Permissionless Consensus from a Common Random String*

Damiano Abram[1][0009−0004−3916−7550], Marshall Ball[2][0000−0002−4236−3710], Juan Garay[3][0000−0003−0366−7110], and Aggelos Kiayias[4][0000−0003−2451−1430]

[1] The University of Edinburgh. abram.damiano@protonmail.com
[2] New York University, marshall.ball@cs.nyu.edu
[3] Texas A&M University, garay@cse.tamu.edu
[4] The University of Edinburgh and IOG, aggelos.kiayias@ed.ac.uk

Abstract. Permissionless consensus enables parties to perform Byzantine agreement without any *a priori* knowledge about who is participating, except for an upper bound on the number of participants running the protocol (no PKI, etc.). Since Nakamoto's Bitcoin paper, it has been widely believed that permissionless consensus is feasible provided the (Byzantine) adversary only controls a fraction of the collective computational power. However, all known protocols, including Nakamoto's, rely on idealized assumptions (or *ad hoc* instantiations).

Is permissionless consensus possible without such assumptions? Surprising little progress had been made towards solving this open question until the recent result by Ball *et al.* (Crypto 2024), which showed how to achieve permissionless consensus from proofs of work (PoWs) based on fine-grained complexity assumptions in a setting where a randomness beacon is available to all parties running the protocol. Their work left open whether it is possible to remove the beacon assumption; this question is the focus of our work, which we resolve via a new consensus protocol construction that relies on a novel class of *distributed samplers* and a common random string (that does not need to be structured or sampled precisely at the onset of the protocol execution).

To prove our protocol secure, we revisit the concept of distributed samplers and adapt it to a setting where multiple sampler executions need to be simultaneously secure. To address this challenge we introduce the primitive we call *d-wise independent distributed samplers* and put forward constructions for such samplers based on DDH and LWE. We then present our consensus protocol via a modular design that utilizes a new moderately hard cryptographic primitive we call *multi-verifier signatures of work*, a sort of "time-based signature" we construct by composing distributed samplers and (fine-grained complexity-based) PoWs, and which may be of independent interest.

1 Introduction

The permissionless consensus problem has attracted significant attention since the introduction of the Bitcoin blockchain [Nak08]. It asks whether it is possible for an set of parties connected via an unauthenticated gossip network (a.k.a. a "diffusion" channel) to reach consensus. While the original Bitcoin protocol aims at a continuously operating protocol (a.k.a. *state machine replication* (SMR) [Sch90]) the "single-shot" case (a.k.a. Byzantine agreement [PSL80,LSP82]) is its core agreement problem [GKL15,LPR24] This form of permissionless consensus is defined as follows (cf. [GK20]): A set of parties without preassigned identities, authenticated channels or even knowledge of each other, each one with their own input, wish to arrive at a common output so that the three properties of consensus hold: Agreement, Validity and Termination. Importantly, the parties wish to do so in the presence of an adversary that controls a minority of the parties and may behave in a *Byzantine* manner with the objective to violate any one of the three properties.

A number of works have analyzed the Bitcoin blockchain that operates in this permissionless setting (e.g., [GKL15,PSS17]), and made the connection to permissionless consensus as defined above. Such protocols assume the existence of an unpredictable beacon/string value that is available at the onset of the protocol

* The abridged version of this paper appears in *Proc. Crypto '26.*

operation. Alternative protocols have also been proposed that operate under more relaxed assumptions about whether such an unpredictable beacon is available at the onset [AD15,GKLP18]. A common downside of all such protocol designs and analyses is that they rely non-trivially on the random oracle model [BR93], leaving open the question whether permissionless consensus in the standard model is at all feasible.

Before delving into our work on this question, let us briefly motivate it. While the random oracle heuristic is widely used in practice, it can lead to a false sense of security. A recent high-profile example of this phenomenon is seen in the work of Khovratovich, Rothblum and Soukhanov [KRS25] which showed certain widely-deployed proof-systems that were analyzed under the random oracle heuristic were, in fact, unsound. While we do not know any such counterexamples for permissionless consensus, even for contrived protocols, that does not mean that they are secure. To be clear, the immediate goal of this work is not practical, secure permissionless consensus. Instead, we believe that non-heuristic provable security, even for impractical protocols, can teach us something about the nature of security in this type of protocols.

In order to tackle the question of the feasibility of permissionless consensus, our starting point is the first provably secure permissionless consensus protocol that was constructed outside of the random oracle model. This protocol, designed by Ball *et al.* [BGH$^+$24a], leveraged fine-grained cryptography in the *random beacon model*. This setting is characterized by an abstract entity, called *beacon*, who regularly and in a timely fashion reveals random values to all protocol participants. The protocol in [BGH$^+$24a] translates the popular consensus blueprint of going from weaker forms of consensus to full consensus (cf. [FM97]) to the permissionless setting via repetitions, needed in their case in order to achieve security amplification. Such weaker forms of consensus are designed in a similar way to how we would build them in the permissioned setting: The protocol participants engage in several voting phases, where votes are first cast, then relayed, and finally tallied. Arguably, the most interesting technique in [BGH$^+$24a] is how the authors managed to *authenticate* the votes and *prevent Sybil attacks* without relying on a PKI. This is where fine-grained cryptography enters the picture: Casting a vote requires work via solving a moderately hard puzzle (formally, a proof-of-work challenge) that has been revealed by the beacon precisely at the beginning of the voting phase. The honest parties can then verify the solutions to these puzzles and discard any vote whose puzzle has not been solved. In other words, an adversary that only controls a fraction of the computational resources in the system can only cast a (suitably related) fraction of all valid votes.

By moving to a model without a beacon, the main issue we encounter is that the parties can no longer agree on hard-to-solve puzzles. Consider the perspective of an honest party that receives a vote associated with the solution of a puzzle. How can the party be sure that who casted the vote has performed work *after the beginning of the voting phase*? Maybe the puzzle was generated with an already known solution; or maybe the solution was preprocessed a long time before the beginning of the voting phase.

A crucial observation here is that perhaps we do not need **all** the parties to agree on the puzzle challenges. It may be enough that whoever verifies the ballot has the guarantee that the solved puzzle looked sufficiently random at the beginning of the voting phase. Similar ideas have been explored before in the random-oracle setting, first in [AD15] and later in [GKLP18], but can they also apply to the standard model?

Our approach. To reach our objective we employ the concept of *distributed samplers* [ASY22,AWZ23]. Recall that this primitive allows a set of parties to securely generate a public sample from a distribution $\mathcal{D}(1^\lambda)$ using a single round of simultaneous communication. As long as at least one of the participants is honest, the output of the distributed sampler has essentially the same properties as a random sample from the underlying distribution: If $\mathcal{D}(1^\lambda)$ generates a random integer, deriving the factorization of the distributed sampler output remains intractable; if $\mathcal{D}(1^\lambda)$ samples random puzzles, solving the puzzle produced by the distributed sampler still requires a lot of work.

At a high level, using the above idea, we can precede any consensus voting phase by requiring all parties to send a distributed-sampler message over the gossip network. Subsequently, to send a vote, the parties solve the puzzle produced by pooling together the distributed sampler messages they received. An honest party accepts a message if it is associated with a correctly solved puzzle that was generated using the verifier's distributed sampler message in combination with an arbitrary set of other messages. In this fashion, the number of votes that may be cast by the adversary are in proportion to its computational power.

Our approach, however, is complicated by the fact that the outputs of the distributed sampler need to be secure even if they are generated *reusing* the same honest parties' messages across multiple sampler instances. In essence, we would like that the outputs of multiple (say, t, corresponding to the number of corrupted parties) distributed sampler executions, which, without loss of generality, share a single honest message (i.e., the verifier's), are all simultaneously secure. It must not be the case that solving the puzzle produced in a single execution allows the adversary to immediately recover the solution for the puzzles generated in any other (possibly correlated) distributed sampler executions.

Previous works on distributed samplers [ASY22,AWZ23] restricted their analysis to *single executions* of the primitive for a good reason: Although it might be perfectly feasible that the outputs of all known constructions of distributed samplers are secure even when the honest parties' messages are reused, the current state of techniques in cryptography leaves little hope for a mathematical proof of security. Suppose that d is an integer bigger than the size of a single distributed-sampler message. How can we relate the security of the samples produced in d correlated distributed sampler executions—which reuse the message of the only honest party—to the security of d random samples from $\mathcal{D}(1^\lambda)$? In the first case, the d outputs will never look random: All the entropy in the adversary's view comes from the message of the only honest party, but this is smaller than d.

In this work, we address this problem by constructing what we call a ***d-wise independent distributed sampler***. We do not achieve provable security for an arbitrary number of correlated distributed sampler executions; we achieve it only for at most d of them, where d is a parameter. More formally, we augment the output derivation algorithm to take as input a label. Then, if we consider any d distributed sampler executions with distinct labels, but potentially reusing the same honest-party messages, the outputs are guaranteed to retain the same security properties as d samples from $\mathcal{D}(1^\lambda)$. To summarize, to solve $t \leq d$ puzzles that are generated using (potentially correlated) distributed sampler executions with distinct labels, the adversary requires work proportional to t times the cost of a single puzzle. If $t > d$, this could in principle be false, but for sure the adversary performed work proportional to d puzzle resolutions.

On the negative side, the approach we just described has two consequences we will need to live with. The first one is that we need to set a polynomial cap on the total number of participants, which we will set to d. Interestingly, for our intended application of permissionless consensus, this is required anyway as such upper bound is necessary for suitably calibrating the PoW difficulty. The second negative consequence is that the size of the distributed sampler messages *will unavoidably grow with* d, as the reduction should hide in the message information about all d challenges generated from $\mathcal{D}(1^\lambda)$, which have entropy proportional to d. Solving this second issue seems to require the existence of a strengthening of a PoW notion developed in Ball *et al.* [BGH$^+$24a]: Seeded Proofs of Work. Essentially, that would allow us to generate simultaneously hard PoW challenges by expanding a small public seed in a PRF fashion, provided that the inputs have sufficiently high entropy. We are aware of no examples of such primitive, with the necessary parameters, aside from the trivial construction in the random oracle model.

A game-based security definition for distributed samplers. Next, let us investigate how to formalize the security guarantees of distributed samplers. One would perhaps imagine that this is done using a simulation-based security definition, as for most MPC protocols. Alas, the one-round nature of distributed samplers complicates this approach, especially outside of the semi-honest model. This claim was formalized in the work of Abram *et al.* [AFOS25]: Simulation-based security definitions for active security are plagued by strong impossibilities that seem circumventable only by relying on non-black-box simulation — to this day, a poorly understood world. That is why, in [AWZ23], Abram, Waters and Zhandry introduced *hardness-preserving* distributed samplers via a game-based definition.

Hardness-preserving distributed samplers are defined following the real-world/ ideal-world paradigm. In the real world, the game reproduces an execution of the distributed sampler: The adversary is provided with an honestly generated message and is asked to select the messages for the other participants. Then, the game provides the result of the sampler as a challenge to the adversary. In the ideal world, on the other hand, the challenge provided by the game is just a random sample from $\mathcal{D}(1^\lambda)$, while the honest-party message is generated by a simulator on input the challenge.

Crucially, we do not ask, as one might expect, that the worlds are indistinguishable — ***clearly, they are not!*** Indeed, in the ideal world, it is likely that the challenge differs from the sampler's output (if that was not the case, the adversary would notice that, by changing the messages of the corrupted parties, and the output would remain unchanged). Instead, we ask that, for every efficient adversary $\mathcal{A}$, the probability of outputting 1 in the real world $p_{\mathsf{real}}(\mathcal{A})$ is bounded from above by an affine function of the probability of outputting 1 in the ideal world $p_{\mathsf{ideal}}(\mathcal{A})$. Concretely,

$$p_{\mathsf{real}}(\mathcal{A}) \leq \eta \cdot (p_{\mathsf{ideal}}(\mathcal{A}) + \nu),$$

where η and ν are fixed functions that characterize the construction.

To see why this definition is meaningful, suppose that $\mathcal{D}(1^\lambda)$ outputs random puzzles, and $\mathcal{A}$ outputs 1 only when it manages to solve the puzzle it is given as a challenge in a short amount of time. The probability ϵ that $\mathcal{A}$ outputs 1 in the ideal world execution is negligible: The challenge it received is random, and solving a random puzzle requires time (we can assume more than the joint cost of running $\mathcal{A}$ and the distributed sampler simulator). So, the probability that the adversary solves the puzzle output by the distributed sampler in a short time is at most $\eta \cdot (\epsilon + \nu)$. Assuming that $\epsilon, \nu \ll \eta^{-1}$, we have the guarantee that the distributed sampler outputs puzzles that are hard to solve.

In this work, we adapt the definition of hardness-preserving distributed sampler to account for d-wise independence: The honest message provided at the beginning of the game will be reused across d (differently labelled) executions where the adversary has control of the labels and of all other messages. Then, the adversary will receive the challenge, which consists of either the outputs of the d distributed sampler executions (real world), or the d random samples from $\mathcal{D}(1^\lambda)$, which were given as input to the simulator (ideal world).

PoWs with exponential direct product amplification. The modified security definition introduces new challenges: If we want $\eta \cdot \nu \leq 1/2$, it must be that $\eta = 2^{\Omega(d)}$. To see that, consider the adversary $\mathcal{A}$ that outputs 1 only when the challenge it receives coincides with the outputs of the d distributed sampler executions. It is easy to see that $p_{\mathsf{real}}(\mathcal{A}) = 1$. However, given that the honestly generated message cannot fix the output of any of the d executions, we have that $p_{\mathsf{ideal}}(\mathcal{A}) \leq 1/2^d$. Since $p_{\mathsf{real}}(\mathcal{A}) \leq \eta \cdot (p_{\mathsf{ideal}}(\mathcal{A}) + \nu)$, the parameter η must grow exponentially in d.

In conclusion, to ensure the hardness of the puzzles output by the distributed sampler, we need $\epsilon \ll \eta^{-1}$, where ϵ is the probability that the adversary solves d *random* puzzles in time significantly smaller than d times the cost of a single puzzle. We therefore need that $\epsilon \leq 2^{-\Omega(d)}$. Notice that not all proofs of work satisfy this property. Moreover, we cannot even hope to rely on *complexity leveraging*: If we select the security parameter of the PoW proportionally to d, also the time required to solve a *single* puzzle grows proportionally to d. That would mean the work performed by each honest party to cast a vote would blow up by a multiplicative factor proportional to the total number of participants. The existence of a PoW with the properties we desire is nevertheless feasible. Indeed, observe that the candidate *ad hoc* constructions that are currently in use are likely to satisfy our needs.

