Towards Permissionless Consensus in the Standard Model via Fine-Grained Complexity

Marshall Ball* Juan Garay† Peter Hall‡ Aggelos Kiayias§
Giorgos Panagiotakos¶

Abstract

We investigate the feasibility of *permissionless* consensus (aka Byzantine agreement) under standard assumptions. A number of protocols have been proposed to achieve permissionless consensus, most notably based on the Bitcoin protocol; however, to date no protocol is known that can be provably instantiated outside of the random oracle model.

In this work, we take the first steps towards achieving permissionless consensus in the standard model. In particular, we demonstrate that worst-case conjectures in fine-grained complexity, in particular the orthogonal vectors conjecture (implied by the Strong Exponential Time Hypothesis), imply permissionless consensus in the random beacon model—a setting where a fresh random value is delivered to all parties at regular intervals. This gives a remarkable win-win result: *either permissionless consensus exists relative to a random beacon, or there are non-trivial worst-case algorithmic speed-ups for a host of natural algorithmic problems* (including SAT).

Our protocol achieves resilience against adversaries that control an inverse-polynomial fraction of the honest computational power, i.e., adversarial power $A = T^{1-\epsilon}$ for some constant $\epsilon > 0$, where T denotes the honest computational power. This relatively low threshold is a byproduct of the slack in the fine-grained complexity conjectures.

One technical highlight is the construction of a *Seeded Proof of Work*: a Proof of Work where many (correlated) challenges can be derived from a single short *public* seed, and yet still no non-trivial amortization is possible.

*NYU. marshall.ball@cs.nyu.edu.
†Texas A&M University. garay@cse.tamu.edu.
‡NYU. pf2184@nyu.edu.
§University of Edinburgh and IOHK. aggelos.kiayias@ed.ac.uk.
¶IOHK. pagio91i@gmail.com.

Contents

1 Introduction

A consensus (aka Byzantine agreement) [50, 44] protocol enables the participants to agree on an input value, even in the presence of a malicious adversary who has the ability to corrupt a number of the parties. In a *permissioned* consensus protocol, as all classical protocols are, it is assumed that the identities of the parties participating in the protocol are known in advance, and, moreover, that participants have reliable, authenticated[1] channels to one another. In the *permissionless* setting, on the other hand, participants only know an upper bound on the number of parties — none of the other participants' identities are known. Furthermore, the adversary is capable of injecting and "spoofing" messages in the network, making the exact number of actual participants hard to determine.

In the most basic formulation of the permissionless consensus problem, three properties are sought: (i) Agreement — all honest parties agree on the same output value, (ii) Validity — if all honest parties start with the same input value, the output is determined from this input, and (iii) Termination — all honest parties produce an output in a finite number of steps. As we will see (cf. Section 1.2), the level of adversity present in the permissionless setting makes it infeasible to solve the Byzantine agreement problem in the permissionless setting even assuming (1) a synchronous model of communication where the protocol proceeds in well defined message passing rounds, (2) a public setup or a beacon that regularly emits random values made available to participants at each round, (3) that the adversary does not control any of the protocol participants (i.e., no party corruptions are allowed), and (4) an upper bound to the number of participants is known to all parties.

Given the extent of this impossibility, it is worth asking whether the permissionless setting is a feasible domain for Byzantine agreement. This question was answered in the affirmative by Nakamoto with his Bitcoin blockchain proposal [48]. Indeed, the premise of Nakamoto's workaround is that it would be possible to reach agreement in the permissionless setting by utilizing "proofs of work" (PoWs), i.e., requiring from the participants to solve a moderately hard problem in order to transmit valid messages to each other. The potential of this approach can be seen immediately in the context of the impossibility outlined below (Section 1.2), since the main argument relies on the adversary's ability to simulate parties "in his head" and a PoW would impose a computational burden to do so. It thus follows that there is a potential for a Byzantine agreement protocol to work in the permissionless setting by imposing a *computational assumption* on the adversary.

Discovering such an assumption has proven to be a remarkably elusive proposition. Based on the beautiful observation suggested by Nakamoto's work, a number of works, starting with [31], have built on it and presented permissionless Byzantine agreement protocols. Despite the progress, we still do not know how to achieve agreement outside the Random Oracle (RO) model [6]. Indeed, in most cases, the protocols are directly analyzed in the RO model [31], or they are based on cryptographic primitives that they themselves have no known instantiation outside the RO model [33, 34]. A main obstacle is the "moderate hardness" of the PoW concept that is much easier to argue in an idealized model such as the RO's, but much more difficult to capture in a standard model of computation.

Motivated by this, we make a decisive step towards realizing permissionless Byzantine agreement in the standard model. Our primary contributions are threefold:

1. A simple notion of a PoW scheme that suffices to construct permissionless consensus in the random beacon model. This, for the first time, shows what flavor of PoWs suffices in the standard model of computation to achieve permissionless agreement assuming public random coins

[1] And in some cases private, such as randomized protocols in the information-theoretic setting.

are available to all participants and a suitable computational assumption. Our methodology first uses our PoW notion to yield a weak agreement variant where agreement holds unconditionally, while validity is allowed to sometimes fail. We then use this weak variant as a building block to obtain *graded consensus* by a suitable amplification technique, which easily yields full consensus in the random beacon model.[2]

2. (Worst-case) Conjectures in fine-grained complexity, such as SETH [40, 11] or the Orthogonal Vectors conjecture [30, 3], suffice to realize this notion of PoW, thus yielding permissionless consensus in the random beacon model from simple worst-case conjectures in complexity.

3. The consensus protocol mentioned above requires beacon outputs of length proportional to the number of parties. We show that if, in addition to the fine-grained conjectures, one assumes the Decisional Diffie Hellman assumption (DDH) is sub-exponentially secure (cf. [41]), there is a permissionless consensus protocol that uses beacon outputs of length *independent* of the number of parties. Critical to achieving this are (1) a novel notion of a *Seeded PoW*, where a short public seed can be expanded into many PoW challenges (this transformation does not require DDH), and (2) a fine-grained approach to zero-knowledge (building on an approach by Ball *et al.* [4]). Both of these contributions may be of independent interest. A byproduct of this transformation are the first PoWs with arbitrary polynomial gap that do not require a random oracle (assuming the k-Orthogonal Vectors Conjecture and sub-exponential DDH).

PoWs inherently require a source of *fresh* randomness to be used safely and avoid precomputation attacks. Famously, Bitcoin relies explicitly on a heuristic source of fresh randomness obtained from the headline of *The Times* newspaper on a date near the deployment of the protocol [8], and implicitly on the ability of the underlying hash function to form chains that are unpredictable: indeed, guessing the hash value of any future chain allows a precomputation attack. Previous works (e.g., [31, 2]) have non-trivially relied on the RO model to argue these properties. Our result drops the RO modeling requirement for permissionless BA showing that, under suitable computational assumptions, a randomness beacon is sufficient. Note that a randomness beacon is no stronger assumption than the RO, as it is possible to realize the former from the latter (cf. [2]). Furthermore, and perhaps more interestingly, it is also a setup assumption that, contrary to the RO, can be realized directly on its own right[3]: for example, via the invocation of a suitable *sunspot* setup [17] that allows for deterministic extraction.

1.1 Our Results

Model and Assumptions. Before summarizing our results, we begin by describing the model in which we construct permissionless consensus and the assumptions from fine-grained complexity we utilize.

The permissionless setting. While message passing is reliable in the permissionless setting, the sources of messages and the number of parties participating in the protocol critically is not. The adversary can inject messages to be delivered in any order, and the actual number of parties is chosen adversarially only subject to a known upper bound on the number of parties. Time is divided into rounds, with each active honest participant allowed a fixed number of computational steps per

[2]The reader may wonder why the PoW scheme presented in [4] doesn't directly give a permissionless consensus protocol. The reason is that in Nakamoto's protocol the PoW proving time must follow a geometric distribution, and it is not clear how to get this property from the scheme in [4] without an RO while retaining hardness. On the other hand, our (classical) approach to consensus directly benefits from deterministic-time PoW provers, as all parties are expected to produce a PoW by the end of the round deadline.

[3]In contrast to a RO, a beacon has a feasibly short description.

round. For convenience, we will use rounds of different length as suited for the underlying protocol and its computational requirements. To ensure that the adversary cannot just flood the honest parties' incoming communication tape, an upper bound on the number of messages the adversary may inject is also imposed. Furthermore, in each round the adversary is allowed to determine the order with which messages are delivered to each honest party.

Random beacon model. All parties have access to a random value that is sampled from a uniform distribution in each round. The adversary is allowed to access it first at the onset of the round — and even influence the way messages are scheduled to be received by each honest party after seeing it.

Conjectures in fine-grained complexity. The Orthogonal Vectors Problem, or OV, is the following simple combinatorial problem:

Problem Orthogonal Vectors (OV)

Input: n vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \{0,1\}^d$

Decide: Does there exist i, j such that $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = 0$?

One can equivalently think of this problem as the following: Given n subsets from a universe of size d, determine if there exists a pair of disjoint sets.

For $c \geq 1$, the fastest existing algorithms for OV with dimension $d = c \log n$ run in time $n^{2-1/O(\log c)}$ [1, 18]. In particular, as d grows beyond $\log n$, the time complexity approaches that of the trivial $O(n^2 d)$-time algorithm. The failure to effectively bypass this n^2 barrier has led to the conjecture that OV requires $n^{2-o(1)}$ time for $d = \omega(\log n)$:

Conjecture 1 (Orthogonal Vectors Conjecture (OVC) [54, 56])**.** For all $\epsilon > 0$, there exists $c \geq 1$ such that OV requires $n^{2-\epsilon}$ time on instances with $d = c \log n$.

We note this is a *worst-case* conjecture. In this work, we rely on slightly stronger variants of this conjecture, that OV does not admit any *non-uniform* (and alternatively, probabilistic two-sided constant-error) algorithm that runs in time $n^{2-\epsilon}$, for $\epsilon > 0$. Violating this conjecture would immediately yield a host of algorithmic improvements to many important problems, perhaps most notably Satisfiability. In particular, the orthogonal vectors conjecture is in fact implied by the Strong Exponential Time Hypothesis or SETH (similarly, its strengthenings are implied by corresponding strengthenings of SETH [40, 11, 54]). The converse is not known to be true.

Conjecture 2 (Strong Exponential Time Hypothesis (SETH) [40, 11, 12])**.** For all $\epsilon > 0$, there exists $k \geq 1$ such that k-SAT requires $2^{(1-\epsilon)n}$ time.

The Orthogonal Vectors problem can in fact be generalized to the *k-Orthogonal Vectors Problem*, which effectively asks: Given n subsets from a universe of size d, is there a way to choose k sets such that their intersection is empty?

Problem k-Orthogonal Vectors (kOV)

Input: n vectors $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^n \in \{0,1\}^d$

Decide: Does there exist $i_1, \ldots, i_k$ such that $\sum_{j=1}^{d} \boldsymbol{v}_j^{i_1} \cdots \boldsymbol{v}_j^{i_k} = 0$?

Note that 2OV = OV. This problem is conjectured to require $n^{k-o(1)}$ for $d = \omega(\log n)$:

Conjecture 3 (k-Orthogonal Vectors Conjecture ($k\mathsf{OVC}$) [54, 56])**.** For all $\epsilon > 0$, there exists $c \geq 1$ such that $k\mathsf{OV}$ requires $n^{k-\epsilon}$ time.

This conjecture is also implied by the Strong Exponential Time Hypothesis (similarly, $k\mathsf{OVC}$'s variants are implied by SETH's variants).

Proofs of Work from Fine-Grained Complexity. A *Proof of Work* (PoW), introduced by Dwork and Naor [25], enables a Prover party to convince, relative to a random challenge, a Verifier party that a certain amount of computation was performed. In particular, no adversary that uses significantly fewer steps than those used by the "honest" Prover algorithm can convince the Verifier to accept. In addition, the Verifier must be much more efficient than the Prover.

It is also useful for PoWs to not be amortizable — that is, given many proofs, a malicious Prover should not be able to convince the Verifier in time less than what it takes to separately prove each instance.

Unlike many other cryptographic primitives, PoW schemes necessitate very fine-grained assumptions because the honest Prover running time should be as close to the bound on adversarial running time as possible.

Protocol-friendly PoWs. Ball *et al.* [3, 4] showed how to construct PoW schemes, satisfying Dwork and Naor's definition, from worst-case assumptions about natural problems in fine-grained complexity (such as OVC mentioned above), yielding an exciting win-win.[4] These schemes had the following non-amortization guarantee: Given m challenges that require t time individually, it is impossible to produce proofs for *all* m challenges in time significantly less than $m \cdot t$. Or, in other words, the naïve strategy of solving each instance one-by-one is effectively optimal.

While this result is exciting, it unfortunately falls short of what one would ultimately desire from a PoW scheme (and what we require in the present work to achieve consensus). Loosely speaking, one should require non-amortization to hold even over partial solutions where the adversary can decide what subset of the challenges to focus on. In particular, we introduce a new "protocol-friendly" PoW promise: Given m challenges that require t time individually, producing $\ell < m$ accepting proofs requires time $\ell \cdot t$, for any (large enough) ℓ. In other words, the naïve strategy of honestly producing proofs for any ℓ instances one-by-one is effectively optimal.

In this work we give a novel analysis to show that the PoW schemes of Ball *et al.* [3, 4], in fact already achieve this notion.

(Protocol-Friendly) PoW Theorem (Informal). *Assuming* OVC *(Conjecture 1 above), there exists a Proof of Work scheme where the honest prover runs in time* $O(n^2)$*, the verifier runs in time* $\tilde{O}(n)$*, and for any* $m = \mathsf{poly}(n)$ *and* $\ell = m^{1-o(1)}$ *and any* $\epsilon > 0$*, no adversary running in time* $\ell \cdot n^{2-\epsilon}$ *that is given* m *random challenges can produce* ℓ *accepting proofs with probability* $1/n^{o(1)}$*.*

Assuming Conjecture 3, this can be extended to yield *interactive* proofs with provers running in time $O(n^k)$ and verifiers running in time $\tilde{O}(n)$, and guarantees against adversaries running in time $\ell \cdot n^{k-\epsilon}$. Additionally, assuming an efficient Correlation-Intractable Hash [15, 13, 16] for Round-by-Round Soundness [14] protocols yields a non-interactive PoW with the same characteristics as the Fiat-Shamir transform.

Ball *et al.* constructed their schemes by (a) providing a worst-case to average-case reduction from $k\mathsf{OV}$ to the problem of evaluating a particular family of polynomials: $\mathcal{F}\mathsf{OV}^k$.[5] They then demonstrated that (b) $\mathcal{F}\mathsf{OV}^k$ admits a *direct sum theorem* (which says solving m instances requires

[4] Dwork and Naor themselves gave some candidates, for example assuming that existing attacks on the Ong-Schnorr-Shamir signature scheme could not be improved.

[5] They additionally observed that this technique applies to a variety of problems in fine-grained complexity. Later, others [36, 9, 35] extended this method to show that the problem of counting k-cliques is *itself* hard on average.

m times as much computation as a single instance), and (c) the problem admits a doubly efficient proof system. Then, to produce a proof of work, the Prover simply evaluates $\mathcal{F}\mathsf{OV}^k$ on a random challenge, and uses in proof system to convince the verifier that the evaluation is correct.

To strengthen their result, it suffices to simply improve upon (b): prove what we term a *robust direct sum theorem* for the problem of evaluating $\mathcal{F}\mathsf{OV}^k$: correctly evaluating any ℓ out of m random instances requires ℓ times as much work as evaluating a single instance.

Robust Direct Sum Theorem (Informal). *Assuming the $k\mathsf{OV}$ conjecture (Conjecture 3 above), for any $m = \mathsf{poly}(n)$ and $\ell = m^{1-o(1)}$ and any $\epsilon > 0$, no algorithm running in time $\ell \cdot n^{2-\epsilon}$ that is given m random independent inputs to $\mathcal{F}\mathsf{OV}^k$ can correctly evaluate ℓ of them with probability $1/n^{o(1)}$.*

We remark that this theorem holds not simply with respect to m random, independent instances, but additionally with respect to a specific joint *pseudorandom* distribution that only has $\tilde{O}(m^{1/k}n)$ bits of entropy. (This "derandomized" robust direct sum theorem is the technical core of our next result.)

Seeded proofs of work. It is often desirable to be able to generate a large number of PoW challenges with very little communication. For example, in the context of this paper, we will use a random beacon to generate many PoW challenges in each epoch. Intuitively, in a model without ROs, instantiating such a beacon with long output is likely to incur significant efficiency overhead.

With this in mind, we introduce the notion of a *Seeded PoW* scheme, where a short public seed can be expanded to any of m PoW challenges with the same (strengthened) promise as above, even when the adversary is holding the seed. It is not clear how to generically compile a PoW scheme into a Seeded PoW scheme, even using cryptographic assumptions, because of the public nature of the seed (the distribution over challenges it specifies *cannot* be pseudorandom in a traditional cryptographic sense).

Nonetheless, we show that Ball *et al.*'s PoW scheme can be adapted into a Seeded PoW scheme, by demonstrating that the *robust direct sum theorem* holds relative to a particular "pseudorandom" joint distribution over batches of m instances (as opposed to simply i.i.d. uniform batches) that admits an invertible sampler.

(Protocol-Friendly) Seeded PoW Theorem (Informal). *Assuming the $k\mathsf{OV}$ conjecture (Conjecture 3 above), there exists a Seeded PoW scheme where the honest prover runs in time $O(n^2)$, the verifier runs in time $\tilde{O}(n)$, and for any $m = \mathsf{poly}(n)$ a seed of length $\tilde{O}(m^{1/k}n)$ can be efficiently expanded to m instances of size $\tilde{O}(n)$ such that for any $\ell = m^{1-o(1)}$ and any $\epsilon > 0$, no adversary running in time $\ell \cdot n^{2-\epsilon}$ that is given the seed can produce ℓ accepting proofs with probability better than $1/n^{o(1)}$.*

Zero-knowledge PoWs. A *zero-knowledge PoW* (zkPoW) [4] satisfies the properties of a PoW as well as a zero-knowledge property. Classical zero-knowledge is trivial in this setting because Proofs of Work can be generated in polynomial time. Instead, we use a more precise notion of zero-knowledge: the simulator should run with time complexity proportional to that of the verifier — in our case $\tilde{O}(n^{1+\delta})$, for arbitrarily small constant $\delta > 0$.

This is an important property in the context of protocol-friendly PoWs as it allows simulation of PoWs in the context of a reduction to an underlying hard computational problem provided the complexity of the simulator is low. More concretely, this gives a generic way to get security for our Seeded PoW scheme against an adversary who not only can try to amortize work across ℓ out of m challenges, but additionally sees a number of (correlated) honest proofs as well. (In the case

of vanilla protocol-friendly PoWs where challenges are independent, this stronger security follows simply from encoding the honest proofs as non-uniform advice.)

Assuming subexponential DDH, we present and realize a concretely efficient zkPoW construction based on our seeded PoW primitive in a way that also satisfies a *Proof of Knowledge* property, i.e., it is possible to extract a witness from any convincing prover. The core of this protocol is a simple Schnorr-based Fiat-Shamir-friendly compiler for making sumcheck-based proofs zero-knowledge. A by-product of this theorem is the first *non-interactive* PoW with a superquadratic prover/verifier gap without a random oracle, from natural assumptions.

(Protocol-Friendly) Seeded ZK-PoW Theorem (Informal). *Assuming the $k\mathsf{OV}$ conjecture (Conjecture 3) and subexponentially-secure DDH, for any $\delta > 0$ there exists an (extractable) Zero-Knowledge Seeded PoW scheme (in the public uniformly random string [URS] model) where the honest prover runs in time $O(n^{k+\delta})$, the verifier runs in time $O(n^{1+\delta})$, and for any $m = \mathsf{poly}(n)$, a seed of length $\tilde{O}(m^{1/k}n)$ can be efficiently expanded to m instances of size $\tilde{O}(n)$ such that for any $\ell = m^{1-o(1)}$ and any $\epsilon > 0$, no adversary running in time $\ell \cdot n^{k-\epsilon}$ that is given the seed can produce ℓ accepting proofs with probability $1/n^{o(1)}$.*

Honest proofs can be simulated in time $O(n^{1+\delta})$, the knowledge extractor runs in time $O(n^{1+\delta})$, and the length of the URS is $O(n^{1+\delta})$.

Permissionless Consensus. Next, armed with our protocol-friendly PoWs, we put them to use in the context of consensus. Our construction follows the versatile weak-to-graded-to-full consensus approach (cf. [43, 26]), and non-trivially adapts it to a setting where the adversary may temporarily control a majority of the messages sent in the protocol. As the name implies, we first design a protocol that achieves *weak consensus*, a Byzantine agreement variant where the Agreement property is relaxed to allow some of the honest parties to fail returning $\perp$.

Our first modification to avoid sybil attacks, is that every time a subprotocol is executed, participants parse the random beacon output as a sequence of PoW instances, associate each instance with an input message (multiple instances are associated with the same message), and then solve a random instance corresponding to the message of their choice and share the related witness with other participants to convey their support for this message.

A complication that arises at this stage is that the adversary can enforce a high level of disparity in the views of honest parties by revealing different PoW witnesses it produced to different parties. For this reason, an additional (short) round of communication is added to ensure that most of the PoW instances exchanged become common in the views of all honest parties. This approach nearly unifies the views, to a degree sufficient to enable (weak) agreement.

A second issue is that the security of the PoW schemes we construct fails with non-negligible probability, while we strive to achieve consensus with only negligible error. The problem is that when the PoW security fails, the adversary is overrepresented by computing more PoWs than expected, and may control the majority of the messages created, thus making it impossible to achieve validity. To circumvent this problem we (i) run the weak consensus subprotocol repeatedly to amplify security and ensure that in the majority of the runs PoW security holds with overwhelming probability, and then (ii) apply an adequate error-correcting technique to correct any errors introduced by the temporary loss of honest majority. As we prove, weak consensus can be indeed "error-corrected" and validity be ensured with overwhelming probability, an interesting result by itself; in the initial conception of weak consensus temporary loss of honest majority was not a concern!

Next, we use the above protocol as a subprotocol to achieve *graded consensus* [26], a variant where the output of each honest party is associated with a grade in $\{0, 1\}$ that signals to an honest party whether all honest parties have reached the same output or not. We show that weak consensus

implies graded consensus by a similar amplification and error-correcting technique. Finally, utilizing the random beacon, we then turn graded consensus into a full-fledged consensus protocol. This brings us to our main result stated informally as follows:

Permissionless Consensus Theorem (Informal). *Assuming the kOV conjecture for $k \geq 2$ (Conjecture 3 above), permissionless consensus with negligible error is feasible in the random beacon model provided that the ratio of the total number of computational steps performed by the honest parties (as a function of n) over that of the adversary is n^ϵ for some $\epsilon > 0$, where n is the security parameter.*

While it is necessary to assume that the total number of computational steps of the honest parties is greater than that of the adversary, as otherwise consensus is impossible, we note that under stronger assumptions, e.g., the RO model, permissionless consensus can be achieved by only assuming that the total honest computational power slightly exceeds that of the adversary. Interestingly, the same assumption suffices in our case if the adversarial gap ϵ in the worst-case conjecture we employ tends to zero.

Further, note that, as described, the consensus protocol requires random beacon outputs of length proportional to the number of honest parties (or their cumulative computational power), something that may be prohibitively long in practice. We overcome this problem by incorporating our Seeded NIZK-PoW: for the resulting consensus protocol, it suffices to have a $O(n^2)$-long reference string that will be interpreted as a ZK-PoW seed. The key difference in the reduction is that we have to use the ZK simulator to ensure its complexity remains within bounds.

1.2 Technical Overview

First, we outline the impossibility result for Byzantine agreement in the permissionless setting with a random beacon and also given an upper bound U on the number of parties, demonstrating that computational assumptions are necessary (even in the random beacon model). For the sake of contradiction, consider a protocol Π capable of solving agreement. Suppose there are two sets of parties S_0, S_1, numbering $n > 0$ participants each with $U \geq 2n$, and each holding initial input $0, 1$, respectively. Define first an execution E where the adversary is not active and the parties in $S_0 \cup S_1$ run protocol Π. Given that Π satisfies Agreement and Termination, all parties are capable of producing the same output $b \in \{0, 1\}$. Next, consider two other executions E_a, $a \in \{0, 1\}$, where S_a consists of honest parties and the adversary spoofs and injects all messages that would have been produced by the parties in S_{1-a} if they were actual participants in the protocol — the adversary is capable of simulating them "in his head" and communicating on their behalf since the network model allows him to inject messages and spoof their source. By Validity it must be the case that the honest parties in S_a in execution E_a terminate with output a, for both cases $a \in \{0, 1\}$. This results in a contradiction, since all three executions we defined, E, E_0, and E_1, are identical.

Observe that the argument would still hold if a public setup was available to the parties in the form of a randomness beacon that is available in each round (but it would not hold in the case of private setup: in this case the parties could acquire access to a public-key directory and hence resolve any uncertainty of "permissionless" participation essentially transforming the setting into the classical permissioned one where parties have authenticated channels to each other). Moreover, it is also easy to verify that the impossibility result holds even if the adversary is computationally bounded with a computation quota on par with n participants. We remark that the above reasoning adapts a classic impossibility argument for honest-majority consensus (cf. [29]) to the permissionless setting. Note that a related impossibility result in the permissionless setting was explored for the problem of state machine replication in [49] arguing that access to a "PoW oracle" is necessary for

that problem.

Given the impossibility above, the open question is thus whether permissionless consensus is feasible once the computational power of the honest parties exceeds that of the adversary.

Protocol-Friendly PoWs. Recall from above that the PoW recipe due to Ball *et al.* [3, 4] works by first defining a specific family of polynomials, $\mathcal{F}\mathsf{OV}^k$. This family of polynomials corresponds to an efficient arithmetization of the k-orthogonal vectors problem. Evaluating $\mathcal{F}\mathsf{OV}^k$ faster in the worst-case immediately implies faster algorithms for $k\mathsf{OV}$. Ball *et al.* then showed that $\mathcal{F}\mathsf{OV}^k$ admits (a) a (tight) doubly efficient proof system and (b) a direct sum theorem.

Part (a) says that there exists a k-round interactive proof system $(\mathsf{Prover}, \mathsf{Verifier})$ for the relation $(X, Y = \mathcal{F}\mathsf{OV}^k(X))$ where Prover runs in time $\tilde{O}(n^k)$ and $\mathsf{Verifier}$ runs in time $\tilde{O}(n)$. In the case of $k = 2$, this proof system is non-interactive (although the verifier is randomized).[6]

Part (b), the direct sum theorem for $\mathcal{F}\mathsf{OV}^k$, says that evaluating $\mathcal{F}\mathsf{OV}^k$ correctly on $m = \mathsf{poly}(n)$ uniformly random instances requires $m \cdot n^{k-\epsilon}$ time, assuming $k\mathsf{OVC}$.

The PoW scheme then amounts to simply treating the challenge, X, as a random input to $\mathcal{F}\mathsf{OV}^k$, evaluating the polynomial to get $\mathcal{F}\mathsf{OV}^k(X) = y$, and generating a doubly efficient proof that the polynomial was evaluated correctly. Thus, the security and efficiency of the PoW is an immediate consequence of the soundness and efficiency (resp.) of the proof system. In particular for security, any cheating Prover that can generate m proofs in time $m \cdot n^{k-\epsilon}$ that convince the $\mathsf{Verifier}$ to accept can be immediately used (by the soundness of the proof system) to violate the consequence of the direct sum theorem, and hence $k\mathsf{OVC}$. Hence, to build a protocol-friendly PoW (with the guarantee that solving ℓ-out-of-m instances requires $\ell \cdot n^{k-\epsilon}$), it suffices to improve the direct sum theorem of Ball *et al.*[7]

To build a *seeded* (protocol-friendly) PoW, we will show that the robust direct sum theorem holds, not simply with respect to random independent instances, but to a particular joint pseudorandom distribution. Because there is an efficient, *invertible* function that samples this distribution using just $\tilde{O}(m^{1/k}n)$ random bits, we can use the sampler to expand the random seed. Because the sampler can be efficiently inverted, including the seed does not help the adversary.

Recall that the robust direct sum theorem for $\mathcal{F}\mathsf{OV}^k$ says that for any $m = \mathsf{poly}(n)$ and $\ell = m^{1-o(1)}$ and any $\epsilon > 0$, any adversary solving ℓ instances in time $\ell \cdot n^{k-\epsilon}$ succeeds with probability at most $n^{-o(1)}$. The proof of this theorem relies on two key properties of $\mathcal{F}\mathsf{OV}^k$:

1. $\mathcal{F}\mathsf{OV}^k$ is a low-degree (degree $kd = \tilde{O}(1)$) multivariate polynomial. This means that the "truth table" of this function corresponds to a Reed-Muller codeword, a locally list-decodable code;
2. $\mathcal{F}\mathsf{OV}^k$ is (worst-case) downward self-reducible: an instance X of size $\tilde{O}(N)$ for $N = m^{1/k}n$ can be reduced to solving a batch of m instances of size $\tilde{O}(n)$. Thus, solving this batch in time $m \cdot n^{k-\epsilon}$ for $\epsilon > 0$ immediately yields an algorithm for the single instance that runs in time $N^{k-\epsilon'}$ for some $\epsilon' > 0$.

Put together, this means that $\mathcal{F}\mathsf{OV}^k$ admits a downward random self-reduction with a list-decoding guarantee.

The hard pseudorandom distribution, $\mathcal{D}_{\mathrm{pr}}$, is obtained by simply evaluating the downward self-reduction on a random instance of the appropriate size.

We now are ready to sketch the high-level proof of this theorem. We will sketch (and ultimately prove) the case of hardness relative to the pseudorandom distribution $\mathcal{D}_{\mathrm{pr}}$. To obtain the robust

[6] More generally, a non-interactive proof system where the Prover runs in time $\tilde{O}(n^k)$ and $\mathsf{Verifier}$ runs in time $\tilde{O}(n^{k/2})$ is possible.

[7] We note that other methods are possible, however we believe our robust direct sum theorem to be interesting in its own right.

direct sum theorem relative to uniform and independent instances, it suffices to run the downward random self-reductions below with independent randomness (as opposed to correlated randomness).

At a very high level, our reduction works by (a) "derandomizing" the list-decoding-based reduction of [4] (based in turn on a reduction from [10] and its fine-grained adaptation in [3]) and (b) taking various steps to make it robust to "partial solvers" while preserving pseudorandomness. Still at a high level, our reduction works as follows:

1. Starting with a large (worst-case hard) instance X, split it into m smaller instances using the problem's downward self-reducibility. Note that these smaller instances will necessarily be in the support of $\mathcal{D}_{\text{pr}}$.

2. Then use another downward random self-reduction with a list-decoding guarantee to sample random T instances for each smaller instance. (Each such reduction will make a few queries to instances half the size of the instances generated in step 1.)

   The reductions in this step have the guarantee that, provided a sufficient fraction of their calls are answered correctly, they will generate a short list of advice strings: one of which encodes the correct answer to the small instance. Determining the correct advice string is a matter of correctly evaluating f on some point in the query space (which is half the size of the starting instance).

