and the outputs of each party must be 1. This implies correctness.

Let x belong to the language L, i.e., the functionality f_L outputs 1. For Honest Verifier Zero Knowledge, we construct a simulator M that simulates the view of an honest verifier as follows: M samples a challenge set of cardinality β of indices chosen uniformly at random among q parties. Let the set be $S'_{\mathsf{Ch}} \triangleq \{i_1, \ldots, i_\beta\}$. Sim simulates the MPC protocol Π_{f_L} in its head using the parties with indices in S'_{Ch}. Hence, M picks strings $w'_1, \ldots, w'_\beta$ uniformly at random and simulates an execution of Π_{f_L} on input $x, w'_1, \ldots, w'_\beta$ by invoking the MPC simulator Sim on input $(S'_{\mathsf{Ch}}, x, (w'_i)_{i \in S'_{\mathsf{Ch}}}, 1)$. Sim outputs a transcript Ξ'.

Recall that the transcript Ξ' consists of the public input, every party's randomness, every party's private input, and every message sent in each round. Along with a commitment to the public input, for all $i \in S'_{\mathsf{Ch}}$, M commits to the ith party's input, randomness, private input, and messages sent and received in Ξ'. Let $\mathsf{com}(S'_{\mathsf{Ch}})$ be defined to be the tuple of commitments listed in the previous sentence. For the remaining values in the transcript Ξ', M commits to 0. M sends all commitments, S'_{Ch}, and openings to all commitments in $\mathsf{com}(S'_{\mathsf{Ch}})$. The opened values of the transcript generated by Sim has an identical (statistically-close) distribution to the view of an *Honest*-Verifier due to the perfect (statistical) t-privacy of Π_{f_L}. Moreover, the hiding property of the commitment scheme implies that the Verifier cannot distinguish between the unopened commitments of 0 from commitments to values in transcript Ξ'.

Lemma 2 (Statistical Soundness). *Let $L \in \mathsf{NP}$ be a language. Let $\mathsf{Com}_{\mathsf{SB}}$ be a statistically-binding commitment scheme. Suppose that protocol Π_{f_L} realizes the q-party functionality f_L with perfect β-robustness (in the malicious model), and perfect, statistical or computational β-privacy (in the honest-but-curious model) for $\beta < \lceil q/2 - 1 \rceil$, then the soundness error in ZK protocol Π_{HVZK} is given by $\mathsf{negl}(q)$.*

Proof. Suppose $x \notin L$ so that there does not exist w such that $(x, w) \in R_L$ for relation R_L on NP language L.

If the Prover commits to inputs, randomness, and messages from an honest execution of Π_{f_L}, all parties output 0 and the Verifier will reject for any choice of S_{Ch}.

Otherwise, there exists a message $m_i^{(j)}$ in Ξ that is not consistent with the previous received messages and the next-message function NEXT. For any party P_i who sends an inconsistent message, we say that P_i is a "corrupted" party. There are two cases to consider: If malicious prover P^* corrupts at most β parties and if P^* corrupts strictly more than β parties. For a fixed execution of Π_{f_L} and its corresponding commitments made by malicious Prover P^*, we let B be the set of the indices of all corrupted parties.

In the first case, the β-perfect robustness property guarantees that for all indices $i \notin B$, the output of P_i is 0. If the Verifier chooses any index $i \notin B$, then the Verifier will observe the output of P_i is 0 and the Verifier will catch the Prover cheating. Therefore, with probability at least $1 - 1/\binom{q}{\beta}$, the Verifier will choose a set of indices of size β that is not contained in set B (if $|B| < \beta$ then the probability that Verifier catches the prover is 1).

In the second case, the Prover has chosen strictly more than β parties to corrupt. Here, we argue that the Verifier will ask for the commitment openings to a corrupted party with overwhelming probability. Suppose the Prover has chosen as little as $\beta+1$ many corrupted parties. The probability that the Verifier chooses a subset of size β that does not contain any of these corrupted parties

is given by

$$\begin{aligned}\frac{\binom{q-\beta-1}{\beta}}{\binom{q}{\beta}} &= \prod_{i=0}^{\beta} \frac{q-\beta-i}{q-i} \\ &= \prod_{i=0}^{\beta} \left(1 - \frac{\beta}{q-i}\right) \\ &\leq \prod_{i=0}^{\beta} e^{-\beta/(q-i)} \\ &\leq \left(e^{-\beta/(q-\beta)}\right)^{\beta+1}\end{aligned}$$