Multi-Verifier Signatures of Work. The tool we construct by composing distributed samplers and PoWs can be viewed as some sort of "time-based signature" with interesting multiparty properties: Verification keys correspond to the distributed sampler messages; signatures correspond to solutions of puzzles produced by the distributed sampler; and the signed messages correspond to the labels that identify the distributed sampler executions.

The scheme is fine-grain in the sense that, if one is willing to invest enough computational power, it is always possible to forge a signature; the operation, however, is guaranteed to require a significant use of resources. Further, forging $t \leq d$ signatures would require spending t times the amount of resources required for a single forgery. Thus, these signatures signify an investment of resources in the underlying messages—i.e., some sort of *signature of work*[5].

[5] We point out that a homonymous and seemingly related primitive was introduced in [GKP20b]. While at a high level the overall objective was similar—i.e., to "certify" that enough effort has been invested—the primitive in [GKP20b] lacks the multi-verifier aspect and was constructed in the random oracle model.

Our construction, however, offers something even stronger: If we want to prove that we invested resources on a message to L parties, we do not need to generate L different signatures of work, but a **single one**. The advantage is clear: Instead of spending a significant amount of resources L times, we do it only once. That is why we call this primitive a ***multi-verifier signature of work*** (MV-SoW)).

From MV-SoW*s to Partial-PKI to Consensus.* A simple yet powerful objective for our purpose is to setup a PKI based on MV-SoWs: Parties "self-certify" their public-keys by issuing a signature of work. This approach to consensus in the permissionless setting was first informally described in [AJK05] and later on formalized and proved in [AD15] assuming an idealized model (i.e., random oracles). A particular challenge to achieving consensus in this fashion is the fact that the PKI is *partial* and inconsistent: Honest parties do not have the same view of the public-key directory. In [AD15], this was solved by showing how to endow the partial PKI keys with numerical ranks and subsequently take advantage of the fact that the Dolev-Strong protocol [DS83] can safely use ranked keys.

This approach, however, does not work for us as it fundamentally relies on the random oracle; nevertheless, it is still possible to achieve consensus via a different pathway: A combination of graded consensus [FM97] and an adaptation of the phase-king consensus protocol [BGP89] that only relies on a ***partial*** PKI. The effective resilience of the resulting consensus protocol is $\beta\gamma$, where β is the "slowdown" factor imposed on adversarial running time due to the fine-grained complexity assumption, and γ is the threshold imposed due to the distributed setting. Our protocol features $\gamma \approx 1/4$, while $\lim_{\lambda\to\infty} \beta = 0$, where λ is the security parameter. We also prove that tolerating a $1/4$ (or greater) threshold is infeasible in the partial-PKI setting for "simple" protocols where parties ignore messages referring to keys they do not recognize.

Organization of the paper. In Section 2 we put forward the notion of d-wise independent distributed samplers, starting with a technical overview, followed by constructions and their analysis. Multi-verifier signatures of work are introduced and realized in Section 3, while our permissionless consensus protocol, including how to setup a partial PKI from signatures of work, is presented in Section 4. We conclude in Section 5 with open questions and directions for future work. For the sake of readability, some of the background definitions and building blocks, as well as some of the proofs appear in the appendix (App. A and B, resp.).

2 d-Wise Independent Distributed Samplers

Recall that a distributed sampler is a primitive that allows n parties to generate a random-looking sample from a distribution $\mathcal{D}(1^\lambda)$ using a single round of simultaneous communication. We strengthen the definition of the primitive used in previous work by Abram *et al.* [ASY22]: While the original definition fixed the number of participants n, we now remove such a constrain; moreover, we augment the procedure for the derivation of the output with an auxiliary input, consisting of a label x. Different labels will correspond to independent-looking outputs.

This is a crucial improvement. The constructions in [ASY22] and [AWZ23] guaranteed the security of the output produced by a *single* distributed sampler execution. However, if we considered multiple executions of the protocol that reused the same subset of honest messages, the sequence of outputs would not necessarily have the same security properties as a sequence of independent samples from $\mathcal{D}(1^\lambda)$. The constructions presented in this work, on the other hand, guarantee some form of security even in this setting, provided that the executions adopt different labels. Specifically, we opt for a d-wise-independence form of security for a polynomial d: The outputs of $t \leq d$ executions, all reusing a subset of the honest messages but labeled differently, retain similar security properties to those of a sample from $\mathcal{D}^t(1^\lambda)$. We first present some notation and a high-level overview.

2.1 Technical Overview

We denote the security parameter by λ. We use $\mathsf{poly}(\lambda)$ to denote a generic polynomial in the security parameter. Similarly, we use $\mathsf{negl}(\lambda)$ to denote a generic negligible function of the security parameter, i.e.,

$\mathsf{negl}(\lambda) = 2^{-\omega(\log \lambda)}$. Given two function of the security parameter $a(\lambda)$ and $b(\lambda)$, we use $a \ll b$ to mean that $a/b \leq \mathsf{negl}(\lambda)$. For any positive integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. Similarly, we use $[a, b[$ to denote the set of integers that are greater or equal to a and strictly smaller than b. Given a set S, we use $|S|$ to denote its size.

We adopt the convention in which vectors are denoted using lowercase bold font (e.g., $\mathbf{v}$), while matrices are represented using uppercase bold font (e.g., $\mathbf{M}$). We use the convention in which vectors are represented using columns. We denote row vectors using the transpose symbol $^\intercal$, e.g., $\mathbf{v}^\intercal$. Given an integer value v and a modulus p, we use $\lceil v \rfloor_p$ to denote the element of $\mathbb{Z}_p$ obtained by rounding v to the closest multiple of p, i.e.,

$$\left| v - \lceil v \rfloor_p \cdot p \right| \in \left[-\frac{p}{2}, \frac{p}{2} \right[.$$

We extend this notation to vectors by rounding entry by entry. Given a scalar $v \in \mathbb{Z}_q$, we use $\mathbf{G}^{-1}(v)$ to denote its bit decomposition, i.e., the binary vector $(v_0, \ldots, v_{\lceil \log q \rceil - 1})$ such that

$$v_0 + v_1 \cdot 2 + \cdots + v_{\lceil \log q \rceil - 1} \cdot 2^{\lceil \log q \rceil - 1} = v.$$

Given a group element $g \in G$, we use $\langle g \rangle$ to denote the cyclic subgroup generated by g, i.e., $\langle g \rangle = \{g^i | i \in \mathbb{Z}\}$. If H is a subgroup of G, we denote the coset of H in G containing g by $gH = \{g \cdot h | h \in H\}$.

We now explain in more detail how we manage to construct d-wise independent, hardness-preserving distributed samplers with the properties we described. Note that previous constructions of distributed samplers [ASY22,AWZ23] generated samples from generic distributions $\mathcal{D}(1^\lambda)$ and, for this reason, required the use of *indistinguishability obfuscation* [BGI+01,GGH+13,JLS21]. Our application, however, requires the distributed sampler to simply produce uniformly random strings. We therefore are able to build the tools we need simply from DDH or from LWE (with subexponential modulus-to-noise ratio).

Lossy distributed samplers. For the moment, imagine that $d = 1$, i.e., we are just interested in a distributed sampler that is secure as long as the messages are not reused. Suppose that we want it to generate a random binary string of length ℓ. Recall that in our application, the adversary would try to perform an attack on this string (e.g., try to solve the puzzle it represents in a short amount of time). Let ϵ be an upper bound on the success probability of the attack when the string is uniformly random. Observe that, typically, we have that $\epsilon \geq 2^{-\ell}$.

Now, consider a hardness-preserving distributed sampler. We have implicitly shown above that if we want $\nu \ll \eta^{-1}$, it must be that $\eta = \Omega(p^{-1})$, where p is an upper bound on the probability that, in the ideal world, the challenge provided to the adversary coincides with the outputs of the distributed sampler. We conclude that, if we want a negligible probability of success for attacks on the output of the sampler, we must have that $p \gg 2^{-\ell}$.

This observation is what motivated the definition of *lossy distributed sampler* [AWZ23]: A distributed sampler where a trapdoor in the CRS allows the honest parties to force the output of the protocol into a relative small (but not necessarily polynomial-sized) subset of values. In other words, if the distributed sampler simulator is provided with an ℓ-bit string z, it can generate messages for the honest parties so that, no matter what other messages the adversary sends, the final output belongs to a subset S of $t \ll 2^\ell$ strings, including z. In these constructions, the adversary cannot help selecting z as output of the protocol with probability $1/t$. In this way, we obtain a hardness-preserving distributed sampler where $\eta \leq t$.

A group-based approach. If we follow the lossy distributed sampler approach, we need to find a way to restrict the extent to which the adversary can influence the output of the protocol, no matter how many corrupted parties it impersonates. We borrow from an idea of Abram, Doerner, Ishai and Narayanan [ADIN24], originally applied in the context of coin tossing extension: We work over a large group G and force the adversary's influence within a proper subgroup H. Specifically, suppose that H is hidden, in the sense that it is hard to distinguish a random element of G from a random element of H. Then, the distributed sampler can simply ask each party P_i to send a random element $h_i \in H$ along with a zero-knowledge proof showing that h_i indeed belongs to H. The final output is $\prod_i h_i$.

This distributed sampler is hardness-preserving because the adversary cannot tell if we substitute the group element h_j sent by an honest party P_j with a uniformly random element of G. Formally, on input a random element $z \in G$, the simulator would sample h_j uniformly at random over zH, the coset of H in G containing z. Then, it would simulate the proof using a trapdoor embedded in the CRS.

Observe that the group elements chosen by the other parties all belong to the subgroup H (by the soundness of the proof), so the final output remains an element of zH. Moreover, observe that h_j contains no information about what element of zH is actually z, so the final output ends up coinciding with z with probability $1/|H|$. This would give a hardness-preserving distributed sampler where $\eta \approx |H|$ and $\nu \cdot \eta = \mathsf{negl}(\lambda)$. Observe that H can be much smaller than G. For example, consider the following instantiation based on a group $\mathcal{G}$ where DDH is hard. We can set $G := \mathcal{G}^n$ and $H := \langle (g_1, \ldots, g_n) \rangle$ where $g_1, \ldots, g_n$ are elements of $\mathcal{G}$ sampled at random. The subgroup H remains hidden inside G under DDH but $|G| = |H|^n$.

A lattice-based approach. The above blueprint can also be applied to the lattice setting. As part of the CRS, we provide a "fat" matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and we ask each party P_i to send a vector $\mathbf{y}_i$ obtained by multiplying $\mathbf{B}$ by a random binary vector $\mathbf{x}_i \in \mathbb{Z}_2^m$. As before, we also ask P_i to provide a zero-knowledge proof showing that $\mathbf{y}_i$ was indeed derived as we described. The final output is obtained by rounding the sum $\sum_i \mathbf{y}_i$ down to a smaller modulus p.

We observe that, if $\mathbf{B}$ is sampled at random, by the leftover hash lemma, each $\mathbf{y}_i$ is also random. If, however, we substitute $\mathbf{B}$ with a batch of LWE samples $\mathbf{S} \cdot \mathbf{M} + \mathbf{E}$ – where $\mathbf{S}$ and $\mathbf{M}$ are sampled uniformly at random over $\mathbb{Z}_q^{n \times k}$ and $\mathbb{Z}_q^{k \times m}$ for $k < n$, and $\mathbf{E}$ is a low-norm matrix in $\mathbb{Z}_q^{n \times m}$ sampled from a noise distribution – each $\mathbf{y}_i$ will be close in norm to $\mathbf{S} \cdot \mathbf{v}_i$ for some vector $\mathbf{v}_i \in \mathbb{Z}_q^k$. In this way, we have confined the contribution of the adversary close to the sublattice of $\mathbb{Z}_q^n$ generated by the columns of $\mathbf{S}$. Notice that this structure behaves similarly to cosets: If we add points that are all close to the sublattice generated by $\mathbf{S}$, we obtain again points that are close to the sublattice generated by $\mathbf{S}$. The simulator for the distributed sampler can therefore repeat what we did for groups: On input a random vector $\mathbf{z} \in \mathbb{Z}_p^n$, it lifts it up to a random element $\mathbf{y} \in \mathbb{Z}_q^n$ such that $\lceil \mathbf{y} \rfloor_p = \mathbf{z}$. Then, it sets the vector of the honest party P_j to be $\mathbf{y}_j = \mathbf{y} - \mathbf{S} \cdot \mathbf{v}$ where $\mathbf{v}$ is sampled at random over $\mathbb{Z}_q^k$. Since the zero-knowledge proofs force the contributions of all other parties close to the sublattice generated by $\mathbf{S}$, we obtain that the output will be close to $\mathbf{y} + \mathbf{S} \cdot (\mathbf{v}' - \mathbf{v})$ for some vector $\mathbf{v}' \in \mathbb{Z}_q^k$. Finally, since $\mathbf{v}$ is hidden by $\mathbf{y}$, the adversary ends up inadvertently choosing $\mathbf{v}' = \mathbf{v}$ with $1/q^k$ probability. In that case, the hope is that, after rounding, the final output would be $\mathbf{z}$. Unfortunately, dealing with any errors introduced by the rounding procedure turns out quite complicated. However, armed with patience, we manage to prove that this distributed sampler is hardness-preserving for $\eta = O(q^{2k})$ and $\nu \cdot \eta = \mathsf{negl}(\lambda)$. Notice that, by choosing a sufficiently big parameter n, the length of the output can be much larger than $2k \log q = \log \eta$.

Achieving d-wise independence. Once we have understood how to construct one-time secure hardness-preserving distributed samplers, it is relatively easy to modify the constructions to achieve d-wise independence. The intuition is simple: The message of each party will now consist of a key for a d-wise independent hash function. To obtain the output of the protocol, we derive messages for a one-time secure distributed sampler by evaluating each party's hash function on the label. At that point, we obtain the final sample by running the one-time secure distributed sampler on the messages we have just derived.