   Additionally, we will feed these downward-self-reductions correlated randomness, to ensure that the joint distribution of their ith queries is distributed according to $\mathcal{D}_{\text{pr}}$. Furthermore, we will ensure that the joint distributions of queries will be q-wise independent for some appropriately chosen q (so we can apply concentration bounds).

3. Next, locally randomly permute each batch of the ith queries from the downward random self-reductions, in a manner that does not alter the residual distribution (of the ith queries, jointly). Then, feed the permuted queries to the partial-batch solver A.

   Intuitively, if the queries were not permuted, A might ignore the first instance and the corresponding downward random self-reduction would have no correctness guarantee.

4. After this step, we simply need to determine the correct advice string in each list. Once it is found, we can correctly solve all m smaller instances and combine their solutions to solve the large instance X.

   Determining the correct advice string in any given list can be done by evaluating f correctly on any point in the query space (instances half the size of the smaller instances) that disambiguates all the advice in the list. The reduction then randomly generates a sample from something in the support of $\mathcal{D}_{\text{pr}}$ that has this property and recursively runs reduction from step 2, until instances are sufficiently small to be solved via brute force.

(Simple) Zero-Knowledge Proofs for "Easy" Problems. The notion of *zero-knowledge PoW* (ZK-PoW), introduced by Ball et al. [3], is a Proof of Work with a zero-knowledge property: transcripts between an honest prover and verifier can be simulated in time proportional to the verifier's. Note that because the PoW prover runs in polynomial time, any PoW trivially satisfies the traditional notion of zero-knowledge.

More importantly, the ability to efficiently simulate honest proofs proves to be especially helpful when using Seeded PoWs in protocols: because the challenges are correlated one cannot use non-uniform advice to simulate honest proofs and while still guaranteeing security on the remaining challenges. However, to use zero-knowledge PoWs in a protocol it is helpful to have stronger properties: we require hardness/soundness to hold even when an adversary has access to simulated proofs (and ideally a short, reusable CRS). To this end, we strengthen the notion of a zero-knowledge

PoW (ZK-PoW) to have the properties of robust zero-knowledge [23], with tight simulation and extraction.[8]

We show how to tightly compile the PoW protocols above into ZK-PoWs. At the core of these, we give a DDH-based compiler for rendering recursive sumcheck protocols zero-knowledge by committing to the prover's polynomials (with homomorphic statistically-binding commitments) and performing a few consistency checks in each round. This technique, reminiscent of Schnorr's ID scheme, may be of independent interest due to its simplicity. To make the protocol extractable (so the prover's commitments can be opened), we simply enforce that polynomials are committed to in binary with additional zero-knowledge proofs.

The results are robust (honest verifier) zero-knowledge proof systems, albeit interactive ones.

Next we turn to making these interactive ZK-PoWs non-interactive using the Fiat-Shamir transformation. To avoid using random oracles or heuristic assumptions, we employ the techniques of a recent line of works on *collision-intractable hashing* (CIH) [13, 41, 45, 20]. CIH have been constructed from a variety of assumptions and have been shown to provably instantiate Fiat-Shamir variety of proof systems. Our ZK protocols remain "Fiat-Shamir friendly," we can apply CIH to collapse our protocols to non-interactive ones. Because we are already using DDH, we will use a CIH which only relies on the sub-exponential hardness of the DDH assumption against polynomial-time adversaries [41].

However, this construction requires a few properties of the underlying proof. The resulting construction is only a CIH for the complexity class TC^0, and so we will need to ensure that any bad challenges can be found in TC^0. To do this, [41] relied on a lossy encryption scheme with low-depth decryption, as well as a trapdoor protocol which can identify the bad challenges in low depth, as well. They require that the underlying proof is witness hiding. Finally, they encrypt the statement in the proof using the decryption scheme and use the trapdoor to hide the secret key to decrypt this. Zero-knowledge comes from the lossy-ness of the encryption scheme, and soundness comes from the witness hiding of the protocol and the trapdoor protocol.

We use this CIH, but due to the "easiness" of our problem, we use slightly different techniques to employ it. Our trapdoor protocol will be a proof of knowledge for $2k-3$ DDH tuples. This will ensure that the total round complexity of this is $4k-5$, the same as the underlying zk-PoW protocol we construct described above. With these having the same round complexity, then, collapsing them into a single OR-proof can then be done using techniques from [21]. Witness hiding will be inherited from this, assuming that the underlying zk-PoW satisfies special soundness. Then, by showing that our Extractor (and by extension, Verifier) runs in TC^0, and the trapdoor protocol can recognize bad challenges in TC^0, we can correctly employ the CIH to collapse these $2k-3$ rounds to just a single Prover-Verifier message. In order to get both of these in low depth, we will give powers of exponents of our trapdoor, as in [41]. By encoding these properly, we can achieve all of this using a common random string.

The end result is a robust (multitheorem) NIZK variant of the PoW schemes above.

Permissionless Consensus from PoWs + Beacon. Given protocol-friendly PoWs, a number of challenges have still to be overcome in order to design a secure consensus protocol.

Message-encoding PoWs. First off, in order to use PoWs effectively in a permissionless setting, parties must be able to encode information with it; a PoW that does not uniquely encode a message is useless as the adversary can replay it and claim that it conveys a different message. This problem is easily solved in the RO setting, since message-encoding PoWs (aka *Signatures of Work* [34])

[8] Looking ahead, the simulator will be used in the reduction from an attacker against the consensus protocol to an attacker against PoW hardness in order to simulate the work of honest parties. Thus, an efficient simulator is important to prove our protocol secure.

of arbitrary message size can be constructed, by solving a PoW instance created by hashing the target message together with a random nonce. However, this technique does not generalize in a straightforward way to the standard model of computation.

Here we deal with the problem in a different way. First, we design a consensus protocol that only requires parties to send messages from a constant-size space, namely, $0, 1$, and $\perp$. Second, we assume that the PoW instances provided by the beacon correspond to some pre-defined message, i.e., instances are split into a constant number of equally sized subsets, X_0, X_1, and $X_\perp$, corresponding to messages $0, 1$, and $\perp$, respectively. Now, if a party wants to send message b at some round of the protocol, it parses the output of the beacon at this round as a sequence of instances $X_0, X_1, X_\perp$, picks at random a PoW instance from X_b, solves it, and communicates the computed witness to all other participants of the protocol. Given a large enough number of instances, honest parties mostly pick different instances to solve, and thus the total number of messages generated by them is proportional to their computational power.

Consensus without a PKI. The next big obstacle we have to overcome, given our limited message-encoding capability, is designing a consensus protocol where parties only send messages from a constant-size space. Previous works in the permissionless setting follow one of the following two approaches: (1) The blockchain approach [31], where chains of PoWs are used to reach consensus, and a sufficiently strong moderate hardness guarantee (from the chain) is required in order to prove security, as shown in [33], or (2) use the PoW primitive to first establish some form of a PKI, and then run a "classical" permissioned consensus protocol [2]. In the latter case, parties must be able to encode public keys in their PoWs, i.e., messages of security-parameter length, to obtain any meaningful level of security.[9] Consequently, neither of the two approaches is a good fit given the PoWs we can construct.

Our approach, instead, is based on the observation that the classical/permissioned technique of gradually building stronger forms of consensus — i.e., weak, graded and finally full consensus (cf. [26] and follow-ups) — does not require the existence of authenticated channels or a PKI. We can thus adapt the relevant protocols to work in the permissionless setting.

In more detail, *weak* and *graded* consensus protocols are basically 2-round protocols, where in the first round parties send their message, $0, 1$ (or $\perp$ in the case of graded consensus), while in the second round the sent messages are gathered and tallied to determine what the protocol output should be. Using these protocols in our setting in the form they have appeared in the literature, however, only allows for tolerating up to $1/3$ of the sent messages being created by the adversary. This is due to the fact that in our setting only an upper bound on the number of parties is known, and the algorithm has to conservatively estimate the total number of parties in order to produce outputs that satisfy the required agreement and validity properties.

To achieve the optimal threshold between honest and corrupted parties, we adapt both protocols to have an *extra third round*, where messages/PoWs received in the second round are re-broadcast for one more round, but no new PoWs are produced in the second round by the honest parties. This allows parties to build a more consistent view on the total number of messages sent in the first round, and thus tolerate the optimal corruption threshold. Now, given graded consensus, achieving full consensus is relatively straightforward in the presence of an unpredictable common coin (cf. [51, 26, 42]), whose functionality in our setting can be emulated by the randomness beacon.

Non-negligible PoW security. The final obstacle we face is that while the PoWs we use are moderately hard with high probability, we want to achieve consensus with *overwhelming* probability.

[9] Furthermore, in this approach digital signatures are also used, and thus the existence of one-way functions must be assumed, an assumption that as we show is not necessary.

First, note that the consensus protocol outlined above is secure at best with high probability, since in case moderate hardness fails the adversary can break security by sending more messages than the honest parties. To deal with this issue we showcase another (previously unnoticed) property of the weak-graded-full consensus methodology, namely, that the outputs of the weak/graded consensus sub-protocols can be error-corrected. That is, if we run the weak (resp., graded) consensus protocol multiple times, and correctness holds in more than 2/3 of the runs, then a single output that satisfies weak (resp., graded) consensus guarantees can be recovered.

It still remains to argue why we expect 2/3 of the runs to be correct with overwhelming probability. Our proof is based on sequential amplification: different invocations of the base protocols are run sequentially one after the other, with independently sampled sets of PoW instances used in each run. Thus, a sequential amplification argument can be made, ensuring that if PoW security holds with probability greater than 2/3, a constant fraction of the protocol instances run will be correct. However, unlike previous results (e.g., [22]), the runtime of the reduction involved in the amplification theorem must be concretely efficient, as our target is breaking the fine-grained security of the PoW primitive. We ensure that this is the case by simulating honestly produced PoWs using PoW instances and the related witnesses provided to the reduction as advice.

PoWs from a beacon with short outputs. To reduce the probability that two honest parties choose the same PoW to solve in the approach above, the number of PoW instances output by the beacon should be proportional to the number of parties. Using NIZK-PoWs (refer to the "zero-knowledge proofs for easy problems" subsection above) we manage to weaken our reliance on the randomness beacon. Namely, we parse its output ($O(n^2)$ bits) as a NIZK-PoW seed. As before, parties again have a number of PoW instances available proportional to their number, due to the expanding nature of the seed. The protocol is built in exactly the same way as before, from weak, to graded, to full consensus. A technical challenge arises here due to the fact that the PoW instances in a given round are correlated, as they are generated from the same seed, and thus honest parties' work cannot be simulated by instances coming from the advice string. Luckily, the NIZK simulator can be used to simulate honest work and avoid the problem.

After all these transformations, the consensus protocol we design is quite tight in terms of security: it suffices that the total number of adversarial messages produced when PoW security holds is less than the number of messages produced by honest parties. For the specific PoW construction we employ in this paper, this condition is implied by assuming that the computational power of the adversary is suitably restricted compared to that of honest parties.

2 Proofs of Work without Random Oracles

In this section, we outline our results concerning Proofs of Work. We propose subtle (yet important) changes to prior definitions as well as novel functionalities. In particular, we outline the key technical contribution: an improved non-amortizability result for a problem in fine-grained complexity. This section focuses exclusively on definitions and theorem statements. A reader looking for full details can skip this section entirely and read the stand-alone sections in Appendix A and Appendix B.

We begin with a new definition of a Proof of Work (PoW) that we believe to be closer to the "correct" definition than that of [24, 4]. Nonetheless, our starting point is the definition from [24, 4], albeit with a slightly more stringent hardness condition. Ball *et al.* (following Dwork and Naor [24]) simply required that any prover attempting to solve m random challenges must work approximately m times as hard relative to the work required for a single random challenge.

On the other hand, we will require that solving even $\ell < m$ random challenges requires approximately ℓ times as much work (provided ℓ is larger than some threshold). Informally, this means that allowing an adversary to choose a (large enough) subset of instances to concentrate on doesn't allow the adversary to correctly solve instances faster in amortization than an honest Prover.

This definition has a number of parameters:

1. t – the amount of computational work required to solve a single instance. That is, t should be a function, and in this work, we consider $t(n) = n^k$ for some constant k.
2. γ – a tightness parameter: if the honest party requires $\ell \cdot t(n)$ work to solve ℓ instances, then no adversary can solve ℓ instances with less than $\ell \cdot \gamma(t(n))$ work.

   In this work we consider $\gamma = t^{1-\Omega(1)}$, i.e., there does not exists a constant $\epsilon > 0$ such that the adversary can solve ℓ instances in time $\ell n^{1-\epsilon}$. We will abuse notation and allow γ to be a class of functions (where the above guarantee should hold for any function in the class).
3. τ – a threshold for when non-amortization holds: solving any ℓ out of m instances, where $m/\ell > \tau$, requires roughly ℓ times as much work.

   In this work, we will consider $\tau = n^{o(1)}$. In particular, it should be the case that $m/\ell > n^{\epsilon}$ for any constant $\epsilon > 0$.
4. δ – a bound on the adversary's success probability.

   In this work, we will consider $1/\delta = n^{o(1)}$. In particular, it should be the case that $\delta(n) > n^{-\epsilon}$ for any constant ϵ.

Definition 1 (Proof of Work)**.** A $(t, \gamma, \tau, \delta)$-PoW consists of two algorithms $(\mathsf{Solve}, \mathsf{Verify})$. These algorithms must satisfy the following properties:

- **Efficiency:**
  - For any $\boldsymbol{c} \in \{0,1\}^n$, $\mathsf{Solve}(\boldsymbol{c})$ runs in time $\tilde{O}(t(n))$.
  - For any $\boldsymbol{c} \in \{0,1\}^n$ and any $\boldsymbol{\pi}$, $\mathsf{Verify}(\boldsymbol{c}, \boldsymbol{\pi})$ runs in time $\tilde{O}(n)$.
- **Completeness:** For any $\boldsymbol{c}$ and any $\boldsymbol{\pi} \leftarrow \mathsf{Solve}(\boldsymbol{c})$,

$$\Pr\left[\mathsf{Verify}(\boldsymbol{c}, \boldsymbol{\pi}) = \mathtt{accept}\right] = 1,$$

  where the probability is taken over Verify's randomness.
- **Hardness:** For any function $\ell(n)$ such that $m(n)/\ell(n) \leq \tau(n)$, and any algorithm Solve^* that runs in time $\ell(n) \cdot \gamma(t(n))$ when given $m(n)$ challenges of size n as input, it holds that:

$$\Pr\left[\begin{array}{l|l} \exists I \subseteq [m], |I| \geq \ell(n) \;\&\; \forall i \in I: & (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m) \leftarrow \mathcal{U}_{n \times m} \\ \qquad \mathsf{Verify}(\boldsymbol{c}_i, \boldsymbol{\pi}_i) = \mathtt{accept} & (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{\ell(n)}) \leftarrow \mathsf{Solve}^*_m(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{m(n)}) \end{array}\right] < \delta(n),$$

  where the probability is taken over random samples as well as Solve^*'s and Verify's randomness.

Remark 1. This definition can be generalized to $(t, \gamma, \tau, \delta)$-*Interactive Proofs of Work* (iPoW) in the usual sense: by allowing $\mathsf{Solve}, \mathsf{Verify}$ to be interactive RAM machines. All efficiency, completeness, and hardness properties remain.

We will ultimately prove the following theorems:

Theorem 2. *For $k \geq 2$, suppose kOV takes $n^{k-o(1)}$ time to decide for all but finitely many inputs lengths for any $d = \omega(\log n)$. Then for any polynomial $m(n)$, there is a $(n^2, n^{o(1)}, t^{1-\Omega(1)}, n^{-o(1)})$-PoW.*

This theorem will be an immediate corollary of the second item of Lemma 9 and Theorem 27.

Theorem 3. *For $k \geq 2$, suppose $k\mathsf{OV}$ takes $n^{k-o(1)}$ time to decide for all but finitely many inputs lengths for any $d = \omega(\log n)$. Then for any polynomial $m(n)$, there is an $(n^k, n^{o(1)}, t^{1-\Omega(1)}, n^{-o(1)})$-$\mathsf{iPoW}$.*

This theorem will be an immediate corollary of the second item of Lemma 9 and Theorem 28.

2.1 Deriving Many Challenges from a Short Public Seed

We additionally augment the above definition with an additional functionality: a short public seed can be expanded into many challenges that collectively remain hard to solve (even in amortization), despite the fact that the expanded challenges are inherently correlated with one another.

Definition 4 (*Seeded* Proof of Work)**.** A (s, m)-*Seeded* $(t, \gamma, \tau, \delta)$-*PoW* consists of four algorithms $(\mathsf{Expand}, \mathsf{Solve}, \mathsf{Verify})$. These algorithms must satisfy the Efficiency and Completeness properties of a Proof of Work (Definition 24), and additionally:

- **Efficiency:** For any $\boldsymbol{\sigma} \in \{0,1\}^{s(n)}$ and $i \in [m(n)]$, $\mathsf{Expand}(\boldsymbol{\sigma}, i)$ runs in deterministic time $\tilde{O}(s(n))$.
- **Hardness:** For any function $\ell(n)$ such that $m(n)/\ell(n) \leq \tau(n)$ any algorithm Solve^* that runs in time $\ell(n) \cdot \gamma(t(n))$ when given $m(n)$ challenges of size n as input,

$$\Pr\left[\begin{array}{l|l} \exists I \subseteq [m], |I| \geq \ell(n) \;\&\; \forall i \in I : & \boldsymbol{\sigma} \leftarrow \mathcal{U}_s \\ \qquad \mathsf{Verify}(\boldsymbol{c}_i, \boldsymbol{\pi}_i) = \mathtt{accept} & (\boldsymbol{c}_i \leftarrow \mathsf{Expand}(\boldsymbol{\sigma}, i))_{i \in [m(n)]} \\ & (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{\ell(n)}) \leftarrow \mathsf{Solve}^*_\ell(\boldsymbol{\sigma}) \end{array}\right] < \delta(n),$$

where the probability is taken over $\mathcal{U}_s$, Solve^*'s, and Verify's randomness.

We provide constructions and analysis that yield the following theorems:

Theorem 5. *For $k \geq 2$, suppose $k\mathsf{OV}$ takes $n^{k-o(1)}$ time to decide for all but finitely many input lengths for any $d = \omega(\log n)$. Then for any polynomial $m(n)$, there is a $(\sqrt[k]{m(n)}n, m(n)n)$-Seeded $(n^2, n^{o(1)}, t^{1-\Omega(1)}, n^{-o(1)})$-PoW.*

This theorem is an immediate corollary of the first item of Lemma 9 and Theorem 27.

Theorem 6. *For $k \geq 2$, suppose $k\mathsf{OV}$ takes $n^{k-o(1)}$ time to decide for all but finitely many inputs lengths for any $d = \omega(\log n)$. Then for any polynomial $m(n)$, there is a $(\sqrt[k]{m(n)}n, m(n)n)$-Seeded $(n^k, n^{o(1)}, t^{1-\Omega(1)}, n^{-o(1)})$-Interactive PoW (iPoW).*

This theorem is an immediate corollary of the first item of Lemma 9 and Theorem 28.

2.2 Key Technique: A Robust Direct Sum Theorem for $f\mathsf{OV}^k$

In this subsection, we present a key technical lemma. For details on the simple DDH-based compiler for sumcheck please see Appendix B. Before describing the key lemma — a robust direct sum theorem — we begin by recalling its context in our PoW schemes.

The starting point of our work is the framework of Ball *et al.* [3, 4], who demonstrated that a PoW can be constructed from worst-case assumptions in a few steps:

1. **Prove a fine-grained worst-case to average-case reduction.** In particular, Ball *et al.* showed that (solving) $k\mathsf{OV}$ reduces to solving an arithmetization (low-degree extension) of $k\mathsf{OV}$ on average (with respect to the uniform distribution). The latter arithmetic problem amounts to simply evaluating a family of specific polynomials, $\mathcal{F}\mathsf{OV}^k = \{f\mathsf{OV}^k : \mathbb{F}_p^{knd} \to \mathbb{F}_p\}_n$[10] over a finite field, $\mathbb{F}_p$, $(p > n^k)$:

$$f\mathsf{OV}^k(U^1, \ldots, U^k) = \sum_{i_1,\ldots,i_k \in [n]} \prod_{j \in [d]} \left(1 - \prod_{\ell \in [k]} U^\ell_{i_\ell, j}\right).$$

   Assuming $k\mathsf{OV}$ requires time $n^{k-o(1)}$ in the worst case ($k\mathsf{OVC}$), then $f\mathsf{OV}^k$ requires time $n^{k-o(1)}$ on average (with respect to the uniform distribution).

2. **Prove a Direct Sum Theorem for the average-case hard problem.** This says (roughly) that an algorithm that can correctly solve a batch of m instances of $f\mathsf{OV}^k$ requires time $m \cdot n^{k-o(1)}$, assuming $k\mathsf{OVC}$.

3. **Construct a (tight) doubly efficient proof system for the average-case hard problem.** Ball *et al.* showed that $f\mathsf{OV}^k$ admits very efficient proof systems that enable a prover running in time $\tilde{O}(n^k)$ to convince a verifier that $f\mathsf{OV}^k(x) = y$: adapting ideas from [55] they gave a non-interactive protocol where the verifier runs in time $\tilde{O}(n^{k/2})$, adapting the sumcheck protocol [46] they gave a k-round interactive protocol where the verifier runs in time $\tilde{O}(n)$.

Thus, the resulting PoWs are formed by simply asking the Prover to compute $f\mathsf{OV}^k(x) = y$ for a random challenge x and then prove the statement that "$f\mathsf{OV}^k(x) = y$" using the doubly efficient proof system. Thus the security of the proof system follows immediately from the soundness guarantees of the proof system and the Direct Sum Theorem (non-amortizing hardness).

Thus, in order to "upgrade" PoW security to the definition we use here, i.e., to hold against dishonest provers that only choose a fraction of challenges, it suffices to strengthen the Direct Sum Theorem (Item 2 above) to a *Robust Direct Sum Theorem*: solving a ℓ out of m uniformly random independent instances requires time $\ell \cdot n^{k-o(1)}$ (for large enough ℓ). To get a Seeded PoW we show that this Robust Direct Sum Theorem can be *derandomized*: namely, that it holds relative to a "pseudorandom" distribution over ℓ correlated instances that can be efficiently sampled from a short seed.[11] The sampler for the pseudorandom distribution is thus the Expand procedure for the Seeded Proof of Work.

Although we state this theorem for the specific case of $f\mathsf{OV}^k$, we note that our techniques can be generalized to hold for any low-degree polynomial that is downward-self-reducible.

Before we state the derandomized Robust Direct Sum theorem, we must describe the pseudorandom distribution. In particular, $\mathcal{D}_{\mathrm{pr}}^{r^k,n,d,p}$ denotes the distribution generated by sampling $\boldsymbol{U}$ uniformly at random and applying the downward-self-reduction. Namely, $\boldsymbol{U}$ is drawn uniformly from $(\mathbb{F}_p^{rn \times d})^k$, viewed as k lists of rn d-dimensional vectors with each list partitioned into r blocks, and the resulting output is simply all combinations formed by concatenating one block from each list:

[10] d is a function n such that $d = \omega(\log n)$ (for hardness) and $d = \tilde{O}(1)$ (for efficiency). Typically we fix the choice of $d = \log^2 n$ for concreteness.

[11] Note that this distribution is not pseudorandom in the traditional cryptographic sense: it is easy to distinguish from uniform (and moveover the adversary is given the seed describing a sample); it is only pseudorandom for the purposes of proving a Threshold Direct Sum theorem for $f\mathsf{OV}^k$.

$$\left(U^{1,j_1},\ldots,U^{k,j_k}\right)_{j\in[r]^k} \qquad \text{where} \qquad \begin{bmatrix} U^{1,1} \\ \vdots \\ U^{1,r} \end{bmatrix},\cdots,\begin{bmatrix} U^{1,1} \\ \vdots \\ U^{1,r} \end{bmatrix} \stackrel{u}{\leftarrow} (\mathbb{F}_p^{rn\times d})^k \tag{1}$$

We can now state our Robust Direct Sum Theorems.

Theorem 7 (Robust Direct Sum Theorem for $f\mathsf{OV}^k$)**.** *Assuming the $k\mathsf{OV}$ conjecture, for any $m=\mathsf{poly}(n)$ and $\ell=m^{1-o(1)}$ and any $\epsilon>0$, no algorithm running in time $\ell\cdot n^{2-\epsilon}$ that is given m random independent inputs to $\mathcal{F}\mathsf{OV}^k$ can correctly evaluate ℓ of them with probability $1/n^{o(1)}$.*

Theorem 8 (Derandomized Robust Direct Sum Theorem for $f\mathsf{OV}^k$)**.** *Assuming the $k\mathsf{OV}$ conjecture, for any $m=\mathsf{poly}(n)$ and $\ell=m^{1-o(1)}$ and any $\epsilon>0$, no algorithm running in time $\ell\cdot n^{2-\epsilon}$ that is given m $\mathcal{F}\mathsf{OV}^k$ instances drawn from $\mathcal{D}_{pr}$ inputs can correctly evaluate ℓ of them with probability $1/n^{o(1)}$.*

Both of these Theorems follow from our key technical lemma: an efficient reduction that shows how to use an algorithm that violates the (derandomized) Threshold Direct Sum Theorem for $f\mathsf{OV}^k$ in order to efficiently solve $f\mathsf{OV}^k$ on any instance.

Lemma 9. *Let $m(n)$ be a polynomial and $\ell(n)$ a function such that $\ell(n)<m(n)$, and p a prime such that $\log(p)=n^{o(1)}$ and $p>6\left(\frac{12m(n)}{\delta\ell(n)}\right)^2 k2^k d\log\log(n)\log(n)m(n)$. Let $dk=\tilde{O}(1)$, and let $\{f:\mathbb{F}_p^{knd}\to\mathbb{F}_p\}$ denote $\mathcal{F}\mathsf{OV}^k$ in what follows.*

1. **Pseudorandom reduction.** *Let A be a randomized algorithm running in time $t_A(n)\le\ell(n)\cdot n^{k-\epsilon}$ that on input $X_1,\ldots,X_m$ drawn from $\mathcal{D}_{pr}$ outputs $\hat{y}_1,\ldots,\hat{y}_m$ such that with probability at least $\delta(n)$, $|\{i\in[m]:f(X_i)=\hat{y}_i\}|\ge\ell(n)$.*[12]

   Then, there is a randomized algorithm A' that on input X, of size $\theta_{d,p}(n)$, computes $f(X)$ in time $\tilde{O}\left(\frac{mn^{k-\epsilon}}{(\delta\frac{\ell}{m})^{\Theta(1)}}\right)$, with probability at least $2/3$.

2. **Uniformly random reduction.** *Similarly, let A be a randomized algorithm running in time $t_A(n)\le\ell(n)\cdot n^{k-\epsilon}$ that on input $X_1,\ldots,X_m$ drawn independently and uniformly at random from $\mathbb{F}_p^{knd}$ outputs $\hat{y}_1,\ldots,\hat{y}_m$ such that with probability at least $\delta(n)$, $|\{i\in[m]:f(X_i)=\hat{y}_i\}|\ge\ell(n)$.*

   Then, there is a randomized algorithm A' that on input X, of size $\theta_{d,p}(n)$, computes $f(X)$ in time $\tilde{O}\left(\frac{mn^{k-\epsilon}}{(\delta\frac{\ell}{m})^{\Theta(1)}}\right)$, with probability at least $2/3$.

2.3 Simulatable, Non-Interactive PoW (with Large Proving Times)

Finally, we turn to traditional (heavy) cryptographic tools (and a CRS) to simultaneously (a) make honest proofs efficiently simulatable (i.e., robustly zero-knowledge) and (b) "collapse" Interactive PoW (iPoW) to *non-interactive* PoW (i.e., Fiat-Shamir) achieving arbitrarily large polynomial gaps between proving time and verification time (as opposed to just quadratic).

To this end, we introduce a new robust notion of zero-knowledge PoW. In contrast with the definition presented by Ball *et al.* [4], this new definition retains soundness/hardness in the presence of simulated proofs for correlated challenges (with a short CRS — independent of the number of challenges).

[12] If it is not the case that $t(n)>>m(n)$, imagine that A outputs $(i,\hat{y}_i)_{i\in S}$ for some $S\subseteq[n]$.

Our definition is simply robust zero-knowledge [23] (generalized from NP relations to interactive proofs) with an additional efficiency requirement applied to a (interactive) PoW. In this generalization of robust zero-knowledge, the extractor (very efficiently) extracts a successful prover strategy (not simply a witness for an NP relation). Moreover, we require that the new proof system should roughly preserve the complexity of the original, and that the simulation should be tightly bounded to an adversarial verifier. (Without the last requirement, zero-knowledge is trivial to achieve for PoW systems where the prover itself runs in polynomial time.)

Definition 10. Given an interactive proof $\Pi = (P, V)$, $\Pi' = (q, \mathsf{P}', \mathsf{V}', \mathsf{S}' = (\mathsf{S}'_1, \mathsf{S}'_2), \mathsf{E}'))$ is a *robust ZK argument* for Π, if $\mathsf{P}', \mathsf{V}', \mathsf{S}', \mathsf{E}' \in PPT$ and $q(\cdot)$ is a polynomial such that the following conditions hold:

- **Efficiency Preserving.** P' runs in time $\tilde{O}(T_P)$ where T_P is the runtime of P, and $\mathsf{V}', \mathsf{S}', \mathsf{E}'$ all run in time $\tilde{O}(T_V)$ where T_V is the runtime of V.
- **Completeness.** For all $x \in L$ of length λ, all w such that $\Pr[\langle P(x,w), V(x)\rangle = 1]$, and all $\Omega \in \{0,1\}^{q(\lambda)}$, $\mathsf{V}'(\Omega, x, \mathsf{P}'(\Omega, w, x))] = 1$.
- **Multi-Theorem Zero-Knowledge.** For all PPT adversaries $\mathcal{A}$, we have that $\textsc{Real}(\lambda) \approx \textsc{Sim}(\lambda)$, where

$$\textsc{Real}(\lambda) = \{\Omega \leftarrow \{0,1\}^{q(\lambda)}; out \leftarrow \mathcal{A}^{\mathsf{P}(\Omega,\cdot,\cdot)}(\Omega); \text{Output } out\},$$

$$\textsc{Sim}(\lambda) = \{(\Omega, tk) \leftarrow \mathsf{S}'_1(1^\lambda); out \leftarrow \mathcal{A}^{\mathsf{S}''_2(\Omega,\cdot,\cdot,tk)}(\Omega); \text{Output } out\},$$

  and $\mathsf{S}''_2(\Omega, x, w, tk) \stackrel{\text{def}}{=} \mathsf{S}'_2(\Omega, x, tk)$ if (x, w) such that $\Pr[\langle P(x,w), V(x)\rangle = 1] \geq 2/3$ and otherwise outputs `failure`.