where we apply the inequality $1 - x \leq e^{-x}$ for all real x. Then observe that by our assumption $\beta = \alpha q$ for some constant $\alpha < 1$, so

$$\left(e^{-\beta/(q-\beta)}\right)^{\beta+1} \leq e^{-c^2 q - c}.$$

Observe this probability forms an upper bound for the probability the Verifier is fooled for when the Prover chooses *at least* $\beta + 1$ many corrupted parties. Formally, for all $i \geq 1$,

$$\binom{q-\beta-i}{\beta} \leq \binom{q-\beta-1}{\beta}.$$

Therefore the probability that the Verifier fails to catch the Prover in this setting is negligible in q and therefore negligible in security parameter λ.

Finally, by a union bound the soundness error is then $e^{-c^2 q - c} + 1/\binom{q}{\beta} = \mathsf{negl}(q)$.

6 Instantiating Fiat-Shamir via Correlation Intractable Hash Functions.

We first reintroduce the notions of Efficient Product Verifiability and Product Sparsity from [29].

Definition 26 (Product Relation). *A relation $R \subset \mathcal{X} \times \mathcal{Y}^t$ is a product relation, if for any x, the set $R_x = \{y \mid (x, y) \in R\}$ is the Cartesian product of several sets $S_{1,x}, S_{2,x}, \ldots, S_{t,x}$,*

$$R_x = S_{1,x} \times S_{2,x} \times \ldots \times S_{t,x}.$$

Definition 27 (Efficient Product Verifiability, Definition 3.3). *A relation R is efficiently product verifiable, if there exists a polynomial-sized circuit C such that, for any x, the sets $S_{1,x}, S_{2,x}, \ldots, S_{t,x}$ (in Definition 26) satisfy for any $i, y_i \in S_{i,x}$ if and only if $C(x, y_i, i) = 1$.*

Definition 28 (Product Sparsity, Definition 3.4). *A relation $R \subseteq \mathcal{X} \times \mathcal{Y}^t$ has sparsity ρ, if for any x, the sets $S_{1,x}, S_{2,x}, \ldots, S_{t,x}$ (in Definition 26) satisfies $|S_{i,x}| \leq \rho|\mathcal{Y}|$.*

Definition 29 (Bad Challenge Set). *For Protocol 1, let* com *be a string containing all commitments the prover sends to the verifier and let V_i denote the view of P_i formed by taking the appropriate subset of decommitments to* com*. We say that V_i is consistent if there exists an honest execution of the the q-party Protocol Π_f with P_i's inputs, randomness, and messages sent and received. Then we have the following set of bad challenges*

$$\mathcal{B} = S_{\mathsf{com}}^{|I|} = \underbrace{S_{\mathsf{com}} \times S_{\mathsf{com}} \times \cdots \times S_{\mathsf{com}}}_{|I| \ times}$$

where $S_{\mathsf{com}} = \{i \mid V_i \text{ is consistent}\}$.

Remark 5. The set S_{com} is efficiently verifiable by the MPC next message function. Also, $|S_{\mathsf{com}}| \leq \alpha q$, for some tiny constant $\alpha \in \{0, 1\}$. Here q is the number of parties involved in the MPC-in-the-Head protocol so the size of the *Bad Challenge Set* is the maximum number of parties in the MPC protocol that can be corrupted.

6.1 Construction of CIH family

Lemma 3 (CIH for Efficient Enumerable Relations [34,7]). *Assuming that $\mathsf{LWE}_{\frac{m}{2\log q}, m, q, \chi}$ holds for the particular parameter settings where χ is a B-bounded distribution for $B = q^{\Omega(1)}$, $q = \mathsf{poly}(m)$. Then, for every triplet 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 relation that is enumerable in time T.*

Lemma 4 ([29]). *Let $R \subseteq \times\mathcal{X} \times \mathbb{Z}_q^n$ be an efficiently verifiable product relation with sparsity α. Moreover, let $\mathcal{C} : \mathcal{M} \to \mathbb{Z}_q^n$ be a code that is $(\alpha q, L)$ list recoverable and $\mathcal{H}$ be a hash function family that is correlation intractable for all efficiently enumerable relations $R' \subseteq \mathcal{X} \times \mathcal{M}$, then $\mathcal{C} \circ \mathcal{H}$ is correlation intractable for R.*