Notice that any d distinct labels are mapped to d independent, one-time secure, distributed sampler messages. This property can be exploited by the simulator for the d-wise independent, hardness preserving distributed sampler, which runs the simulator for the one-time secure distributed sampler d times, tries to guess the d labels chosen by the adversary, and finally generates the honest party hash key so that the guessed labels are mapped to the simulated messages for the one-time distributed sampler. For this last step, observe that we need the d-wise independent hash function to be programmable, a property that we will get quite naturally, using Lagrange interpolation. As we have already mentioned, with these modifications, the parameters of the hardness-preserving distributed sampler will inevitably move from η to $\eta^d \cdot \mu^d$, where μ is the size of the label space.

DDH-based solution. Moving from the general intuition to the actual construction still requires some work. We need to design d-wise independent hash functions that are compatible with the algebraic structure on

which the one-time distributed samplers are based. For the DDH-based construction, it means that we need two types of indistinguishable hash keys: The first one will be used by all parties to generate random elements in the subgroup H; the second one will be used by the simulator to generate random elements in $G = \mathcal{G}^n$. This is not too hard to build. We obtain the first type of key by putting the coefficients of a random degree-$(d-1)$ polynomial

$$a(X) := a_0 + a_1 \cdot X + \cdots + a_{d-1} \cdot X^{d-1}$$

in the exponent of the generator $(g_1, \ldots, g_n)$ of the subgroup H. In other words, if g is a generator of $\mathcal{G}$, $(g_1, \ldots, g_n)^\intercal = g^{\mathbf{r}}$ for an vector $\mathbf{r} \in \mathbb{Z}_p^n$ and $\mathbf{a}^\intercal = (a_0, \ldots, a_{d-1})$, the key is

$$\begin{bmatrix} g_1^{a_0} & \cdots & g_1^{a_{d-1}} \\ \vdots & & \vdots \\ g_n^{a_0} & \cdots & g_n^{a_{d-1}} \end{bmatrix} = g^{\mathbf{r} \cdot \mathbf{a}^\intercal},$$

To evaluate the hash function on some label x, we exploit the fact that the polynomial evaluation is linear in the coefficients, i.e., we derive

$$g^{\mathbf{r} \cdot \mathbf{a}^\intercal \cdot \overline{\mathbf{x}}} = \begin{pmatrix} g_1^{a(x)} \\ \vdots \\ g_n^{a(x)} \end{pmatrix} \in H,$$

where $\overline{\mathbf{x}}^\intercal = (1, x, x^2, \ldots, x^{d-1})$.

The second type of key consists instead of a random $n \times d$ matrix of group elements $g^{\mathbf{R}}$ for a random $\mathbf{R} \in \mathbb{Z}_p^{n \times d}$. It is easy to see that the two types of keys are indistinguishable under DDH.

LWE-based solution. For the lattice-based solution, we would like the d-wise independent hash function to output random elements of $\mathbb{Z}_q^n$ when the matrix $\mathbf{B}$ in the CRS is uniformly random. If instead $\mathbf{B} = \mathbf{S} \cdot \mathbf{M} + \mathbf{E}$, we would like it to output elements that are close to the sublattice generated by $\mathbf{S}$.

We start by observing that, given vectors $\mathbf{a}_0, \ldots, \mathbf{a}_{d-1} \in \mathbb{Z}_q^n$ that are close to the sublattice generated by $\mathbf{S}$, for any low magnitude scalar x, the vector

$$\mathbf{a}_0 + \mathbf{a}_1 \cdot x + \cdots + \mathbf{a}_{d-1} \cdot x^{d-1}$$

is close to the sublattice generated by $\mathbf{S}$. If instead $\mathbf{a}_0, \ldots, \mathbf{a}_{d-1}$ are random (and q is prime), we obtain a d-wise independent hash function over $\mathbb{Z}_q^n$. To solve our issue, it is therefore sufficient to ask each party to set $\mathbf{a}_i = \mathbf{B} \cdot \mathbf{x}_i$ for a random binary vector $\mathbf{x}_i \in \mathbb{Z}_2^m$ (we enforce this condition one again using a zero-knowledge proof).

Our final solution will actually be an improved version of this hash function. In order to get better parameters, set no cap on the magnitude of x and avoid the noise magnitude from growing exponentially in d, we rely on bit decomposition $\mathbf{G}^{-1}$: the vectors $\mathbf{a}_i$ will be substituted by matrices $\mathbf{A}_i = \mathbf{B} \cdot \mathbf{X}_i$ where $\mathbf{X}_i \in \mathbb{Z}_2^{m \times \log q}$. To evaluate the hash function on x, we compute

$$\mathbf{A}_0 \cdot \mathbf{G}^{-1}(1) + \mathbf{A}_1 \cdot \mathbf{G}^{-1}(x) + \cdots + \mathbf{A}_{d-1} \cdot \mathbf{G}^{-1}(x^{d-1}).$$

To see that, when $\mathbf{A}_0, \ldots, \mathbf{A}_{d-1}$ are random, this is a d-wise independent hash function, check what happens if we substitute $\mathbf{A}_i$ with $\mathbf{R}_i + \mathbf{a}_i \cdot \mathbf{g}^\intercal$, where $\mathbf{g}^\intercal = (1, 2, \ldots, 2^{\lceil \log q \rceil - 1})$ is the gadget vector [MP13], and the matrix $\mathbf{R}_i \in \mathbb{Z}_q^{n \times \log q}$ and the vector $\mathbf{a}_i \in \mathbb{Z}_q^n$ are random. The evaluation on a scalar x would produce

$$\Big(\mathbf{R}_0 \cdot \mathbf{G}^{-1}(1) + \mathbf{R}_1 \cdot \mathbf{G}^{-1}(x) + \cdots + \mathbf{R}_{d-1} \cdot \mathbf{G}^{-1}(x^{d-1})\Big) + \Big(\mathbf{a}_0 + \mathbf{a}_1 \cdot x + \cdots + \mathbf{a}_{d-1} \cdot x^{d-1}\Big).$$

Notice that the term on the right is a d-wise independent hash function.

2.2 d-wise Independent Distributed Samplers: The Details

We begin by introducing the notion of *labeled* distributed samplers.

Definition 1 (*Labeled* Distributed Sampler). *Let $\mathcal{D}(1^\lambda)$ be an efficiently sampleable distribution. A* labeled distributed sampler *for $\mathcal{D}$ and label space $[\mu]$ consists of a triple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sample})$*, with the following syntax:*

$\mathsf{Setup}(1^\lambda)$**:** *This setup algorithm is randomized and takes as input the security parameter 1^λ. The output is a CRS σ.*

$\mathsf{Gen}(\sigma)$**:** *The generation algorithm is randomized and takes as input the CRS σ. The output is a distributed sampler message U.*

$\mathsf{Sample}(\sigma, U_1, \ldots, U_L, x)$**:** *The sampling algorithm is deterministic and takes as input a CRS σ, distributed sampler messages $U_1, \ldots, U_L$ for any $L \in \mathbb{N}$, and a label $x \in [\mu]$. The output is a sample z.*

We now strengthen the definition of a hardness-preserving distributed sampler [AWZ23] for our modified context. In its original form, a hardness-preserving distributed sampler is a distributed sampler that provides meaningful security guarantees against active adversaries. Specifically, imagine that no adversary can successfully perform an attack to a sample from $\mathcal{D}(1^\lambda)$ with probability greater than ϵ (for example, this could be the probability that the adversary is able to factor the output from $\mathcal{D}(1^\lambda)$). A hardness-preserving distributed sampler guarantees that the probability that an adversary successfully performs an attack on its output is at most $(\epsilon + \nu) \cdot \eta$, where η and ν are some fixed functions that characterize the construction. Although this seems a rather unnatural and perhaps weak game-based security property, it turns out this is the best we are currently able to achieve in the plain model (refer to [AFOS25] for an overview of the difficulties facing simulation-based security definitions).

Formally, our definition relies on the ideal world-real world paradigm. In the real world, we provide the adversary with a distributed sampler CRS and a honestly generated message U that will be used across $t \leq d$ distributed sampler executions with distinct labels. For each of these t executions, the adversary is allowed to select the label and the remaining (possibly corrupted) distributed-sampler messages. At the end, the challenger asks the adversary to attack the outputs of the t executions it chose. The ideal world proceeds similarly, but the t samples the adversary is required to attack are sampled from $\mathcal{D}(1^\lambda)$, whereas the CRS and U are produced by a simulator on input the t challenges.

Crucially, however, we do not require the two worlds to be indistinguishable. Actually, this would be impossible as, in the ideal world, there is at least a $1 - 1/\mathsf{poly}(\lambda)^t$ probability that the t samples the adversary is asked to break are different from the outputs of the chosen distributed sampler executions (if the CRS and U fix the output, the adversary would notice that by changing the corrupted parties' messages, the result does not change). What we ask, instead, is that the probability of the adversary's success in the real world is not significantly bigger than η times that in the ideal world.

Definition 2 (d-wise Independent Hardness-Preserving Distributed Sampler). *Let $\mathsf{DS} \stackrel{\text{def}}{=} (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sample})$ be a labeled distributed sampler for the distribution $\mathcal{D}(1^\lambda)$, let $\eta = \eta(t, \lambda)$ and $N = N(\lambda)$ be (not-necessarily polynomial) functions, and let $d = d(\lambda)$ be a polynomial function. Let $\nu : \mathbb{N} \times \mathbb{N} \to [0, 1]$ be a real function.*

We say that an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ is t-admissible *if it is PPT and $\mathcal{A}_0$ always outputs a tuple $((\ell_k, (U_{j,k})_{j \in [L_k] \setminus \{\ell_k\}}, x_k)_{k \in [t]}, \mathsf{aux})$, where $\ell_k \in [L_k]$, $L_k \leq N$ and $r_k \neq r_j$ for all $k \neq j \in [t]$.*

We say that DS is (η, ν, d, N)-hardness-preserving*, if there exist PPT simulators $(\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that, for every t-admissible adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ with $t \leq d$, and sufficiently large $\lambda \in \mathbb{N}$, it holds that*

$$Adv^{\mathsf{ideal}}_{\mathcal{A}}(\lambda) \geq \frac{1}{\eta(t, \lambda)} \cdot Adv^{\mathsf{real}}_{\mathcal{A}}(\lambda) - \nu(t, \lambda),$$

where

$$Adv^{\mathsf{real}}_{\mathcal{A}}(\lambda) \stackrel{\text{def}}{=} \Pr\left[\mathcal{A}_1((z_k)_{k\in[t]}, \mathsf{aux}) = 1 \;\middle|\; \begin{array}{l} \sigma \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^\lambda) \\ U \stackrel{\$}{\leftarrow} \mathsf{Gen}(\sigma) \\ ((\ell_k, (U_{j,k})_{j\neq\ell_k}, x_k)_{k\in[t]}, \mathsf{aux}) \stackrel{\$}{\leftarrow} \mathcal{A}_0(1^\lambda, \sigma, U) \\ \forall k \in [t] : U_{\ell_k,k} \leftarrow U \\ \forall k \in [t] : z_k \leftarrow \mathsf{Sample}(\sigma, U_{1,k}, \ldots, U_{L_k,k}, x_k) \end{array}\right]$$

$$Adv^{\mathsf{ideal}}_{\mathcal{A}}(\lambda) \stackrel{\text{def}}{=} \Pr\left[\mathcal{A}_1((z_k)_{k\in[t]}, \mathsf{aux}) = 1 \;\middle|\; \begin{array}{l} (\sigma, \tau) \stackrel{\$}{\leftarrow} \mathsf{Sim}_0(1^\lambda) \\ \forall k \in [t] : z_k \stackrel{\$}{\leftarrow} \mathcal{D}(1^\lambda) \\ U \stackrel{\$}{\leftarrow} \mathsf{Sim}_1(\tau, (z_k)_{k\in[t]}) \\ ((\ell_k, (U_{j,k})_{j\neq\ell_k}, x_k)_{k\in[t]}, \mathsf{aux}) \stackrel{\$}{\leftarrow} \mathcal{A}_0(1^\lambda, \sigma, U) \end{array}\right]$$

2.3 A DDH-Based Construction

We now present our first construction: A d-wise independent, hardness-preserving distributed sampler for the uniform distribution based on the hardness of DDH. The rationale behind this solution has been described in Section 2.1.

In a nutshell, we take a group $\mathcal{G}$ of prime order p over which DDH is hard. We work over its direct product $G \stackrel{\text{def}}{=} \mathcal{G}^n$. As part of the CRS, we provide a random element $\mathbf{g} \in G$. We consider the subgroup $H = \langle \mathbf{g} \rangle$. The distributed sampler message of each party consists of d random elements of H along with NIZKs proving their membership to H. Notice that H is in bijection with $\mathbb{Z}_p$, so we can view the d group elements published by each party as the coefficients of a random degree-$(d-1)$ polynomial over $\mathbb{Z}_p$: the only unusual detail is that the coefficient are encoded in the exponent of $\mathbf{g}$. To evaluate the distributed sampler under a label $x \in \mathbb{Z}_p$, we derive, from each party's message, a new element of H by evaluating the degree-$(d-1)$ polynomial in the exponent on x. Then, we simply multiply the elements we obtained in this way.

Intuitively, this consists of a hardness preserving distributed sampler because, under DDH, the message of each honest party looks like d random elements in $G = \mathcal{G}^n$ (we can simulate the NIZK using a trapdoor hidden in the CRS). In other words, by exploiting the trapdoor in the CRS, we can make an honest party contribute to the output with a random element of G, whereas all other parties (including the corrupted ones) will contribute with elements in H. Since the output is just the product of all these contributions, the chosen honest party will have much more influence on the final result than all other participants together. The construction is also d-wise independent because the random degree-$(d-1)$ polynomial maps d distinct labels to d independent contributions.

This result is formalized in the following theorem. Note that, under a suitable instantiation of the NIZK proof, the CRS required by the construction can be unstructured.

Construction 1: DDH-Based Hardness-Preserving Distributed Sampler

$\mathsf{Setup}(1^\lambda)$**:** Sample $\mathbf{g} \stackrel{\$}{\leftarrow} \mathcal{G}^n_\lambda$, Then, generate $\omega \stackrel{\$}{\leftarrow} \mathsf{NIZK.Setup}(1^\lambda)$. Output $\sigma := (\mathbf{g}, \omega)$.

$\mathsf{Gen}(\sigma)$**:** For $i \in [d]$, sample $a_i \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, set $\mathbf{h}_i \leftarrow \mathbf{g}^{a_i}$ and derive

$$\pi_i \stackrel{\$}{\leftarrow} \mathsf{NIZK.Prove}(\omega, (\mathbf{g}, \mathbf{h}_i), a_i).$$

Output $U := (\mathbf{h}_i, \pi_i)_{i\in[d]}$.