  (We say Π' is robust *honest verifier* zero-knowledge if the above condition holds simply for V'.)
- **Extractability.** For all PPT $\mathcal{A}$,

$$\Pr\left[(\Omega, tk) \leftarrow \mathsf{S}_1(1^\lambda); \Pr[\langle \mathsf{E}'^{\mathcal{A}^{\mathsf{S}_2(\Omega,(x),tk)}(\Omega,x)}(\Omega, x, tk), V(x)\rangle = 1] \leq 2/3 \quad \right] \leq \mathsf{negl}(\lambda),$$

If Π' is *non-interactive* we say that it is a robust non-interactive zero-knowledge argument for Π.

Definition 11. A protocol Π is *robust zero-knowledge* $(t, \gamma, \tau, \delta)$*-(i)PoW (ZK-PoW)* if it is a robust ZK argument for a $(t, \gamma, \tau, \delta)$-iPoW.

We say such a protocol (with an additional Expand algorithm) is a *seeded ZK-PoW* if it is a robust ZK argument for a seeded iPoW. Finally, we say such a protocol is a *NIZK-PoW* if it is non-interactive.

We show that by additionally assuming subexponentially-secure DDH one can construct robust seeded NIZK-PoW.

Definition 12. The DDH assumption states that there exists some $\mathcal{G} = \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}}$ a group ensemble with efficient representation, where each $\mathbb{G}_\lambda$ is a cyclic group of prime order $p(\lambda)$ such that, for any PPT $\mathcal{A}$, we have

$$|\Pr[\mathcal{A}(1^\lambda, g, g^a, g^b, g^{ab}) = 1 : a, b \stackrel{\$}{\leftarrow} \mathbb{Z}_{p(\lambda)}] -$$
$$\Pr[\mathcal{A}(1^\lambda, g, g^a, g^b, g^c) = 1 : a, b, c \stackrel{\$}{\leftarrow} \mathbb{Z}_{p(\lambda)}]| \leq \mathsf{negl}(n),$$

where g is a generator for $\mathbb{G}_\lambda$.

The Sub-exponential DDH assumption instead assumes that the above is true for all non-uniform $\mathcal{A}$ that run in time $\lambda^{O((\log\log\lambda)^3)}$.

Theorem 13. *Let $k \in \mathbb{N}$, if DDH is hard for non-uniform $\lambda^{O(\log^3 \log \lambda)}$ distinguishers and $k\mathsf{OV}$ takes $n^{k-o(1)}$ time to decide for all but finitely many input lengths (when $d = \log^2 n$), then there exists a robust seeded $(n^k, n^{o(1)}, t^{1-\Omega(1)}, n^{-o(1)})$-NIZK-PoW.*

3 Consensus from PoW and a Beacon

In this section we show how to achieve consensus in the permissionless setting given access to a random beacon and assuming the existence of a hard PoW scheme. We start with some more details on the setting in which we analyze security.

3.1 Protocol Execution and Security Model

We will analyze the Consensus protocol in a concrete complexity, synchronous, and permissionless model, similarly to [33]. In more detail:

Security model. The set of n parties $\{P_1, \ldots, P_n\}$ running the protocol is fixed, and both the parties and the adversary are modeled as Interactive Word RAM machines. Throughout, we consider algorithms in the Word RAM model of computation with $O(log(\lambda))$-bit words, where λ is used to denote the security parameter. The adversary is active/byzantine and can corrupt up to t parties (statically) in order to break security.

Communication model. The protocol advances in rounds and communication happens through a diffusion functionality $\mathcal{F}_{\text{DIFF}}$. Messages sent by honest parties through $\mathcal{F}_{\text{DIFF}}$ arrive to all other parties within a round. On the other hand, the adversary may send messages to an adaptively chosen subset of the parties. Communication is not authenticated, and thus parties cannot directly tell which was the sender of a message.

Setup. Parties have access to a beacon functionality $\mathcal{F}_{\text{BEACON}}$, which when queried at any round outputs a uniformly random string from $\{0,1\}^{\mathsf{poly}(\lambda)}$; all queries of some round get the same response.

Concrete computation model. Each party has a concrete upper bound of c computational steps per round (including corrupted parties), following the formulation of [33]. The adversary has an upper bound θ on the number of messages it can send per round (similarly to [2]).[13]

Next, we recite the classical definition of Byzantine agreement (aka consensus).

Definition 14. A protocol Π implements byzantine agreement among n parties iff the following two properties are satisfied in the presence of an adversary $\mathcal{A}$ who might corrupt some of them:

- **Validity:** If all parties have the same input $b \in \{0,1\}$, then they all output b;
- **Agreement:** All parties output the same value $b \in \{0,1\}$.

We will arrive at our final protocol through a series a transformations, starting with a protocol providing weaker security guarantees, namely, Weak Consensus [43].

3.2 Weak Consensus

Next, we show how to achieve Weak Consensus in our setting. First, we provide the relevant security definition; note the relaxation of the Agreement property with respect to the original definition of Consensus.

[13] This choice is made in [2] to avoid deniable-of-service attacks that may deplete the round-bounded computational power of the honest parties.

Definition 15 (Weak Consensus)**.** A protocol Π implements Weak Consensus iff the following two properties are satisfied:

- **Weak Agreement:** There exists $y \in \{0,1\}$ such that all honest parties output $y_i \in \{y, \perp\}$.
- **Validity:** If all honest parties have the same input $b \in \{0,1\}$, they all output $y_i = b$.

The main idea of our 3-round Weak Consensus protocol is as follows. We are going to interpret the output of the beacon at the first round of the protocol as a sequence of PoW instances $(x_i)_i$. The parties will then send messages by solving PoW instances. Sending a witness of one the instances in the first half of the sequence, will correspond to sending 0, while sending a witness of one of the instances from the other half will correspond to sending 1. To avoid solving the same instances, and thus being misrepresented, honest parties are going to pick at random and solve a PoW instance encoding their message, which they will subsequently diffuse together with the respective witness to all other parties.

Given that in our setting parties are not aware of their total number, and to ensure that their views about this number is somewhat consistent, parties are going to resend valid PoWs for one more round, and then estimate their total number based on the PoWs received. Note that while only messages sent during the first round are counted as votes (0 or 1), the total number of votes estimation (k) is based on the number of PoWs sent in the *first and second* rounds. This allows our protocol to have maximal security (i.e., $t < n/2$, where t is the number of corrupted parties) even if the total number of votes is not known. Essentially, late votes do not help the adversary swing the result in one direction or the other.

Finally, rounds in our protocol have different "duration"[14] (parameters r_p, r_v), in order to allow parties to execute the relevant required operations: In the first round they are expected to solve a PoW, while in the second and third rounds they should be able to verify all PoWs received. As we explain later, these parameters should be carefully picked, as they also determine the available time the adversary has to solve more PoWs than the parties it controls.

Protocol $\text{WeakConsensus}_{r_p,r_v}(P_i, b_i)$

Initialization:

- Initialize sets $P_i^0, P_i^1, P_i^{\text{late}}$ to $\emptyset$.

Round 1 (round duration := r_p):

- Let $(x_i)_{i\in[m]}$ be the output of $\mathcal{F}_{\text{BEACON}}$ at this round, parsed as a sequence of PoW instances. Let $X_0 = (x_i)_{i\in[m/2]}$ denote the first half of the instances, and X_1 the second half.
- Compute $w_j := \mathsf{PoW.Solve}(x_j)$, where $x_j \leftarrow X_{b_i}$.
- Send (x_j, w_j) to $\mathcal{F}_{\text{DIFF}}$ and proceed to the next round.

Round 2 (round duration := r_v):

- Fetch messages from $\mathcal{F}_{\text{DIFF}}$. For every message of the form (x, w), if $x \in X_b$ (for some $b \in \{0,1\}$) and $\mathsf{PoW.Verify}(x, w) = 1$, add (x, w) to P_i^b.
- Send P_i^0, P_i^1 to $\mathcal{F}_{\text{DIFF}}$ and proceed to the next round.

Round 3 (round duration := r_v):

- Fetch messages from $\mathcal{F}_{\text{DIFF}}$. For every message of the form (x, w), if $x \in X_b$ (for some $b \in \{0,1\}$), $(x, w) \notin P_i^0 \cup P_i^1$, and $\mathsf{Pow.Verify}(x, w) = 1$, add (x, w) to P_i^{late}.
- Let $k_i = |P_i^0| + |P_i^1| + |P_i^{\text{late}}|$.

[14] We overload the term "round" here, and assume that in a round of duration x each party can take up to $x \cdot c$ computational steps.

– P_i outputs $y_i := \begin{cases} 0 & \text{if } |P_i^0| > k_i/2; \\ 1 & \text{if } |P_i^1| > k_i/2; \\ \perp & \text{otherwise.} \end{cases}$

Next, we analyze the protocol presented above. At this stage, we are going to assume a lower and an upper bound on the number of messages produced by the honest parties and the adversary, respectively; later on we will show how we can get rid of this assumption. Having these bounds in place, the analysis of the protocol boils down to: (i) a simple counting argument about the PoWs generated showing that both validity and the weak consensus property are satisfied, and (ii) that honest parties have enough time to perform any operations required by the protocol.

Lemma 16. *Protocol* $\text{WeakConsensus}_{r_p,r_v}$ *achieves Weak Agreement unconditionally, and Validity assuming that the number of distinct PoW witnesses produced by the honest parties exceeds that produced exclusively by* $\mathcal{A}$*, for* $t_v \geq 2\theta/\sigma$*,* $t_p := t_v^2$ *,* $r_p := t_p/c$ *and* $r_v := r_p \cdot \sigma/2$*, for some* $\sigma \in (0,1)$*, where* t_p, t_v *denote the proving and verification costs of the PoW primitive, respectively.*

Proof. We start by arguing that honest parties have enough time to process and forward all valid messages received in round 2.

Claim 1. *Each honest party has enough computational power to process all the PoWs received during round 2 of the protocol.*

Proof of claim. By assumption, each honest party is able to take c computational steps per unit of time, while the adversary takes $t \cdot c$. Let θ be an upper bound on the total messages sent in each round.

We want to show that (i) honest parties have enough time to compute a PoW in round 1, and (ii) honest parties are able to process all θ messages they receive at the beginning of round 2. These conditions are described by inequalities $t_p \leq r_p \cdot c$ and $\theta t_v \leq r_v \cdot c$, respectively. It holds that:

$$r_p c \geq c t_p/c = t_p,$$

and

$$r_v c = r_p c \sigma/2 = \frac{t_v^2 \sigma c}{2c} = t_v^2 \sigma/2 > \frac{t_v 2\theta\sigma}{2\sigma} = \theta t_v.$$

Hence, the claim follows. ⊣

Next, we show that the above claim about the network conditions is sufficient to prove that protocol WeakConsensus achieves Weak Agreement unconditionally. For the sake of contradiction, assume that there exists two honest parties P_i, P_j such that P_i outputs 0 while P_j outputs 1. It follows that:

$$|P_i^0| > (|P_i^0| + |P_i^1| + |P_i^{\text{late}}|)/2 \Leftrightarrow |P_i^0| > |P_i^1| + |P_i^{\text{late}}|,$$

and, symmetrically, that $|P_j^1| > |P_j^0| + |P_j^{\text{late}}|$. Adding both inequalities, we have that

$$|P_j^1| + |P_i^0| > |P_i^1| + |P_i^{late}| + |P_j^0| + |P_j^{\text{late}}|. \tag{2}$$

On the other hand, by the guarantees provided by $\mathcal{F}_{\text{DIFF}}$, it should hold that

$$|P_i^0| \leq |P_j^0| + |P_j^{late}| \text{ and } |P_j^1| \leq |P_i^1| + |P_i^{\text{late}}|,$$

since by the above claim any valid PoW witness seen by P_i in the first round will be received by P_j in the second round. Adding the two inequalities, we get that

$$|P_j^1| + |P_i^0| \leq |P_i^1| + |P_i^{\text{late}}| + |P_j^0| + |P_j^{\text{late}}|,$$

which contradicts inequality 5. Thus, Weak Agreement holds unconditionally.

Next, we turn our attention to Validity. Let b be the common input of all honest parties. Let x denote the number of distinct PoW witnesses computed by the honest parties, and y the number of any other PoW witnesses produced by the adversary. By our assumption, it holds that $x > y$. For any party P_i, it should hold that $k_i \leq x + y$, which implies by our assumption that $x > k_i/2$. Given that all honest PoWs are received in the second round, it follows that all parties are going to output b. Thus, Validity follows. □

Security amplification. Protocol WeakConsensus achieves Validity as long as honest parties produce more PoWs than the adversary. Next, we showcase a protocol that ensures that this condition holds with overwhelming probability given two assumptions: (i) That there exists a secure PoW scheme (Definition 24), and (ii) that the number of parties the adversary can corrupt is sufficiently bounded.

Assumption 1 (PoW). There exists a $(\lambda^2, \gamma(\cdot), \lambda^{o(1)}, \lambda^{-o(1)})$-PoW.

Assumption 2 (Corruption bound). There exists $\sigma \in (0, 1)$, such that:

$$t < (n - t) \cdot (1 - \sigma)\frac{\gamma(\lambda^2)}{\lambda^2} - \sigma.$$

Note, that Assumption 2 can be stated equivalently as a restriction on the total computational power controlled by the adversary, i.e, $t \cdot c$ steps per round, compared to that controlled by honest parties, i.e., $(n - t)c$ steps per round. Moreover, note that if the computational advantage of the adversary over honest parties on computing PoWs approaches zero, i.e, $\gamma(\lambda^2) \approx \lambda^2$, then Assumption 2 is equivalent to a necessary condition for consensus: $2t < n$.

The main idea behind the security amplification protocol is to run WeakConsensus multiple times sequentially, and then try to error-correct the outputs. Our assumption about PoW hardness holding with good probability, together with Weak Agreement holding unconditionally for the base protocol, are going to be sufficient to ensure that we can error-correct the output in a consistent manner with overwhelming probability in the security parameter.

Protocol $\text{AmpedWeakConsensus}_{r_p, r_v}(P_i, b_i)$

Initialization:
- Initialize counters $t_i^0, t_i^1, t_i^\perp$ to 0.

Rounds 1 to l:
- $c := \text{WeakConsensus}_{r_p, r_v}(P_i, b_i)$;
- $t_i^c := t_i^c + 1$.

Round $l + 1$:
- Output $y_i := \begin{cases} 0 & \text{if } t_i^0 > 2l/3; \\ 1 & \text{if } t_i^1 > 2l/3; \\ \perp & \text{otherwise.} \end{cases}$

Lemma 17. *If Assumptions 1 and 2 hold, Protocol* AmpedWeakConsensus$_{r_p,r_v}$ *achieves Weak Consensus with overwhelming probability in λ, for $l = \log^2(\lambda)$, and r_p, r_v, t_p, t_v as in Lemma 16.*[15]

Proof. We start by proving some initial claims that are going to help with our analysis. First, we show that in more than 5/6 of the WeakConsensus invocations, honest parties are going to produce close to $n - t$ distinct PoW witnesses.

Claim 2. *Given any constant $\sigma \in (0, 1)$, for a large enough λ, honest parties are going to produce more than $(1 - \sigma)(n - t)$ distinct PoWs in 5/6 of the* WeakConsensus *invocations with overwhelming probability in λ.*

Proof of claim. We start by analyzing the probability that honest parties produce enough distinct PoW witnesses in a single instance of WeakConsensus. Let $k := n - t$, be the number of honest parties and $m := 2r$, be the number of different PoW instances considered by parties running WeakConsensus. In the worst case, all parties will have the same input and will select which instance to solve among r different ones. As shown earlier, all the honest parties have enough computational steps available per round to finish computing a PoW witness.

Since each of the honest parties picks a PoW instance at random, some of them may end up picking the same one. Making r large enough would in general minimize the number of collisions among honest parties. However, in order to preserve the hardness of PoWs we are restricted in how many PoW instances should be available to the adversary, compared to how many it solves. Given that the number of instances solved by the adversary is going to be proportional to that solved by honest parties, the condition we want to satisfy is that $r/k \leq \tau(\lambda) = \lambda^{o(1)}$.

We split our analysis into two cases. In the first one, assume that $k = O(1)$ and set $r = 7k^2$. In the second one, we assume that $k = \omega(1)$, and set $r = ak$, for some large enough $a \in \mathbb{N}$. Note that in both cases it holds that $r/k = O(1) = O(\lambda^{1/\log(\lambda)}) \in \lambda^{o(1)}$.

In the first case, no collision will happen with probability at most

$$\frac{k(k-1)}{2r} \leq \frac{k^2}{7k^2} = 1/7,$$

as stated by our claim.

In the second case, we will analyze the number of collisions as a balls and bins process. Namely, we are going to use the Poisson approximation [47], where the event of interest is analyzed in a setting (the "Poisson" setting) where the load in each bin is assumed to be an independent Poisson variable with mean k/r. Results obtained in this setting can then be translated back to the "exact" setting, where the are dependencies between the loads of different bins, with some loss on probability.

The event E we care about upper-bounding is the number of full bins being greater than a target value. This event depends entirely on the number of balls in each bin. Furthermore, as the number of balls increases, $\Pr[E]$ increases. By [47], if $\Pr[E] = p$ in the Poisson setting, $\Pr[E] \leq 2p$ in the exact setting.

We proceed to bound the probability of E occurring. Assume that we have t i.i.d. discrete Poisson random variables $(X_i)_i$, each with parameter $\mu = k/t$, denoting the number of balls in the i-th bin. We have that $\Pr[X_i > 0] = 1 - e^{-\mu}$. Let random variable Y_i be equal to 1 iff $X_i > 0$, and let $Y = \sum_{i=1}^{t} Y_i$. It then holds that $\mathbb{E}[Y_i] = 1 - e^{-\mu}$ and $\mathbb{E}[Y] = t(1 - e^{-k/r})$. For $r = ak$, for some constant $a > 2$, we have by the Chernoff bound that for any $\delta \in (0, 1)$:

$$\Pr[Y \leq (1-\delta)\mathbb{E}[Y]] = \Pr[Y \leq (1-\delta)t(1-e^{-k/r})] \leq e^{-\frac{\delta^2 ck(1-e^{-1/a})}{3}} = e^{-O(k)}.$$

[15]The σ parameter used in the definitions of r_p, r_v, t_p, t_v is the one from Assumption 2. This will also be the case for the rest of the statements proved in this section.

Firstly, notice that since $k \in \omega(1)$, for any selection of a there exists a large enough λ such that the probability of our desired event becomes smaller than any constant, we choose in particular $1/14$. By the Poisson approximation this implies that the event in the exact setting happens with probability at most $1/7$.

Secondly, $a(1 - e^{-1/a})$ tends to 1 as a goes to infinity. Thus, for any constant ϵ, there exists a large enough constant a, such that $a(1 - e^{-1/a}) \geq 1 - \epsilon$. This easily implies that for any $\sigma \in (0, 1)$, there exists a, δ such that $(1 - \delta)t(1 - e^{-k/t}) \geq (1 - \sigma)k$. Hence, for any $\sigma \in (0, 1)$, there exists a $a \in \mathbb{N}$, such that with probability at least $6/7$ honest parties mine at least $(1 - \sigma)k$ different PoWs.

To finish proving our claim, let E_i the probability of event E happening in the i-th invocation of protocol WeakConsensus. Note that $\{E_i\}_{i \in [l]}$ is a sequence of independent events. By a standard Chernoff bound argument we can show that with overwhelming probability in λ, in less than $1/6$ of the WeakConsensus invocations honest parties will produce less than $(1 - \sigma)k$ PoWs. Thus, the claim follows. ⊣

Next, we show that in more than $5/6$ of the WeakConsensus protocol invocations the adversary is going to produce less than $n - t$ PoWs with overwhelming probability in λ.

Claim 3. *The number of PoW witnesses produced by $\mathcal{A}$ (different from those produced by the honest parties) in more than $5/6$ of the* WeakConsensus *invocations is at most $t' := (1 - \sigma/2)(n - t)$, for any $\sigma \in (0, 1)$, with overwhelming probability in λ.*

Proof. In contradiction, assume that there exists and adversary $\mathcal{A}$ such that in $1/6$ of the WeakConsensus invocations produces more than t' PoWs with non-negligible probability. We are going to use $\mathcal{A}$ to construct another adversary that breaks the security of the PoW scheme. First, we argue that if $\mathcal{A}$ produces more than t' PoWs in a WeakConsensus invocation with probability at most $\epsilon := 1/6 - \sigma$, then it will produce more than t' PoWs in less than $1/6$ of the invocations with overwhelming probability.

Let $\mathcal{T}$ be the protocol's execution tree when we fix $\mathcal{A}$. The tree has nodes $n_{i,j}$, where $n_{i,j}$ reflects the execution state just before $\mathcal{A}$ receives the i-th beacon output. Index j runs over all possible coin-flip histories up to that point. Wlog, assume that between receiving any two consecutive challenges the adversary and the honest parties perform exactly l coins flips. Thus, for any level $i \in [m]$, there are at most 2^{il} nodes, i.e., $j \in [2^{il}]$.

Next, we argue that if for every subtree defined by $n_{i,j}$ ($i \in [m]$), at most $\epsilon 2^l$ paths are successful for the adversary, in the sense that the adversary generates more than t' PoWs, then the fraction of paths with at least $m/6$ successes is negligible in λ. W.l.o.g., assume that exactly $\epsilon 2^l$ paths are successful in any such subtree. Namely, we will show that in that case there are at most $2^{ml} \cdot \mathsf{negl}(\lambda)$ paths with at least $m/6$ successes in $\mathcal{T}$.

Let $a_{i,c}$ denote the number of paths ending at some node at level $i + 1$ that have exactly c successes. By our assumptions it holds that $a_{1,0} = 2^l(1 - \epsilon), a_{1,1} = 2^l\epsilon$ for the first level, and

$$a_{n,c} = a_{n-1,c-1}2^l\epsilon + a_{n-1,c}2^l(1 - \epsilon)$$

for any subsequent level, where $a_{n,-1} = a_{n,n+1} = 0$. The equalities follow by the fact that, at every node, $2^l\epsilon$ of the paths are going to increase their successes by one, and the rest are going to retain the same value of successes. It is easy to see that the solution of this recursion is

$$a_{n,c} = 2^{nl}\binom{n}{c}\epsilon^c(1 - \epsilon)^{n-c}.$$

We are interested in bounding the sum $\sum_{i=m/6}^{m} a_{m,i}$. Note that for any i, $r_i = a_{m,i}/2^{m \cdot l}$ is equal to the probability of i successes in m independent Bernoulli trials, where each trial succeeds with

probability ϵ. Thus, we can use the Chernoff bound to upper-bound the probability that $\sum_{i=m/6}^{m} r_i$ is less than $m/6$ by

$$e^{-((1-6\epsilon)/(6\epsilon))^2 \epsilon m/3} \leq e^{-\sigma^2 m/108\epsilon} \leq \lambda^{-\Omega(\log(\lambda))} \leq \mathsf{negl}(\lambda),$$

where we have used the fact that $m = \log^2(\lambda)$. Hence, $\sum_{i=m/2}^{m} a_{m,i} \leq 2^{ml}\mathsf{negl}(\lambda)$, as we have claimed.

Therefore, since we have assumed that $\mathcal{A}$ produces more than t' PoWs in at least 1/6 of the WeakConsensus invocations with non-negligible probability, by the analysis above it must be the case that there exists an $n_{i,j}$ where the adversary computes more than t' PoWs with probability greater than ϵ. We are going to use the state of node $n_{i,j}$ to construct an adversary $\mathcal{A}'$ that contradicts our assumption about the PoW scheme being secure.

Let $2r$ be the size of the output of $\mathcal{F}_{\text{BEACON}}$. $\mathcal{A}'$ works as follows: It takes as input a sequence of $2r$ PoW instances $(x_i)_i$ and some non-uniform advice. We choose the advice to contain $n-t$ randomly sampled PoW instances $(x'_i, w'_i)_i$ together with their respective witnesses. $\mathcal{A}'$ is going to construct a "fake" beacon output for $\mathcal{A}$. Given the input bits of honest parties at node $n_{i,j}$, it is going to replace randomly selected PoW instances from $(x'_i)_i$ with instances from $(x'_i)_i$. Specifically, if an honest party has input 0 (resp. 1), then a randomly selected instance from the first half (resp. second half) of $(x_i)_i$ is replaced. Note, that this process preserves the possibility that two honest parties choose to solve the same PoW instance, thus perfectly mimicking the real world. We denote by $Z = (z_i)_i$ the resulting sequence of instances.

Next, $\mathcal{A}'$ is going to initialize $\mathcal{A}$ to the state described by $n_{i,j}$, and provide Z as the output of the beacon. At the end of the first round, it is going to send $\mathcal{A}$ the witnesses $(w'_i)_i$ that it got as advice, simulating the behavior of the honest parties. Then, it is going to verify the messages $\mathcal{A}$ sent in the first round, and forward any valid PoWs it produces. Finally, it is going to verify the messages $\mathcal{A}$ sent in the second round. Finally, $\mathcal{A}'$ outputs any PoW witnesses produced by $\mathcal{A}$ that do not correspond to the pre-solved instances it has planted in Z.

We will first analyze the running time of $\mathcal{A}'$. As before, let $t_v \geq 2\theta/\sigma$, $t_p = t_v^2$, $r_p := t_p/c$ and $r_v := r_p \cdot \sigma/2$. $\mathcal{A}$ takes a total of $(r_v + r_p)tc$ steps. In addition, $\mathcal{A}'$ takes an additional $2t_v\theta$ steps in order to verify messages sent by $\mathcal{A}$ in the first and second rounds. Hence, $\mathsf{Steps}_{\mathcal{A}'} \leq (r_v+r_p)tc+2t_v\theta$. Now, for $\mathcal{A}'$ to be breaking PoW's security it must be that $\mathsf{Steps}_{\mathcal{A}'} < \gamma(t_p)t' = \gamma(t_p)(1-\sigma/2)(n-t)$. It holds that:

$$\begin{aligned}
\mathsf{Steps}_{\mathcal{A}'} \leq (r_v + r_p)tc + 2t_v\theta &\leq \frac{t_p(1+\sigma/2)tc}{c} + 2t_v\theta \\
&\leq t_p(1+\sigma/2)t + t_v^2\sigma \\
&\leq t_p((1+\sigma/2)t + \sigma).
\end{aligned}$$

On the other hand, by our assumption about the number of corruptions, we have that:

$$\begin{aligned}
(1-\sigma)(n-t)\gamma(\lambda^2)/\lambda^2 - \sigma > t &\Rightarrow \frac{(1-\sigma/2)}{(1+\sigma/2)}(n-t)\gamma(\lambda^2)/\lambda^2 - \sigma > t \\
&\Leftrightarrow (1+\sigma/2)(t+\sigma)t_p < \gamma(\lambda^2)(1-\sigma/2)(n-t) \\
&\Rightarrow t_p((1+\sigma/2)t+\sigma) < \gamma(\lambda^2)(1-\sigma/2)(n-t).
\end{aligned}$$

Combing the two inequalities we get our desired relation about the running time of $\mathcal{A}'$.

Next, we analyze the success probability of $\mathcal{A}'$. First, notice that the execution in the eyes of $\mathcal{A}$ is indistinguishable in the reduction and in the actual protocol, as honest parties are perfectly simulated. Thus, by our assumption, $\mathcal{A}$ is going to produce t' PoWs different from the ones produced

by the honest parties with probability at least ϵ. This further implies that probability ϵ, $\mathcal{A}'$ is going to solve t' instances from $(x_i)_i$ in a total of $t' \cdot \gamma(t_p)$ steps. Since $\epsilon \in \Omega(1) > \lambda^{-o(1)}$, this is a contradiction to our initial assumption about the hardness of PoW, and thus, in more than 5/6 of the WeakConsensus invocations, the adversary is going to produce at most t' PoWs with overwhelming probability in λ. ⊣

Combining the above two claims we easily get that in more than 2/3 of the WeakConsensus invocations honest parties will produce more PoWs than the adversary with overwhelming probability. This fact will be sufficient to prove our lemma.

First, we argue that Validity holds. For the sake of contradiction, assume that all parties have the same input b and there exists an honest party that outputs $y_i \neq b$. Due to Validity being satisfied by Protocol WeakConsensus when honest parties produce more PoWs than the adversary, it follows that t_i^b must be greater than $2l/3$. Thus, Validity follows.

Regarding Weak Agreement, for the sake of contradiction, assume that there exist honest parties P_i, P_j that output $y_i = 0, y_j = 1$, respectively. Since $y_i = 0$, it should hold that $t_i^0 > 2l/3$. By Weak Agreement of protocol WeakConsensus and the fact that in less than 1/3 of the WeakConsensus invocations $\mathcal{A}$ may produce as many PoWs as the honest parties, it follows that $t_j^1 < 2l/3$. This is a contradiction and the lemma follows. □

We have shown that we can amplify the security of Weak Consensus to be overwhelming in the security parameter. Next, we turn our attention to realizing a primitive known as *Graded Consensus* from Weak Consensus.

3.3 Graded Consensus

We start by providing the relevant security definition (cf. [26]).

Definition 18 (Graded Consensus)**.** A protocol Π implements Graded Consensus iff the following two properties are satisfied:

- **Graded Agreement:** If some honest party output $y_i \in \{0, 1\}$ and $g_i = 1$, then all honest parties output $y_j = y_i$ and $g_j \in \{0, 1\}$;
- **Validity:** If all honest parties have the same input x, they all output $(y_i, g_i) = (x, 1)$.

In our analysis, we will also make use of an additional weaker agreement property, that will hold for our protocol when the security of the PoW collapses.

- **Weak Graded Agreement:** If some honest party output $y_i \in \{0, 1\}$ and $g_i = 1$, then all honest parties output $y_j \in \{y_i, \perp\}$ and $g_j \in \{0, 1\}$.

We are going to follow a similar strategy as above for Graded Consensus, by first designing a protocol which, given that inputs satisfy the Weak Consensus definition and assuming honestly produced PoW witnesses exceed those produced by the adversary, achieves both Graded Agreement and Validity, and then designing a second protocol that amplifies the security of the first one by repeated execution.

Protocol GradedConsensus takes 3 rounds. In the first round, each party sends a message based on its input by solving a PoW instance, similarly to protocol WeakConsensus. Here, unlike the Weak Consensus protocol, the message may be 0,1, or $\perp$. Fortunately, we can extend the idea for encoding messages from that protocol, by splitting the output of the beacon into 3 parts: the first one corresponding to the 0 message, the second one to 1, and the third one to $\perp$. The

protocol then proceeds as protocol WeakConsensus, and establishes Graded Consensus by proper counting of the received PoW witnesses. We also argue that in case the PoW security collapses, the protocol satisfies the Weak Graded Agreement property–essentially the equivalent of the Weak Agreement property for Graded Consensus. This property will be useful to guarantee the security of the amplification protocol.

Protocol $\text{GradedConsensus}_{r_p,r_v}(P_i, z_i)$

Initialization:

- Initialize sets $P_i^0, P_i^1, P_i^\perp, P_i^{\text{late}}$ to $\emptyset$.