Theorem 7. *Let $\mathcal{C}_{\mathsf{concat}} = \mathcal{C}_{\mathsf{alg}} \circ \mathscr{C} : \mathbb{Z}_Q^{k+1} \to \mathbb{Z}_q^{\eta q}$, $\eta < 1$ be the* Recurrent List Recoverable Code *with parameters as in Theorem 6. Let $\mathcal{H}$ be a Correlation Intractable Hash Function Family for an efficiently enumerable relation as in Lemma 3. Then the hash function family $\mathscr{C}_{\mathsf{concat}} \circ \mathcal{H}$ is a correlation intractable hash function family for the efficiently verifiable relation $\mathcal{B}$.*

Proof. From Theorem 6, the recurrent list recovery of $\mathcal{C}_{\mathsf{concat}}$ tells us that a list of size $O(k^2 \log^{4\varepsilon - \delta} k)$, for arbitrary constants $0 < \delta < \varepsilon < 1$ can be efficiently recovered. This is indeed bound by a polynomial, hence is certainly efficiently enumerable. Thus, from Lemma 3 and Lemma 4, we conclude that $\mathscr{C} \circ \mathcal{H}$ is indeed Correlation Intractable for the relation $\mathcal{B}$.

This leads to our final theorem.

Theorem 8. *Assuming that* $\mathsf{LWE}_{\frac{m}{2\log q},m,q,\chi}$ *holds for the particular parameter settings where* χ *is a* B*-bounded distribution for* $B = q^{\Omega(1)}$*,* $q = \mathsf{poly}(k)$*,* k *is the security parameter, and a MPC protocol with perfect* αn*-robustness and perfect, statistical, or computational security, where* $\alpha \in (0, 1/2)$ *is a constant and* n *is the size of the challenge set in the interactive protocol, there exists NIZKs with computational soundness for all of* NP *whose proof size is*

$$\mathcal{O}(|C| + q \cdot \mathrm{depth}(C)) + \mathsf{poly}(\lambda)$$

where C *is an arithmetic circuit for the* NP *verification function at* $q = k \log^{1+\epsilon} k$ *for any* $\epsilon > 0$*.*

This theorem is a direct consequence of the following results:

- Theorems 3 and 7 combine to provide a hash function family which is Fiat-Shamir compatible with parameters aligning with the "MPC-in-the-Head" paradigm.
- Applying the Fiat-Shamir compatible hash to Protocol 1 gives us a computational sound NIZK from the MPC-in-the-Head model without parallel repetition.
- There exists perfect αn-robust MPC protocols with the aforementioned communication complexity for $\alpha < 0.5$ (Theorem 4).

7 Acknowledgements

This research was supported in part from a Simons Investigator Award, DARPA SIEVE award, NTT Research, NSF Frontier Award 1413955, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024. This work was done [in part] while PL was visiting the Simons Institute for the Theory of Computing. The authors would like to thank Alexis Korb for useful discussions and help with proof reading.

References

1. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 2087–2104. ACM Press (Oct / Nov 2017)
2. Asharov, G., Ephraim, N., Komargodski, I., Pass, R.: On perfect correctness without derandomization. Cryptology ePrint Archive, Report 2019/1025 (2019), `https://eprint.iacr.org/2019/1025`