$\mathsf{Sample}(\sigma, U_1, \ldots, U_L, x)$**:** For any $j \in [L]$, rewrite U_j as $(\mathbf{h}_{i,j}, \pi_{i,j})_{i\in[d]}$. If, for any $j \in [L]$ and $i \in [d]$, we have

$$\mathsf{NIZK.Verify}(\omega, (\mathbf{g}, \mathbf{h}_{i,j}), \pi_{i,j}) = 0,$$

set $\mathbf{h}_{i,j} \leftarrow \mathbf{1}$. For any $j < k \in [L]$ such that $(\mathbf{h}_{i,j})_{i\in[d]} = (\mathbf{h}_{i,k})_{i\in[d]}$, set $\mathbf{h}_{i,k} \leftarrow \mathbf{0}$ for every $i \in [d]$. Then, for every $j \in [L]$, derive

$$\mathbf{h}_j \leftarrow \prod_{i\in[d]} \mathbf{h}_{i,j}^{x^i}.$$

Then, output

$$\mathbf{z} \leftarrow \prod_{j\in[L]} \mathbf{h}_j.$$

Theorem 1. *Let $n = n(\lambda)$ and $d = d(\lambda)$ be polynomial functions of the security parameter. Let $N = N(\lambda)$ be a (not-necessarily polynomial) function of the security parameter. Suppose that $\mathcal{G} = (\mathcal{G}_\lambda)_{\lambda\in\mathbb{N}}$ is a family of groups of prime order $p(\lambda)$. Assume also that $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ is a simulation-extractable NIZK (see Appendix A.4) for the relation*

$$\mathcal{R}_\lambda := \{x = (\mathbf{g}, \mathbf{h}), w | \mathbf{g}, \mathbf{h} \in \mathcal{G}_\lambda^n, w \in \mathbb{Z}_p, \mathbf{h} = \mathbf{g}^w\}.$$

Then, under the hardness of DDH over $\mathcal{G}$, Construction 1 is a (η, ν, d, N)-hardness-preserving labeled distributed sampler for the uniform distribution over $\mathcal{G}^n$ and with label space $[p]$, where

$$\eta(t, \lambda) := p^{2t}, \qquad \nu(t, \lambda) := \eta^{-1}(t, \lambda) \cdot \mathsf{negl}(\lambda).$$

We prove this theorem in Appendix B.1.

By choosing a sufficiently large n in Construction 1, we obtain the following corollary. We also need a crucial observation, that there exists an explainable function[6] that allows us to map random elements of $\mathcal{G}$ to random binary strings of length $\ell \geq 1$.

Corollary 1. *Assume the existence of simulation-extractable NIZKs for all NP relations. Under the DDH assumption, there exists a (η, ν, d, N)-hardness-preserving distributed sampler for the uniform distribution over $\{0,1\}^n$ where N is superpolynomial in λ, $\nu(t, \lambda) = \mathsf{negl}(\lambda) \cdot \eta^{-1}(t, \lambda)$, and $\eta(t, \lambda) = 2^{t \cdot n^\gamma}$ for a constant $\gamma < 1$.*

2.4 An LWE-Based Construction

Our second construction is an LWE-based d-wise independent, hardness-preserving distributed sampler for the uniform distribution. The rationale behind this solution is similar to that of our DDH-based construction, with the exception that we move from "linear algebra in the exponent" to "(low-norm) linear algebra plus noise". In this setting, we need however to work harder to take into account how the adversarially generated noise can affect rounding in the unlucky event in which the value is close to a threshold, i.e. any odd multiple of $q/2p$.

In a nutshell, the construction provides as part of the CRS the description of a lossy function [PW08]. This consists of $f_{\mathbf{B}} : \mathbb{Z}_2^{m\times \log q} \to \mathbb{Z}_q^{n\times \log q}$, $\mathbf{X} \mapsto \mathbf{B} \cdot \mathbf{X}$, where $\mathbf{B}$ is a matrix in $\mathbb{Z}_q^{n\times m}$, with $m \geq 2n \log q$. When $\mathbf{B}$ is sampled at random, by the leftover hash lemma, the function maps the uniform distribution over $\mathbb{Z}_2^{m\times \log q}$ to the uniform distribution over $\mathbb{Z}_q^{n\times \log q}$. If instead we substitute $\mathbf{B}$ with a batch of LWE samples $\mathbf{S} \cdot \mathbf{M} + \mathbf{E}$, where $\mathbf{S}$ and $\mathbf{M}$ are random over $\mathbb{Z}_q^{n\times k}$ and $\mathbb{Z}_q^{k\times m}$, and $\mathbf{E}$ comes from a low-norm distribution, the function $f_{\mathbf{B}}$ outputs matrices where all columns are close in norm to elements of the lattice generated by $\mathbf{S}$. We call this the *lossy mode* of $f_{\mathbf{B}}$.

[6] An *explainable function* is a keyed function $f_{\mathsf{k}} : \mathcal{G} \to \{0,1\}^\ell$ for which there exists a PPT simulator f_{k}^{-1} such that the distribution of $(\mathsf{k}, f_{\mathsf{k}}^{-1}(y), y)$, for $y \xleftarrow{\$} \{0,1\}^\ell$ and a random k, is computationally indistinguishable from $(\mathsf{k}, x, f_{\mathsf{k}}(x))$, where $x \xleftarrow{\$} \mathcal{G}$ and k are random. When $\ell = O(\log \lambda)$, we can easily build an explainable function using a randomness extractor: $f_{\mathsf{k}}^{-1}(y)$ will simply brute force search for an element x such that $f_{\mathsf{k}}(x) = y$.

The construction requires each party to generate d matrices $\mathbf{A}_1, \ldots, \mathbf{A}_d$ in $\mathbb{Z}_q^{n \times \log q}$ by evaluating $f_{\mathbf{B}}$ on a random input. As before, we use NIZKs to enforce this condition. The d matrices are seen as the components of the d-wise independent hash function that maps a scalar $x \in \mathbb{Z}_q$ to the $\mathbb{Z}_q^n$ vector

$$\mathbf{A}_1 \cdot \mathbf{G}^{-1}(1) + \mathbf{A}_2 \cdot \mathbf{G}^{-1}(x) + \cdots + \mathbf{A}_d \cdot \mathbf{G}^{-1}(x^{d-1}),$$

where $\mathbf{G}^{-1}(x)$ denotes the bit decomposition of x. As we explained in Section 2.1, this function is d-wise independent because it hides under the hood the evaluation of (a batch of n) random degree-$(d-1)$ polynomials. Compared to the typical instantiation of d-wise independent hashing, however, this solution has an important advantage in the lattice setting: the output is a *low-norm* linear function of the key. That means that when the function $f_{\mathbf{B}}$ is in lossy mode, the output of the d-wise independent hash function is close in norm to an element of the lattice generated by $\mathbf{S}$.

We obtain the output of the distributed sampler with an approach similar to the DDH-based solution: we derive from each party's message a new vector in $\mathbb{Z}_q^n$ by evaluating the d-wise independent hash function on the label x. Then, we add the contributions of all parties and we round the result down onto $\mathbb{Z}_p^n$, where p is a modulus sufficiently smaller than q.

Intuitively, this is construction is hardness-preserving for reasons similar to those we have discussed in the previous section: we can pick an honest party and sample the components of their hash key at random (we simulate their NIZK proofs using a trapdoor in the CRS). Then, we can switch $f_{\mathbf{B}}$ to lossy mode. This affects the contribution of all parties except the selected honest one, forcing their contribution close to the lattice generated by $\mathbf{S}$. Of course, the adversary can still try to influence the final output of the protocol through the noise, however, with high probability (actually, this is where the complex bit lies), any attempt of this kind will be invalidated by rounding. In conclusion, the selected honest party will have much more influence on the output than all other parties together.

Notice that, as for the DDH-based construction, with an appropriate choice of the NIZK, we can make the CRS in our construction unstructured.

Construction 2: LWE-Based Hardness-Preserving Distributed Sampler

$\mathsf{Setup}(1^\lambda)$**:** Sample $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and $\omega \xleftarrow{\$} \mathsf{NIZK.Setup}(1^\lambda)$. Output $\sigma := (\mathbf{B}, \omega)$.

$\mathsf{Gen}(\sigma)$**:** For $i \in [d]$, sample $\mathbf{X}_i \xleftarrow{\$} \mathbb{Z}_2^{m \times \log q}$, set $\mathbf{A}_i \leftarrow \mathbf{B} \cdot \mathbf{X}_i$ and derive $\pi_i \xleftarrow{\$} \mathsf{NIZK.Prove}(\omega, (\mathbf{B}, \mathbf{A}_i), \mathbf{X}_i)$. Output $U := (\mathbf{A}_i, \pi_i)_{i \in [d]}$.

$\mathsf{Sample}(\sigma, U_1, \ldots, U_L, x)$**:** For any $j \in [L]$, rewrite U_j as $(\mathbf{A}_{i,j}, \pi_{i,j})_{i \in [d]}$. If, for any $j \in [L]$ and $i \in [d]$, we have

$$\mathsf{NIZK.Verify}(\omega, (\mathbf{B}, \mathbf{A}_{i,j}), \pi_{i,j}) = 0,$$

set $\mathbf{A}_{i,j} \leftarrow \mathbf{0}$. For any $j < h \in [L]$ such that $(\mathbf{A}_{i,j})_{i \in [d]} = (\mathbf{A}_{i,h})_{i \in [d]}$, set $\mathbf{A}_{i,h} \leftarrow \mathbf{0}$ for every $i \in [d]$. Then, for every $j \in [L]$, derive

$$\mathbf{y}_j \leftarrow \sum_{i \in [d]} \mathbf{A}_{j,i} \cdot \mathbf{G}^{-1}(x^{i-1})$$

Then, output

$$\mathbf{z} \leftarrow \left\lfloor \sum_{i \in [L]} \mathbf{y}_i \right\rceil_p$$

Theorem 2. *Let $k = k(\lambda), n = n(\lambda), m = m(\lambda)$ and $d = d(\lambda)$ be polynomial functions of the security parameter. Let $N = N(\lambda)$ be a (not-necessarily polynomial) function of the security parameter. Suppose that $q = q(\lambda)$ is a prime modulus and let $s = \alpha(\lambda) \cdot q$ where $\alpha : \mathbb{N} \to [0, 1]$ is a real function of the security parameter. Suppose that $m \geq 2n \log q$. Let $p = p(\lambda)$ be a (not-necessarily polynomial) modulus such that $N \cdot \alpha \cdot p = \mathsf{negl}(\lambda)$. Assume also that $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ is a simulation-extractable NIZK (see Appendix A.4) for the relation*

$$\mathcal{R}_\lambda := \left\{ x = (\mathbf{B}, \mathbf{A}), w = \mathbf{X} \middle| \mathbf{B} \in \mathbb{Z}_q^{n \times m}, \mathbf{A} \in \mathbb{Z}_q^{n \times \log q}, \mathbf{X} \in \mathbb{Z}_2^{m \times \log q}, \mathbf{A} = \mathbf{B} \cdot \mathbf{X} \right\}.$$

Then, under the hardness of LWE with $\alpha(\lambda)$ modulus-to-noise ratio, Construction 2 is a (η, ν, d, N)-hardness-preserving labeled distributed sampler for the uniform distribution over $\mathbb{Z}_p^n$ and with label space $\mathbb{Z}_q$, where

$$\eta(t,\lambda) := q^{2t(k+1)}, \qquad \nu(t,\lambda) := \eta^{-1}(t,\lambda) \cdot \mathsf{negl}(\lambda).$$

We prove this theorem in Appendix B.2.

By instantiating Construction 2 with a sufficiently large n and with $p = 2^\ell$ for some $\ell \in \mathbb{N}$, we obtain the following.

Corollary 2. *Assume the existence of simulation-extractable NIZKs for all* NP *relations. Under the LWE assumption with subexponential modulus-to-noise ratio, there exists a (η, ν, d, N)-hardness-preserving distributed sampler for the uniform distribution over $\{0,1\}^n$ where $\nu(t,\lambda) = \mathsf{negl}(\lambda) \cdot \eta^{-1}(t,\lambda)$, N is superpolynomial in λ and $\eta(t,\lambda) = 2^{t \cdot n^\gamma}$ for a constant $\gamma < 1$.*

3 Multi-Verifier Signatures of Work

Next, we formalize the notion of a *multi-verifier signature of work* (MV-SoW). This is a primitive that aims to provide (a weak form of) authentication in a world without a PKI. Specifically, it allows parties to efficiently prove to a verifier that they invested a significant amount of resources on a message m. The proof takes the form of a signature which can be checked against the verifier's key. We will then show how to construct a MV-SoW by composing a d_0-wise independent, hardness-preserving distributed sampler with a proof of work (see Appendix A.3 for syntax and security definitions of a proof of work).

The MV-SoW construction guarantees that to sign multiple messages (up to a polynomial threshold d), the prover (signer) needs to invest resources proportionally to the number of messages. The costs can, however, be amortized over the number of verifiers: We can generate a signature that validates against multiple verifier keys at roughly the same cost as a signature that validates against a single key.

Definition 3. *A* multi-verifier signature of work *(*MV-SoW*) consists of a tuple of PPT algorithms* (Setup, Gen, Sign, Verify) *with the following syntax:*

$\mathsf{Setup}(1^\lambda, 1^s)$: *This algorithm is randomized and takes as input the security parameter 1^λ and a time parameter 1^s. The output is a CRS σ.*

$\mathsf{Gen}(1^\lambda, \sigma)$: *This algorithm is randomized and takes as input the security parameter and the CRS σ. The output is a key K. We require this algorithm to run in time $\mathsf{poly}(\lambda, \log s)$.*

$\mathsf{Sign}(1^s, \sigma, K_1, \ldots, K_L, m)$**:** *This algorithm takes as input the time parameter 1^s, the CRS σ, an arbitrary number of keys $K_1, \ldots, K_L$ and a message m. The output is a signature of work w. We require this algorithm to run in time $r_{\mathsf{SoW}}(s, \lambda, L) = \mathsf{poly}(s, \lambda) + \mathsf{poly}(L, \lambda)$. We call L the rank of the signature.*

$\mathsf{Verify}(\sigma, K, m, v)$**:** *This algorithm takes as input the CRS σ, a key K, a message m and a signature v. The output is a bit $b \in \{0,1\}$ representing whether the signature of work is accepted or not. We require this algorithm to run in time $\mathsf{poly}(L, \lambda, \log s)$.*

The following definition formalizes the correctness property: Honestly generated signatures are always accepted by the verifier.