Round 1 (round duration $:= r_p$):

- Let $(x_i)_{i\in[m]}$ be the output of $\mathcal{F}_{\text{BEACON}}$ in this round, parsed as a sequence of PoW instances. Let $X_0 = (x_i)_{i\in[m/3]}$, $X_1 = (x_i)_{i\in\{m/3+1,\ldots,2m/3\}}$ and $X_\perp = (x_i)_{i\in\{2m/3+1,\ldots,m\}}$, denote the PoW instances corresponding to messages $0, 1$ and $\perp$, respectively.
- Compute $w_j := \mathsf{PoW.Solve}(x_j)$, where $x_j \leftarrow X_{z_i}$.
- Send (x_j, w_j) to $\mathcal{F}_{\text{DIFF}}$ and proceed to the next round.

Round 2 (round duration $:= r_v$):

- Fetch messages from $\mathcal{F}_{\text{DIFF}}$. For every message of the form (x, w), if $x \in X_b$ (for some $b \in \{0, 1, \perp\}$), and $\mathsf{Pow.Verify}(x, w) = 1$, add (x, w) to P_i^b.
- Send $P_i^0, P_i^1, P_i^\perp$ to $\mathcal{F}_{\text{DIFF}}$ and proceed to the next round.

Round 3 (round duration $:= r_p$):

- Fetch messages from $\mathcal{F}_{\text{DIFF}}$. For every message of the form (x, w), if $x \in X_b$ (for some $b \in \{0, 1\}$), $(x, w) \notin P_i^0 \cup P_i^1 \cup P_i^\perp$, and $\mathsf{Pow.Verify}(x, w) = 1$, add (x, w) to P_i^{late}.
- Let $k_i := |P_i^0| + |P_i^1| + |P_i^\perp| + |P_i^{\text{late}}|$.
- Output $y_i := \begin{cases} 0, \text{ if } |P_i^0| + |P_i^\perp| > k_i/2; \\ 1, \text{ if } |P_i^1| + |P_i^\perp| > k_i/2; \\ \perp, \text{ otherwise.} \end{cases}$ and $g_i := \begin{cases} 1, \text{ if } |P_i^{y_i}| > k_i/2; \\ 0, \text{ otherwise.} \end{cases}$

Lemma 19. *Assume that the inputs of honest parties $(z_i)_i$ satisfy the Weak Consensus properties (cf. Definition 15). Then, protocol $\text{GradedConsensus}_{r_p,r_v}$ achieves Weak Graded Agreement (unconditionally) and Graded Agreement and Validity (Definition 18), provided that the number of distinct PoW witnesses produced by the honest parties exceeds that produced exclusively by $\mathcal{A}$, for r_p, r_v, t_p, t_v as in Lemma 16.*

Proof. As in the case of protocol WeakConsensus, it also holds here that honest parties have sufficient time to process any valid messages they receive. This is sufficient to show that protocol GradedConsensus achieves Weak Graded Agreement unconditionally. For the sake of contradiction, assume that there exists two honest parties P_i, P_j such that P_i outputs (wlog) $(0, 1)$ while P_j outputs $(1, 0)$. It follows that:

$$|P_i^0| > (|P_i^0| + |P_i^1| + |P_i^\perp| + |P_i^{\text{late}}|)/2 \Leftrightarrow |P_i^0| > |P_i^1| + |P_i^\perp| + |P_i^{\text{late}}|.$$

Similarly, we have that $|P_j^1| + |P_j^\perp| > |P_j^0| + |P_j^{\text{late}}|$. Adding both inequalities:

$$|P_j^1| + |P_j^\perp| + |P_i^0| > |P_i^1| + |P_i^\perp| + |P_i^{\text{late}}| + |P_j^0| + |P_j^{\text{late}}|. \tag{3}$$

On the other hand, by the guarantees provided by $\mathcal{F}_{\text{DIFF}}$, it should hold that

$$|P_i^0| \leq |P_j^0| + |P_j^{\text{late}}| \text{ and } |P_j^1| + |P_j^\perp| \leq |P_i^1| + |P_i^\perp| + |P_i^{\text{late}}|,$$

since any valid PoW witness seen by P_i in the first round, will be received by P_j in the second round. Adding the two inequalities, we obviously get a contradiction to Inequality 6. Thus, Weak Graded Agreement holds unconditionally.

Next, we focus on Graded Agreement. Let x denote the number of distinct PoW witnesses computed by the honest parties, and y the number of any other PoW witnesses produced by the adversary. By our assumption, it should hold that $x > y$. For any party P_i, it should hold that $k_i = x + y$, which implies by our assumption that $x > k_i/2$. For the sake of contradiction, assume that there exists two honest parties P_i, P_j such that P_i outputs (wlog) $(0, 1)$ while P_j outputs $y_j \neq 0$. As before, we have that $|P_i^0| > |P_i^1| + |P_i^{\perp}| + |P_i^{\text{late}}|$. By the weak agreement of the inputs of honest parties, and the fact that honest parties solve in total more PoWs that the adversary, it easily follows that there exists an honest party with input 0, and thus no honest party has input 1. Otherwise,

$$|P_i^0| \leq y < x \leq |P_i^1| + |P_i^{\perp}|,$$

which is a contradiction. It thus follows that

$$|P_j^0| + |P_j^{\perp}| \geq x > k_j/2,$$

and y_j must be 0, which is a contradiction to our initial hypothesis.

Finally, we turn our attention to Validity. Let b be the common input of all honest parties. Given that all honest PoWs are received in the round 2 of the protocol and that $x > k_i/2$, for any party P_i, it follows that all parties will output b. Thus, Validity follows. □

Security amplification. We can use a similar strategy as in Weak Consensus in order to "error correct" the output of the Graded Consensus protocol we constructed above. Before running Protocol GradedConsensus, we have to run protocol AmpedWeakConsensus to ensure that the inputs of honest parties to the Graded Consensus protocol satisfy the Weak Consensus properties. Next, we provide a detailed analysis of the security of the protocol.

Protocol $\text{AmpedGradedConsensus}_{r_p,r_v}(P_i, b_i)$

Initialization:
- Initialize counters $t_i^{0,1}, t_i^{0,0}, t_i^{1,1}, t_i^{1,0}, t_i^{\perp,0}, t_i^{\perp,0}, t_i^{\perp,1}$ to 0.

Round 1:
- $z_i := \text{AmpedWeakConsensus}_{r_p,r_v}(P_i, b_i)$.

Rounds 2 to $l + 1$:
- $(c_i, f_i) := \text{GradedConsensus}_{r_p,r_v}(P_i, z_i)$.
- Increment $t_i^{c_i,f_i}$.

Round $l + 2$:
- Output $y_i := \begin{cases} 0 & \text{if } t_i^{0,1} + t_i^{0,0} + t_i^{\perp,1} + t_i^{\perp,0} > 2l/3; \\ 1 & \text{if } t_i^{1,1} + t_i^{1,0} + t_i^{\perp,1} + t_i^{\perp,0} > 2l/3; \\ \perp & \text{otherwise.} \end{cases}$ and $g_i := \begin{cases} 1 & \text{if } t_i^{y_i,1} > 2l/3; \\ 0 & \text{otherwise.} \end{cases}$

Lemma 20. *If Assumptions 1 and 2 hold, protocol* $\text{AmpedGradedConsensus}_{r_p,r_v}$ *achieves Graded Consensus with overwhelming probability in* λ*, for* $l = \log^2(\lambda)$ *and* r_p, r_v, t_p, t_v *as in Lemma 16.*

Proof. First, by Lemma 17, protocol AmpedWeakConsensus achieves Weak Consensus with overwhelming probability. This implies that the precondition about the input of protocol GradedConsensus will be satisfied. Second, in the same way as in Lemma 17, we can show that in

more than 2/3 of the GradedConsensus invocations the honest parties will produce more PoWs than the adversary with overwhelming probability. Hence, by Lemma 19, in more than 2/3 of these invocations, protocol GradedConsensus achieves Graded Consensus. This fact suffices to prove the lemma.

We first argue that Validity holds. For the sake of contradiction, assume that all parties have the same input b and there exists an honest party P_i that outputs $y_i \neq b$. By the previous observation about protocol GradedConsensus, it follows that $t_i^{b,1}$ must be greater than $2l/3$, which is a contradiction. Thus, Validity follows.

Next, we argue why Graded Agreement holds. For the sake of contradiction, assume that there exist honest parties P_i, P_j that output $(y_i = 0, g_i = 1)$, and $y_j = 1$ or $y_j = \perp$. Since $y_i = 0, g_i = 1$, it should hold that $t_i^{0,1} > 2l/3$. By Weak Graded Agreement holding unconditionally, it follows that $t_j^{0,1} + t_j^{0,0} + t_j^{\perp,1} + t_j^{\perp,0} > 2l/3$. On the other hand, since Graded Agreement holds in more than $2l/3$ of the GradedConsensus invocations, it follows that $t_j^{1,1} + t_j^{1,0} < l/3$. Thus, P_j is going to output either $(0, 0)$ or $(0, 1)$. This is a contradiction and the lemma follows. □

Having achieved Graded Consensus with overwhelming probability, we can now focus achieving full-fledged Consensus.

3.4 Consensus

Full-fledged Consensus can be obtained from Graded Consensus using a (oblivious) common coin (cf. [26]), which can be emulated by our beacon. In more detail, the protocol consists of running $\log^2(\lambda)$ times the following sub-protocol: Parties first run protocol AmpedGradedConsensus. If the output has grade 1, then they choose this to be their input for the next iteration; otherwise, they choose the output of the common coin to be their input for the next iteration.

Protocol $\text{Consensus}_{r_p,r_v}(P_i, b_i)$

Rounds 1 to l:
- $(b_i, g_i) := \text{AmpedGradedConsensus}_{r_p,r_v}(P_i, b_i)$.
- Let b'_R be the first bit of the output of $\mathcal{F}_{\text{BEACON}}$ in this round.
- Output $y_i := \begin{cases} b_i & \text{if } (g_i = 1); \\ b'_R & \text{otherwise.} \end{cases}$

Round $l + 1$:
- Output y_i.

The correctness of the protocol above is based on the following observations: (i) If all honest parties have the same input at the beginning of an iteration, then due to Graded Validity they will all have the same output; (ii) if no honest party gets an output with grade 1, then all honest parties are going to agree on the value of the common coin; and (iii) if an honest party gets an output with grade 1 at some iteration, then all parties are going to get the same output (possibly with grade 0), and if the common coin has the same value they will reach Agreement. Since the output of the coin is unpredictable before the Graded Consensus protocol ends, this event happens with probability at least 1/2. Thus, after $\log^2(\lambda)$ rounds all parties will reach Agreement with overwhelming probability in λ.

Theorem 21. *If Assumptions 1 (PoW) and 2 (corruption bound) hold, then protocol* Consensus *achieves Consensus with overwhelming probability in λ, for $l = \log^2(\lambda)$ and r_p, r_v, t_p, t_v as in Lemma 16.*

Proof. We start by analyzing a single iteration of the protocol. First, note that if all honest parties have the same input b, due to Lemma 20 and Graded Validity, they are all going to output $(b, 1)$ with overwhelming probability. Thus, at the end of this iteration they are all going to set $y_i := b$. Further, the protocol preserves Validity across iterations: once parties agree, this state persists. Thus, Validity follows.

Next, we focus on Agreement. First, we show that the probability that all parties agree at the end of an iteration is at least 1/2. We split the analysis into two cases based on the grades that are output by AmpedGradedConsensus: (1) no party outputs $g_i = 1$, and (2) at least one party outputs $g_i = 1$. In case (1), all parties set $y_i = b'_R$, and thus they reach Agreement in this iteration. In case (2), since at least one party has output $b_i \in \{0, 1\}$,$g_i = 1$, by Graded Agreement it follows that no party P_j has output $b_j \neq b_i$ and $g_j = 1$. Thus, all parties with a different b_j than b_i are going to output b'_R. Given that the adversary learns b'_R *after* the AmpedGradedConsensus invocation finishes, its actions are independent from b'_R. Since with probability at least 1/2, $b'_R = b_i$, it follows that with probability at least 1/2 all parties reach agreement in this iteration.

Next, we analyze Agreement in the full protocol. Since in each iteration there is probability at least 1/2 of all parties agreeing, and the events of interest are independent, the probability that parties have not agreed in at least one round is at most $(1 - 1/2)^l = 2^{-\log^2(\lambda)} = \mathsf{negl}(\lambda)$. Moreover, in case they agree in one iteration, as argued earlier, Agreement persists. Thus, Agreement is achieved with overwhelming probability. □

Combining the above theorem with the results of Section 2, we get the following corollary. We remark that our result does not depend on the existence of one-way functions. As long as the adversarial power is less than an inverse polynomial fraction of the total power, we can achieve consensus in the permissionless setting.

Corollary 22. *For $k \geq 2$, suppose $k\mathsf{OV}$ takes $\lambda^{k-o(1)}$ time to decide for all but finitely many inputs lengths for any $d = \omega(\log \lambda)$. Then, assuming that for some $\epsilon > 0$,*

$$\frac{\textit{total honest power}}{\textit{total adversarial power}} \equiv \frac{(n-t)\cdot c}{t \cdot c} > \lambda^{\epsilon}$$

and the existence of a randomness beacon, there exists a protocol that achieves Consensus in the permissionless setting with overwhelming probability in λ.

4 Consensus from NIZK-PoW and a Beacon with $O(\lambda^2)$ Output

We have shown how to achieve consensus in the presence of a beacon whose output length is proportional to the number of parties in Section 3. In this subsection, we show how to use the Seeded NIZK-PoW construction developed in the previous sections, to relax the assumption regarding the size of the output of the beacon. Namely, we show how to achieve Consensus with a beacon that has an output whose size is independent of the number of parties, i.e., it produces $O(\lambda^2)$ bits each time. Note, that such a beacon is strictly weaker than a beacon that produces $O(poly(\lambda))$ outputs each round, as by the time $poly(\lambda)$ bits will be generated by the "short" output beacon some of the generated randomness will be fairly old, essentially allowing the adversary to learn part of the output a lot earlier than honest parties.

Our construction is quite similar to the one extensively presented earlier in Sections 3.2, 3.3, and 3.4, hence here we only describe the necessary modifications to these protocols. Firstly, we are going to interpret beacon outputs as consisting of a NIZK-PoW seed (implying the same number of compressed instances as in the original protocols). Instead of parties selecting a random PoW from

the instance sequence to solve, in the modified protocols they are going to generate and solve the related PoW instance implied by the seed. Finally, instead of PoW witnesses, parties are going to generate NIZK-PoW proofs (and later verify them in the respective steps of the protocols).

The protocol described above is thus sufficient to prove Theorem **??** in Section 3. We only provide a sketch of the proof as it mostly follows that presented in the previous sections.

Theorem 23. *For $k \geq 2$, suppose $k\mathsf{OV}$ takes $\lambda^{k-o(1)}$ time to decide for all but finitely many inputs lengths for any $d = \omega(\log \lambda)$, and that DDH is sub-exponentially hard. Then, assuming that for some $\epsilon > 0$,*

$$\frac{\textit{total honest power}}{\textit{total adversarial power}} \equiv \frac{(n-t)\cdot c}{t \cdot c} > \lambda^{\epsilon}$$

and the existence of a randomness beacon with output size $O(\lambda^2)$, there exists a protocol that achieves Consensus in the permissionless setting with overwhelming probability in λ.

Sketch. The security analysis of the respective protocols is exactly the same, except of the reduction presented in Lemma 17. Note, that we cannot simulate honest PoWs by witnesses coming from the advice string, as PoW instances are all generated from the same seed and are thus correlated. Instead, we have to resort to the Zero-Knowledge property of the NIZK-PoW and simulate honest proofs. This does not change our main argument, as (i) the simulated proofs are indistinguishable from honestly produced ones, and (ii) extractability still holds in the presence of simulated proofs. The running time of the reduction is slightly increased, due to the need to run the NIZK-PoW simulator and extractor. Note, that the running time of these algorithms is extremely efficient compared to the computing NIZK-PoW, thus our result is essentially unaffected. □

5 Conclusions and Directions for Future Work

We have shown how to achieve permissionless consensus in the standard model assuming a computationally bounded adversary, the orthogonal vectors conjecture, and the existence of a randomness beacon. A number of interesting questions are left open by our work.

The main question is whether we can further weaken or entirely eliminate our reliance on the randomness beacon for permissionless consensus. Currently we need at least $\Omega(\log^2(n))$ queries for amplifying security and the beacon samples a uniform distribution in each round. Interesting directions for relaxation include allowing for biased distributions such as for example the "sunspot" model [17], or allowing all common randomness to be determined ahead of time as in the uniform string model. Eliminating the need for randomness may also be a possibility via some form of permissionless coin flipping (currently known to exist in the RO model (cf. [2, 32]).

Outside of consensus numerous other questions remain. Can one construct a PoW scheme with a super-quadratic prover/verifier gap from natural assumptions that don't imply one-way functions? Is there generic way to transform any PoW into a Seeded PoW? Note that our Seeded PoW gives a means of expanding s bits into $\mathsf{poly}(s)$ bits which have some form of pseudoentropy without using Nisan-Wigderson designs. Does this construction (tailored to hard problems with a particular structure) have applications in derandomization?

As remarked above, our robust direct sum theorem can be extended to a wide class of problems. Do problems outside of this class admit robust direct sum theorems? Perhaps more importantly, can we show non-amortizability results for problems conjectured to be hard for sequential algorithms, such as repeated squaring (outside of the generic group model and its cousins)?

Finally, we give a simple zero-knowledge compiler for a particular sumcheck-based doubly efficient proof system that preserves the prover and verifier complexity and admits simulation propor-

tional to verifier complexity. Is it possible to construct such a generic tight zero-knowledge compiler for any doubly-efficient interactive proof system?

References

[1] A. Abboud, R. R. Williams, and H. Yu. More applications of the polynomial method to algorithm design. In P. Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 218–230. SIAM, 2015.

[2] M. Andrychowicz and S. Dziembowski. Distributed cryptography based on the proofs of work. Cryptology ePrint Archive, Report 2014/796, 2014. `http://eprint.iacr.org/`.

[3] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan. Average-case fine-grained hardness. In H. Hatami, P. McKenzie, and V. King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 483–496. ACM, 2017.

[4] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan. Proofs of work from worst-case assumptions. In *CRYPTO (1)*, volume 10991 of *Lecture Notes in Computer Science*, pages 789–819. Springer, 2018.

[5] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan. Proofs of work from worst-case assumptions. In *Advances in Cryptology – CRYPTO 2018*, 2018.

[6] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73, 1993.

[7] M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 276–287. IEEE Computer Society, 1994.

[8] Bitcoinwiki. Genesis block. https://en.bitcoin.it/wiki/Genesis_block.

[9] E. Boix-Adserà, M. S. Brennan, and G. Bresler. The average-case complexity of counting cliques in erdős-rényi hypergraphs. In D. Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1256–1280. IEEE Computer Society, 2019.

[10] J. Cai, A. Pavan, and D. Sivakumar. On the hardness of permanent. In C. Meinel and S. Tison, editors, *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, volume 1563 of *Lecture Notes in Computer Science*, pages 90–99. Springer, 1999.

[11] C. Calabro, R. Impagliazzo, and R. Paturi. The complexity of satisfiability of small depth circuits. In J. Chen and F. V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 75–85. Springer, 2009.

[12] C. Calabro, R. Impagliazzo, and R. Paturi. On the exact complexity of evaluating quantified k-cnf. In V. Raman and S. Saurabh, editors, *Parameterized and Exact Computation - 5th International Symposium, IPEC 2010, Chennai, India, December 13-15, 2010. Proceedings*, volume 6478 of *Lecture Notes in Computer Science*, pages 50–59. Springer, 2010.

[13] R. Canetti, Y. Chen, J. Holmgren, A. Lombardi, G. Rothblum, R. Rothblum, and D. Wichs. Fiat-shamir: from practice to theory. In *STOC*, pages 1082–1090, 2019.

[14] R. Canetti, Y. Chen, J. Holmgren, A. Lombardi, G. N. Rothblum, and R. D. Rothblum. Fiat-shamir from simpler assumptions. Cryptology ePrint Archive, Paper 2018/1004, 2018. `https://eprint.iacr.org/2018/1004`.

[15] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *STOC*, pages 209–218, 1998.

[16] R. Canetti, A. Lombardi, and D. Wichs. Fiat-shamir: From practice to theory, part ii (nizk and correlation intractability from circular-secure fhe). In *STOC*, 2019.

[17] R. Canetti, R. Pass, and A. Shelat. Cryptography from sunspots: How to use an imperfect reference string. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 249–259. IEEE Computer Society, 2007.

[18] T. M. Chan and R. Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In R. Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255. SIAM, 2016.

[19] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Advances in Cryptology – CRYPTO 1992*, pages 89–105, 1992.

[20] A. R. Choudhuri, S. Garg, A. Jain, Z. Jin, and J. Zhang. Correlation intractability and snargs from sub-exponential ddh. In *Advances in Cryptology – CRYPTO 2023*, 2023.

[21] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology – CRYPTO 1994*, pages 174–187, 1994.

[22] I. Damgård and B. Pfitzmann. Sequential iteration of interactive arguments and an efficient zero-knowledge argument for np. In *Automata, Languages and Programming: 25th International Colloquium, ICALP'98 Aalborg, Denmark, July 13–17, 1998 Proceedings 25*, pages 772–783. Springer, 1998.

[23] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *Advances in Cryptology—CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings 21*, pages 566–598. Springer, 2001.

[24] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In E. F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 1992.

[25] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 139–147, London, UK, UK, 1993. Springer-Verlag.

[26] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.

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

[28] C. M. Fiduccia. Polynomial evaluation via the division algorithm: The fast fourier transform revisited. In P. C. Fischer, H. P. Zeiger, J. D. Ullman, and A. L. Rosenberg, editors, *Proceedings of the 4th Annual ACM Symposium on Theory of Computing, May 1-3, 1972, Denver, Colorado, USA*, pages 88–93. ACM, 1972.

[29] M. Fitzi. *Generalized communication and security models in Byzantine agreement.* PhD thesis, ETH Zurich, 2002.

[30] J. Gao, R. Impagliazzo, A. Kolokolova, and R. R. Williams. Completeness for first-order properties on sparse structures with algorithmic applications. In P. N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2162–2181. SIAM, 2017.

[31] J. A. Garay, A. Kiayias, and N. Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 281–310, 2015.

[32] J. A. Garay, A. Kiayias, N. Leonardos, and G. Panagiotakos. Bootstrapping the blockchain — directly. Cryptology ePrint Archive, Report 2016/991, 2016. `http://eprint.iacr.org/2016/991`.

[33] J. A. Garay, A. Kiayias, and G. Panagiotakos. Blockchains from non-idealized hash functions. In R. Pass and K. Pietrzak, editors, *Theory of Cryptography*, pages 291–321, Cham, 2020. Springer International Publishing.

[34] J. A. Garay, A. Kiayias, and G. Panagiotakos. Consensus from signatures of work. In S. Jarecki, editor, *Topics in Cryptology - CT-RSA 2020 - The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24-28, 2020, Proceedings*, volume 12006 of *Lecture Notes in Computer Science*, pages 319–344. Springer, 2020.

[35] O. Goldreich. On counting t-cliques mod 2. *Electron. Colloquium Comput. Complex.*, TR20-104, 2020.

[36] O. Goldreich and G. N. Rothblum. Counting t-cliques: Worst-case to average-case reductions and direct interactive proof systems. In M. Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 77–88. IEEE Computer Society, 2018.

[37] W. Hesse, E. Allender, and D. A. M. Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. In *Journal of Computer and System Sciences*, volume 65, 2014.

[38] E. Horowitz. A fast method for interpolation using preconditioning. *Inf. Process. Lett.*, 1(4):157–163, 1972.

[39] R. Impagliazzo, R. Jaiswal, V. Kabanets, and A. Wigderson. Uniform direct product theorems: simplified, optimized, and derandomized. In C. Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 579–588. ACM, 2008.

[40] R. Impagliazzo and R. Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[41] A. Jain and Z. Jin. Non-interactive zero knowledge from sub-exponential ddh. In *EUROCRYPT 2021*, pages 3–32. Springer, 2021.

[42] J. Katz and C.-Y. Koo. On expected constant-round protocols for byzantine agreement. *Journal of Computer and System Sciences*, 75(2):91 – 112, 2009.

[43] L. Lamport. The weak byzantine generals problem. *J. ACM*, 30(3):668–676, 1983.

[44] L. Lamport, R. E. Shostak, and M. C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[45] A. Lombardi and V. Vaikuntanathan. Correlation-intractable hash functions via shift-hiding. In *ITCS*, 2022.

[46] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 2–10. IEEE Computer Society, 1990.

[47] M. Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104, 2001.

[48] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf, 2008.

[49] R. Pass and E. Shi. Rethinking large-scale consensus. Cryptology ePrint Archive, Paper 2018/302, 2018. `https://eprint.iacr.org/2018/302`.

[50] M. C. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

[51] M. O. Rabin. Randomized byzantine generals. In *FOCS*, pages 403–409. IEEE Computer Society, 1983.

[52] R. M. Roth and G. Ruckenstein. Efficient decoding of reed-solomon codes beyond half the minimum distance. *IEEE Trans. Inf. Theory*, 46(1):246–257, 2000.

[53] R. Shaltiel. Towards proving strong direct product theorems. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 107–117. IEEE Computer Society, 2001.

[54] R. Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005.

[55] R. R. Williams. Strong ETH breaks with merlin and arthur: Short non-interactive proofs of batch evaluation. In R. Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 2:1–2:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[56] V. V. Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the international congress of mathematicians: Rio de janeiro 2018*, pages 3447–3487. World Scientific, 2018.

A Proofs of Work without Random Oracles

We begin with a new definition of a Proof of Work (PoW) that we believe to be closer to the "correct" definition than that of [24, 4]. Nonetheless, our starting point is the definition from [24, 4], albeit with a slightly more stringent hardness condition. Ball *et al.* (following Dwork and Naor [24]) simply required that any prover attempting to solve m random challenges must work approximately m times as hard relative to the work required for a single random challenge.

On the other hand, we will require that solving even $\ell < m$ random challenges requires approximately ℓ times as much work (provided ℓ is larger than some threshold). Informally, this means that allowing an adversary to choose a (large enough) subset of instances to concentrate on doesn't allow the adversary to correctly solve instances faster in amortization than an honest prover.

This definition has a number of parameters:

1. t – the amount of computational work required to solve a single instance.
   t should be a function, and in this work, we consider $t(n) = n^k$ for some constant k.
2. γ – a tightness parameter: if the honest party requires $\ell \cdot t(n)$ work to solve ℓ instances, then no adversary can solve ℓ instances with less than $\ell \cdot \gamma(t(n))$ work.
   In this work we consider $\gamma = t^{1-\Omega(1)}$, i.e., there does not exists a constant $\epsilon > 0$ such that the adversary can solve ℓ instances in time $\ell n^{1-\epsilon}$. We will abuse notation and allow γ to be a class of functions (where the above guarantee should hold for any function in the class).
3. τ – a threshold for when non-amortization holds: solving any ℓ out of m instances, where $m/\ell > \tau$, requires roughly ℓ times as much work.
   In this work, we will consider $\tau = n^{o(1)}$. In particular, it should be the case that $m/\ell > n^{\epsilon}$ for any constant $\epsilon > 0$.
4. δ – a bound on the adversary's success probability.
   In this work, we will consider $1/\delta = n^{o(1)}$. In particular, it should be the case that $\delta(n) > n^{-\epsilon}$ for any constant ϵ.

Definition 24 (Proof of Work)**.** A $(t, \gamma, \tau, \delta)$-PoW consists of two algorithms $(\mathsf{Solve}, \mathsf{Verify})$. These algorithms must satisfy the following properties:

- **Efficiency:**
  - For any $\boldsymbol{c} \in \{0,1\}^n$, $\mathsf{Solve}(\boldsymbol{c})$ runs in time $\tilde{O}(t(n))$.
  - For any $\boldsymbol{c} \in \{0,1\}^n$ and any $\boldsymbol{\pi}$, $\mathsf{Verify}(\boldsymbol{c}, \boldsymbol{\pi})$ runs in time $\tilde{O}(n)$.
- **Completeness:** For any $\boldsymbol{c}$ and any $\boldsymbol{\pi} \leftarrow \mathsf{Solve}(\boldsymbol{c})$,

$$\Pr\left[\mathsf{Verify}(\boldsymbol{c}, \boldsymbol{\pi}) = \mathtt{accept}\right] = 1,$$

  where the probability is taken over Verify's randomness.
- **Hardness:** For any function $\ell(n)$ such that $m(n)/\ell(n) \leq \tau(n)$, and any algorithm Solve^* that runs in time $\ell(n) \cdot \gamma(t(n))$ when given $m(n)$ challenges of size n as input, it holds that:

$$\Pr\left[\begin{array}{r|l} \exists I \subseteq [m], |I| \geq \ell(n) \;\&\; \forall i \in I: & (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m) \leftarrow \mathcal{U}_{n \times m} \\ \mathsf{Verify}(\boldsymbol{c}_i, \boldsymbol{\pi}_i) = \mathtt{accept} & (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{\ell(n)}) \leftarrow \mathsf{Solve}^*_m(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{m(n)}) \end{array}\right] < \delta(n),$$

  where the probability is taken over random samples as well as Solve^*'s and Verify's randomness.

Remark 2. This definition can be generalized to $(t, \gamma, \tau, \delta)$-*Interactive Proofs of Work* (iPoW) in the usual sense: by allowing $\mathsf{Solve}, \mathsf{Verify}$ to be interactive RAM machines. All efficiency, completeness, and hardness properties remain.

We will ultimately prove the following theorems:

Theorem 2. *For $k \geq 2$, suppose $k\mathsf{OV}$ takes $n^{k-o(1)}$ time to decide for all but finitely many inputs lengths for any $d = \omega(\log n)$. Then for any polynomial $m(n)$, there is a $(n^2, n^{o(1)}, t^{1-\Omega(1)}, n^{-o(1)})$-$\mathsf{PoW}$.*

This theorem will be an immediate corollary of the second item of Lemma 9 and Theorem 27.

Theorem 3. *For $k \geq 2$, suppose $k\mathsf{OV}$ takes $n^{k-o(1)}$ time to decide for all but finitely many inputs lengths for any $d = \omega(\log n)$. Then for any polynomial $m(n)$, there is an $(n^k, n^{o(1)}, t^{1-\Omega(1)}, n^{-o(1)})$-$\mathsf{iPoW}$.* This theorem will be an immediate corollary of the second item of Lemma 9 and Theorem 28.

A.1 Proof of Work Schemes, the Orthogonal Vectors Problem and Its Arithmetization

In this section, we will show that the PoW constructions by Ball *et al.* [4] achieve our stronger definition. However, before we can describe their schemes, recall the *kOV* problem which we will rephrase as follows (this version is at least as hard as the version defined previously)[16]:

Definition 25 (k-Orthogonal Vectors)**.** The $k\mathsf{OV}$ problem on vectors of dimension d (denoted $k\mathsf{OV}_d$) is to determine, given k sets $U^1, \ldots, U^k$ of n vectors from $\{0,1\}^{d(n)}$ each, whether there exist $u^1 \in U^1, \ldots, u^k \in U^k$ such that $\sum_{\ell \in [d(n)]} u^1_\ell \cdots u^k_\ell = 0$, where the arithmetic is over $\mathbb{Z}$. If left unspecified, d is taken to be $\lceil \log^2 n \rceil$.