3. Beck, G., Goel, A., Jain, A., Kaptchuk, G.: Order-C secure multiparty computation for highly repetitive circuits. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part II. LNCS, vol. 12697, pp. 663–693. Springer, Heidelberg (Oct 2021)
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: 20th ACM STOC. pp. 1–10. ACM Press (May 1988)
5. Brakerski, Z., Koppula, V., Mour, T.: NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 738–767. Springer, Heidelberg (Aug 2020)
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (2000), `https://eprint.iacr.org/2000/067`
7. Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D., Wichs, D.: Fiat-Shamir: from practice to theory. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC. pp. 1082–1090. ACM Press (Jun 2019)
8. Canetti, R., Chen, Y., Reyzin, L.: On the correlation intractability of obfuscated pseudorandom functions. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part I. LNCS, vol. 9562, pp. 389–415. Springer, Heidelberg (Jan 2016)
9. Canetti, R., Chen, Y., Reyzin, L., Rothblum, R.D.: Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 91–122. Springer, Heidelberg (Apr / May 2018)
10. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: 30th ACM STOC. pp. 209–218. ACM Press (May 1998)
11. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. Cryptology ePrint Archive, Report 2017/279 (2017), `https://eprint.iacr.org/2017/279`
12. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: 20th ACM STOC. pp. 11–19. ACM Press (May 1988)
13. Damgård, I., Ishai, Y.: Scalable secure multiparty computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 501–520. Springer, Heidelberg (Aug 2006)
14. Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly secure multiparty computation and the computational overhead of cryptography. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 445–465. Springer, Heidelberg (May / Jun 2010)
15. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (Aug 1987)
16. Forney Jr, G.D.: Concatenated codes. research monograph no. 37 (1966)
17. Gentry, C., Groth, J., Ishai, Y., Peikert, C., Sahai, A., Smith, A.: Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. Journal of Cryptology 28(4), 820–843 (2015)
18. Giacomelli, I., Madsen, J., Orlandi, C.: ZKBoo: Faster zero-knowledge for Boolean circuits. In: Holz, T., Savage, S. (eds.) USENIX Security 2016. pp. 1069–1083. USENIX Association (Aug 2016)
19. Goldreich, O.: Foundations of Cryptography: Basic Tools, vol. 1. Cambridge University Press, Cambridge, UK (2001)

20. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press (May 1987)
21. Goldreich, O., Micali, S., Wigderson, A.: How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 171–185. Springer, Heidelberg (Aug 1987)
22. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th ACM STOC. pp. 291–304. ACM Press (May 1985)
23. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on computing 18(1), 186–208 (1989)
24. Goyal, V., Polychroniadou, A., Song, Y.: Unconditional communication-efficient MPC via hall's marriage theorem. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 275–304. Springer, Heidelberg, Virtual Event (Aug 2021)
25. Guruswami, V.: Algorithmic results in list decoding. Now Publishers Inc (2007)
26. Guruswami, V., Indyk, P.: Expander-based constructions of efficiently decodable codes. In: FOCS. pp. 658–667. IEEE Computer Society (2001)
27. Guruswami, V., Sudan, M.: Improved decoding of reed-solomon and algebraic-geometric codes. In: FOCS. pp. 28–39. IEEE Computer Society (1998)
28. Holmgren, J., Lombardi, A.: Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In: Thorup, M. (ed.) 59th FOCS. pp. 850–858. IEEE Computer Society Press (Oct 2018)
29. Holmgren, J., Lombardi, A., Rothblum, R.D.: Fiat-shamir via list-recoverable codes (or: Parallel repetition of gmw is not zero-knowledge). STOC (2021), `https://eprint.iacr.org/2021/286`
30. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) 39th ACM STOC. pp. 21–30. ACM Press (Jun 2007)
31. Kalai, Y.T., Rothblum, G.N., Rothblum, R.D.: From obfuscation to the security of Fiat-Shamir for proofs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 224–251. Springer, Heidelberg (Aug 2017)
32. Lombardi, A., Schaeffer, L.: A note on key agreement and non-interactive commitments. Cryptology ePrint Archive, Report 2019/279 (2019), `https://eprint.iacr.org/2019/279`
33. Parvaresh, F., Vardy, A.: Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In: 46th FOCS. pp. 285–294. IEEE Computer Society Press (Oct 2005)
34. Peikert, C., Shiehian, S.: Noninteractive zero knowledge for NP from (plain) learning with errors. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 89–114. Springer, Heidelberg (Aug 2019)
35. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. Journal of the society for industrial and applied mathematics 8(2), 300–304 (1960)
36. Regev, O.: New lattice based cryptographic constructions. In: 35th ACM STOC. pp. 407–416. ACM Press (Jun 2003)
37. Sudan, M.: Decoding of reed solomon codes beyond the error-correction bound. J. Complex. 13(1), 180–193 (1997)
38. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986)

A Supplementary Materials

A.1 Comparison with Prior Work

We explicitly present the calculations depicting the parameters which are achievable by [29] in our setting.

Let us briefly introduce some definitions needed for their construction:

Definition 30 (Parvaresh-Vardy Code [33].). *There is an explicit code $C : \{0,1\}^k \to [q'^s]^{q'}$, parameterized by integers $s, k, q' \in \mathbb{Z}^+$ (with q' a power of two) such that the code is (efficiently) (ℓ', L)-list recoverable in time $\mathsf{poly}((2s)^s, q', \ell')$ as long as*

$$\ell' \leq \left(\frac{1}{s+1}\right)^{s+1} \left(\frac{q'}{k}\right)^s$$

and

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

for some constant c.