Definition 4 (Correctness). *A multi-verifier signature of work is* correct *if, for every $\lambda, s \in \mathbb{N}$, every message m, every number of keys L and index $j \in [L]$, we have*

$$\Pr\left[\mathsf{Verify}(\sigma, K_j, m, v) = 1 \;\middle|\; \begin{array}{l} \sigma \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^s) \\ \forall i \in [L] : K_i \xleftarrow{\$} \mathsf{Gen}(1^\lambda, \sigma) \\ v \xleftarrow{\$} \mathsf{Sign}(1^s, \sigma, K_1, \ldots, K_L, m) \end{array}\right] = 1.$$

Next, we formalize security with a definition along the lines of the *unforgeability* property for signature schemes. The property states that it is hard for an adversary to convince an honest verifier about $t \leq d_0$ distinct messages without investing resources proportionally to t. This holds even if the adversary has given access to a signing oracle that can be queried up to d_1 times.

Definition 5 (Moderate Unforgeability). *Consider a multi-verifier signature of work* $\mathsf{MV\text{-}SoW} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$. *Let* d_0, d_1 *be polynomial functions and let* $\epsilon : \mathbb{N} \times \mathbb{N} \to [0,1]$ *and* $\beta : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to [0,1]$ *be real functions.*

Let $\mathcal{O}^s_{\sigma,K}$ *be an oracle that on input an index* ℓ*, a tuple* $(K_i)_{i\in[L]\setminus\{\ell\}}$ *and a message* m*, sets* $K_\ell \leftarrow K$ *and outputs* $\mathsf{Sign}(1^s, \sigma, K_1, \ldots, K_L, m)$*. Let* $\mathcal{Q}$ *be the set of messages* m *that are queried by the adversary to* $\mathcal{O}^s_{\sigma,K}$*.*

We say that a PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *is* (t, d_1, β, L)-admissible *if* $\mathcal{A}_1$ *runs in time at most* $t(\lambda) \cdot \beta(s, \lambda, L) \cdot r_{\mathsf{SoW}}(s, \lambda, L)$*, queries* $\mathcal{O}^s_{\sigma,K}$ *at most* $d_1(\lambda)$ *times and always for signatures of rank at most* L*, and outputs a tuple* $(m_h, v_h)_{h\in[t]}$*, where* v_h *is a signature of rank at most* L*.*

Then, MV-SoW *is* $(d_0, d_1, \beta, \epsilon)$-moderately unforgeable *if, for every* (t, d_1, β, L)*-admissible adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *with* $t \leq d_0$*, and every sufficiently large* $\lambda, s \in \mathbb{N}$*,*

$$\Pr\left[\begin{array}{c} \forall h \in [t]: \\ \mathsf{Verify}(\sigma, K, m_h, v_h) = 1 \\ m_h \notin \mathcal{Q} \cup \{m_j\}_{j\neq h} \end{array} \middle| \begin{array}{l} \sigma \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^s) \\ \mathsf{aux} \xleftarrow{\$} \mathcal{A}_0(1^\lambda, 1^s, \sigma) \\ K \xleftarrow{\$} \mathsf{Gen}(1^\lambda, \sigma) \\ (m_h, v_h)_{h\in[t]} \xleftarrow{\$} \mathcal{A}_1^{\mathcal{O}^s_{\sigma,K}}(1^\lambda, K, \mathsf{aux}) \end{array}\right] \leq \epsilon(t, \lambda).$$

3.1 A Concrete MV-SoW Construction

We now show how to construct a MV-SoW by composing a d_0-wise independent, hardness-preserving distributed sampler with a proof of work (see Appendix A.3 for syntax and security definitions).

Construction 3: A Multi-Verifier Signature of Work

$\mathsf{Setup}(1^\lambda, 1^s)$**:** Compute $\sigma_{\mathsf{PoW}} \xleftarrow{\$} \mathsf{PoW.Setup}(1^\lambda, 1^s)$ and $\sigma_{\mathsf{DS}} \xleftarrow{\$} \mathsf{DS.Setup}(1^\lambda)$. Then, output $\sigma := (\sigma_{\mathsf{PoW}}, \sigma_{\mathsf{DS}})$.
$\mathsf{Gen}(1^\lambda, \sigma)$**:** Compute $U \xleftarrow{\$} \mathsf{DS.Gen}(1^\lambda, \sigma_{\mathsf{DS}})$. Output $K := U$.
$\mathsf{Sign}(1^s, \sigma, K_1, \ldots, K_L, m)$**:** Compute $z \leftarrow \mathsf{DS.Sample}(\sigma_{\mathsf{DS}}, U_1, \ldots, U_L, m)$. Then, derive

$$w := \mathsf{PoW.Solve}(1^s, \sigma_{\mathsf{PoW}}, z)$$

and output $v := (w, K_1, \ldots, K_L)$.
$\mathsf{Verify}(\sigma, K, m, v)$**:** If $K \notin \{K_1, \ldots, K_L\}$, output 0. If not, compute

$$z \leftarrow \mathsf{DS.Sample}(\sigma_{\mathsf{DS}}, U_1, \ldots, U_L, m).$$

Then, output $\mathsf{PoW.Verify}(\sigma_{\mathsf{PoW}}, z, w)$.

Theorem 3. *Let* $\mathsf{PoW} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Solve}, \mathsf{Verify})$ *be a* $(d_0, d_1, \beta, \epsilon)$*-secure PoW scheme (Appendix A.3). Suppose that* $\mathsf{DS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sample})$ *is a* (η, ν, d_0, N)*-hardness-preserving labeled distributed sampler for the distribution* PoW.Gen*, where* N *is superpolynomial in* λ*.*

Then, Construction 3 is a correct and $(d_0, d_1, \beta', \eta \cdot (\epsilon + \nu))$*-moderately unforgeable* MV-SoW*, where*

$$\beta'(s, \lambda, L) = \frac{\beta(s, \lambda) \cdot r_{\mathsf{PoW}}(s, \lambda) - T_{\mathsf{DS.Sim}_1}(\lambda) - d_1 \cdot T_{\mathsf{DS.Sample}}(\lambda, L)}{T_{\mathsf{DS.Sample}}(\lambda, L) + r_{\mathsf{PoW}}(s, \lambda)}.$$

Above, $T_{\mathsf{DS.Sim}_1}(\lambda)$ denotes the running time of the simulator Sim_1 for the hardness-preserving distributed sampler DS; $T_{\mathsf{DS.Sample}}(\lambda, L)$ denotes the running time of $\mathsf{DS.Sample}$ when there are at most L parties; and $r_{\mathsf{PoW}}(s, \lambda)$ denotes the time required by an honest party to solve a proof of work of time parameter s.

Proof. It is easy to verify that the MV-SoW scheme is correct, we therefore focus on security. Suppose that our claim is false and there exists a (t, d_1, β', L)-admissible adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ with $t \leq d_0$, that wins the security game of the multi-verifier signature of work with probability $\delta(\lambda) > \eta(t,\lambda) \cdot (\epsilon(t,\lambda) + \nu(t,\lambda))$. In other words, $\delta(\lambda) =$

$$\Pr\left[\begin{array}{c} \forall h \in [t]: \\ \mathsf{PoW.Verify}(\sigma_{\mathsf{PoW}}, z_h, w_h) = 1 \\ m_h \notin \mathcal{Q} \cup \{m_j\}_{j \neq h} \end{array} \middle| \begin{array}{l} \sigma_{\mathsf{PoW}} \xleftarrow{\$} \mathsf{PoW.Setup}(1^\lambda, 1^s) \\ \sigma_{\mathsf{DS}} \xleftarrow{\$} \mathsf{DS.Setup}(1^\lambda) \\ \mathsf{aux} \xleftarrow{\$} \mathcal{A}_0(1^\lambda, 1^s, (\sigma_{\mathsf{PoW}}, \sigma_{\mathsf{DS}})) \\ U \xleftarrow{\$} \mathsf{DS.Gen}(1^\lambda, \sigma_{\mathsf{DS}}) \\ (\ell_h, (U_{j,h})_{j \neq \ell_h}, m_h, w_h)_{h \in [t]} \xleftarrow{\$} \mathcal{A}_1^{\mathcal{O}^s_{\sigma,U}}(1^\lambda, U, \mathsf{aux}) \\ \forall h \in [t]: U_{\ell_h, h} \leftarrow U \\ \forall h \in [t]: z_h \leftarrow \mathsf{DS.Sample}(\sigma_{\mathsf{DS}}, (U_{j,h})_j, m_h) \end{array}\right]$$

Define the probability $\delta_1(\lambda)$ as

$$\Pr\left[\begin{array}{c} \forall h \in [t]: \\ \mathsf{PoW.Verify}(\sigma_{\mathsf{PoW}}, z_h, w_h) = 1 \\ m_h \notin \mathcal{Q} \cup \{m_j\}_{j \neq h} \end{array} \middle| \begin{array}{l} \sigma_{\mathsf{PoW}} \xleftarrow{\$} \mathsf{PoW.Setup}(1^\lambda, 1^s) \\ (\sigma_{\mathsf{DS}}, \tau) \xleftarrow{\$} \mathsf{DS.Sim}_0(1^\lambda) \\ \mathsf{aux} \xleftarrow{\$} \mathcal{A}_0(1^\lambda, 1^s, (\sigma_{\mathsf{PoW}}, \sigma_{\mathsf{DS}})) \\ \forall h \in [t]: z_h \xleftarrow{\$} \mathsf{PoW.Gen}(1^\lambda) \\ U \xleftarrow{\$} \mathsf{DS.Sim}_1(\tau, (z_h)_{h \in [t]}) \\ (\ell_h, (U_{j,h})_{j \neq \ell_h}, m_h, w_h)_{h \in [t]} \xleftarrow{\$} \mathcal{A}_1^{\mathcal{O}^s_{\sigma,U}}(1^\lambda, U, \mathsf{aux}) \end{array}\right]$$

We claim that, by the security of the (η, ν, d_0, N)-hardness-preserving distributed sampler, $\delta_1(\lambda) \geq \delta(\lambda)/\eta(t,\lambda) - \nu(t,\lambda)$. To see that, consider the following distributed sampler adversary $\mathcal{A}_{\mathsf{DS}} = (\mathcal{A}_{\mathsf{DS},0}, \mathcal{A}_{\mathsf{DS},1})$:

- $\mathcal{A}_{\mathsf{DS},0}$ receives σ_{DS} and U from the challenger. It begins by sampling $\sigma_{\mathsf{PoW}} \xleftarrow{\$} \mathsf{PoW.Setup}(1^\lambda, 1^s)$ and $\mathsf{aux} \xleftarrow{\$} \mathcal{A}_0(1^\lambda, 1^s, (\sigma_{\mathsf{PoW}}, \sigma_{\mathsf{DS}}))$. Then, it runs

$$(\ell_h, (U_{j,h})_{j \in [L_h] \setminus \{\ell_h\}}, m_h, w_h)_{h \in [t]} \xleftarrow{\$} \mathcal{A}_1^{\mathcal{O}^s_{(\sigma_{\mathsf{PoW}}, \sigma_{\mathsf{DS}}), U}}(1^\lambda, U, \mathsf{aux}).$$

  The adversary $\mathcal{A}_{\mathsf{DS},0}$ outputs $(\ell_h, (U_{j,h})_{j \in [L_h] \setminus \{\ell_h\}}, m_h)_{h \in [t]}$ along with the auxiliary information $\mathsf{aux}_{\mathsf{DS}}$ consisting of σ_{PoW}, $(w_h)_{h \in [t]}$, U and the queries $\mathcal{Q}$.
- $\mathcal{A}_{\mathsf{DS},1}$ receives $(z_h)_{h \in [t]}$ and $\mathsf{aux}_{\mathsf{DS}}$. It checks that the elements $(m_h)_{h \in [t]}$ are all distinct and not in $\mathcal{Q}$. It also checks that, for every $h \in [t]$

$$\mathsf{PoW.Verify}(\sigma_{\mathsf{PoW}}, z_h, w_h) = 1.$$

  If all these checks pass, $\mathcal{A}_{\mathsf{DS},1}$ outputs 1; otherwise, it outputs 0.

Note that in the real world execution of the distributed sampler game, $\mathcal{A}_{\mathsf{DS}}$ outputs 1 with probability $\delta(\lambda)$. On the other hand, in the ideal world, $\mathcal{A}_{\mathsf{DS}}$ outputs 1 with probability at most $\delta_1(\lambda)$. We conclude that $\delta_1(\lambda) \geq \delta(\lambda)/\eta(t,\lambda) - \nu(t,\lambda)$.

Now consider the PoW adversary $\mathcal{A}_{\mathsf{PoW}} = (\mathcal{A}_{\mathsf{PoW},0}, \mathcal{A}_{\mathsf{PoW},1})$:

- $\mathcal{A}_{\mathsf{PoW},0}$ receives σ_{PoW} from the challenger. It begins by sampling $(\sigma_{\mathsf{DS}}, \tau) \xleftarrow{\$} \mathsf{DS.Setup}(1^\lambda)$ and $\mathsf{aux} \xleftarrow{\$} \mathcal{A}_0(1^\lambda, 1^s, (\sigma_{\mathsf{PoW}}, \sigma_{\mathsf{DS}}))$. Finally, it outputs the auxiliary information $\mathsf{aux}_{\mathsf{PoW}}$ consisting of aux and τ.
- $\mathcal{A}_{\mathsf{PoW},1}$ receives $(z_h)_{h\in[t]}$ and $\mathsf{aux}_{\mathsf{PoW}}$. It computes $U \xleftarrow{\$} \mathsf{DS.Sim}_1(\tau, (z_h)_{h\in[t]})$. Finally, it runs

$$(\ell_h, (U_{j,h})_{j\in[L_h]\setminus\{\ell_h\}}, m_h, w_h)_{h\in[t]} \xleftarrow{\$} \mathcal{A}_1^{\widetilde{\mathcal{O}}}(1^\lambda, U, \mathsf{aux}),$$

  where $\widetilde{\mathcal{O}}$ is the oracle that, on input any tuple $(\ell, (U_j)_{j\in[L]\setminus\{\ell\}}, m)$, sets $U_\ell \leftarrow U$, computes $z \leftarrow \mathsf{DS.Sample}(\sigma_{\mathsf{DS}}, U_1, \ldots, U_L, m)$ and outputs the response of $\mathcal{O}^s_{\sigma_{\mathsf{PoW}}}$ when queried with z. Finally, $\mathcal{A}_{\mathsf{PoW},1}$ outputs $(w_h)_{h\in[t]}$.