This problem is conjectured to take effectively n^k time, and thus the variants of the OVC can be generalized to non-uniform (resp., constant-error probabilistic) variants of the $k\mathsf{OVC}$: For any $d = \omega(\log n)$, any non-uniform (resp., constant-error probabilistic) algorithm for $k\mathsf{OV}_d$ requires $n^{k-o(1)}$ time. This conjecture is in turn implied by the non-uniform (resp., constant-error probabilistic) variants of the Strong Exponential Time Hypothesis, nuSETH (and BPSETH, resp.) [3].

We are now ready to recall the arithmetized variants of these problems: low-degree extensions of $k\mathsf{OV}$ that can still be computed in $\tilde{O}(n^k)$ time (for any k). For any n, let $p(n)$ be the smallest prime number larger than n^2, and $d(n) = \lceil \log^2 n \rceil$ (for brevity, we shall write just p and d). We define polynomials $f\mathsf{OV}^k_n : \mathbb{F}_p^{knd} \to \mathbb{F}_p$ over knd variables.

We will think of our knd-ary input space as comprised of k matrices, $U^1, \ldots, U^k$, of dimension $n \times d$. We take $U^\ell_{i,j}$ to denote the (i,j)th entry of the ℓth such matrix, and, moreover, we allow $U^\ell_{i,:}$ and $U^\ell_{:,j}$ to denote the ith row and jth column (respectively) of the ℓth matrix. The polynomial $f\mathsf{OV}^k_n$ is then defined as follows:

$$f\mathsf{OV}_n(U^1, \ldots, U^k) = \sum_{i_1, \ldots, i_k \in [n]} \prod_{j \in [d]} \left(1 - \prod_{\ell \in [k]} U^\ell_{i_\ell, j}\right)$$

Define the family of polynomials $\mathcal{F}\mathsf{OV}^k = \{f\mathsf{OV}_n\}$. It is not difficult to observe that $\mathcal{F}\mathsf{OV}^k$ agrees with $k\mathsf{OV}$ on binary valued inputs. Thus, evaluating $\mathcal{F}\mathsf{OV}^k$ it is just as hard as $k\mathsf{OV}$ in the worst-case. However, as it was shown in [3], uniformly random inputs (from $\mathbb{F}_p^{knd}$) are just as hard:

Theorem 26 ([3])**.** *For any k, if $\mathcal{F}\mathsf{OV}^k$ can be computed in $O(n^{1+\alpha})$ time on average for some $\alpha > 0$, then $k\mathsf{OV}$ can be decided in $\tilde{O}(n^{1+\alpha})$ time in the worst case. Thus, assuming $k\mathsf{OVC}$, $\mathcal{F}\mathsf{OV}^k$ requires $\Theta(n^{k-o(1)})$ time.*

[16] To reduce a single list of n vectors to k lists of n vectors, simply set each of the k lists to be the same as the original.

Moreover, in a followup work [4] it was shown that the problem admits an even more stringent hardness property: it is no easier to solve *batches* of instances (in amortization) than single instances. In other words, the $\mathcal{F}\mathsf{OV}^k$ problem admits an *(average-case) direct sum theorem*: given m instances, solving them all requires m times as much computation as a single instance. Or in other words, the naive approach of solving each instance one-by-one is effectively optimal.

This prior result is an all-or-nothing direct sum theorem: the guarantee only holds with respect to adversaries attempting to solve all instances given. In this work, we strengthen this to show that the naive approach of solving each instance one-by-one is effectively optimal for $\mathcal{F}\mathsf{OV}^k$, even if you only wish to solve a fraction of the instances. In particular, given m instances, solving solving any (large enough, i.e. $\ell = m^{1-o(1)}$) ℓ instances requires ℓ times as much computation as a single instance.

We remark that our reduction *only utilizes two aspects* of $\mathcal{F}\mathsf{OV}^k$, and thus our robust direct sum theorem will generalize to other problems with these properties: (1) it is an efficiently computable low degree polynomial, (2) it is tightly downward self-reducible. For the sake of clarity however, we constrain our discussion in this section to $\mathcal{F}\mathsf{OV}^k$.

A.2 A Proof of Work

We are now almost ready to describe the proof of work protocol from [3]. First, we introduce yet another polynomial from [3]. Suppose, for convenience, that k is even. Then for any fixed $U^{k/2+1}, \ldots, U^k$, we can define

$$f\mathsf{OV}^k_{U^{k/2+1},\ldots,U^k}(x_1,\ldots,x_d) \stackrel{\text{def}}{=} \sum_{i_{k/2+1},\ldots,i_k \in [n]} \prod_{\ell \in [d]} (1 - x_\ell \prod_{j \in [k/2]} U^{k/2+j}_{i_\ell,\ell}).$$

Note that we can write $f\mathsf{OV}^k$ as follows:

$$f\mathsf{OV}^k(U^1,\ldots,U^k) = \sum_{i_1,\ldots,i_{k/2} \in [n]} f\mathsf{OV}^k_{U^{k/2+1},\ldots,U^k}(\prod_{j \in [k/2]} U^j_{i_j,1}, \ldots, \prod_{j \in [k/2]} U^j_{i_j,d}).$$

Next, fix some enumeration of $[n]^{k/2}$, $\psi : [n^{k/2}] \to [n]^{k/2}$ (where if $\psi : i \mapsto (i_1,\ldots,i_{k/2})$ then $\psi_j : i \mapsto i_j$). Let $\phi_1,\ldots,\phi_d : \mathbb{F}_p \to \mathbb{F}_p$ denote the unique degree $n^{k/2}-1$ degree univariate polynomials such that for $i \in [n^{k/2}]$, $\phi_\ell(i) = \prod_{j \in [k/2]} U^j_{\psi_j(i),\ell}$. Now we define:

$$R_{U^1,\ldots,U^k}(x) \stackrel{\text{def}}{=} f\mathsf{OV}^k_{U^{k/2+1},\ldots,U^k}(\phi_1(x),\ldots,\phi_d(x)).$$

Note that $R_{U^1,\ldots,U^k}$ is a univariate polynomial of degree at most $d(n^{k/2}-1)$. Additionally, it holds that

$$f\mathsf{OV}^k_n(U^1,\ldots,U^k) = \sum_{i \in [n^{k/2}]} R_{U^1,\ldots,U^k}(i).$$

So given the coefficients of $R_{U^1,\ldots,U^k}$ one can evaluate it on $[n^{k/2}]$ in time $\tilde{O}(n^{k/2}d)$ (via [28]), then one can compute $f\mathsf{OV}^k_n(U^1,\ldots,U^k)$ in time $n^{k/2}$. Thus, $R_{U^1,\ldots,U^k}$ comprises the (honest) proof of [3]'s scheme. (Producing such a proof in time $n^{k-o(1)}$ would thus yield a fast method of evaluating $f\mathsf{OV}^k$ and violate the $k\mathsf{OV}$ conjecture.)

We now recall how to check that such a tentative proof, R^*, is genuine, i.e., $R^* \equiv R_{U^1,\ldots,U^k}$. By the Schwartz-Zippel lemma, if $R^* \not\equiv R_{U^1,\ldots,U^k}$, then with high probability $R^*(\alpha) \neq R_{U^1,\ldots,U^k}(\alpha)$ for a random $\alpha \in \mathbb{F}_p$. So it suffices to evaluate these polynomials efficiently on a random point.

Note that $R^*(\alpha)$ can be evaluated in time $\tilde{O}(n^{k/2}d)$. Next, to evaluate $R_{U^1,\ldots,U^k}(\alpha)$ we will use its structure. First note that the coefficients of $\phi_1, \ldots, \phi_d$ can be recovered in time $\tilde{O}(dn^{k/2})$ [38]. So, for any $\alpha \in \mathbb{F}_p$, $\phi_1(\alpha), \ldots, \phi_\ell(\alpha)$ can be computed in time $\tilde{O}(dn^{k/2})$. Then, $f\mathsf{OV}^k_{U^{k/2+1},\ldots,U^k}$ can be evaluated on these values in time $\tilde{O}(n^{k/2}d)$. Thus, we can check $R^*(\alpha) = R_{U^1,\ldots,U^k}(\alpha)$ in time $\tilde{O}(n^{k/2}d)$ for any $\alpha \in \mathbb{F}_p$.

Protocol A Proof of Work via $\mathcal{F}\mathsf{OV}^k$ [3]

- $\mathsf{Solve}((U^1, \ldots, U^k))$ compute and output the coefficients of $R_{U^1,\ldots,U^k}$.
- $\mathsf{Verify}((U^1, \ldots, U^k), R^*)$ check that $R^*(x) = R_{U^1,\ldots,U^k}(x)$ for a random $x \in \mathbb{F}_p$.

Theorem 27 ([3])**.** *The protocol above is a single-round doubly efficient interactive proof for proving that $y = \mathcal{F}\mathsf{OV}^k(x)$. This protocol has perfect completeness and soundness error at most $\left(\frac{2nd}{p}\right)$. The prover runs in time $\tilde{O}(n^2 d \log p)$, and the verifier in time $\tilde{O}(nd \log p)$.*

Ball *et al.* later showed how to use the sumcheck protocol to construct *interactive* PoWs where producing a proof requires order n^k work, but the verifier still runs in time n.

Theorem 28 ([4])**.** *There is a k-round doubly efficient interactive proof for proving that $y = \mathcal{F}\mathsf{OV}^k(x)$. This protocol has perfect completeness and soundness error at most $\left(\frac{2nd}{p}\right)$. The prover runs in time $\tilde{O}(n^k d \log p)$, and the verifier in time $\tilde{O}(nd \log p)$.*

A.3 *Seeded* Proofs of Work from Fine-Grained Complexity

We begin by introducing the new notion of a *Seeded Proof of Work*, a PoW scheme such that a short public seed can be expanded into many challenges that collectively remain hard in the sense defined in the prior section, despite the fact that challenges are correlated with one another (in a known manner).

Definition 29 (Seeded Proof of Work)**.** A (s, m)-*Seeded* $(t, \gamma, \tau, \delta)$-*PoW* consists of four algorithms $(\mathsf{Expand}, \mathsf{Solve}, \mathsf{Verify})$. These algorithms must satisfy the efficiency and completeness properties of a Proof of Work (Definition 24), and additionally:

- **Efficiency:** For any $\boldsymbol{\sigma} \in \{0,1\}^{s(n)}$ and $i \in [m(n)]$, $\mathsf{Expand}(\boldsymbol{\sigma}, i)$ runs in deterministic time $\tilde{O}(s(n))$.
- **Hardness:** For any function $\ell(n)$ such that $m(n)/\ell(n) \leq \tau(n)$ any algorithm Solve^* that runs in time $\ell(n) \cdot \gamma(t(n))$ when given $m(n)$ challenges of size n as input,

$$\Pr\left[\begin{array}{l|l} \exists I \subseteq [m], |I| \geq \ell(n) \;\&\; \forall i \in I : & \boldsymbol{\sigma} \leftarrow \mathcal{U}_s \\ \qquad \mathsf{Verify}(\boldsymbol{c}_i, \boldsymbol{\pi}_i) = \mathtt{accept} & (\boldsymbol{c}_i \leftarrow \mathsf{Expand}(\boldsymbol{\sigma}, i))_{i \in [m(n)]} \\ & (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{\ell(n)}) \leftarrow \mathsf{Solve}^*_\ell(\boldsymbol{\sigma}) \end{array}\right] < \delta(n),$$

  where the probability is taken over $\mathcal{U}_s$, Solve^*'s, and Verify's randomness.

Ball *et al.* [4] proved their PoW hardness condition by in turn proving a *direct sum theorem* for $f\mathsf{OV}$: if solving 1 random instance of size n is $n^{2-o(1)}$-hard, then solving m random, independent instances is m times as hard. Note that to specify the distribution of m random instances requires mn random bits. In this language, we can think of our analysis as corresponding to proving a

derandomized direct sum theorem; we want to show that there is a "pseudorandom" distribution over m instances that still requires m times as much work as a single random instance, and yet specifying these m instances can be done with far fewer than mn random bits. Note that the difficulty here is that the distribution is clearly distinguishable from random as the adversary is given a concise description of it.

Ultimately, we will utilize the downward self-reducible structure of $f\mathsf{OV}$. This is also what [4] used to prove their direct sum theorem. Our analysis can be viewed as further exploiting the structure of $f\mathsf{OV}$ to remove some slack from the reduction to independent uniform instances.[17] Any small polynomial improvement on our derandomization will in fact yield a fine-grained hard problem with a *bigger* gap between hardness and verification time than what is currently known from just worst-case assumptions (and without interaction).

Remark 3. More generally, derandomized direct sum theorems present an intriguing possibility for escalating or magnifying computational intractability that, to our knowledge, has not been extensively explored. General derandomized direct *product* theorems are known [39]. Note that while general direct product theorems hold for arbitrary problems, direct sum theorems to do not hold in general [53].

The basic idea in our construction stems from a simple divide and conquer approach. Consider, for the moment, the problem of *counting* the number of k-orthogonal vectors in $U^1, \ldots, U^k \in \{0,1\}^{n \times d}$, i.e., $|\{u^1 \in U^1, \ldots, u^k \in U^k : \sum_{\ell \in [d]} u^1_\ell \cdots u^k_\ell = 0\}|$. Note that if we partition each list of n vectors, U^i, into r lists of size n/r, to compute the number of k-orthogonal vectors in the big lists, $U^1, \ldots, U^k$, it suffices to count all k-orthogonal vectors in each subproblem formed by taking exactly one partial list from each big list. Any sequence, $u^1, \ldots, u^k$, of k-orthogonal vectors occurs in exactly one subproblem. The case of the arithmetized version works similarly. For convenience, we will assume that $m = r^k$ for some $r \in \mathbb{Z}$ and that r divides N.

- $\mathsf{Split}^m : \mathbb{F}_p^{k \times N \times d} \to \left(\mathbb{F}_p^{k \times N/\sqrt[k]{m} \times d}\right)^m$. On input $U^1, \ldots, U^k$, partition the rows of each array U^i into $\sqrt[k]{m}$ subarrays, $U^{i,1}, \ldots, U^{i,\sqrt[k]{m}}$, of height $N/\sqrt[k]{m}$ such that

$$U^i = \begin{bmatrix} U^{i,1} \\ U^{i,2} \\ \vdots \\ U^{i,\sqrt[k]{m}} \end{bmatrix}$$

  Then output all combinations of exactly one subarray, $U^{i,j}$, from each array, U^i:

$$(U^{1,j_1}, \ldots, U^{k,j_k})_{j \in [\sqrt[k]{m}]^k}.$$

- $\mathsf{Merge}^m : \mathbb{F}_p^m \to \mathbb{F}_p$. On input $y^1, \ldots, y^m$, simply output the sum:

$$\sum_{i=1}^{m} y^i.$$

Notice that because each term of $f\mathsf{OV}^k(U^1, \ldots, U^k)$, $\prod_{j \in [d]} \left(1 - \prod_{\ell \in [k]} U^\ell_{i_\ell, j}\right)$ occurs in exactly one polynomial of the form $f\mathsf{OV}^k(U^{1,j_1}, \ldots, U^{k,j_k})$ (where $U^{i,j}$ correspond to the subarrays defined

[17] One can view [4] as reducing an N^2-hard problem on inputs of length N to an N^2-hard problem on inputs of length ℓN (solving ℓ^2 random instances of size N/ℓ), for some ℓ. We, on the other hand, reduce to a problem that does not incur this ℓ-factor loss (we reduce to a pseudorandom distribution over ℓ^2 instances of size N/ℓ; this distribution has support size 2^N as opposed to $2^{\ell N}$).

above), we have that

$$f\mathsf{OV}^k(U^1,\ldots,U^k) = \sum_{j\in[\sqrt[k]{m}]^k} f\mathsf{OV}^k(U^{1,j_1},\ldots,U^{k,j_k})$$
$$= \mathsf{Merge}^m(f\mathsf{OV}^k(\mathsf{Split}_1^m(U^1,\ldots,U^k)),\ldots,f\mathsf{OV}^k(\mathsf{Split}_m^m(U^1,\ldots,U^k))).$$

Protocol A Seeded Proof of Work via $\mathcal{F}\mathsf{OV}^k$

Let m be the number of PoW instances desired and n a security parameter such that proofs (conditionally) require $\tilde{O}(n^2)$ time to compute and $\tilde{O}(n)$ time to verify. Let p be a prime number of order $n^{\log n}$ and $d = \lceil \log^2 n \rceil$.

- Seed σ is sampled uniformly at random from $\mathbb{F}_p^{k\times \sqrt[k]{n^2 m}\times d}$.
- $\mathsf{Expand}(\sigma, i)$ outputs $\mathsf{Split}_i^m(S)$, the ith (in $\mathbb{F}_p^{k\times n^{2/k}\times d}$) output of $\mathsf{Split}^m(S)$.
- Solve : On input $U^1,\ldots,U^k \in \mathbb{F}_p^{k\times n^{2/k}\times d}$, output the coefficients of $R_{U^1,\ldots,U^k}$
- Verify : On input $U^1,\ldots,U^k \in \mathbb{F}_p^{k\times n^{2/k}\times d}$ and R^*, a degree $k(n-1)d$ univariate polynomial over $\mathbb{F}_p$, check that $R^*(x) = R_{U^1,\ldots,U^k}(x)$ for a random $x \in \mathbb{F}_p$.

Theorem 5: *For $k \geq 2$, suppose $k\mathsf{OV}$ takes $n^{k-o(1)}$ time to decide for all but finitely many input lengths for any $d = \omega(\log n)$. Then for any polynomial $m(n)$, the protocol above is a $(\sqrt[k]{m(n)}n, m(n)n)$-Seeded $(n^2, n^{o(1)}, t^{1-\Omega(1)}, n^{-o(1)})$-PoW.*

This theorem is an immediate corollary of the first item of Lemma 9 and Theorem 27.

Theorem 6: *For $k \geq 2$, suppose $k\mathsf{OV}$ takes $n^{k-o(1)}$ time to decide for all but finitely many inputs lengths for any $d = \omega(\log n)$. Then for any polynomial $m(n)$, there is a $(\sqrt[k]{m(n)}n, m(n)n)$-Seeded $(n^k, n^{o(1)}, t^{1-\Omega(1)}, n^{-o(1)})$-Interactive PoW (iPoW).* This theorem is an immediate corollary of the first item of Lemma 9 and Theorem 28.

A.4 A (Derandomized) Robust Direct Sum Theorem for Arithmetized k-OV

We now present our main technical contribution for the section: the arithmetized variant of k-OV, $f\mathsf{OV}^k$, described above admits a *robust direct sum theorem*. In particular, given m random instances, it requires roughly ℓn^k work to correctly evaluate **any** ℓ of the m instances. Moreover, we show that this result can also be *derandomized*—i.e., the robust direct sum guarantee applies even when the instances are drawn from a pseudorandom joint distribution that requires polynomially fewer random bits to sample. Because the sampler for this distribution is efficiently invertible, the guarantee holds even when the seed is given in conjunction with the pseudorandom distribution.

A hard pseudorandom distribution. Before we state Lemma 9 that describes our reduction, we describe the hard pseudorandom distribution over instances. This distribution is simply the output of Split on a uniformly random instance. Concretely, for the case of $\mathcal{F}\mathsf{OV}^k$, this is as follows:

Let $\mathcal{D}_{\mathrm{pr}}^{r^k,n,d,p}$ denote the distribution generated by sampling $\boldsymbol{U}$ uniformly at random from $(\mathbb{F}_p^{rn\times d})^k$, k lists of rn d-dimensional vectors with each list partitioned into r blocks, and outputting $\mathsf{Split}^{r^k}(\boldsymbol{U})$.

$$\left(U^{1,j_1},\ldots,U^{k,j_k}\right)_{j\in[r]^k} \qquad \text{where} \qquad \begin{bmatrix} U^{1,1} \\ \vdots \\ U^{1,r} \end{bmatrix},\cdots,\begin{bmatrix} U^{1,1} \\ \vdots \\ U^{1,r} \end{bmatrix} \xleftarrow{u} (\mathbb{F}_p^{rn\times d})^k \tag{4}$$

Direct sum reduction. We are now prepared to state this sections main technical lemma.

Lemma 9. *Let $m(n)$ be a polynomial and $\ell(n)$ a function such that $\ell(n) < m(n)$, and p a prime such that $\log(p) = n^{o(1)}$ and $p > 6\left(\frac{12m(n)}{\delta\ell(n)}\right)^2 k2^k d\log\log(n)\log(n)m(n)$. Let $dk = \tilde{O}(1)$, and let $\{f : \mathbb{F}_p^{knd} \to \mathbb{F}_p\}$ denote $\mathcal{F}\mathsf{OV}^k$ in what follows.*

1. **Pseudorandom reduction.** *Let A be a randomized algorithm running in time $t_A(n) \leq \ell(n) \cdot n^{k-\epsilon}$ that on input $X_1, \ldots, X_m$ drawn from $\mathcal{D}_{pr}$ outputs $\hat{y}_1, \ldots, \hat{y}_m$ such that with probability at least $\delta(n)$, $|\{i \in [m] : f(X_i) = \hat{y}_i\}| \geq \ell(n)$.*[18]

   Then, there is a randomized algorithm A' that on input X, of size $\theta_{d,p}(n)$, computes $f(X)$ in time $\tilde{O}\left(\frac{mn^{k-\epsilon}}{(\delta\frac{\ell}{m})^{\Theta(1)}}\right)$, with probability at least $2/3$.

2. **Uniformly random reduction.** *Similarly, let A be a randomized algorithm running in time $t_A(n) \leq \ell(n) \cdot n^{k-\epsilon}$ that on input $X_1, \ldots, X_m$ drawn independently and uniformly at random from $\mathbb{F}_p^{knd}$ outputs $\hat{y}_1, \ldots, \hat{y}_m$ such that with probability at least $\delta(n)$, $|\{i \in [m] : f(X_i) = \hat{y}_i\}| \geq \ell(n)$.*

   Then, there is a randomized algorithm A' that on input X, of size $\theta_{d,p}(n)$, computes $f(X)$ in time $\tilde{O}\left(\frac{mn^{k-\epsilon}}{(\delta\frac{\ell}{m})^{\Theta(1)}}\right)$, with probability at least $2/3$.

Remark 4. In the above lemma (and its proof), for simplicity we assume $\ell(n)$ is known to the reduction precisely (similarly $\delta(n)$). Because the reduction requires just $\log(n)$ such values (each of length $O(\log(n))$), this can be efficiently encoded as advice.

However, if f can be evaluated on a single instance in time n^k, and $\frac{m}{\ell} \leq \tau(n)$ for some $\ell n^{k-\epsilon}$ time computable $\tau = n^{o(1)}$, then, alternatively, $\ell(n)$ can be efficiently approximated up to a constant factor in time $\tilde{O}(\ell n^{k-\epsilon})$ given $\delta(n)$ (such that $1/\delta(n) = n^{o(1)}$). This follows by repeated empirically testing the output of A at $n^{o(1)}$ random points. By a Chernoff bound, it follows that with overwhelming probability each test will yield a candidate value $\hat{\ell}$ that is precise up to a constant factor (if the algorithm solved at least τ instances correctly). Then simply output the smallest estimate that is greater than a δ fraction of all estimates. By another Chernoff bound, this estimate will be a constant factor approximation of $\ell(n)$.

The proof of Lemma 9 can be adapted to work with constant factor approximations of ℓ by simply adjusting constants.

Remark 5. The lemma can be extended to apply to any low-degree polynomial that is downward-self-reducible (in the sense that it admits some efficient, correct $\mathsf{Split}, \mathsf{Merge}$ pair).

Technical tools. To complete the proof of Lemma 9 we first need to introduce a number of technical tools. First, we require an efficient list-decoding algorithm, related to Sudan's, from Roth and Ruckenstein [52].

Lemma 30 ([52])**.** *List decoding for $[n, k]$ Reed-Solomon (RS) codes over $\mathbb{F}_p$ given a code word with almost $n - \sqrt{2kn}$ errors (for $k > 5$), can be performed in*

$$O\left(n^{3/2}k^{-1/2}\log^2 n + (n-k)^2\sqrt{n/k} + (\sqrt{nk} + \log p)n\log^2(n/k)\right)$$

operations over $\mathbb{F}_p$.

By restricting the choice of parameters, the following is obtained:

[18] If it is not the case that $t(n) >> m(n)$, imagine that A outputs $(i, \hat{y}_i)_{i \in S}$ for some $S \subseteq [n]$.

Corollary 31. *For $T = \Omega(D), \Omega(\log^{2/3}(p))$, list decoding $[T, D]$ Reed-Solomon codes of $\mathbb{F}_p$ can be performed in $\tilde{O}(T^{2.5})$ operations.*

We also need a bound on list size.

Lemma 32 ([10])**.** *Let q be a polynomial over $\mathbb{F}_p$, and define* $\mathrm{Graph}(q) := \{(i, q(i)) \mid i \in [p]\}$*. Let $c > 2$, $\delta/2 \in (0, 1)$, and $m \leq p$ such that $m > \frac{c^2(d-1)}{\delta^2(c-2)}$ for some d. Finally, let $I \subseteq [p]$ such that $|I| = m$. Then, for any set $S = \{(i, y_i) \mid i \in I\}$, there are less than $\lceil c/\delta \rceil$ polynomials q of degree at most d that satisfy $|\mathrm{Graph}(q) \cap S| \geq m\delta/2$.*

Corollary 33. *Let S be as in Lemma 32 with $I = \{K + 1, \ldots, K + T\}$, for any K, T such that $K + T < p$. Then for $T > 9D/\delta^2$, there are at most $3/\delta$ polynomials, q, of degree at most D such that $|\mathrm{Graph}(q) \cap S| \geq T\delta/2$.*

We additionally require high-moment bounds on random variables with bounded independence.

Lemma 34 (Concentration for sum of q-wise independent random variables [7])**.** *Let q be an even integer and X the sum of n q-wise independent random variables taking values in $[0, 1]$. Let $\mu = \mathbb{E}[X]$ and $a > 0$. Then*

$$\Pr[|X - \mu| > a] < 1.1 \left(\frac{nq}{a^2}\right)^{q/2}.$$

Proof of Lemma 9. Let $r = m^{1/k}$ assume this value is integral for simplicity. Similarly, assume n is a power of 2 for simplicity. Let $X \in \mathbb{F}_p^{k \times r \cdot n \times d}$, $q = \log\log n$, $T = \frac{(m \log n)^{2/q} q + 9k^2 d^2 q^2 2^{2k}}{(\frac{\delta \ell}{m})^2}$, and $D = kdq2^k$.

We begin by presenting the reduction that computes $f(X)$ for any X using A. Let $X_1, \ldots, X_m = \mathsf{Split}^m(X)$, where each $X_i \in \mathbb{F}_p^{k \times r \cdot n \times d}$. Note that $(X_1, \ldots, X_m) \in \mathrm{Supp}(\mathcal{D}_{\mathrm{pr}})$. Now invoke the following procedure, B, that takes $X_1, \ldots, X_m$ as input:

1. If $X_i \in \mathbb{F}_p^{k \times n' \times d}$ for $n' \leq N$ for a fixed constant N, simply evaluate $f(X_1), \ldots, f(X_m)$.
2. For each $i \in [m]$, parse X_i as follows:

$$X_i = (\begin{bmatrix} X_i^{1,0} \\ \hline X_i^{1,1} \end{bmatrix}, \cdots, \begin{bmatrix} X_i^{k,0} \\ \hline X_i^{k,1} \end{bmatrix}) \in (\mathbb{F}_p^{n \times d})^k$$

3. Sample $R_0, \ldots, R_q$ independently and uniformly at random from $\mathbb{F}_p^{k \times r \cdot n/2 \times d}$. Compute $(R_{j,1}, \ldots, R_{j,m}) := \mathsf{Split}^m(R_i)$ for $j \in \{0, \ldots, q\}$.
4. For each $i \in [m]$, define the univariate polynomial

$$P_i(t) = \sum_{v \in \{0,1\}^k} \delta_v(t) \cdot (X_i^{1,v_1}, \ldots, X_i^{k,v_k}) + \left(\prod_{u=0}^{2^k - 1} (t - u)\right) \cdot \left(\sum_{j=0}^{q} t^j \cdot R_{j,i}\right),$$

   where $\delta_v(t)$ denotes the unique degree $2^k - 1$ such that for $t \in \{0, \ldots, 2^k - 1\}$, if t is represented as a binary vector in $\{0, 1\}^k$: $\delta_v(t) = 1$ if $t = v$ and $\delta_v(t) = 0$ otherwise.

   Note that $f(P_i(t))$ is a univariate polynomial of degree at most $D = kdq2^k$.
5. For $t = 2^k, \ldots, T + 2^k$:
   - Choose k random permutations $\pi_{t,j} : [r] \xrightarrow{\mathcal{S}} [r]$.
   - Interpreting each $j \in [m]$ as a vector in $[r]^k$, consistent with the ordering of Split, define $\pi : [m] \xrightarrow{\mathcal{S}} [m]$ such that $\pi_t : (j_1, \ldots, j_k) \mapsto (\pi_{t_1}(j_1), \ldots, \pi_{t,k}(j_k))$.

- Call $A(P_{\pi_t^{-1}(1)}(t), \ldots, P_{\pi_t^{-1}(m)}(t)) = (\hat{y}^t_{\pi_t^{-1}(1)}, \ldots, \hat{y}^t_{\pi_t^{-1}(m)})$.

6. For $i \in [m]$, run the efficient local list decoding for Reed-Solomon Codes from Lemma 30 on $(y_i^{2^k}, \ldots, y_i^{T+2^k})$ to get a list L_i of candidate univariate polynomials of degree at most D.
7. Choose uniformly random and independent $t^* \in \mathbb{F}_p$.
8. Recursively call B on $(P_1(t^*), \ldots, P_m(t^*))$ to get $(\hat{y}'_1, \ldots, \hat{y}'_m)$.
9. For each $i \in [m]$, let $\hat{h}_i$ be the first polynomial in L_i such that $\hat{h}_i(P_i(t^*)) = \hat{y}_i$.
10. For $i \in [m]$, set $\hat{y}_i = \sum_{j=0}^{2^k-1} \hat{h}_i(j)$.
11. Output $(\hat{y}_1, \ldots, \hat{y}_m)$.

Given $B(X_1, \ldots, X_m) = (\hat{y}_1, \ldots, \hat{y}_m)$, return $\mathsf{Merge}(\hat{y}_1, \ldots, \hat{y}_m) = \sum_{i=1}^m \hat{y}_i$.

Correctness. Now let us analyze the correctness of the reduction. First, we will observe that the reduction maintains the invariant that the input to each recursive call to B is in the support of $\mathcal{D}_{\mathrm{pr}}$ (i.e., the output of $\mathsf{Split}(X')$ for some X').