Definition 31 (Random Codes). *There exists a constant $c \geq 0$ such that for any q, q', a, ℓ, ℓ' (all of which are functions of n), a random function $f : [q'^s] \to [q]^n$ is combinatorially (ℓ, ℓ')-list recoverable with probability $1 - 2^{-\Omega(\ell')}$, as long as,*

$$\ell' \geq c \cdot (q'^s \rho + \ell n \log(q/\ell)),$$

where the parameter $\rho = \left(\frac{\ell}{q}\right)^n$. This list recovery can be done (by brute force) in time $O(q'^s \cdot n \cdot \ell \cdot \log q)$. Evaluation of f can be done in time $O(q'^s \cdot n \cdot \log q)$.

As was done throughout the paper, assume that k is the security parameter. We ask whether it is possible to achieve a block size $q' = \tilde{O}(k)$ for the Paravresh-Vardy code. Minimizing this block size results in a smaller number of challenges, and therefore a smaller proof.

Necessary parameter relations for MPC-in-the-head Recall that in the MPC-in-the-head setting that q is the number of parties and ℓ is the maximum size of the bad challenge set (also the maximum number of parties that can be corrupted) so $\ell = \alpha q$ for some constant $0 < \alpha < 1$. Moreover, $q' \cdot n$ is the number of verifier challenges produced by the CI hash function obtained by composing the error-correcting code with the CI hash function from [34] (the output of this composed function is in $\mathbb{Z}_q^{q' \cdot n}$ so it must be that $q' \cdot n < q$) and this directly determines the number of $\mathbb{Z}_q$ field elements necessary in the NIZK proof.

Necessary parameter relations for code concatenation It is a fact that the concatenated code $C \circ f$ is (ℓ, L) list recoverable if C is (ℓ', L) list recoverable and f is (ℓ, ℓ') list recoverable. From the two definitions above, the following relation must be true for list recoverability of the concatenation of the Paravesh-Vardy code with random codes:

$$\left(\frac{1}{s+1}\right)^{s+1}\left(\frac{q'}{k}\right)^{s} \geq \ell' \geq c \cdot (q'^{s}\rho + \ell n \log(q/\ell)).$$

The LHS above is $O\left(\left(\frac{q'}{k}\right)^{s}\right)$. The RHS is equivalently (by substitution) given by $q'^{s} \cdot \alpha^{n} + \alpha q n \log(1/\alpha)$ for constant $0 < \alpha < 1$. Then observe that if the RHS is dominated by $q'^{s}\alpha^{n}$, satisfying this inequality is doomed because the LHS is $O\left(\left(\frac{q'}{k}\right)^{s}\right)$. Therefore, assume that the RHS is dominated by the second term which is $O(qn)$. Now we directly compute to see if certain block sizes q' for the PV code can be used to instantiate this code concatenation.

1. **The case of** $q' = k^{1+\varepsilon}$: To use a PV code with block length $q' = k^{1+\varepsilon}$, we observe that setting $s \geq \frac{1+\varepsilon}{\varepsilon}$ gives a satisfying solution. However, we note that this setting comes at a cost. PV codes group outputs into blocks of constant size s, which it then treats as an element of the extension field $\mathbb{F}_{q^s}$. The headline result from HLR which achieves block length growing with k^2 works by setting $s = 2\log_k(\ell)$. If we were to try to carry out the same with block length $O(k^{1+\varepsilon})$ (as stated in a remark on page 29 after the proof of Proposition 5.2 of their paper), this would require, roughly, setting $s > (1+\epsilon)/\epsilon$, which would yield finite fields that are enormously large. Furthermore, one would need to compensate for a loss of roughly $(1/(s+1))^{(s+1)}$ in the list recovery size, which although a constant, would also be very large – at least $(1/\epsilon)^{(1/\epsilon)}$ – and incur a massive degradation of parameters.
2. **The case of** $q' = \tilde{O}(k)$: Setting $q' = k\log^{1+\varepsilon} k$ has no solution as the LHS is only polylogarithmic in k whereas the RHS is at least $O(k \log k)$. Thus, [29] can not achieve quasi linear block size even with the optimizations mentioned in their paper.

We finally note that a major advantage of our work is that we get our results by using simple polynomial reconstruction instead of the complex extension-field based PV codes used by HLR.