Observe that $\mathcal{A}_{\mathsf{PoW}}$ wins the proof-of-work security game with probability at least $\delta_1(\lambda) \geq \delta(\lambda)/\eta(t,\lambda) - \nu(t,\lambda) > \epsilon(t,\lambda)$. Notice that $\mathcal{A}_{\mathsf{PoW},1}$ runs in time at most $t(\lambda)\cdot\beta(t,\lambda)\cdot r_{\mathsf{PoW}}(t,\lambda)$. We reached a contradiction. □

The following corollary formalizes the conditions the proof of work needs to satisfy to obtain a MV-SoW. The result is obtained by instantiating Theorem 3 with the hardness-preserving distributed samplers of Corollary 1 and Corollary 2.

Corollary 3. *Suppose there exists a $(d_0, d_1, \beta, \epsilon)$-moderately hard proof of work (see Appendix A.3) with challenge space $\{0,1\}^{n(\lambda)}$, where $\epsilon(t,\lambda) = 2^{-t\cdot n^\gamma}$ for a constant $\gamma > 0$. Assume also the existence of simulation-extractable NIZKs for all NP relations. Then, under the hardness of LWE with subexponential modulus-to-noise ratio or DDH, there exists a $(d_0, d_1, \beta', \mathsf{negl}(\lambda))$-moderately unforgeable multi-verifier signature of work, where $\beta'(s,\lambda,L) \geq \beta(s,\lambda)/2$ when s is a sufficiently large polynomial function of λ and L.*

Unfortunately, existing constructions of PoWs from fine-grained hardness assumptions [BGH+24a], fall short of the subexponential soundness required in Corollary 3. We note that given any function f that is moderately-hard (Def. 14) with parameters as required by Corollary 3 but *not* necessarily efficiently verifiable, can be turned into a proof of work via any subexponentially sound SNARG for $\mathcal{P}$. Thus, by the work of Choudhuri *et al.* [CJJ21], it suffices to assume the existence of this moderately (very) hard function and subexponentially secure LWE.

4 Permissionless Consensus from Multi-Verifier Signatures of Work

In this section we show how to achieve consensus in the permissionless setting assuming a CRS and the existence of commitments, digital signatures and a MV-SoW scheme. We start with additional details on the setting in which we analyze security.

4.1 Protocol Execution and Security Model

We will analyze our protocols in a concrete complexity, synchronous, and permissionless model, similarly to [BGH+24a].

The set of n parties $\{P_1, \ldots, P_n\}$ running the protocol is fixed, albeit unknown (both their number and network identities) subject to a condition $n \leq n_{\mathsf{max}}$ which is a protocol parameter. We restrict ourselves to consensus protocols that do not use the network identity or the number of parties n in the protocol logic (as this typifies permissionless consensus protocols, cf. [GKL15]).

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 the security parameter. The adversary is active/Byzantine and can corrupt up to t parties (statically) in order to break security (the exact value of t will be specified later).

The protocol advances in rounds and communication happens through a "diffusion" functionality $\mathcal{F}_{\mathsf{diff}}$. Messages sent by honest parties through $\mathcal{F}_{\mathsf{diff}}$ arrive to all other parties at the beginning of the next round.

On the other hand, the adversary at any point may decide in which order messages arrive to different parties and can send messages to an adaptively chosen *subset* of them. Communication is not authenticated, and thus parties cannot tell who was the sender of a message.

Given that in our setting parties are not aware of their total number, parties will be making use of the moderately hard primitive (MV-SoW) to ensure that their views about this number are somewhat consistent. As a result, and similarly to [BGH+24a] r_{rcv} (sufficient to verify all receiving MV-SoWs) and r_{sow} (sufficient for a single party to execute a MV-SoW), in order to allow parties to execute the relevant operations required by the protocol.

Each party (including corrupted parties) has a concrete upper bound of c computational steps per round, following the formulation in [GKP20a]. The adversary has an upper bound θ on the number of messages it can send per round (similarly to [AD14]).[7] In order to capture the fact that standard model assumptions for moderate hardness are not tight, we use a "slowdown" parameter β that can be imposed on adversarial parties. This is used in the following way: When an honest party requires r steps to compute a moderately hard function output, then no matter the strategy of the adversary $\beta \cdot r$ steps can at best produce at most one such output.

4.2 Partial PKI from Multi-Verifier Signatures of Work

In a (γ, ϵ)-*partial PKI*, each (honest) party, P_i, is associated with a run-time identity which consists of (a) public key for a signature scheme, pk_i (for which the party knows the corresponding secret key sk_i), as well as (b) an identifier $r_i \in [m(n)]$, for some polynomial $m(n)$ which is a public parameter.

What makes the PKI 'partial' is that honest parties have a set of possibly different "valid" identities that they recognize, G_i. (note G_i includes P_i's run-time identity, (pk_i, r_i).) Although the honest parties do not necessarily agree on these sets, the sets have certain properties that ensure a high overlap. Specifically, let H denote the set of run-time identities corresponding to honest parties. We require the following with respect to the parameter γ:

1. (Honest identifiers are almost always valid.) $\forall i : |H \cap G_i| \geq (1 - \epsilon)|H|$.
2. (Overall, honest identifiers are not too few.) $|H| \geq (1 - \gamma)|\cup_i G_i|$.

Next, we describe a protocol that enables a set of participants communicating over $\mathcal{F}_{\mathsf{diff}}$ to set up a partial PKI, as described above. The key tool is the MV-SoW scheme (Section 3): Participants initially sample and exchange keys for the primitive. In round 2 of the protocol the parties perform the signature of work on a *commitment* of their public key and chosen identifier which is selected from a (small polynomial) range of values $[m]$. The round concludes with parties sending their MV-SoWs to each other. In the next round, parties collect all signatures and assemble a list of committed keys. Upon completion of this step, each party transmits the opening of their commitment. In the final, termination round, the parties collect all commitment openings and attempt to open the commitments in the list of committed keys. Public keys that are successfully recovered in this way, are entered in the partial PKI set of valid keys for each party.

Construction 4: PARTIALPKISETUP: PARTIAL PKI FROM MV-SoW

Public Parameters: m (m is the domain of the random identifiers in the partial PKI, and should be set such that $m = 4n_{\mathsf{max}}^2$; n_{max} is an upper bound on the number of parties; round durations r_{rcv} (sufficient to verify all receiving SoWs) and r_{sow} (sufficient for a single party to execute a SoW).

Cryptographic Primitives: Digital signature DSig: $\langle \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify} \rangle$; Multi-Verifier Signature of Work MVSoW: $\langle \mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify} \rangle$; commitment scheme Com: $\langle \mathsf{Gen}, \mathsf{Commit}, \mathsf{Verify} \rangle$

[7] This choice is made in [AD14] to avoid deniable-of-service (DoS) attacks that may deplete the round-bounded computational power of the honest parties. We remark that handling DoS attacks this way is not the only option. See [BGH+24b] for more details.

Precondition: $\mathsf{MVSoW.Setup}(1^\lambda)$ is executed to produce σ_0, as well as $\mathsf{Com.Setup}(1^\lambda)$ producing σ_1. The CRS (σ_0, σ_1) is available to all parties.

Party P_i:

Round 1: Exchange keys (duration r_{rcv}).
- Run $\mathsf{Dsig.Gen}(1^\lambda)$ to produce $\mathsf{pk}_i, \mathsf{sk}_i$ and a random number r_i from $[m]$.
- Run $\mathsf{MVSoW.Gen}(1^\lambda, 1^m)$ to produce K_i. Sends K_i to $\mathcal{F}_{\mathsf{diff}}$.

Round 2: Perfom signature of work (duration r_{sow}).
- Receives from $\mathcal{F}_{\mathsf{diff}}$ messages $\{K_i\}$ and populates the set F_i.
- Computes $v_i := \mathsf{MVSoW.Sign}(\psi_i)$, where $\psi_i = \mathsf{Com.Commit}(\mathsf{pk}_i, r_i)$ to obtain v_i.
- Sends (ψ_i, v_i) to $\mathcal{F}_{\mathsf{diff}}$.

Round 3: Collect signatures of work (duration r_{rcv}).
- Receives from $\mathcal{F}_{\mathsf{diff}}$ a sequence of messages (ψ, v), and populates the set C_i with those pairs that satisfy $\mathsf{MVSoW.Verify}(\psi, v) = \mathsf{true}$.
- Sends to $\mathcal{F}_{\mathsf{diff}}$ the opening ρ_i of its own commitment ψ_i.

Round 4: Termination (duration r_{rcv}).
- Receives from $\mathcal{F}_{\mathsf{diff}}$ openings to commitments in C_i.
- For each such opening (pk', r', ρ'), it matches it with corresponding commitment ψ', and if $\mathsf{Com.Verify}(\psi', \rho') = \mathsf{true}$, it adds the corresponding key (pk', r') to G_i. If two or more keys are assigned the same identifier, only one is kept arbitrarily.

We next prove the security of our construction. The main ideas behind the security proof are (i) the MV-SoW throttles the ability of the adversary to register valid public keys in the lists of honest parties in proportion to its total computational power, and (ii) the commitment prevents the adversary from adaptively choosing the same identities as the honest participants. Formally:

Theorem 4. *Protocol* `PartialPKISetup` *realizes a (γ, ϵ)-partial PKI assuming a commitment scheme that is perfectly binding and computationally hiding with distinguishing advantage ϵ_{hid}, as well as a $(t_{\mathsf{max}}, n_{\mathsf{max}}, \beta, \mathsf{negl})$-moderately unforgeable* MV-SoW *scheme, as long as $t \leq t_{\mathsf{max}}$ is the number of parties controlled by the adversary, $n \leq n_{\mathsf{max}}$ is the total number of parties, and it holds that $t < \gamma n$, negl is a negligible function and β is the slowdown factor.*

We start by proving a simple combinatorial lemma that ensures that all but a small fraction of honest parties will acquire a unique identifier via the above partial PKI protocol. This ensures that all but a small ϵ fraction of honest keys are retained in the valid set of keys of each honest participant.

Lemma 1. *Let $\epsilon \in (0, 1)$ be a constant. Consider randomly tossing h balls into m bins of which z are tainted, where $2h(h + z) < m$. Let X denote the number of balls that end up in an untainted bin that contains no other balls. Then,*

$$\Pr[X < (1 - \epsilon)h] \leq \exp(-\Omega(h)).$$

Proof. Let U_i denote the event that ith ball lands in a unique untainted bin. Notice that $\Pr[U_i] = (1 - z/m)(1 - 1/m)^{n-1}$. Using a Taylor expansion ($\ln(1 + x) \geq x - x^2$ for $x \geq -.68$) and the fact that $h < m$, we can show

$$\begin{aligned}
\ln(\Pr[U_i]) &= (n-1)\ln(1 - 1/m) + \ln(1 - z/m) \\
&\geq (h-1)\left(-\frac{1}{m} - \frac{1}{m^2}\right) + \left(-\frac{z}{m} - \frac{z^2}{m^2}\right) \\
&\geq -\frac{h+z}{m} - \frac{h+z^2}{m^2}.
\end{aligned}$$

Thus

$$\Pr[U_i] \geq \exp\left(-\frac{h+z}{m} - \frac{h+z^2}{m^2}\right).$$

Applying $e^x \geq 1 + x$,

$$\Pr[U_i] \geq 1 - \frac{h+z}{m} - \frac{h+z^2}{m^2}.$$

Now let U be the random variable counting the number of balls in unique untainted bins. Abusing notation, take U_i as the corresponding indicator random variable, $U = \sum_{i=1}^{h} U_i$. Thus, by linearity of expectation and the constraint $2h^2 < m$, we have $\mathbb{E}[U]$ is equal to:

$$\mathbb{E}[\sum_{i=1}^{h} U_i] = \sum_{i=1}^{h} \mathbb{E}[U_i] \geq h(1 - \frac{h+z}{m} - \frac{h+z^2}{m^2}) \geq h - \frac{h(h+z)}{m} - \frac{h(h+z^2)}{m^2} \geq h - 1.$$

Next, we will use McDiarmid's inequality [McD97] to analyze the probability that U is significantly smaller than its expected value.

Lemma 2 ([McD97]). *Let $f : \mathcal{X}^n \to \mathbb{R}$ such that for any $x_1, \ldots, x_n \in \mathcal{X}$, any $i \in [n]$, and any $x_i' \in \mathcal{X}$*

$$|f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n)| \leq c.$$

Then, for any random variables $X_1, \ldots, X_n$ and any $\epsilon > 0$,

$$\Pr[\mathbb{E}[f(X_1, \ldots, X_n)] - f(X_1, \ldots, X_n) \geq \epsilon] \leq \exp\left(\frac{-2\epsilon^2}{nc^2}\right).$$

In more detail, let $\mathcal{X} = [m]$, X_i uniform over $[m]$ and f the function that counts the number of distinct integers that are not tainted (e.g., f can be parameterized by the locations of the z tainted locations). From our analysis above, $\mathbb{E}[f(X_1, \ldots, X_h)] \geq h - 1$. Moreover, for any $x_1, \ldots, x_h, x_i' \in [m]$, $|f(x_1, \ldots, x_i, \ldots, x_h) - f(x_1, \ldots, x_i', \ldots, x_h)| \leq 2$ (because changing one integer changes the count of distinct integers by at most 2: two unique integers become identical or two identical integers become unique). Thus, we can apply Lemma 2 to conclude

$$\Pr[U \leq (1-\epsilon)h] = \Pr[f(X_1, \ldots, X_h) \leq (1-\epsilon)h] \leq \exp(\frac{-(\epsilon h - 1)^2}{4h}) = \exp(-\Omega(h)).$$

This concludes the proof of Lemma 1. □

Proof. (of Theorem 4 — sketch) Consider an execution of the protocol in the presence of an adversary $\mathcal{A}$ that violates the (γ, ϵ)-partial PKI conditions with non-negligible probability. We first substitute all honest commitments with commitments to a predefined value incurring a distinguishing difference of $n_{\mathsf{max}}\epsilon_{\mathsf{hid}}$. Now observe that in the resulting execution, due to the perfectly binding property of the underlying commitment, the choices of adversarial identifiers are independent of the choices of honest parties, as a result we can considered those to be the tainted positions in the context of Lemma 1. It follows that the number of honest parties with unique identifiers will be at least $(1-\epsilon)h$. Suppose, for the sake of contradiction, that it holds that $|H| < (1-\gamma)|\cup_i G_i|$. We know that $|H| > (1-\gamma)n$ and, as a result, $|\cup_i G_i| - |H| > n - h = t$, hence the number of SoWs produced by the adversary exceeds t, which violates moderate unforgeability. □

4.3 Phase King-style Consensus from a Partial PKI

We follow the graded consensus to full consensus approach, suitably adapting it to the partial PKI setting. We restrict ourselves to protocols where honest parties sign any message they send and ignore any message that they cannot fully verify, either due to the fact that the signer is unknown (to the recipient) or because the message contains a reference to a signed message by an unknown signer. We call such protocols "simple."