Claim 4. *$(P_1(t), \ldots, P_m(t)) \in \mathrm{Supp}(\mathcal{D}_{pr})$ for all $t \in \mathbb{F}_p$.*

The claim follows from inspection.

Next, we observe that the calls made to A in each recursive step are q-wise independent draws from $\mathcal{D}_{\mathrm{pr}}$, enabling us to extract strong empirical guarantees on A's responses.

Claim 5. *$(P_{\pi_t^{-1}(1)}(t), \ldots, P_{\pi_t^{-1}(m)}(t))_{t=2^k}^{T+2^k}$ is q-wise independent with residual distribution $\mathcal{D}_{pr}$.*

Again, assuming the input to B is in the support of $\mathcal{D}_{\mathrm{pr}}$, this is clear from the definition of P_i's and π_t's. The claim then follows by induction.

We are now prepared to demonstrate that the list-decoding subroutines almost always recover "good" lists.

Claim 6. *With probability at most $1/6$, there exist a level of recursion and index i such that the L_i list recovered in step 6 of B (in that recursive call) either:*

1. *It does not contain the polynomial $Q_i^*(t) := f(P_i(t))$, or*
2. *it consists of more than $6/\delta'$ polynomials, for $\delta' = \frac{\delta \ell}{2m}$.*

Proof. Consider any fixed $i \in [m]$ and level of recursion. Because each recursive call cuts the number of rows in each array in half, there are most $\log n$ levels of recursion. Thus, by the promise on the list-decoding algorithm (Lemma 30 and Corollary 33), it suffices to show that with probability $\leq \frac{1}{6m \log n}$, less than a δ'-fraction of $(\hat{y}_i^{2^k}, \ldots, y_i^{T+2^k})$ returned by A (in larger batch calls) agree with $f(P_i(2^k)), \ldots, f(P_i(T+2^k))$. Then, the claim follows by a union bound over $i \in [m]$ and all $\log n$ levels of recursion. This is because $T > \frac{9k^2d^2q^2 2^{2k}}{(\frac{\delta \ell}{m})^2}$, so $\delta' T > \sqrt{2TD}$ (and thus it satisfies the error threshold for efficient list decoding).

To analyze this, let E_t be the indicator random variable for the event that $\hat{y}_i^t = f(P_i(t))$. We can lower-bound the probability that $E_t = 1$ by the probability that A correctly solves ℓ out of m instances in the tth batch, which happens with probability at least δ, *and* the probability that π_t mapped $P_i(t)$ to one of the of the ℓ instances solved correctly. Because $(P_1(t), \ldots, P_m(t))$, for fixed t, is simply a random sample of $\mathcal{D}_{\mathrm{pr}}$, these two events are independent. Thus,

$$p := \mathbb{E}[E_t] \geq \delta \cdot \frac{\ell}{m}.$$

Now, by the claim above we have that $E_{2^k}, \ldots, E_{T+2^k}$ are q-wise independent. Applying the tail bound of Lemma 34, and the choice of $T \geq (m \log n)^{2/q} \frac{q}{(\frac{\delta\ell}{m})^2}$, we have:

$$\begin{aligned}\Pr[\sum E_t \leq \delta' \cdot T] \leq \Pr\left[\left|\sum E_t - \frac{\delta\ell}{m}T\right| \geq \left(\frac{\delta\ell}{m} - \delta'\right)T\right] &\leq 1.1\left(\frac{Tq}{\left(\frac{\delta\ell}{m} - \delta'\right)^2 T^2}\right)^{q/2} \\ &\leq \frac{q^{q/2}}{\left(\frac{\delta\ell}{m} - \delta'\right)^q T^{q/2}} \\ &\leq \frac{1}{6m \log n}.\end{aligned}$$

□

Next, to complete the proof of correctness, we simply need to demonstrate that, with sufficiently high probability, recursing on a random point works.

Claim 7. *Given $L_1, \ldots, L_m$, each a list of $L = \frac{12m}{\delta\ell}$ polynomials of degree at most D, for a uniformly random choice of $t^* \xleftarrow{u} \mathbb{F}_p$, the probability that there exists some L_i such that there are $p \neq q \in L_i$ where $p(t^*) = q(t^*)$ is at most $\frac{mL^2D}{p}$.*

Proof. For any L_i there are at most $\binom{L}{2} \leq L^2$ pairs of polynomials $p \neq q \in L_i$. Because $p \neq q$ and the degree is at most D, they can agree on at most D points. The claim follows from a union bound over all such pairs in all such lists. □

The probability that the event described in Claim 7 happens in any recursive call is at most $\frac{mL^2D}{p}$. By our choice of p, this is less than $\frac{1}{6}$. Thus, because the reduction only fails if (a) any list-decoding call fails to recover the correct polynomial, or (b) a recursive call is made on some t^* which fails to uniquely identify the correct polynomial in all lists, we can conclude that the probability that the reduction fails to compute $f(X)$ with probability at most $\frac{1}{6} + \frac{1}{6} = \frac{1}{3}$. This completes the proof of correctness.

Efficiency. Finally, let us analyze efficiency of the reduction. Note that in each recursive call, the number of rows in each array is cut in half. Therefore, the reduction will make at most $\log n$ recursive calls before the base case in the first step is triggered (which takes $O(mkd)$ time).

In step 5, computing $P_i(2^k), \ldots, P_i(T)$ can be performed in time $\tilde{O}(TD)$ via batch univariate polynomial evaluation [28], taking time $\tilde{O}(mTD)$ overall. The calls to A collectively take $T \cdot t_A \leq T\ell \cdot n^{k-\epsilon}$ time, per level of recursion.

In step 6, list decoding takes time $\tilde{O}(mT^{2.5})$ overall, by Corollary 31. In step 9, finding the first such polynomial in each list takes time at most $\tilde{O}(mD/(\delta\ell/m))$. All other steps clearly have cost at most linear in input size.

So, all in all, the total complexity (plugging in $T = \frac{(m \log n)^{2/q} q}{(\delta \frac{\ell}{m})^2}$) becomes:

$$\begin{aligned}\tilde{O}(T^{2.5}m + T\ell \cdot n^{k-\epsilon})) &= \tilde{O}\left(\frac{m^6}{\delta^5\ell^5} + \frac{m^2 n^{k-\epsilon}}{\delta\ell}\right) \\ &= \tilde{O}\left(\frac{mn^{k-\epsilon}}{(\delta\frac{\ell}{m})^{\Theta(1)}}\right)\end{aligned}$$

This completes the proof of Lemma 9. □

B Zero-Knowledge Proofs of Work

B.1 Definitions

We give the definitons needed to extend proofs of work both to be zero-knowledge and, soon, noninteractive. First, we give the definition of a robust non-interactive zero-knowledge proof system, as introduced in [23], which will be the eventual goal of this section.

Definition 35. Given an $\mathcal{NP}$ relation R, let $L = \{x : \exists w \text{ s.t. } R(x,w) = 1\}$. $\Pi = (q, \mathsf{P}, \mathsf{V}, \mathsf{S} = (\mathsf{S}_1, \mathsf{S}_2), \mathsf{E}))$ is a *robust NIZK argument* for L, if $\mathsf{P}, \mathsf{V}, \mathsf{S}, \mathsf{E} \in PPT$ and $q(\cdot)$ is a polynomial such that the following conditions hold:

- **Completeness.** For all $x \in L$ of length λ, all w such that $R(x,w) = 1$, and all $\Omega \in \{0,1\}^{q(\lambda)}$, $\mathsf{V}(\Omega, x, \mathsf{P}(\Omega, w, x))] = 1$.
- **Multi-Theorem Zero-Knowledge.** For all PPT adversaries $\mathcal{A}$, we have that $\textsc{Real}(\lambda) \approx \textsc{Sim}(\lambda)$, where

$$\textsc{Real}(\lambda) = \{\Omega \leftarrow \{0,1\}^{q(\lambda)}; out \leftarrow \mathcal{A}^{\mathsf{P}(\Omega,\cdot,\cdot)}(\Omega); \text{Output } out\},$$

$$\textsc{Sim}(\lambda) = \{(\Omega, tk) \leftarrow \mathsf{S}_1(1^\lambda); out \leftarrow \mathcal{A}^{\mathsf{S}_2'(\Omega,\cdot,\cdot,tk)}(\Omega); \text{Output } out\},$$

  and $\mathsf{S}_2'(\Omega, x, w, tk) \stackrel{\text{def}}{=} \mathsf{S}_2(\Omega, x, tk)$ if $(x,w) \in R$ and otherwise outputs `failure` if $(x,w) \notin R$.
- **Extractability.** There exists a PPT algorithm E such that, for all PPT $\mathcal{A}$,

$$\Pr\left[\begin{matrix}(\Omega, tk) \leftarrow \mathsf{S}_1(1^\lambda); (x,\pi) \leftarrow \mathcal{A}^{\mathsf{S}_2(\Omega,\cdot,tk)}(\Omega); w \leftarrow \mathsf{E}(\Omega, (x,\pi), tk) : \\ R(x,w) \neq 1 \wedge (x,\pi) \notin \mathcal{Q} \wedge \mathsf{V}(\Omega, x, \pi) = 1\end{matrix}\right] \leq \mathsf{negl}(\lambda),$$

  where $\mathcal{Q}$ contains the successful pairs (x_i, π_i) that $\mathcal{A}$ has queried to S_2.

Theorem 36 ([23])**.** *Assuming trapdoor permutations and a dense cryptosystem exist, robust NIZK arguments exist for all languages in $\mathcal{NP}$.*

We now introduce all the tools that will help us reach this goal, starting with the definition of a zero-knowledge proof of work.

Definition 37. An *interactive zero-knowledge proof of work* (zk-PoW) is an (interactive) $(t(n), \epsilon(n), \delta(n))$-PoW $(\mathsf{Gen}^{\mathtt{crs}}, \Pi = \langle P, V\rangle)$ if there exists a simulator S which runs in time $\tilde{O}(n)$ and, for any $x \leftarrow \mathsf{Gen}(1^n)$, we have

$$\mathsf{View}_{P,V,\mathsf{Gen}^{\mathtt{crs}}}(x) \approx \mathcal{S}(x),$$

where $\mathsf{View}_{P,V,\mathsf{Gen}^{\mathtt{crs}}}(x)$ denotes the distribution of the `crs` output by $\mathsf{Gen}^{\mathtt{crs}}$ in addition to the transcript generated by the (honest) prover P interacting with the (honest) verifier V when given that `crs`.

We will also prove extractability, necessary for a (zero-knowledge) proof of knowledge. Extraction is a stronger property than soundness. Essentially, extraction requires the existence of an extractor, which knows the trapdoor and uses this alongside an arbitrary prover P^* to extract out the secret that P^* is trying to prove.

Definition 38. An *interactive proof system* between Prover P and Verifier V is an (interactive) proof of knowledge for relation R with knowledge error $\epsilon(n)$ if there exists an extractor $\mathcal{E}$ which runs in time $\tilde{O}(n)$ and, for any w and arbitrary Prover P^*, we have:

$$\left|\Pr\left[(x,w) \in R : x \leftarrow \mathcal{E}^{P^*}(w)\right] - \Pr\left[\langle P^*, V\rangle(w) = 1\right]\right| \leq \epsilon(n).$$

If $\epsilon(n) = \mathsf{negl}(n)$, then we say $\langle P, V\rangle$ is a *proof of knowledge* for R.

In order to make our resulting proof of knowledge noninteractive, we will employ the Fiat-Shamir transform using *collision intractable hash functions*. In particular, to use these hash functions with our $2k$-round protocol, we will need a stronger property than the above soundness property, which is known as *round-by-round soundness*. We give its definition below.

Definition 39. Let $\Pi = (P, V)$ be a $2r$-message public coin interactive proof system for a language L. For any $x \in \{0,1\}^*$, and any prefix τ of a protocol transcript, let $V(x, \tau)$ denote the distribution of the next message (or output) of V when the transcript so far is τ and V was executed on input x.

We say that Π has *round-by-round soundness error* $\varepsilon(\cdot)$ if there exists a deterministic (not necessarily efficiently computable) function State that takes as input an instance x and a transcript prefix τ and outputs either accept or reject such that the following properties hold:

1. If $x \notin L$, then $\mathsf{State}(x, \emptyset) = \mathsf{reject}$, where $\emptyset$ denotes the empty transcript.
2. If $\mathsf{State}(x, \tau) = \mathsf{reject}$ for a transcript prefix τ, then for every potential prover message α, it holds that
$$\Pr\left[\mathsf{State}(x, \tau|\alpha|\beta) = \mathsf{accept}\right] \leq \varepsilon(n).$$
3. For any full transcript τ, if $\mathsf{State}(x, \tau) = \mathsf{reject}$, then $V(x, \tau) = 0$.

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

Luckily, for our protocol specifically, this property is very closely related to extractability. Finally, we present definitions related to collision intractable hashing.

Definition 40. For a given relation ensemble $R = \{R_\lambda \subseteq (\{0,1\}^{\nu(\lambda)})^{t(\lambda)} \times (\{0,1\}^{\mu(\lambda)})^{t(\lambda)}\}$, a hash family $\mathcal{H} = \{h_\lambda : \{0,1\}^{\kappa(\lambda)} \times \{0,1\}^{\nu(\lambda) \to \{0,1\}^{\mu(\lambda)}}\}$ is said to be *R-correlation intractable with security (s, δ)* if for every s-size adversary $\mathcal{A} = \{A_\lambda\}$,
$$\Pr\left[(\mathbf{x}, \mathbf{y} = (h(x_1), \ldots, h(x_t))) \in R\right] = O(\delta(\lambda)).$$

We say $\mathcal{H}$ is *R-correlation intractable with security δ* if it is (λ^c, δ)-correlation intractable for all $c > 1$. Finally, we say that $\mathcal{H}$ is *R-correlation intractable* if it is $(\lambda^c, 1/\lambda^c)$-correlation intractable for all $c > 1$.

For most existing schemes, this requires an additional property on the underlying relation known as *sparsity*. Informally, sparsity requires there to be negligibly many witnesses for any given statement in the language.

Definition 41 ([45])**.** A relation $R = \{R_\lambda \subseteq (\{0,1\}^{\nu(\lambda)})^{t(\lambda)} \times (\{0,1\}^{\mu(\lambda)})^{t(\lambda)}\}$ is *sparse* if for every $\mathbf{x} \in (\{0,1\}^{\nu(\lambda)})^{t(\lambda)}$,
$$\Pr[(\mathbf{x}, \mathbf{y}) \in R] \leq \mathsf{negl}(\lambda).$$

Finally, for our constructions, we will use the Decisional Diffie-Hellman (DDH) assumption, which we state here.

Definition 42. The DDH assumption states that there exists some $\mathcal{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}}$ a group ensemble with efficient representation, where each $\mathbb{G}_\lambda$ is a cyclic group of prime order $p(\lambda)$ such that, for any PPT $\mathcal{A}$, we have
$$\begin{aligned}&|\Pr[\mathcal{A}(1^\lambda, g, g^a, g^b, g^{ab}) = 1 : a, b \xleftarrow{\$} \mathbb{Z}_{p(\lambda)}] - \\ &\Pr[\mathcal{A}(1^\lambda, g, g^a, g^b, g^c) = 1 : a, b, c \xleftarrow{\$} \mathbb{Z}_{p(\lambda)}]| \leq \mathsf{negl}(n),\end{aligned}$$

where g is a generator for $\mathbb{G}_\lambda$.

The Sub-exponential DDH assumption instead assumes that the above is true for all non-uniform $\mathcal{A}$ that run in time $\lambda^{O((\log\log\lambda)^3)}$.

In Section B.3, we present a sub-protocol we will use for knowledge extraction. In Section B.4, we give our full zero-knowledge proof of work protocol for the k-orthogonal vectors problem and compose it with this sub-protocol to make a zero-knowledge proof of knowledge. We also refer to Section B.2 for a warm-up protocol, which more elegantly shows the techniques used by our main proof of knowledge at the cost of only having a quadratic Prover-Verifier gap. In Section B.5, we add a trapdoor to the proof of knowledge so that we can make our proof non-interactive using techniques from Jain and Jin [41] in Section B.6.

B.2 Warm-Up

As a warm-up, we first present a variant of the zero-knowledge proof of work (zk-PoW) from [4] based on the hardness of the k-orthogonal vectors problem $k\mathsf{OV}_{d(n)}$ ($d(n) = \mathsf{polylog}(n)$) and DDH for q-order Schnorr group G (with $q > n^2$).

For $\boldsymbol{x} = (x_1, \ldots, x_t)$, $\boldsymbol{h} = (h_1, \ldots, h_t)$, g and y, let $g^{\boldsymbol{x}} = (g^{x_1}, \ldots, g^{x_t})$, $\boldsymbol{h}^{\boldsymbol{x}} = (h_1^{x_1}, \ldots, h_t^{x_t})$ and $\boldsymbol{h}^y = (h_1^y, \ldots, h_t^y)$.

Protocol A Zero-Knowledge Proof of Work via $\mathcal{F}\mathsf{OV}^k$ [3]

CRS: $\mathsf{Gen}^{\mathrm{CRS}}(1^n) \to (g, h)$, where g generates G and $h = g^y$ for a uniformly random $y \xleftarrow{u} \mathbb{Z}_q$.

Challenge: $\mathrm{Gen}(1^n) \to (U^1, \ldots, U^k)$, where each U_i is i.d.d. uniform from G.

Protocol:

- Prover P computes the coefficients of $R_{U^1,\ldots,U^k}$ $\boldsymbol{\alpha}$, samples $\boldsymbol{r} \xleftarrow{u} \mathbb{Z}_q^{dn^{k/2}\log q}$, and $t \xleftarrow{u} \mathbb{Z}_q$.
  Let $\mathrm{bits}(\boldsymbol{\alpha}) = \boldsymbol{\beta} = (\beta_{1,0}, \ldots, \beta_{1,\log q}, \ldots, \beta_{dn^{k/2},0}, \beta_{dn^{k/2},\log q})$ denote the binary representation of α, such that $\sum_{i=0}^{\log q} 2^i(\beta_{1,i}, \ldots, \beta_{dn^{k/2},i}) = \boldsymbol{\alpha}$.
  Prover then sends $(a, b, \boldsymbol{c}, \boldsymbol{d}) = (g^t, h^t, g^{\boldsymbol{r}}, g^{\mathrm{bits}(\boldsymbol{\alpha})}h^{\boldsymbol{r}})$ to Verifier V.
- Verifier samples $x \xleftarrow{u} \mathbb{Z}_q$ and $s \xleftarrow{u} \mathbb{Z}_q$ and sends (x, s) to Prover.
  Let $\boldsymbol{x}$ denote $(1, x, x^2, \ldots, x^{dn^{k/2}-1})$.
- Prover computes $\boldsymbol{r}' = \sum_{i=0}^{\log q} 2^i(r_{1,i}, \ldots, r_{dn^{k/2},i})$ and sends $w = t + s\langle \boldsymbol{r}', \boldsymbol{x}\rangle$ to Verifier.
- Verifier computes $z = R_{U^1,\ldots,U^k}(x)$ as well as $\boldsymbol{c}' = (\prod_{i=0}^{\log q}(c_{1,i})^{2^i}, \ldots, \prod_{i=0}(c_{dn^{k/2},i})^{2^i})$ and $\boldsymbol{d}' = (\prod_{i=0}^{\log q}(d_{1,i})^{2^i}, \ldots, \prod_{i=0}(d_{dn^{k/2},i})^{2^i})$. It accepts if and only if

$$g^w = a \cdot (\boldsymbol{c}')^{s\cdot\boldsymbol{x}} \qquad \bigwedge \qquad h^w = b \cdot (\boldsymbol{d}')^{s\cdot\boldsymbol{x}}/g^{sz}$$

Theorem 43. *Suppose $k\mathsf{OV}$ takes $n^{k-o(1)}$ time to decide for all but finitely many input lengths and $d = \omega(\log n)$. Then, the protocol above is a $(N^2, \epsilon(n), \delta(n))$-zk-PoW for any $\epsilon(n) > 0$ and any $\delta(n) > 1/n^{o(1)}$.*

It remains to show that the extractor from Section B.3 does succeed. As this protocol will be used for the full, $2k$-round protocol, it is most useful to isolate it as a tool first to better understand it.

Proof. We prove efficiency, completeness, hardness, and zero-knowledge separately. Note that our efficiency goals are in terms of $N = n^{k/2}$.

Efficiency: Efficiency of Gen is clear. The verifier must only generate two random values in the first step, then generate knd random points for $(U^1, \ldots, U^k)$.

For the Prover, we see that it takes $O(dn^k)$ to compute the coefficients of R [5]. Breaking up each bit of this polynomial takes $O(dn^{k/2})$ time. Computing the bit encryptions only takes $O(dn^{k/2})$ time as well. Finally, the prover must calculate w, which it can do in $O(dn^{k/2})$ time.

Finally, for the Verifier, aside from generating and sending $O(dn^{k/2})$ random challenges, the only computation performed is recomputing $\mathbf{c}$ and $\mathbf{d}$, and computing $z = R_{U^1,\ldots,U^k}(x)$. All of these take $O(dn^{k/2})$ time as well, in particular the computation of the polynomial $R_{U^1,\ldots,U^k}$ on a single value takes this much time, as shown by [5].

Completeness: We have $R_{U_1,\ldots,U_k}(x) = f\mathsf{OV}^k_{U^{k/2+1},\ldots,U^k}(\phi_1(x)\ldots,\phi_d(x))$ on all inputs $x \in \mathbb{Z}_q$ when the Prover is honest. Additionally, in the case of honest Prover and Verifier, we have $\boldsymbol{r}' = \boldsymbol{r}$, $\boldsymbol{c}' = \boldsymbol{c}$, and $\boldsymbol{d}' = \boldsymbol{d}$, and so we see the Verifier accepts when given the following by the honest Prover:

$$\begin{aligned} g^w = g^{t+x\langle \boldsymbol{r}', \boldsymbol{x}\rangle} &= a \cdot (\boldsymbol{c}')^{x\cdot\boldsymbol{x}} \\ &= [t]_g \cdot \left(\prod_{i=0}^{\log q} (c_{j,i})^{2^i}_{j\in[dn^{k/2}]}\right)^{x\cdot\boldsymbol{x}} \\ &= [t + \boldsymbol{r}\cdot x\boldsymbol{x}]_g \\ &= [t + x\langle \boldsymbol{r}, \boldsymbol{x}\rangle]_g \end{aligned}$$

and

$$\begin{aligned} h^w = h^{t+x\langle \boldsymbol{r}', \boldsymbol{x}\rangle} &= b \cdot (\boldsymbol{d}')^{x\cdot\boldsymbol{x}} / g^{xz} \\ &= h^t \cdot (\prod_{i=0}^{\log q} (d^{2^i}_{j,i})_{j\in[dn^{k/2}]})^{x\cdot\boldsymbol{x}} / g^{xz} \\ &= [t]_h \cdot \left(\prod_{i=0}^{\log q} (d_{j,i})^{2^i}_{j\in[dn^{k/2}]}\right)^{x\cdot\boldsymbol{x}} / [xz]_g \\ &= [yt]_g \cdot [bits(\boldsymbol{\alpha}) + y\cdot\boldsymbol{r}]_g^{x\cdot\boldsymbol{x}} / [xz]_g \\ &= [yt + x\cdot\boldsymbol{\alpha}\cdot\boldsymbol{x} + yx(\boldsymbol{r}\cdot\boldsymbol{x}) - x\cdot R_{U_1,\ldots,U_k}(x)]_g \\ &= [yt + yx(\boldsymbol{r}\cdot\boldsymbol{x})]_g \\ &= [t + x(\boldsymbol{r}\cdot\boldsymbol{x})]_h\,. \end{aligned}$$

The correctness of the last checks relies on the correctness of the underlying bit protocol, assuming that each $\beta_{i,j}$ is in fact a single bit.

Zero-knowledge: Our simulator is given $(U_1, \ldots, U_k)$, where each U_i is drawn uniformly at random from G. We will design $\mathcal{S}$ as follows:

1. Let $\mathtt{crs}$ be (g, g^y), where g generates G and $y \xleftarrow{u} \mathbb{Z}_q$.
2. Sample $x \xleftarrow{u} \mathbb{Z}_q$. Sample $\boldsymbol{r} \xleftarrow{u} \mathbb{Z}_q^{dn^{k/2}}$ and set $\boldsymbol{c} = g^{\boldsymbol{r}}$.
3. Let $\beta = (\beta_{1,0}, \ldots, \beta_{1,\log q}, 0, \ldots, 0)$ be a $dn^{k/2}$-long binary vector, where $(\beta_{1,0}, \ldots, \beta_{1,\log q})$ is the binary representation of $R_{U_1,\ldots,U_k}(x)$ and all other bits are 0. Define $\boldsymbol{d} = (g^{\beta+y\boldsymbol{r}})$.

4. Sample $\mathbf{M} \xleftarrow{u} \mathbb{Z}_q^{dn^{k/2}\log q}$. Perform the proofs of bit encryption normally using $h = g^y$, $\boldsymbol{d}$ as above, and $\mathbf{M}$, receiving $\mathbf{E} = \{e_{i,j}\}$, $\mathbf{F} = \{f_{i,j}\}$, $\mathbf{U}_1 = \{u^1_{i,j}\}$, $\mathbf{U}_2 = \{u^2_{i,j}\}$, $\mathbf{V}_1 = \{v^1_{i,j}\}$, and $\mathbf{V}_2 = \{v^2_{i,j}\}$.
5. Sample $w \xleftarrow{u} \mathbf{Z}_q$ and set $a = g^w / g^{x\langle \boldsymbol{r}, \boldsymbol{x}\rangle}$ and $b = g^{yw} / g^{yx\langle \boldsymbol{r}, \boldsymbol{x}\rangle}$, where $\boldsymbol{x}$ is defined similarly as in the protocol.
6. Output the following:

$$((g, g^y), (a, b, \boldsymbol{c}, \boldsymbol{d}, \mathbf{E}, \mathbf{F}), (x, \mathbf{M}), (w, \mathbf{U}_1, \mathbf{U}_2, \mathbf{V}_1, \mathbf{V}_2)).$$

Efficiency of the simulator is clear for most steps as running in time $O(n^{k/2})$. Note that $R_{U_1,\ldots,U_k}(x)$ may be computed in time $O(n^{k/2})$ [5]. All other steps are sampling $O(n^{k/2})$ random values and exponentiating, so the overall simulator's time is $O(n^{k/2})$.

To see that the transcript is computationally indistinguishable from that of a real Prover and Verifier, we first note that many of the elements are distributed exactly as in the real interaction. Since $x, s, \boldsymbol{r}$, and w are all sampled uniformly at random, a and b are also distributed identically to the real protocol. The only remaining elements to check are $\boldsymbol{d}$. For this, we reduce to the DDH problem. Suppose there were an algorithm A which could distinguish between the output of $\mathcal{S}$ and the view of the real transcript. We construct an algorithm which distinguishes the tuple (g_1, g_2, g_3) as either (g^a, g^b, g^c) or (g^a, g^b, g^{ab}) as follows: Given (g_1, g_2, g_3), set $g^y = g_1$ and sample a random index $(i, j) \xleftarrow{u} \mathbb{Z}_q^{dn^{k/2}(\log q+1)}$ such that $i \geq 2$ and sent element (i, j) of $\boldsymbol{c}$ to be g^b. Perform the rest of the simulation as normal.

We did not prove soundness above because, as mentioned, extraction is a strictly stronger property than basic soundness. In order to extract, we use a proof of knowledge for bit encryption. First, we show that, assuming there exists an extractor for the bit encryptions, we may extract the entire secret.

Extractability: Let P^* be an arbitrary prover who makes V accept with probability 1. Given a CRS (g, h) where $\mathcal{E}$ knows the exponent t such that $g^t = h$, we design $\mathcal{E}^{P^*}$ as follows:

1. $\mathcal{E}$ runs P^* until receiving $(a, b, \boldsymbol{c}, \boldsymbol{d})$ from P^*.
2. Running the extractor from Section B.3 twice for each index of $(\boldsymbol{c}, \boldsymbol{d})$, $\mathcal{E}$ recovers the bits $\beta' = (\beta'_{i,j})_{i\in[dn^{k/2}], j\in[\log q]}$.
3. Define R' to be the polynomial of degree at most $dn^{k/2}$ defined by coefficients $R'_i = \sum_{j\in[0,\log q]} 2^j \beta'_{i,j}$. Then, let $A = \sum_{i=1}^{n^{k/2}} R'(i)$. Output A.

First, note that the extractor strategy is well-formed—indeed, A is exactly $f\mathsf{OV}^k$ if $P^* = P$. Then, from the extractability of the bit proofs, we have that each $\beta'_{i,j}$ must be a bit. So, for any prover behavior, we may extract a polynomial of degree at most $dn^{k/2}$. To confirm A is in particular a polynomial satisfying the orthogonal vectors relation, we may continue running the prover on different challenges and confirming it is giving back the polynomial R evaluated on those challenges, but in either case, we output A. □

B.3 Proofs of Knowledge of Bit Encryption

In order to construct a proof of knowledge for $k\mathsf{OV}$, we will first construct a zero-knowledge proof of work where the Prover sends the proof bit by bit to the Verifier. By composing this with a zero-knowledge proof of knowledge for bit encryption, we can then transform our proof into a proof

of knowledge for $k\mathsf{OV}$. To this end, we present Protocol B.3, adapted from [21] and relying on the discrete logarithm problem.

Protocol Proof of Bit Encryption

CRS: $\mathsf{Gen}^{\mathrm{CRS}}(1^n) \to (g, h)$, where g generates G and $h = g^y$ for a uniformly random $y \xleftarrow{u} \mathbb{Z}_q$.

Challenge: $\mathrm{Gen}(1^n) \to g^\beta h^r$, where $r \xleftarrow{u} \mathbb{Z}_q$ and $\beta \in \{0, 1\}$.

Protocol:

- Prover P samples and computes the following:
  - If $\beta = 0$, Prover samples $w, u^1, v^1 \xleftarrow{u} \mathbb{Z}_q$ and computes $e := h^w$ and $f := h^{u^1}(d/g)^{-v^1}$.
  - If $\beta = 1$, Prover samples $w, u^2, v^2 \xleftarrow{u} \mathbb{Z}_q$ and computes $e := h^{u^2}(d)^{-v^2}$ and $f := h^w$.

  In either case, Prover sends e, f to Verifier V.
- Verifier samples $m \xleftarrow{u} \mathbb{Z}_q$ and sends m to Prover.
- Prover computes the following:
  - If $\beta = 0$, set $v^2 := m - v^1$ and $u^2 := w + rv^2$.
  - If $\beta = 1$, set $v^1 := m - v^2$ and $u^1 := w + rv^1$.