Graded consensus. First, recall the definition of graded consensus (cf. [FM97]).

Definition 6 (Graded Consensus). *Let Π be a protocol where each party P_i inputs a bit $b_i \in \{0,1\}$ and outputs a pair of bits (b'_i, g_i) consisting of an 'output' bit $b'_i \in \{0,1\}$ and a 'grade' $g_i \in \{0,1\}$. We say Π achieves* graded consensus *if the following two properties are satisfied:*

- **Validity:** *If all honest parties input the same bit b, then all honest parties output $(b,1)$.*
- **Graded Agreement:** *If some honest party outputs $b'_i \in \{0,1\}$ and $g_i = 1$, then all honest parties output $b'_j = b'_i$ and $g_j \in \{0,1\}$.*

We next demonstrate that a simple, one-message approach achieves graded consensus in the partial PKI setting. In this protocol, a party simply counts votes (from recognized parties in its set G_i) and uses the majority as the output. If there is a strong majority, the grade is set to 1, and otherwise it is set to 0.

Construction 5: GradedConsensus: Graded Consensus from Partial PKI

On input b_i, party P_i (with set of identities G_i) does the following:

- Send b_i to $\mathcal{F}_{\mathsf{diff}}$.
- Receive votes $\hat{b}_j$ from $\mathcal{F}_{\mathsf{diff}}$. Ignore votes from parties not in G_i (and any other messages and any party that votes twice). Let $\hat{b}_i := \frac{\sum_j \hat{b}_j}{|G_i|}$.
- Output (b'_i, g_i) where $g_i = 0$ if and only if $\hat{b}_i \in [1/3, 2/3]$ and $b'_i = 1 \iff \hat{b}_i > 1/2$.

Lemma 3. *Protocol* `GradedConsensus` *achieves graded consensus in the (γ,ϵ)-partial PKI setting, for $\gamma < 1/4(1-3\epsilon)$ and $\epsilon \in (0, 1/3)$.*

Proof. We first prove Validity. Suppose that all honest parties have as input the same input bit b. The number of inputs the adversary can contribute in the view of any party is $x < 3h$, where h is the number of honest parties. It follows that more than $3/4 > 2/3$ of the votes received by all honest parties will be for b, and hence all honest parties will output $(b,1)$.

To show that Graded Agreement holds, suppose for the sake of contradiction that honest parties P_i and P_j output $(b,1)$ and $(1-b,0)$, respectively, for some b. We will assume $b = 1$, as the other case is symmetric. Let h_0 and h_1 denote the number of honest votes for 0 and 1 respectively, and let $h = h_0 + h_1$. Let x denote the number of adversarial votes for 1 sent to P_i, and y denote the number of adversarial votes for 0 sent to P_j. Using our notation, we can rewrite our assumption via the following inequalities:

$$h_0 + x > 2/3(h + x),$$
$$h_1 + y \geq 1/2(h + y),$$

from where we can deduce that $h + x + y > \frac{7}{6}h + \frac{2}{3}x + \frac{1}{2}y$ which implies $2x + 3y > h$.

Given the (γ,ϵ)-partial PKI properties, we have $h \geq (1-\gamma)(h(1-\epsilon)+x+y)$, which implies $(\gamma/(1-\gamma)+\epsilon)h \geq x+y$. On the other hand, observe that by the choice of γ, ϵ, we have $\gamma/(1-\gamma) < 1/3-\epsilon$, and hence $3x+3y < h$, which results in a contradiction. □

Full consensus with a partial PKI. Next, we show how to use our graded consensus protocol to reach consensus with only a partial PKI.[8]

In the protocol, each party maintains a current "guess" v_i as to what the consensus output should be. The parties iteratively run the graded consensus protocol using their (current) values, for m phases, where $m = 4n^2_{\mathsf{max}}$ is the domain of the random identifiers in the partial PKI.[9] Such identifier (if it exists) corresponding to the phase number determines the possible "king" for phase r. If at any point during the

[8] Recall that the standard definition of consensus is stronger than that of graded consensus (Def. 6): There is no grade, and Agreement among honest parties is complete.

[9] Recall from Section 4.2 that n_{max} is an upper bound on the number of parties. The specific value m is set to follows from Lemma 1.

execution a party isn't confident in the output of the graded consensus (i.e., grade $g_i = 0$) and (from its perspective) there is a *unique* king for that round (i.e., exactly one good/valid party associated with round r), then the party will update its current guess v_i to be what was received from the current king in the graded consensus protocol.

The analysis of this protocol is very similar to that of the classical phase king protocol [BGP89] except for one difference: There will be rounds where honest parties may disagree about the identity of the king or where the king **may not even exist**; however, there will also be rounds where an honest party will be recognized as king by all other honest parties, and this will allow all honest parties to converge to consensus after this round, if they haven't already. Moreover, once parties reach consensus in some phase, consensus persists regardless of who is the king after that phase.

Construction 6: PARTIALPHASEKING: PHASE KING WITH A PARTIAL PKI

Public Parameter: m (**Note:** m is the domain of the random identifiers in the partial PKI; it should be set such that $m = 4n_{\mathsf{max}}^2$.)

On input b_i, party P_i (with set of identities G_i) does the following:

- Set $v_i := b_i$.
- For phases $r = 1, \ldots, m$, do the following:
  1. Run `GradedConsensus` with v_i as input to get $(\hat{b}_i, g_i)$.
  2. If $r = r_i$ (party P_i is currently king), send $\hat{b}_i$ to $\mathcal{F}_{\mathsf{diff}}$ signed with pk_i.
  3. Receive from $\mathcal{F}_{\mathsf{diff}}$ the pair $(\hat{b}, \hat{\sigma})$.
  4. Update v_i as follows:
     (a) If $g_i = 1$, set $v_i := \hat{b}_i$;
     (b) else, if $g_i = 0$ and there is a unique identity in G_i such that $r_j = r$, set $v_i := \hat{b}_r$ as long as $(\hat{b}, \hat{\sigma})$ is a valid message signature pair under pk_j.
     (c) else, set $v_i := \hat{b}_i$.
  5. Output v_i.

To see why the protocol works, we first demonstrate that with overwhelming probability, there will be some honest party associated with a round r that is not associated with any other identity recognized by any honest party.

Let h denote the number of honest parties and let R denote the set of values r_i associated with these parties. We are interested in bounding the size of R. In particular, we can apply Lemma 1 to deduce that for any constant ϵ (provided m is sufficiently large), $Pr[|R| < (1-\epsilon)h] \leq \exp(-\Omega(h))$.

In what follows, we assume this high probability event has occurred: I.e., at least one honest party is associated with a unique phase (not associated with any other identifier) r.

First, note that if the current values, v_i, of the honest parties are ever all b, then graded consensus will output $(b, 1)$ to all parties. Hence, the current values will remain the same in the next phase and their current values will continue to agree. From this we can immediately deduce that if all honest parties have input b, the values v_i will all be initialized to b and will remain b. Hence all parties will output b.

Now, to see that all honest parties will output the same value regardless, consider the phase r associated with a unique honest party P_i. Because this party is honest, its run-time identity is in every set of good/valid identifiers, G_j. We now split the analysis into two cases and show that in either case, parties will be in agreement after phase r:

1. *Some honest party gets output* $(b, 1)$ *(for some* b*) from graded consensus.* In this, case all parties, including P_j, will have output $(b, 0)$ or $(b, 1)$. Thus, P_j will diffuse b and all parties will set v_i to b (either because they got grade 1 with b or they are following P_j).
2. *No honest party gets output* $(b, 0)$ *(for some* b*) from graded consensus.* In this case, P_j will get $(b, 0)$ as output for some b and will send b. As no party has grade 1 and all parties recognize P_j as the unique king, all parties will set v_i to b.

Putting the above together, we obtain:

Theorem 5. *Protocol* `PartialPhaseKing` *achieves consensus in the* (γ, ϵ)*-partial PKI setting, for* $\gamma < 1/4(1-3\epsilon)$ *and* $\epsilon \in (0, 1/3)$.

Combining the above Theorem with Theorem 4 and Corollary 3, we obtain our permissionless consensus protocol in the CRS model (and without a random oracle nor a beacon).

4.4 Impossibility of Consensus in the Partial PKI Setting for $\gamma \geq 1/4$

Can the resiliency bound in the partial PKI setting be improved? Next, we prove that tolerating a 1/4 (or greater) threshold is infeasible for simple protocols, hence establishing that the bound achieved by `PartialPhaseKing` from the previous section is essentially tight for this class of protocols.

Theorem 6. *There is no "simple" consensus protocol in the partial PKI setting with* $\gamma = 1/4, \epsilon = 0$ *satisfying both Agreement and Validity.*

5 Extensions and Open Questions

We conclude the paper by discussing a number of directions that remain open.

The theoretical limit for resiliency (according to the known lower bound for consensus [Fit02]) would be (less than) 1/2; so achieving it in our setting constitutes an interesting question for future work. We surmise that new techniques would be needed for this that go beyond multi-verifier signatures of work. Indeed, one key "deficiency" of MV-SoWs is that they are not transferable: One honest party could perceive a proof as valid, while a different honest party would not. With transferable proofs one could come closer to building a *ranked* partial PKI, from which we know from [AD15] that we can get consensus with 1/2 resilience via an adaptation of the Dolev-Strong protocol [DS83].

As mentioned above, one drawback of our protocol is the high communication complexity, as messages are at least as long as the lower bound on the number of parties. A potential means for mitigating this would be to use a *seeded proof of work* as defined by Ball *et al.* [BGH+24a]. Unfortunately, the stretch and soundness of their construction is incompatible with the loss encountered with distributed samplers.

A second drawback is our consensus protocol's high round complexity, as the number of rounds also scales with the bound on the number of parties. To remedy this, one could hope to replace the deterministic Phase-King protocol with a randomized, expected-constant-round protocol (e.g., [FG03,KK09]), for which the partial PKI should allow the parties to toss an (even information-theoretic!) asynchronous oblivious common coin (cf. [Fel89,Nie02,CFG+23]). Even though adapting an existing asynchronous agreement protocol to our partial PKI setting might seem straightforward, the parties' PKI uncertainty may take a toll on the resiliency threshold. Indeed, suppose each party runs the protocol treating their G_i set as the total party set and uses an α threshold for the adversary in order to determine quorum sizes. A common objective is that any two quorums of two honest parties should contain at least one honest party. Recall that the potential of the partial PKI is to make two honest parties disagree on their views of keys, something that implies that one party's quorum can be as large as $(1-\alpha)n$, while another party's quorum can be as small as $(1-\alpha)(1-\gamma)n$. Combining these two facts with the requirement to contain at least one honest party's key, we get that $\gamma < (1-2\alpha)/(2-\alpha)$. In the particular case of $\alpha = 1/3$, the choice for γ would be $\gamma = 1/5$.

The sequential composition of multiple instances of our consensus problem is another interesting direction for future work. Of particular interest also would be to eliminate the need for continuously providing an upper bound on the number of parties and allow the parties/protocol to dynamically adjust the PoW difficulty, as done in the Bitcoin blockchain and other related protocols that operate in the setting of dynamic availability where participation fluctuates over time [GKL17,BGK+18,LPR24,GKS25].

Another interesting connection is to protocols with "asymmetric trust" (cf. [DDFN07,LGM19,ACKV25]), where different parties have their own personal view on the set of acceptable quorums. We also remark that our partial PKI setting is "orthogonal" to the partially authenticated setting considered in [LLSW26].

In Section 3, we observe that a sufficiently moderately hard function can be combined with subexponentially secure LWE to yield a proof of work, constructing this moderately (very) hard function from standard

complexity conjectures is an interesting open problem in hardness amplification. For example, can one show a (tight) "dream" strong direct product theorem for some function assuming, say, NSETH?

Finally, two pivotal questions for future work are to (a) remove all setup and show feasibility in the plain model (without a CRS), and (b) to minimize assumptions. Accomplishing either of these goals will require novel ideas.

Acknowledgments

D.A. was supported by a fellowship from the Simons Institute for the Theory of Computing, and was partly conducted at the institute. M.B. was supported in part by NSF Award CCF-2443735. J.G. was supported in part by NSF Award CCF-2055694. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work was also supported by Input|Output (iog.io) through their funding of the Edinburgh Blockchain Technology Lab.

References

ACKV25. Ignacio Amores-Sesar, Christian Cachin, Simon Holmgaard Kamp, and Juan Villacis. Weaker assumptions for asymmetric trust. In Andrei Arusoaie, Emanuel Onica, Michael Spear, and Sara Tucci Piergiovanni, editors, *29th International Conference on Principles of Distributed Systems, OPODIS 2025, Iaşi, Romania, December 3-5, 2025*, LIPIcs, pages 8:1–8:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

AD14. Marcin Andrychowicz and Stefan Dziembowski. Distributed cryptography based on the proofs of work. Cryptology ePrint Archive, Report 2014/796, 2014. `http://eprint.iacr.org/`.

AD15. Marcin Andrychowicz and Stefan Dziembowski. Pow-based distributed cryptography with no trusted setup. In *Annual cryptology conference*, pages 379–399. Springer, 2015.