  In either case, Prover sends u^1, u^2, v^1, v^2 to Verifier.
- Verifier accepts if and only if

$$m = v^1 + v^2 \qquad \bigwedge \qquad h^{u^1} = f(d/g)^{v^1} \qquad \bigwedge \qquad h^{u^2} = ed^{v^2}.$$

Consider the game between a dishonest prover and honest verifier as the prover sending (e^*, f^*), the verifier sending m generated uniformly at random, and the prover sending $u^{1*}(m), v^{1*}(m), u^{2*}(m)$, $v^{2*}(m)$. Suppose the prover can convince the verifier to accept on two different transcripts with the same first message (e^*, f^*). That is, let (m_1, m_2) be two challenges to the prover generated uniformly at random, and let $(u_1^{1*}, v_1^{1*}, u_1^{2*}, v_1^{2*})$ and $(u_2^{1*}, v_2^{1*}, u_2^{2*}, v_2^{2*})$ be the responses from the prover to challenges m_1 and m_2, respectively. For the verifier to accept, the following must be true:

$$\begin{array}{lll}
(1)\ m_1 = v_1^{1*} + v_1^{2*}, & (2)\ h^{u_1^{1*}} = f^*(d/g)^{v_1^{1*}}, & (3)\ h^{u_1^{2*}} = e^* d^{v_1^{2*}}, \\
(4)\ m_2 = v_2^{1*} + v_2^{2*}, & (5)\ h^{u_2^{1*}} = f^*(d/g)^{v_2^{1*}}, & (6)\ h^{u_2^{2*}} = e^* d^{v_2^{2*}}.
\end{array}$$

Let $\Delta = m_1 - m_2$. We derive the following from (1), (3), and (4):

$$\begin{aligned}
h^{u_1^{2*}} &= e^* d^{m_1 - v_1^{1*}} \\
&= e^* d^{\Delta + m_2 - v_1^{1*}} \\
&= h^{u_2^{2*}} d^{v_2^{1*} + \Delta - v_1^{1*}}, \text{ and} \\
h^{u_1^{2*} - u_2^{2*}} &= d^{\Delta + v_2^{1*} - v_1^{1*}} \\
h^{u_1^{2*} - u_2^{2*}} g^{-v_2^{1*}} &= d^{\Delta - v_1^{1*}} (d/g)^{v_2^{1*}} \\
&= d^{\Delta - v_1^{1*}} h^{u_2^{1*}} (f^*)^{-1}.
\end{aligned}$$

Rearranging terms, we have

$$f^* h^{u_1^{2*} - u_2^{2*}} g^{-v_2^{1*}} = d^{\Delta - v_1^{1*}} h^{u_2^{1*}}.$$

By multiplying both sides by $(d/g)^{v_1^{1*}}$ and using (2), we get

$$h^{u_1^{1*}+u_1^{2*}-u_2^{2*}} g^{-v_2^{1*}} = d^{\Delta} h^{u_2^{1*}} g^{-v_1^{1*}}$$
$$h^{u_1^{2*}-u_2^{2*}} g^{v_1^{1*}-v_2^{1*}} = d^{\Delta} h^{u_2^{1*}-u_1^{1*}}.$$

Substituting $v_1^{1*} - v_2^{1*} = m_1 - v_1^{2*} - m_2 + v_2^{2*} = \Delta + v_2^{2*} - v_1^{2*}$, we have

$$\begin{aligned}
h^{u_1^{2*}-u_2^{2*}} g^{\Delta+v_2^{2*}-v_1^{2*}} &= d^{\Delta} h^{u_2^{1*}-u_1^{1*}} \\
h^{u_1^{2*}-u_2^{2*}} g^{v_2^{2*}-v_1^{2*}} &= (d/g)^{\Delta} h^{u_2^{1*}-u_1^{1*}} \\
h^{u_1^{1*}+u_1^{2*}} g^{v_2^{2*}-v_1^{2*}} &= (d/g)^{\Delta} h^{u_2^{1*}+u_2^{2*}} \\
f^*(d/g)^{v_1^{1*}} e^* d^{v_1^{2*}} &= (d/g)^{\Delta} f^*(d/g)^{v_2^{1*}} e^* d^{v_2^{2*}} \\
(d/g)^{v_1^{1*}} d^{v_1^{2*}} &= (d/g)^{\Delta} (d/g)^{v_2^{1*}} d^{v_2^{2*}}.
\end{aligned}$$

From this, we can derive d^{Δ} as

$$\begin{aligned}
d^{\Delta} &= \frac{d^{v_1^{1*}+v_1^{2*}} g^{v_1^{2*}-v_2^{1*}-v_2^{2*}}}{d^{v_2^{2*}}} \\
&= \frac{d^{v_1^{1*}+v_1^{2*}-v_2^{2*}}}{g^{v_2^{1*}+v_2^{2*}-v_1^{2*}}} \\
&= \frac{d^{m_1}}{g^{m_2}} \cdot \frac{g^{v_1^{2*}}}{d^{v_2^{2*}}}.
\end{aligned}$$

Substituting back $m_1 - m_2 = \Delta$, we may then simplify

$$d^{-m_2} = g^{v_1^{2*}-m_2} d^{-v_2^{2*}}.$$

This implies $d^{v_2^{1*}+v_2^{2*}} = d^{v_2^{2*}} g^{m_2-v_1^{2*}}$, which in turn implies

$$d^{v_2^{1*}} = g^{m_2-v_1^{2*}}.$$

Because $d = g^{\beta^*} h^{r^*}$ for some (β^*, r^*), and in particular we are given $c = g^r$, the extractor (which knows t such that $g^t = h$) may find $d/c^t = g^{\beta^*}$. So, we have that

$$g^{\beta^*} = d^{v_2^{1*}} / c^{t v_2^{1*}} = g^{m_2-v_1^{2*}} / c^{t v_2^{1*}}.$$

Thus, the extractor may find β^* by calculating g^{β^*} and determining whether it equals 1 or g.

B.4 The Full zkPoW Protocol

In this section, for simplicity we will use the notation $[a]_g$ to refer to g^a for a generator g and field element a. When g is obvious from context, we will just drop it and simply use $[a]$.

The full protocol achieves a gap of $\tilde{O}(n^k)$ prover time and $\tilde{O}(n)$ verifier time (where coefficients logarithmic in the security parameter p are absorbed into the big-O constant). The protocol is as follows:

Protocol zkPoW

CRS: $\mathsf{Gen}^{\mathrm{CRS}}(1^n) \to (g, [x])$, where g generates G and $[x] = [x]_g$ for a uniformly random $x \xleftarrow{u} \mathbb{Z}_q$.

Challenge: $\mathrm{Gen}(1^n) \to (U^1, \ldots, U^k)$, where each U_i is i.d.d. uniform from G.

Protocol:

- Prover P computes the coefficients of $R_{U^1,\ldots,U^k}(q)$ and samples $r^1 \xleftarrow{u} \left(\mathbb{Z}_p^*\right)^D$ and $t_1, t_1' \xleftarrow{u} \mathbb{Z}_p^*$. Prover then sends the following to Verifier V:

$$[r^1], m[q] + r^1[x], [t_1], [t_1'], (t_1' - t_1)[x].$$

- For each of $s = 2, \ldots, k-1$:
  - Verifier samples $\alpha_{s-1} \xleftarrow{u} \mathbb{Z}_p$ and sends α_{s-1} to Prover.
  - Prover samples $r^s \xleftarrow{u} \left(\mathbb{Z}_p^*\right)^D$ and $t_s, t_s' \xleftarrow{u} \mathbb{Z}_p^*$. Prover then sends the following to Verifier V:

$$[r^s], m[q^{s,\alpha_1,\ldots,\alpha_{s-1}}] + r^s[x], [t_s], [t_s'], (t_s' - t_s)[x].$$

  - Verifier samples $c_{s-1} \xleftarrow{u} \mathbb{Z}_p^*$ and sends c_{s-1} to Prover.
  - Prover computes $u_{s-1} := t_{s-1}' + c_{s-1} \sum_{i=0}^{q} r^s(i)$ and $v_{s-1} := t_{s-1} + c_{s-1} r^{s-1}(\alpha_{s-1})$ and sends (u_{s-1}, v_{s-1}) to Verifier.
  - Verifier continues if and only if:

$$[u_{s-1}] = [t_{s-1}'] + c_{s-1} \sum [r^s(i)] \qquad \bigwedge \qquad [v_{s-1}] = [t_{s-1}] + c_{s-1}[r^{s-1}(\alpha_{s-1})]$$

    and

$$(u_{s-1} - v_{s-1})[x] = c_{s-1}^2 \left(\sum [m q^{s,\alpha_1,..,\alpha_{s-1}} + x r^s(x)] \right) / [m q^{s-1,\alpha_1,..,\alpha_{s-2}}(\alpha_{s-1}) + x r^{s-1}(\alpha_{s-1})] + [(t_{s-1}' - t_s)x].$$

- Verifier samples $\alpha_{k-1} \xleftarrow{u} \mathbb{Z}_p$ and $c_{k-1} \xleftarrow{u} \mathbb{Z}_p^*$ and sends (α_{k-1}, c_{k-1}) to Prover.
- Prover computes $w := t_{k-1} + c_{k-1} r^{k-1}(\alpha_{k-1})$ and sends w to Verifier.
- Verifier accepts if and only if:

$$[w] = [t_{k-1}] + c_{k-1}[r^{k-1}(\alpha_{k-1})]$$

  and

$$w[x] = [x t_{k-1}] + c_{k-1}[m q^{k-1,\alpha_1,..,\alpha_{k-2}}(\alpha_{k-1}) + x r^{k-1}(\alpha_{k-1})] - [q^{k-1,\alpha_1,..,\alpha_{k-2}}(\alpha_{k-1})].$$

Essentially, this protocol may be thought of as the prover first committing to the polynomial q, followed by a series of sum-checks using the fact that, for any sequence of $\alpha_1, \ldots, \alpha_{s-1}$, $\sum q^{s,\alpha_1,\ldots,\alpha_{s-1}(i)} = q^{s-1,\alpha_1,\ldots,\alpha_{s-2}}(\alpha_{s-1})$. This allows the verifier's time to be reduced by offloading much of the work of checking consistency of polynomials to the prover. We now present our main theorem for this section.

Theorem 44. *Suppose $\mathcal{F}$OV takes $n^{k-o(1)}$ time to decide for all but finitely many input lengths and $d = \omega(\log n)$. Then, Protocol* zkPoW *is a $(N^k, \epsilon(n), \delta(n))$-zk-PoW for any $\epsilon(n) > 0$ and any $\delta(n) > 1/n^{o(1)}$. Additionally, when composed with Protocol B.3, the resultant protocol is additionally a proof of knowledge and round-by-round sound.*

The exact composition of Protocol B.3 with Protocol B.4 is described in the proof of Theorem 44. Essentially, we will be giving the bits for the coefficients of each of the q-polynomials in Protocol B.4. We do not provide this full protocol as explained above for clarity. We also prove completeness and zero-knowledge, and use the extractor to straightforwardly prove round-by-round soundness.

Completeness: If the prover is honest, then we want the verifier to accept every intermediate check as well as the final acceptance condition.

For the intermediate condition, the verifier checks:

$$\begin{aligned}[u_{s-1}] &= [t'_{s-1} + c_{s-1} \sum_i r^s(i)] \\ &= [t'_{s-1}][c_{s-1} \sum_i r^s(i)] \\ &= [t'_{s-1}] \prod_i [r^s]^{i \cdot c_{s-1}}\end{aligned}$$

and

$$\begin{aligned}[v_{s-1}] &= [t_{s-1} + c_{s-1} r^{s-1}(\alpha_{s-1})] \\ &= [t_{s-1}][c_{s-1} r^{s-1}(\alpha_{s-1})] \\ &= [t_{s-1}][r^{s-1}(\alpha_{s-1})]^{c_{s-1}}\end{aligned}$$

and, letting $b^s = m[\tilde{q}^s] + r^s[x]$ for all s,

$$\begin{aligned}(u_{s-1} - v_{s-1})[x] &= \left(\left(t'_{s-1} + c_{s-1} \sum r^s(i)\right) - \left(t_{s-1} + c_{s-1} r^{s-1}(\alpha_{s-1})\right)\right)[x] \\ &= (t'_{s-1} - t_{s-1})[x] + c_{s-1}[x] \left(\sum r^s(i) - r^{s-1}(\alpha_{s-1})\right) \\ &= (t'_{s-1} - t_{s-1})[x] + c_{s-1} \left(\sum b^s(i) - m[\tilde{q}^s](i) - b^{s-1}(\alpha_{s-1}) + m[\tilde{q}^{s-1}](\alpha_{s-1})\right) \\ &= (t'_{s-1} - t_{s-1})[x] + c_{s-1} \left(\sum b^s(i) - m[\tilde{q}^s(i)] - b^{s-1}(\alpha_{s-1}) + m[\tilde{q}^{s-1}](\alpha_{s-1})\right) \\ &= (t'_{s-1} - t_{s-1})[x] + c_{s-1} \left(\sum b^s(i) - m[\tilde{q}^{s-1}(\alpha_{s-1})] - b^{s-1}(\alpha_{s-1}) + m[\tilde{q}^{s-1}](\alpha_{s-1})\right) \\ &= (t'_{s-1} - t_{s-1})[x] + c_{s-1} \left(\sum b^s(i) - b^{s-1}(\alpha_{s-1})\right) \\ &= c_{s-1} \left(\sum \left(m[\tilde{q}^s] + r^s[x]\right)(i)\right) - c_{s-1} \left(m[\tilde{q}^{s-1}] + r^{s-1}[x]\right)(\alpha_{s-1}) + (t'_{s-1} - t_{s-1})[x].\end{aligned}$$

The final check proceeds similarly.

Zero Knowledge: Our simulator is given $(U_1, \ldots, U_k)$, where each U_i is drawn uniformly at random from G. We will design Sim as follows:

1. Let crs be g, where g generates G, and $[x]$ for $x \xleftarrow{u} \mathbb{Z}_q$.
2. Sample $\alpha_0, \ldots, \alpha_{k-1} \leftarrow \mathbb{Z}_p$, $c_0, \ldots, c_{k-1} \leftarrow \mathbb{Z}_p^*$.
3. Sample $r^0, r^1, \ldots, r^{k-1}$, $\tilde{u}_1, \ldots, \tilde{u}_{k-2} \xleftarrow{u} \mathbb{Z}_q$, and $\tilde{v}_1, \ldots, \tilde{v}_{k-2} \xleftarrow{u} \mathbb{Z}_q$. For each $i = 1, \ldots, k-2$, set
   - $\tilde{t}_i = \tilde{v}_i - c_i r^i(\alpha_i)$
   - $\tilde{t}'_i = \tilde{u}_i - c_i(\sum r^i(j))$
4. Finally, sample $\tilde{w} \xleftarrow{u} \mathbb{Z}_q$ and set $\tilde{t}_{k-1} = \tilde{w} - c_{k-1} r^{k-1}(\alpha_{k-1})$. Set $\tilde{q}$ to be the polynomial whose constant term is $q^{k,\alpha_1,\ldots,\alpha_{k-1}}(x)$ and all other bits are zero. Output

$$[r^1], m[\tilde{q}] + r^1[x], [\tilde{t}_1], [\tilde{t}'_1], (\tilde{t}_1 - \tilde{t}'_1)[x],$$

the following for $s = 2, \ldots, k-1$:

$$((\alpha_{s-1}), ([r^s], m[\tilde{q}] + r^s[x], [\tilde{t}_s], [\tilde{t}'_s], (\tilde{t}_s - \tilde{t}'_s)[x]), (c_{s-1}), (\tilde{u}_{s-1}, \tilde{v}_{s-1}))$$

and, in addition,

$$((\alpha_{k-1}, c_{k-1}), (\tilde{w})).$$

Efficiency of the simulator is straightforward. Zero-knowledge follows straightforwardly from the analysis presented in the warm-up. The actual verifier is public-coin, so the simulator samples its values exactly as in the real proof. For the prover, notice that the r^s terms, α_s and c_s are all distributed exactly as in the real proof. Because r^s is a random polynomial, $\sum_j r^s(j)$ is distributed as a random value, and because the c_s are drawn only from group generators, we have that $c_s r^s(\alpha)$ and $c_s(\sum_j r^s(j)$ are distributed uniformly at random. So $\tilde{t}_s$ and $\tilde{t}'_s$ are the difference of uniformly random values, which is also uniform. By the same reasoning, each $\tilde{u}_s$, $\tilde{v}_s$, and $\tilde{w}$ are also distributed uniformly in the actual proof, so all of the prover's outputs are distributed as in the actual proof with the potential exception of the $m[\tilde{q}] + r^s[x]$ terms. As in the warm-up, we again reduce to the DDH problem. We refer to that analysis in Section B.2.

Extractability with proofs of bit encryption: In a similar way to the warm-up, we want to introduce bit proofs in order to bootstrap extractibility to the protocol. In this case, we will encode the bits of each bit of each $q^{s,\alpha_1,\ldots,\alpha_{s-1}}$.

Extractability implies soundness. We will further use this extractor to prove round-by-round soundness (Definition 39).

Let $\beta_s = \{\beta_{s,i,j}\}_{i\in[D],j\in[\log q]}$ be the binary representation of $q^{s,\alpha_1,\ldots,\alpha_{s-1}}$ for each s, such that the sum $\sum_{j=0}^{\log q} 2^j(\beta_{s,1,j}, \ldots, \beta_{s,D,j})$ equals the vector of coefficients of $q^{s,\alpha_1,\ldots,\alpha_{s-1}}$. The new proof of work will be the same as before with the following additions:

1. At the beginning, P generates $r^0 \leftarrow \left(\mathbb{Z}_p^*\right)^{D\times\log q}$, calculates $\beta_1 = bits(q')$ and, instead of sending $[r^1], m[q'] + r^1[x]$, sends $[r^1], m[\beta_1] + r^1[x]$.
2. For each of $s = 2, \ldots, k-1$, P now generates $r^s \leftarrow \left(\mathbb{Z}_p^*\right)^{D\times\log q}$, finds $\beta_s = bits(q^{s-1,\alpha_1,\ldots,\alpha_{s-1}})$, and sends $[r^s], m[\beta_s] + r^s[x], [t_s], [t'_s], (t'_s - t_s)[x]$. Additionally, for each bit $\beta_{s,i,j}$:
   - If $\beta_{s,i,j} = 0$, P samples $w_{s,i,j}, u^1_{s,i,j}, v^1_{s,i,j} \xleftarrow{u} \mathbb{Z}_q$ and sets $e_{s,i,j} = w_{s,i,j}[x]$ and $f_{s,i,j} = u^1_{s,i,j}[x] - mv^1_{s,i,j}[\beta_{s,i,j} + r^s_{i,j}x - 1]$.
   - If $\beta_{s,i,j} = 1$, P samples $w_{s,i,j}, u^2_{s,i,j}, v^2_{s,i,j} \xleftarrow{u} \mathbb{Z}_q$ and sets $e_{s,i,j} = u^2_{s,i,j}[x] - mv^2_{s,i,j}[\beta_{s,i,j} + r^s_{i,j}x]$ and $f_{s,i,j} = w_{s,i,j}[x]$.

   In either case, P sends $e_{s,i,j}, f_{s,i,j}$ to V.
3. For each of $s = 2, \ldots, k-1$, V instead of sending just c_{s-1} now additionally samples $\mathbf{ch}_s = (ch_{s,i,j})_{i\in[D],j\in[\log q]}$ and sends these challenges alongside c_{s-1} to P.
4. For each of $s = 2, \ldots, k-1$, P additionally computes the following:
   - If $\beta_{s,i,j} = 0$, P sets $v^2_{s,i,j} = ch_{s,i,j} - v^1_{s,i,j}$ and $u^2_{s,i,j} = w_{s,i,j} + r^s_{i,j}v^2_{s,i,j}$.
   - If $\beta_{s,i,j} = 1$, P sets $v^1_{s,i,j} = ch_{s,i,j} - v^2_{s,i,j}$ and $u^1_{s,i,j} = w_{s,i,j} + r^s_{i,j}v^1_{s,i,j}$.

   In either case, P additionally sends $(u^1_{s,i,j}, v^1_{s,i,j}, u^2_{s,i,j}, v^2_{s,i,j})$ for each i, j to P.
5. In each intermediate check, V in addition only continues if, for each i, j:

$$ch_{s,i,j} = v^1_{s,i,j} + v^2_{s,i,j} \wedge u^1_{s,i,j}[x] = f_{s,i,j}\left(a^s_{2,i,j}/g\right)^{v^1_{s,i,j}} \wedge u^2_{s,i,j}[x] = e_{s,i,j}\left(a^s_{2,i,j}\right)^{v^2_{s,i,j}}.$$

As before, we may now use the bit proofs to extract out each bit $\beta_{s,i,j}$, assuming we know the secret exponent x. So, the extractor runs as follows:

1. The extractor may generate its set of challenges from the start uniformly at random. We assume it generates $\alpha_1, \ldots, \alpha_{k-1}$ and $c_1, \ldots, c_{k-1}$ uniformly at random at the start of running. It also may generate the challenge vectors $\mathbf{ch}_s$ for each s, as well.
2. Begin running P^*, receiving $[r^1], m[\beta_1] + r^1[x], [t_1], [t'_1], (t'_1 - t_1)[x]$. The extractor will run as the verifier, responding with the values it generated for each $s = 2, \ldots, k-1$ as in the protocol. In addition, the extractor, who is given as input the secret exponent x, will also compute $(\left(a^s_{2,i,j}\right)^{1/x} - a^s_{1,i,j}) \cdot x$ and compare this to g. If it is equal, then the extractor will set $\beta'_{s,i,j} = 1$. Else, it will set it to 0.
3. For each $s = 2, \ldots, k-1$, let $q^{s,*}$ be the reconstructed polynomial whose coefficients are defined by $R'_{s,i} \sum_{j=0}^{\log q} 2^j (\beta'_{s,i,j}$.

Next, we prove the following claim.

Claim 8. *For any $s = 2, \ldots, k-1$, if the extractor cannot extract $q^{s,\alpha_1,\ldots,\alpha_{s-1}}$, then with all but negligible probability, the verifier check for the same setting of s will fail.*

Proof. We will prove this by induction on s. First, let $s = 2$. Then, the extractor has received the following from P^*:

$$[r^1], m[q^*] + r^1[x], [t_1], [t'_1], (t'_1 - t_1)[x],$$
$$[r^2], m[q^{2*}] + r^2[x], [t_2], [t'_2], (t'_2 - t_2)[x],$$

and

$$u_1, v_1,$$

as well as having access to the values α_1, c_1. Let $\tilde{q^2}$ be the extractor's guess (from the bit proof extraction) for q^{2*}. Suppose toward a contradiction that $\tilde{q^2} \neq q^{2,\alpha_1}$, but V will continue with its checks. Then, we see that necessarily

$$\begin{aligned}
(u_1 - v_1)[x] &= c_1[\sum mq^{2*}(i) + xr^2(i)] - c_1[mq^{1*} + xr^1(\alpha_1)] + (t'_1 - t_2)[x] \\
&= [\sum mq^{2*}(i) - mq^{1*}][x] + c_1^2[\sum xr^2(i) - xr^1(\alpha_1)] + (t'_1 - t_2)[x] \\
(u_1 - v_1) - (t'_1 - t_1))[x] &= c_1[\sum mq^{2*}(i) - mq^{1*}][x] + c_1^2[\sum xr^2(i) - xr^1(\alpha_1)] \\
[u_1 - v_1 - t'_1 + t_1] &= [\sum mq^{2*}(i) - mq^{1*}] + c_1^2[\sum r^2(i) - r^1(\alpha_1)].
\end{aligned}$$

The first two checks require that $[u_1 - t'_1] = c_1 \sum [r^2(i)]$ and $[v_1 - t_1] = c_1[r^1(\alpha_1)]$. Substituting these in the above, we see that it must be that $[(u_1 - t'_1) - (v_1 - t_1)] = [\sum mq^{2*}(i) - mq^{1*}] \cdot [(u_1 - t'_1) - (v_1 - t_1)]$.

Dividing both sides by $[(u_1 - t'_1) - (v_1 - t_1)]$, we see that $[0] = [\sum mq^{2*}(i) - mq^{1*}]$. However, because $q^{2*} \neq q^{2,\alpha_1}$, with overwhelming probability these polynomials are not equal, and so their difference will not equal 0. This completes the base case.

The inductive case is similar, but this time we leverage the fact that, for some setting of $2 \leq j < k-1$, we have that with high probability $q^{j*} = q^{j,\alpha_1,\ldots,\alpha_{j-1}}$ but $q^{(j+1)*} \neq q^{j+1,\alpha_1,\ldots,\alpha_j}$. Notice that the only terms in the verifier's checks not dependent soleley on values for $s - 1 = j$ are the coefficients of $q^{j+1,\alpha_1,\ldots,\alpha_j}$. By the same reasoning as above, then, we will eventually get

$$[u_j - v_j - t'_j + t_j] = [\sum mq^{(j+1)*}(i) - mq^{j*}] + c_1^2[\sum r^{j+1}(i) - r^j(\alpha_1)].$$

□

The above claim leads naturally to the proof of round-by-round soundness. This is to be expected, as our protocol essentially puts in the round-by-round verifications into the verifier.

Corollary 45. *Protocol* zkPoW *is round-by-round sound.*

Proof. We show our extractor has a good round-by-round soundness State (Definition 39). Because the extractor runs as the verifier (including in particular its checks), taking $k-2$ union bounds over the claims above gives us the extractor satisfies the second property. The extractor will reject if the cheating prover does not send any information, giving us the first property. The third property comes for free. □

B.5 Adding a Trapdoor

In order to make our proof non-interactive and multi-theorem, we want to add a trapdoor for the DDH triple we use in the CRS. We first present our trapdoor protocol, which is based on the digital signature scheme by Craum and Pedersen [19]. We will inherit much of their analysis for the properties of the proof.

Protocol $(2k-3)$-Proof of Knowledge of DDH Triple

CRS: $\mathsf{Gen}^{\mathrm{CRS}}(1^n) \to g$, where g generates G.

Challenge: $\mathrm{Gen}(1^n) \to \{a_i, b_i, c_i\}_{i=1}^{2k-3}$), where each $a_i = g^{y_{1,i}}$, $b_i = g^{y_{2,i}}$ for uniformly random $y_{1,i}, y_{2,i} \xleftarrow{u} \mathbb{Z}_q$, and each $c_i = g^{y_{3,i}}$ for either $y_{3,i} = y_{1,i}y_{2,i}$ or uniformly random $y_{3,i} \xleftarrow{u} \mathbb{Z}_q$ with even probability.

Protocol:

- Prover P samples $z_1 \xleftarrow{u} \mathbb{Z}_q$ and computes $A_1 := a_1^{z_1}$ and $B_1 := b_1^{z_1}$. Prover then sends (A_1, B_1) to the Verifier V.
- Verifier samples $r_1 \xleftarrow{u} \mathbb{Z}_q$ and sends r_1 to Prover.
- For each of $s = 2, \ldots, 2k-3$:
  - Prover computes $x_{s-1} := z_{s-1} + r_{s-1}y_{3,s-1}$. In addition, Prover samples $z_s \xleftarrow{u} \mathbb{Z}_q$ and computes $A_s := a_s^{z_s}$ and $B_s := b_s^{z_s}$. Prover then sends (x_{s-1}, A_s, B_s) to Verifier.
  - Verifier continues if and only if

$$a_{s-1}^{x_{s-1}} = A_{s-1}c_{s-1}^{r_{s-1}} \qquad \bigwedge \qquad b_{s-1}^{r_{s-1}} = B_{s-1}c_{s-1}^{r_{s-1}}.$$

  Then, Verifier samples $r_s \xleftarrow{u} \mathbb{Z}_q$ and sends r_s to Prover.

Verifier accepts if and only if

$$a_{2k-3}^{x_{2k-3}} = A_{2k-3}c_{2k-3}^{r_{2k-3}} \qquad \bigwedge \qquad b_{2k-3}^{r_{2k-3}} = B_{2k-3}c_{2k-3}^{r_{2k-3}}.$$

Essentially, we are performing $2k-3$ proofs of knowledge of DDH triple exponents in sequence. The underlying proof of knowledge (take the 3-round protocol for any fixed s above) is a standard and well-known result. As we are just chaining independent instances of this a constant number of times, efficiency, completeness, zero-knowledge, and extractability should be immediate. Additionally, extractability here implies special soundness of the protocol. Additionally, by structuring the protocol in this way, the total round complexity of Protocol B.5 is $2(2k-3)+1 = 4k-5$, the same as Protocol B.4. This is crucial for us, as we will now combine Protocols B.5 and B.4 as in [21]. We state our main corollary (of Theorem 8 from [21]):

Corollary 46. *Let R_1 be the relation consisting of instances of the $\mathcal{F}\mathsf{OV}$ problem and its witnesses, and let R_2 be the relation consisting of instances of sequences of $2k-3$ DDH tuples and their witnesses. Suppose that Protocol B.5 is a $4k-5$-round zero-knowledge proof of knowledge satisfying the special soundness property, and Protocol B.4 (with the extractability augmentation) is a $4k-5$-round zero-knowledge proof of knowledge satisfying the special soundness property. Then, there is a $4k-5$-round public coin, witness indistinguishable proof of knowledge for the relation $R_1 \vee R_2$.*

Proof. Note that both Protocol B.5 and Protocol zkPoW are public coin. Then, the proof of knowledge follows directly from Theorem 8 from [21] when instantiating the secret-sharing scheme with a 1-out-of-2 secret sharing. □

B.6 Making The Proof Non-interactive

In order to make our protocol noninteractive, we will use the standard Fiat-Shamir transform [27]. Recall that, at a high level, if the verifier of a proof is public-coin, then the proof can be turned into a noninteractive argument for the same language by using a random oracle on the transcript in place of the verifier's responses. With high likelihood, then, the prover cannot cheat on its messages, as their challenge will be out of its control.

As our work aims to avoid random oracles, though, we clearly cannot use them in doing Fiat-Shamir. Thankfully, a line of recent works—e.g. [13, 45, 41, 20]—have constructed *collision-intractable hashes*, which allow one to instantiate the Fiat-Shamir transform under various assumptions. We will use in particular the collision-intractable hash construction given by Jain and Jin [41], which relies on the subexponential-DDH assumption and a common random string.

Specifically, Jain and Jin [41] showed that, assuming sub-exponential hardness of DDH, there is a CIH for TC^0. They then used this to construct a (statistical) multi-theorem NIZK with two main ingredients:

1. A lossy encryption scheme with low-depth decryption, specifically where decryption can be done in TC^0, and
2. A trapdoor Σ-protocol which can identify "bad" challenges in TC^0 and allows for knowledge extaction from a single transcript.

At a high level, the encryption scheme allows them to hide the statement using the trapdoor Σ-protocol (where the trapdoor is the scheme's secret key). The CIH then allows this trapdoor protocol to be collapsed into a single message. The lossy property allows for witness indistinguishability, and the knowledge extraction allows for extraction of the decryption key. This will get one as far as a NIWI. To then turn this into a multi-theorem NIZK, they then hide a discrete logarithm instance in the CRS in order to prove either the NIWI instance or knowledge-of-exponent. This additionally requires that the underlying protocol in the NIWI is an argument of knowledge. We refer to [41] for more details.

For our purposes, we have already proven that our underlying scheme is an argument of knowledge (as the underlying scheme is a proof of knowledge). Zero-knowledge allows us to forgo the lossy encryption scheme, but we will need a way to run our extractor (in particular, we will need a way to get the DDH exponent). Thankfully, Section B.5 supplies us a trapdoor protocol, so using $x = y_{3,1}$ suffices (where, for clarity, x is the secret exponent in Protocol B.4 and $y_{3,1}$ is the exponent of the first DDH triple in Protocol B.5). Finally, by using the CRS switching as described in [41], we are able to get a multi-theorem (and therefore robust) NIZK in the CRS model without random oracles.

Theorem 47. *Let R_1 and R_2 be as in Corollary 46, and let $R = R_1 \vee R_2$. Suppose that DDH is sub-exponentially hard for polynomial-time adversaries. Then, there exists a robust NIZK argument for L_R, where $L_R = \{x : \exists w \text{ s.t. } R(x, w) = 1\}$.*

Proof. We have already proven witness indistinguishability and extractability of our protocol by Corollary 46. So we only have left to prove that the bad challenges of the underlying Protocol B.4 belong to a sparse set, and then, further, that our extractor (and therefore the verifier as well) is in TC^0.

For the bit protocol (Section B.3) alone, this is clear. In fact, there is no bad challenge in the sense that the adversary may never commit to $\beta \notin \{0, 1\}$, as for every input $g^{a^*}, a^* \in \mathbb{Z}_q$, there is some $r^* \in \mathbb{Z}_q, \beta^* \in \{0, 1\}$ such that $g^{a^*} = g^{\beta^*} h^{r^*}$ (specifically, $r^* = (a^* - \beta^*)/y$). So, there is always some bit that the prover is committing to, though it remains to be seen if this ties into the general protocol.

For the main protocol, let us for a moment consider the warm-up, Protocol B.2. That is, $\mathbf{c}, \mathbf{d}$ sent in the first message are now $dn^{k/2}$-long vectors, rather than $(dn^{k/2} \times \log q)$ matrices across bits. The changes this implies to the rest of the protocol should be clear. Now, the adversarial prover wins if $\alpha^* = \log_g(\mathbf{d} - \mathbf{c}y)$ does not equal the coefficients of $R_{U^1,\ldots,U^k}$, but the adversary is able to choose $w(x)$ such that the verifier accepts (for now, only the first two conditions).

Let $a = [t^*]_g, b = [t'^*]_h, \mathbf{c} = [\mathbf{r}^*]_g$, and $\mathbf{d} = [\alpha^* + \mathbf{r}^* y]_g$, and let x be the challenge given to the prover by the verifier. Finally, let w^* be the response from the prover, and let $c = \prod \mathbf{c}$ and $d = \prod \mathbf{d}$. From the verifier checks, we see that

$$[w^* y]_g = \left[y t'^* + x\langle \mathbf{x}, \alpha^* \rangle + yx\langle \mathbf{x}, \mathbf{r}^* \rangle - xz\right]_g$$

and

$$[w^*]_g = \left[t^* + x \langle \mathbf{x}, \mathbf{r}^* \rangle\right]_g,$$

where $z = R_{U^1,\ldots,U^k}(x)$ is honestly calculated independently of the adversary. This naturally yields

$$[yt^* + yx\langle \mathbf{x}, \mathbf{r}^* \rangle]_g = \left[y t'^* + x\langle \mathbf{x}, \alpha^* \rangle + yx\langle \mathbf{x}, \mathbf{r}^* \rangle - xz\right]_g,$$

which is true if and only if $[yt^*]_g = [yt'^* + x\langle \mathbf{x}, \alpha^* \rangle - xz]_g$. Solving for α^*, we see that $[x\langle \mathbf{x}, \alpha^* \rangle]_g = [xz + y(t^* - t'^*)]_g$. In order for this equality to hold, the prover must have $\alpha^* - z$ be a set of coefficients for an equation of which x is a root. As the degree of this equation is at most $dn^{k/2}$, we see that there are at most $dn^{k/2} - 1$ bad challenges.

This reasoning can be extended to the first stage of the overall Protocol B.4. Compiling these two analyses together, we see there are at most $dn^{k/2} - 1$ bad challenges for any initial prover message, which is negligible in the size of the group. This proves sparsity.

Next, we must show that our extractor runs in TC^0. Note that TC^0 contains n-bit multiplication and integer division [37]. The main concern then is that we will have to be doing many sums in the exponent of groups. To get around this, we will introduce one final augmentation to our proof system. After generating the CRS, Prover and Verifier will additionally pre-compute (by whatever means) exponents of the form $[x], [2x], [4x], \ldots$. These increasing powers of the secret exponent will allow our extractor to more efficiently perform the verifier checks and other computations.

Concretely, we get that an individual proof of bit encryption may be extracted in $O(1)$ exponentiations and $O(1)$ multiplications (exact numbers may be computed by following Section B.3). In Protocol B.4, we see that each intermediate verifier check also takes a constant number of exponentiations and multiplications, as well. The final check also takes a constant number of multiplications and exponentiations. While we must also compute the $D \log q$ bit extractions, note that these may

all be run in parallel in the same depth. In total, because each exponentiation may be completed in $O(\log q)$ multiplications given the pre-computation, we have $(k-1)O(\log q)$ multiplications to account for. As k is a constant, we now must simply choose q such that $O(\log q) = O(1)$ (in the length of the input). This can be done by selecting q small enough, say, $q = o(2^{n^2})$.

Finally, we must also show that the trapdoor function extractor is in TC^0, however this follows from its similarity to the bit encryptions. □

C Other Proofs Omitted from the Main Body

C.1 Proof of Lemma 16

Proof. We start by arguing that honest parties have enough time to process and forward all valid messages received in round 2.

Claim 9. *Each honest party has enough computational power to process all the PoWs received during round 2 of the protocol.*

Proof of claim. By assumption, each honest party is able to take c computational steps per unit of time, while the adversary takes $t \cdot c$. Let θ be an upper bound on the total messages sent in each round.

We want to show that (i) honest parties have enough time to compute a PoW in round 1, and (ii) honest parties are able to process all θ messages they receive at the beginning of round 2. These conditions are described by inequalities $t_p \leq r_p \cdot c$ and $\theta t_v \leq r_v \cdot c$, respectively. It holds that:

$$r_p c \geq c t_p / c = t_p,$$

and

$$r_v c = r_p c \sigma / 2 = \frac{t_v^2 \sigma c}{2c} = t_v^2 \sigma / 2 > \frac{t_v 2\theta\sigma}{2\sigma} = \theta t_v.$$

Hence, the claim follows. ⊣

Next, we show that the above claim about the network conditions is sufficient to prove that protocol WeakConsensus achieves Weak Agreement unconditionally. For the sake of contradiction, assume that there exists two honest parties P_i, P_j such that P_i outputs 0 while P_j outputs 1. It follows that:

$$|P_i^0| > (|P_i^0| + |P_i^1| + |P_i^{\text{late}}|)/2 \Leftrightarrow |P_i^0| > |P_i^1| + |P_i^{\text{late}}|,$$

and, symmetrically, that $|P_j^1| > |P_j^0| + |P_j^{\text{late}}|$. Adding both inequalities, we have that

$$|P_j^1| + |P_i^0| > |P_i^1| + |P_i^{late}| + |P_j^0| + |P_j^{\text{late}}|. \tag{5}$$

On the other hand, by the guarantees provided by $\mathcal{F}_{\text{DIFF}}$, it should hold that

$$|P_i^0| \leq |P_j^0| + |P_j^{late}| \text{ and } |P_j^1| \leq |P_i^1| + |P_i^{\text{late}}|,$$

since by the above claim any valid PoW witness seen by P_i in the first round will be received by P_j in the second round. Adding the two inequalities, we get that

$$|P_j^1| + |P_i^0| \leq |P_i^1| + |P_i^{\text{late}}| + |P_j^0| + |P_j^{\text{late}}|,$$

which contradicts inequality 5. Thus, Weak Agreement holds unconditionally.

Next, we turn our attention to Validity. Let b be the common input of all honest parties. Let x denote the number of distinct PoW witnesses computed by the honest parties, and y the number of any other PoW witnesses produced by the adversary. By our assumption, it holds that $x > y$. For any party P_i, it should hold that $k_i \leq x + y$, which implies by our assumption that $x > k_i/2$. Given that all honest PoWs are received in the second round, it follows that all parties are going to output b. Thus, Validity follows. □

C.2 Proof of Lemma 17

Proof. We start by proving some initial claims that are going to help with our analysis. First, we show that in more than 5/6 of the WeakConsensus invocations, honest parties are going to produce close to $n - t$ distinct PoW witnesses.

Claim 10. *Given any constant $\sigma \in (0, 1)$, for a large enough λ, honest parties are going to produce more than $(1 - \sigma)(n - t)$ distinct PoWs in 5/6 of the* WeakConsensus *invocations with overwhelming probability in λ.*

Proof of claim. We start by analyzing the probability that honest parties produce enough distinct PoW witnesses in a single instance of WeakConsensus. Let $k := n - t$, be the number of honest parties and $m := 2r$, be the number of different PoW instances considered by parties running WeakConsensus. In the worst case, all parties will have the same input and will select which instance to solve among r different ones. As shown earlier, all the honest parties have enough computational steps available per round to finish computing a PoW witness.

Since each of the honest parties picks a PoW instance at random, some of them may end up picking the same one. Making r large enough would in general minimize the number of collisions among honest parties. However, in order to preserve the hardness of PoWs we are restricted in how many PoW instances should be available to the adversary, compared to how many it solves. Given that the number of instances solved by the adversary is going to be proportional to that solved by honest parties, the condition we want to satisfy is that $r/k \leq \tau(\lambda) = \lambda^{o(1)}$.

We split our analysis into two cases. In the first one, assume that $k = O(1)$ and set $r = 7k^2$. In the second one, we assume that $k = \omega(1)$, and set $r = ak$, for some large enough $a \in \mathbb{N}$. Note that in both cases it holds that $r/k = O(1) = O(\lambda^{1/\log(\lambda)}) \in \lambda^{o(1)}$.

In the first case, no collision will happen with probability at most

$$\frac{k(k-1)}{2r} \leq \frac{k^2}{7k^2} = 1/7,$$

as stated by our claim.

In the second case, we will analyze the number of collisions as a balls and bins process. Namely, we are going to use the Poisson approximation [47], where the event of interest is analyzed in a setting (the "Poisson" setting) where the load in each bin is assumed to be an independent Poisson variable with mean k/r. Results obtained in this setting can then be translated back to the "exact" setting, where the are dependencies between the loads of different bins, with some loss on probability.

The event E we care about upper-bounding is the number of full bins being greater than a target value. This event depends entirely on the number of balls in each bin. Furthermore, as the number of balls increases, $\Pr[E]$ increases. By [47], if $\Pr[E] = p$ in the Poisson setting, $\Pr[E] \leq 2p$ in the exact setting.

We proceed to bound the probability of E occurring. Assume that we have t i.i.d. discrete Poisson random variables $(X_i)_i$, each with parameter $\mu = k/t$, denoting the number of balls in the i-th bin. We have that $\Pr[X_i > 0] = 1 - e^{-\mu}$. Let random variable Y_i be equal to 1 iff $X_i > 0$, and

let $Y = \sum_{i=1}^{t} Y_i$. It then holds that $\mathbb{E}[Y_i] = 1 - e^{-\mu}$ and $\mathbb{E}[Y] = t(1 - e^{-k/r})$. For $r = ak$, for some constant $a > 2$, we have by the Chernoff bound that for any $\delta \in (0, 1)$:

$$\Pr[Y \leq (1-\delta)\mathbb{E}[Y]] = \Pr[Y \leq (1-\delta)t(1-e^{-k/r})] \leq e^{-\frac{\delta^2 ck(1-e^{-1/a})}{3}} = e^{-O(k)}.$$

Firstly, notice that since $k \in \omega(1)$, for any selection of a there exists a large enough λ such that the probability of our desired event becomes smaller than any constant, we choose in particular $1/14$. By the Poisson approximation this implies that the event in the exact setting happens with probability at most $1/7$.

Secondly, $a(1 - e^{-1/a})$ tends to 1 as a goes to infinity. Thus, for any constant ϵ, there exists a large enough constant a, such that $a(1 - e^{-1/a}) \geq 1 - \epsilon$. This easily implies that for any $\sigma \in (0, 1)$, there exists a, δ such that $(1 - \delta)t(1 - e^{-k/t}) \geq (1 - \sigma)k$. Hence, for any $\sigma \in (0, 1)$, there exists a $a \in \mathbb{N}$, such that with probability at least $6/7$ honest parties mine at least $(1 - \sigma)k$ different PoWs.

To finish proving our claim, let E_i the probability of event E happening in the i-th invocation of protocol WeakConsensus. Note that $\{E_i\}_{i\in[l]}$ is a sequence of independent events. By a standard Chernoff bound argument we can show that with overwhelming probability in λ, in less than $1/6$ of the WeakConsensus invocations honest parties will produce less than $(1 - \sigma)k$ PoWs. Thus, the claim follows. ⊣

Next, we show that in more than $5/6$ of the WeakConsensus protocol invocations the adversary is going to produce less than $n - t$ PoWs with overwhelming probability in λ.

Claim 11. *The number of PoW witnesses produced by $\mathcal{A}$ (different from those produced by the honest parties) in more than $5/6$ of the* WeakConsensus *invocations is at most $t' := (1 - \sigma/2)(n - t)$, for any $\sigma \in (0, 1)$, with overwhelming probability in λ.*

Proof. In contradiction, assume that there exists and adversary $\mathcal{A}$ such that in $1/6$ of the WeakConsensus invocations produces more than t' PoWs with non-negligible probability. We are going to use $\mathcal{A}$ to construct another adversary that breaks the security of the PoW scheme. First, we argue that if $\mathcal{A}$ produces more than t' PoWs in a WeakConsensus invocation with probability at most $\epsilon := 1/6 - \sigma$, then it will produce more than t' PoWs in less than $1/6$ of the invocations with overwhelming probability.

Let $\mathcal{T}$ be the protocol's execution tree when we fix $\mathcal{A}$. The tree has nodes $n_{i,j}$, where $n_{i,j}$ reflects the execution state just before $\mathcal{A}$ receives the i-th beacon output. Index j runs over all possible coin-flip histories up to that point. Wlog, assume that between receiving any two consecutive challenges the adversary and the honest parties perform exactly l coins flips. Thus, for any level $i \in [m]$, there are at most 2^{il} nodes, i.e., $j \in [2^{il}]$.

Next, we argue that if for every subtree defined by $n_{i,j}$ ($i \in [m]$), at most $\epsilon 2^l$ paths are successful for the adversary, in the sense that the adversary generates more than t' PoWs, then the fraction of paths with at least $m/6$ successes is negligible in λ. W.l.o.g., assume that exactly $\epsilon 2^l$ paths are successful in any such subtree. Namely, we will show that in that case there are at most $2^{ml} \cdot \mathsf{negl}(\lambda)$ paths with at least $m/6$ successes in $\mathcal{T}$.

Let $a_{i,c}$ denote the number of paths ending at some node at level $i + 1$ that have exactly c successes. By our assumptions it holds that $a_{1,0} = 2^l(1 - \epsilon), a_{1,1} = 2^l\epsilon$ for the first level, and

$$a_{n,c} = a_{n-1,c-1}2^l\epsilon + a_{n-1,c}2^l(1 - \epsilon)$$

for any subsequent level, where $a_{n,-1} = a_{n,n+1} = 0$. The equalities follow by the fact that, at every node, $2^l\epsilon$ of the paths are going to increase their successes by one, and the rest are going to retain

the same value of successes. It is easy to see that the solution of this recursion is

$$a_{n,c} = 2^{nl}\binom{n}{c}\epsilon^c(1-\epsilon)^{n-c}.$$

We are interested in bounding the sum $\sum_{i=m/6}^{m} a_{m,i}$. Note that for any i, $r_i = a_{m,i}/2^{m\cdot l}$ is equal to the probability of i successes in m independent Bernoulli trials, where each trial succeeds with probability ϵ. Thus, we can use the Chernoff bound to upper-bound the probability that $\sum_{i=m/6}^{m} r_i$ is less than $m/6$ by

$$e^{-((1-6\epsilon)/(6\epsilon))^2\epsilon m/3} \leq e^{-\sigma^2 m/108\epsilon} \leq \lambda^{-\Omega(\log(\lambda))} \leq \mathsf{negl}(\lambda),$$

where we have used the fact that $m = \log^2(\lambda)$. Hence, $\sum_{i=m/2}^{m} a_{m,i} \leq 2^{ml}\mathsf{negl}(\lambda)$, as we have claimed.

Therefore, since we have assumed that $\mathcal{A}$ produces more than t' PoWs in at least 1/6 of the WeakConsensus invocations with non-negligible probability, by the analysis above it must be the case that there exists an $n_{i,j}$ where the adversary computes more than t' PoWs with probability greater than ϵ. We are going to use the state of node $n_{i,j}$ to construct an adversary $\mathcal{A}'$ that contradicts our assumption about the PoW scheme being secure.

Let $2r$ be the size of the output of $\mathcal{F}_{\text{BEACON}}$. $\mathcal{A}'$ works as follows: It takes as input a sequence of $2r$ PoW instances $(x_i)_i$ and some non-uniform advice. We choose the advice to contain $n-t$ randomly sampled PoW instances $(x'_i, w'_i)_i$ together with their respective witnesses. $\mathcal{A}'$ is going to construct a "fake" beacon output for $\mathcal{A}$. Given the input bits of honest parties at node $n_{i,j}$, it is going to replace randomly selected PoW instances from $(x'_i)_i$ with instances from $(x'_i)_i$. Specifically, if an honest party has input 0 (resp. 1), then a randomly selected instance from the first half (resp. second half) of $(x_i)_i$ is replaced. Note, that this process preserves the possibility that two honest parties choose to solve the same PoW instance, thus perfectly mimicking the real world. We denote by $Z = (z_i)_i$ the resulting sequence of instances.

Next, $\mathcal{A}'$ is going to initialize $\mathcal{A}$ to the state described by $n_{i,j}$, and provide Z as the output of the beacon. At the end of the first round, it is going to send $\mathcal{A}$ the witnesses $(w'_i)_i$ that it got as advice, simulating the behavior of the honest parties. Then, it is going to verify the messages $\mathcal{A}$ sent in the first round, and forward any valid PoWs it produces. Finally, it is going to verify the messages $\mathcal{A}$ sent in the second round. Finally, $\mathcal{A}'$ outputs any PoW witnesses produced by $\mathcal{A}$ that do not correspond to the pre-solved instances it has planted in Z.

We will first analyze the running time of $\mathcal{A}'$. As before, let $t_v \geq 2\theta/\sigma$, $t_p = t_v^2$, $r_p := t_p/c$ and $r_v := r_p \cdot \sigma/2$. $\mathcal{A}$ takes a total of $(r_v + r_p)tc$ steps. In addition, $\mathcal{A}'$ takes an additional $2t_v\theta$ steps in order to verify messages sent by $\mathcal{A}$ in the first and second rounds. Hence, $\mathsf{Steps}_{\mathcal{A}'} \leq (r_v+r_p)tc+2t_v\theta$. Now, for $\mathcal{A}'$ to be breaking PoW's security it must be that $\mathsf{Steps}_{\mathcal{A}'} < \gamma(t_p)t' = \gamma(t_p)(1-\sigma/2)(n-t)$. It holds that:

$$\begin{aligned}\mathsf{Steps}_{\mathcal{A}'} \leq (r_v + r_p)tc + 2t_v\theta &\leq \frac{t_p(1+\sigma/2)tc}{c} + 2t_v\theta \\ &\leq t_p(1+\sigma/2)t + t_v^2\sigma \\ &\leq t_p((1+\sigma/2)t + \sigma).\end{aligned}$$

On the other hand, by our assumption about the number of corruptions, we have that:

$$\begin{aligned}(1-\sigma)(n-t)\gamma(\lambda^2)/\lambda^2 - \sigma > t &\Rightarrow \frac{(1-\sigma/2)}{(1+\sigma/2)}(n-t)\gamma(\lambda^2)/\lambda^2 - \sigma > t \\ &\Leftrightarrow (1+\sigma/2)(t+\sigma)t_p < \gamma(\lambda^2)(1-\sigma/2)(n-t) \\ &\Rightarrow t_p((1+\sigma/2)t+\sigma) < \gamma(\lambda^2)(1-\sigma/2)(n-t).\end{aligned}$$

Combing the two inequalities we get our desired relation about the running time of $\mathcal{A}'$.

Next, we analyze the success probability of $\mathcal{A}'$. First, notice that the execution in the eyes of $\mathcal{A}$ is indistinguishable in the reduction and in the actual protocol, as honest parties are perfectly simulated. Thus, by our assumption, $\mathcal{A}$ is going to produce t' PoWs different from the ones produced by the honest parties with probability at least ϵ. This further implies that probability ϵ, $\mathcal{A}'$ is going to solve t' instances from $(x_i)_i$ in a total of $t' \cdot \gamma(t_p)$ steps. Since $\epsilon \in \Omega(1) > \lambda^{-o(1)}$, this is a contradiction to our initial assumption about the hardness of PoW, and thus, in more than 5/6 of the WeakConsensus invocations, the adversary is going to produce at most t' PoWs with overwhelming probability in λ. ⊣

Combining the above two claims we easily get that in more than 2/3 of the WeakConsensus invocations honest parties will produce more PoWs than the adversary with overwhelming probability. This fact will be sufficient to prove our lemma.

First, we argue that Validity holds. For the sake of contradiction, assume that all parties have the same input b and there exists an honest party that outputs $y_i \neq b$. Due to Validity being satisfied by Protocol WeakConsensus when honest parties produce more PoWs than the adversary, it follows that t_i^b must be greater than $2l/3$. Thus, Validity follows.

Regarding Weak Agreement, for the sake of contradiction, assume that there exist honest parties P_i, P_j that output $y_i = 0, y_j = 1$, respectively. Since $y_i = 0$, it should hold that $t_i^0 > 2l/3$. By Weak Agreement of protocol WeakConsensus and the fact that in less than 1/3 of the WeakConsensus invocations $\mathcal{A}$ may produce as many PoWs as the honest parties, it follows that $t_j^1 < 2l/3$. This is a contradiction and the lemma follows. □

C.3 Proof of Lemma 19

Proof. As in the case of protocol WeakConsensus, it also holds here that honest parties have sufficient time to process any valid messages they receive. This is sufficient to show that protocol GradedConsensus achieves Weak Graded Agreement unconditionally. For the sake of contradiction, assume that there exists two honest parties P_i, P_j such that P_i outputs (wlog) $(0, 1)$ while P_j outputs $(1, 0)$. It follows that:

$$|P_i^0| > (|P_i^0| + |P_i^1| + |P_i^\perp| + |P_i^{\text{late}}|)/2 \Leftrightarrow |P_i^0| > |P_i^1| + |P_i^\perp| + |P_i^{\text{late}}|.$$

Similarly, we have that $|P_j^1| + |P_j^\perp| > |P_j^0| + |P_j^{\text{late}}|$. Adding both inequalities:

$$|P_j^1| + |P_j^\perp| + |P_i^0| > |P_i^1| + |P_i^\perp| + |P_i^{\text{late}}| + |P_j^0| + |P_j^{\text{late}}|. \tag{6}$$

On the other hand, by the guarantees provided by $\mathcal{F}_{\text{DIFF}}$, it should hold that

$$|P_i^0| \leq |P_j^0| + |P_j^{\text{late}}| \text{ and } |P_j^1| + |P_j^\perp| \leq |P_i^1| + |P_i^\perp| + |P_i^{\text{late}}|,$$

since any valid PoW witness seen by P_i in the first round, will be received by P_j in the second round. Adding the two inequalities, we obviously get a contradiction to Inequality 6. Thus, Weak Graded Agreement holds unconditionally.

Next, we focus on Graded Agreement. Let x denote the number of distinct PoW witnesses computed by the honest parties, and y the number of any other PoW witnesses produced by the adversary. By our assumption, it should hold that $x > y$. For any party P_i, it should hold that $k_i = x + y$, which implies by our assumption that $x > k_i/2$. For the sake of contradiction, assume that there exists two honest parties P_i, P_j such that P_i outputs (wlog) $(0, 1)$ while P_j outputs $y_j \neq 0$. As before, we have that $|P_i^0| > |P_i^1| + |P_i^\perp| + |P_i^{\text{late}}|$. By the weak agreement of the inputs of honest

parties, and the fact that honest parties solve in total more PoWs that the adversary, it easily follows that there exists an honest party with input 0, and thus no honest party has input 1. Otherwise,

$$|P_i^0| \leq y < x \leq |P_i^1| + |P_i^\perp|,$$

which is a contradiction. It thus follows that

$$|P_j^0| + |P_j^\perp| \geq x > k_j/2,$$

and y_j must be 0, which is a contradiction to our initial hypothesis.

Finally, we turn our attention to Validity. Let b be the common input of all honest parties. Given that all honest PoWs are received in the round 2 of the protocol and that $x > k_i/2$, for any party P_i, it follows that all parties will output b. Thus, Validity follows. □

C.4 Proof of Lemma 20

Proof. First, by Lemma 17, protocol AmpedWeakConsensus achieves Weak Consensus with overwhelming probability. This implies that the precondition about the input of protocol GradedConsensus will be satisfied. Second, in the same way as in Lemma 17, we can show that in more than 2/3 of the GradedConsensus invocations the honest parties will produce more PoWs than the adversary with overwhelming probability. Hence, by Lemma 19, in more than 2/3 of these invocations, protocol GradedConsensus achieves Graded Consensus. This fact suffices to prove the lemma.

We first argue that Validity holds. For the sake of contradiction, assume that all parties have the same input b and there exists an honest party P_i that outputs $y_i \neq b$. By the previous observation about protocol GradedConsensus, it follows that $t_i^{b,1}$ must be greater than $2l/3$, which is a contradiction. Thus, Validity follows.

Next, we argue why Graded Agreement holds. For the sake of contradiction, assume that there exist honest parties P_i, P_j that output $(y_i = 0, g_i = 1)$, and $y_j = 1$ or $y_j = \perp$. Since $y_i = 0, g_i = 1$, it should hold that $t_i^{0,1} > 2l/3$. By Weak Graded Agreement holding unconditionally, it follows that $t_j^{0,1} + t_j^{0,0} + t_j^{\perp,1} + t_j^{\perp,0} > 2l/3$. On the other hand, since Graded Agreement holds in more than $2l/3$ of the GradedConsensus invocations, it follows that $t_j^{1,1} + t_j^{1,0} < l/3$. Thus, P_j is going to output either $(0, 0)$ or $(0, 1)$. This is a contradiction and the lemma follows. □

C.5 Proof of Theorem 21

Proof. We start by analyzing a single iteration of the protocol. First, note that if all honest parties have the same input b, due to Lemma 20 and Graded Validity, they are all going to output $(b, 1)$ with overwhelming probability. Thus, at the end of this iteration they are all going to set $y_i := b$. Further, the protocol preserves Validity across iterations: once parties agree, this state persists. Thus, Validity follows.

Next, we focus on Agreement. First, we show that the probability that all parties agree at the end of an iteration is at least 1/2. We split the analysis into two cases based on the grades that are output by AmpedGradedConsensus: (1) no party outputs $g_i = 1$, and (2) at least one party outputs $g_i = 1$. In case (1), all parties set $y_i = b'_R$, and thus they reach Agreement in this iteration. In case (2), since at least one party has output $b_i \in \{0, 1\}$,$g_i = 1$, by Graded Agreement it follows that no party P_j has output $b_j \neq b_i$ and $g_j = 1$. Thus, all parties with a different b_j than b_i are going to output b'_R. Given that the adversary learns b'_R *after* the AmpedGradedConsensus invocation

finishes, its actions are independent from b'_R. Since with probability at least $1/2$, $b'_R = b_i$, it follows that with probability at least $1/2$ all parties reach agreement in this iteration.

Next, we analyze Agreement in the full protocol. Since in each iteration there is probability at least $1/2$ of all parties agreeing, and the events of interest are independent, the probability that parties have not agreed in at least one round is at most $(1-1/2)^l = 2^{-\log^2(\lambda)} = \mathsf{negl}(\lambda)$. Moreover, in case they agree in one iteration, as argued earlier, Agreement persists. Thus, Agreement is achieved with overwhelming probability. □

D Consensus from NIZK-PoW and a Beacon with $O(\lambda^2)$ Output

We have shown how to achieve consensus in the presence of a beacon whose output length is proportional to the number of parties in Section 3. In this subsection, we show how to use the Seeded NIZK-PoW construction developed in the previous sections, to relax the assumption regarding the size of the output of the beacon. Namely, we show how to achieve Consensus with a beacon that has an output whose size is independent of the number of parties, i.e., it produces $O(\lambda^2)$ bits each time. Note, that such a beacon is strictly weaker than a beacon that produces $O(poly(\lambda))$ outputs each round, as by the time $poly(\lambda)$ bits will be generated by the "short" output beacon some of the generated randomness will be fairly old, essentially allowing the adversary to learn part of the output a lot earlier than honest parties.

Our construction is quite similar to the one extensively presented earlier in Sections 3.2, 3.3, and 3.4, hence here we only describe the necessary modifications to these protocols. Firstly, we are going to interpret beacon outputs as consisting of a NIZK-PoW seed (implying the same number of compressed instances as in the original protocols). Instead of parties selecting a random PoW from the instance sequence to solve, in the modified protocols they are going to generate and solve the related PoW instance implied by the seed. Finally, instead of PoW witnesses, parties are going to generate NIZK-PoW proofs (and later verify them in the respective steps of the protocols).

The protocol described above is thus sufficient to prove Theorem **??** in Section 3. We only provide a sketch of the proof as it mostly follows that presented in the previous sections.

Theorem 48. *For $k \geq 2$, suppose $k\mathsf{OV}$ takes $\lambda^{k-o(1)}$ time to decide for all but finitely many inputs lengths for any $d = \omega(\log \lambda)$, and that DDH is sub-exponentially hard. Then, assuming that for some $\epsilon > 0$,*

$$\frac{\textit{total honest power}}{\textit{total adversarial power}} \equiv \frac{(n-t)\cdot c}{t \cdot c} > \lambda^{\epsilon}$$

and the existence of a randomness beacon with output size $O(\lambda^2)$, there exists a protocol that achieves Consensus in the permissionless setting with overwhelming probability in λ.

Sketch. The security analysis of the respective protocols is exactly the same, except of the reduction presented in Lemma 17. Note, that we cannot simulate honest PoWs by witnesses coming from the advice string, as PoW instances are all generated from the same seed and are thus correlated. Instead, we have to resort to the Zero-Knowledge property of the NIZK-PoW and simulate honest proofs. This does not change our main argument, as (i) the simulated proofs are indistinguishable from honestly produced ones, and (ii) extractability still holds in the presence of simulated proofs. The running time of the reduction is slightly increased, due to the need to run the NIZK-PoW simulator and extractor. Note, that the running time of these algorithms is extremely efficient compared to the computing NIZK-PoW, thus our result is essentially unaffected. □