ADIN24. Damiano Abram, Jack Doerner, Yuval Ishai, and Varun Narayanan. Constant-round simulation-secure coin tossing extension with guaranteed output. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part V*, volume 14655 of *LNCS*, pages 122–154. Springer, Cham, May 2024.

AFOS25. Damiano Abram, Serge Fehr, Maciej Obremski, and Peter Scholl. On the impossibility of actively secure distributed samplers. In Benny Applebaum and Huijia (Rachel) Lin, editors, *TCC 2025, Part IV*, volume 16271 of *LNCS*, pages 547–581. Springer, Cham, December 2025.

AJK05. James Aspnes, Collin Jackson, and Arvind Krishnamurthy. Exposing computationally-challenged byzantine impostors. Technical report, Technical Report YALEU/DCS/TR-1332, Yale University Department of Computer . . . , 2005.

ASY22. Damiano Abram, Peter Scholl, and Sophia Yakoubov. Distributed (correlation) samplers: How to remove a trusted dealer in one round. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 790–820. Springer, Cham, May / June 2022.

AWZ23. Damiano Abram, Brent Waters, and Mark Zhandry. Security-preserving distributed samplers: How to generate any CRS in one round without random oracles. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 489–514. Springer, Cham, August 2023.

BGH+24a. Marshall Ball, Juan A. Garay, Peter Hall, Aggelos Kiayias, and Giorgos Panagiotakos. Towards permissionless consensus in the standard model via fine-grained complexity. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part II*, volume 14921 of *LNCS*, pages 113–146. Springer, Cham, August 2024.

BGH+24b. Marshall Ball, Juan A. Garay, Peter Hall, Aggelos Kiayias, and Giorgos Panagiotakos. Towards permissionless consensus in the standard model via fine-grained complexity. *IACR Cryptol. ePrint Arch.*, page 637, 2024.

BGI+01. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Berlin, Heidelberg, August 2001.

BGK+18. Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 913–930. ACM Press, October 2018.

BGP89. Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Towards optimal distributed consensus (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 410–415. IEEE Computer Society, 1989.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, pages 62–73. ACM, 1993.

CFG+23. Ran Cohen, Pouyan Forghani, Juan A. Garay, Rutvik Patel, and Vassilis Zikas. Concurrent asynchronous byzantine agreement in expected-constant rounds, revisited. In Guy N. Rothblum and Hoeteck Wee, editors, *Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part IV*, volume 14372 of *Lecture Notes in Computer Science*, pages 422–451. Springer, 2023.

CJJ21. Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for NP from standard assumptions. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 394–423, Virtual Event, August 2021. Springer, Cham.

DDFN07. Ivan Damgård, Yvo Desmedt, Matthias Fitzi, and Jesper Buus Nielsen. Secure protocols with asymmetric trust. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, Lecture Notes in Computer Science, pages 357–375. Springer, 2007.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

DS83. Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

Fel89. Paul Feldman. Asynchronous byzantine agreement in constant expected time. Unpublished manuscript, 1989.

FG03. Matthias Fitzi and Juan A. Garay. Efficient player-optimal protocols for strong and differential consensus. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *PODC*, pages 211–220. ACM, 2003.

Fit02. Matthias Fitzi. *Generalized communication and security models in Byzantine agreement*. PhD thesis, ETH Zurich, 2002.

FM97. Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.

GGH+13. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

GK20. Juan A. Garay and Aggelos Kiayias. Sok: A consensus taxonomy in the blockchain era. In Stanislaw 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 284–318. Springer, 2020.

GKL15. Juan A. Garay, Aggelos Kiayias, and Nikos 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.

GKL17. Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol with chains of variable difficulty. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 291–323. Springer, Cham, August 2017.

GKLP18. Juan A. Garay, Aggelos Kiayias, Nikos Leonardos, and Giorgos Panagiotakos. Bootstrapping the blockchain, with applications to consensus and fast PKI setup. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 465–495. Springer, 2018.

GKP20a. Juan A. Garay, Aggelos Kiayias, and Giorgos Panagiotakos. Blockchains from non-idealized hash functions. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography*, pages 291–321, Cham, 2020. Springer International Publishing.

GKP20b. Juan A. Garay, Aggelos Kiayias, and Giorgos Panagiotakos. Consensus from signatures of work. In Stanislaw 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.

GKS25. Juan A. Garay, Aggelos Kiayias, and Yu Shen. State machine replication among strangers, fast and self-sufficient. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part II*, volume 16001 of *Lecture Notes in Computer Science*, pages 3–36. Springer, 2025.

JLS21. Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 60–73. ACM Press, June 2021.

KK09. Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. *Journal of Computer and System Sciences*, 75(2):91 – 112, 2009.

KRS25. Dmitry Khovratovich, Ron D. Rothblum, and Lev Soukhanov. How to prove false statements: Practical attacks on fiat-shamir. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 3–26. Springer, Cham, August 2025.

LGM19. Giuliano Losa, Eli Gafni, and David Mazières. Stellar consensus by instantiation. In Jukka Suomela, editor, *33rd International Symposium on Distributed Computing, DISC 2019, Budapest, Hungary, October 14-18, 2019*, LIPIcs, pages 27:1–27:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

LLSW26. Christoph Lenzen, Julian Loss, Kecheng Shi, and Benedikt Wagner. Byzantine consensus in the partially authenticated setting. *IACR Cryptol. ePrint Arch.*, 2026:470, 2026.

LPR24. Andrew Lewis-Pye and Tim Roughgarden. Permissionless consensus. arXiv 2304.14701 [cs.DC], 2024.

LSP82. Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

McD97. Colin McDiarmid. Centering sequences with bounded differences. *Comb. Probab. Comput.*, 6(1):79–86, 1997.

MP13. Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 21–39. Springer, Berlin, Heidelberg, August 2013.

Nak08. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf, 2008.

Nie02. Jesper Buus Nielsen. A threshold pseudorandom function construction and its applications. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 401–416. Springer, 2002.

PSL80. Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

PSS17. Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 643–673, 2017.

PW08. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 187–196. ACM Press, May 2008.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

Sch90. Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, 1990.

A Definitions and Building Blocks

In this section, we recall basic assumptions and primitives we use in this work.

A.1 Computational Assumptions

We recall the two main computational assumptions on which our constructions are based: Decisional Diffie-Hellman (DDH) [DH76] and Learning with Errors (LWE) [Reg05].

Definition 7 (Decisional Diffie-Hellman). *Let $\mathcal{G} = (\mathcal{G}_\lambda)_{\lambda\in\mathbb{N}}$ be a family of cyclic groups of order $p(\lambda)$. Let $g = g_\lambda$ be a generator for $\mathcal{G}_\lambda$. The decisional Diffie-Hellman assumption (DDH) over $\mathcal{G}$ states that, for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ for which*

$$\left|\Pr\left[\mathcal{A}(1^\lambda, g^a, g^b, g^{a\cdot b}) = 1 \middle| a, b \xleftarrow{\$} \mathbb{Z}_p\right] - \Pr\left[\mathcal{A}(1^\lambda, g^a, g^b, g^c) = 1 \middle| a, b, c \xleftarrow{\$} \mathbb{Z}_p\right]\right| \leq \mathsf{negl}(\lambda).$$

Definition 8 (Learning with Errors). *Let $k = k(\lambda)$ and $m = m(\lambda)$ be polynomial functions. Let $q = q(\lambda)$ be a (not-necessarily) polynomial modulus and let χ_s be a distribution over $\mathbb{Z}$ having standard deviation $s = s(\lambda)$. The learning with errors assumption (LWE) relative to (k, m, q, χ_s) states that, for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ for which*

$$\left|\Pr\left[\mathcal{A}(1^\lambda, \mathbf{M}, \mathbf{s}^\intercal\cdot\mathbf{M} + \mathbf{e}^\intercal) = 1 \middle| \begin{array}{l}\mathbf{M} \xleftarrow{\$} \mathbb{Z}_q^{k\times m}\\ \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^k\\ \mathbf{e} \xleftarrow{\$} \chi_s^m\end{array}\right] - \Pr\left[\mathcal{A}(1^\lambda, \mathbf{M}, \mathbf{v}^\intercal) = 1 \middle| \begin{array}{l}\mathbf{M} \xleftarrow{\$} \mathbb{Z}_q^{k\times m}\\ \mathbf{v} \xleftarrow{\$} \mathbb{Z}_q^m\end{array}\right]\right| \leq \mathsf{negl}(\lambda).$$

We call $\alpha(\lambda) := s(\lambda)/q(\lambda)$ the modulus-to-noise ratio.

A.2 Signature Schemes

Next, we recall the syntax and correctness and security properties for signature schemes.

Definition 9 (Signature scheme). *A signature scheme consists of a triple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *with the following syntax:*

$\mathsf{KeyGen}(1^\lambda)$**:** *The key generation algorithm takes as input the security parameter and outputs a pair* $(\mathsf{sk}, \mathsf{pk})$ *where* sk *is a secret key and* pk *is a verification key.*

$\mathsf{Sign}(\mathsf{sk}, m)$**:** *The signature algorithm takes as input the secrte key* sk *and a message m. The output is a signature s.*

$\mathsf{Verify}(\mathsf{pk}, m, s)$**:** *The verification algorithm takes as input a public key* pk*, a message m and a signature s. The output is a bit b, which represented whether the signature has been accepted or rejected.*

Definition 10 (Correctness). *A signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is correct if, for every $\lambda \in \mathbb{N}$ and every message m, we have*

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, m, s) = 1 \middle| \begin{array}{l}(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)\\ s \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, m)\end{array}\right] = 1.$$

Definition 11 (Unforgeability). *A signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is unforgeable if, for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that, for every $\lambda \in \mathbb{N}$, we have*

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{pk}, m, s) = 1\\ m \notin \mathcal{Q}\end{array} \middle| \begin{array}{l}(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)\\ (m, s) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{sk}}}(1^\lambda, \mathsf{pk})\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Above, $\mathcal{O}_{\mathsf{sk}}$ denotes the oracle that, on input any message m, replies with $s \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, m)$. The oracle can be queried by the adversary an unbounded number of times. We use $\mathcal{Q}$ to denote the set of queries issued by the adversary.

A.3 Proofs of Work

In this section, we recall the syntax, correctness and security of proofs of work. This primitive specifies how to generate puzzles that, in order to be solved, require a significant use of computational resources. On the other hand, once the solution is known, this can be easily verified.

Definition 12 (Proof of Work). *A proof of work consists of a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Solve}, \mathsf{Verify})$ *with the following syntax:*

$\mathsf{Setup}(1^\lambda, 1^s)$**:** *The setup algorithm takes as input the security parameter λ and the time parameter s. The output is a CRS σ.*

$\mathsf{Gen}(1^\lambda)$**:** *The generation algorithm takes as input the security parameter and outputs and outputs a proof-of-work challenge z.*

$\mathsf{Solve}(1^s, \sigma, z)$**:** *The resolution algorithm takes as input the time parameter s, the CRS σ and the proof-of-work challenge z. The output is a solution w. We denote the running time of this algorithm by $r_{\mathsf{PoW}}(s, \lambda)$.*

$\mathsf{Verify}(\sigma, z, w)$**:** *The verification algorithm takes as input the CRS σ, a proof-of-work challenge z and a candidate solution w. The output is a bit b representing whether the solution w has been accepted or rejected. We require that this algorithm runs in time* $\mathsf{poly}(\lambda, \log s)$.

Definition 13 (Correctness). *We say that a proof of work* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Solve}, \mathsf{Verify})$ *is correct if, for every $\lambda, s \in \mathbb{N}$, we have that*

$$\Pr\left[\mathsf{Verify}(\sigma, z, w) = 1 \;\middle|\; \begin{array}{l} \sigma \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^s) \\ z \xleftarrow{\$} \mathsf{Gen}(1^\lambda) \\ w \leftarrow \mathsf{Solve}(1^s, \sigma, z) \end{array}\right] = 1.$$

We opt for a multi-instance security definition: solving $t \leq d_0$ puzzles requires a use of resources proportional to t. We ask this property to hold even if the adversary has given oracle access to a resolution oracle that can be queried at most d_1 times (if we want to give unbounded access to the oracle, we can simply set $d_1 = 2^\lambda$).

Definition 14 (Moderate Hardness). *Consider a proof of work* $\mathsf{PoW} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Solve}, \mathsf{Verify})$. *Let $d_0 = d_0(\lambda)$ and $d_1 = d_1(\lambda)$ be polynomial functions, and let $\epsilon : \mathbb{N} \times \mathbb{N} \to [0, 1]$ and $\beta : \mathbb{N} \times \mathbb{N} \to [0, 1]$ be real functions. Let $\mathcal{O}_\sigma^s$ be an oracle that on input a proof-of-work challenge z, responds with* $\mathsf{Solve}(1^s, \sigma, z)$. *We say that a PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ is (t, d_1, β)-admissible if $\mathcal{A}_1$ runs in time $t(\lambda) \cdot \beta(s, \lambda) \cdot r_{\mathsf{PoW}}(s, \lambda)$, makes at most $d_1(\lambda)$ queries to $\mathcal{O}_\sigma^s$ and always outputs t candidate solutions $(w_h)_{h \in [t]}$.*

The proof of work is $(d_0, d_1, \beta, \epsilon)$-moderately hard if, for every (t, d_1, β)-admissible adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ with $t \leq d_0$, and every sufficiently large $\lambda, s \in \mathbb{N}$, we have

$$\Pr\left[\forall h \in [t] : \mathsf{Verify}(\sigma, z_h, w_h) = 1 \;\middle|\; \begin{array}{l} \sigma \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^s) \\ \mathsf{aux} \xleftarrow{\$} \mathcal{A}_0(1^\lambda, 1^s, \sigma) \\ \forall h \in [t] : z_h \xleftarrow{\$} \mathsf{Gen}(1^\lambda) \\ (w_h)_{h \in [t]} \xleftarrow{\$} \mathcal{A}_1^{\mathcal{O}_\sigma^s}((z_h)_{h \in [d_0]}, \mathsf{aux}) \end{array}\right] \leq \epsilon(t, \lambda).$$

A.4 Simulation Extractable Non-Interactive Zero-Knowledge

We finally recall the definition of non-interactive zero-knowledge proof (NIZK). This primitive allows a prover to convince a verifier about the validity of an NP statement without leaking any additional information about the witness.

Definition 15 (NIZK). *A non-interactive zero-knowledge proof (NIZK) for an* NP *relation $\mathcal{R} = (\mathcal{R}_\lambda)_{\lambda \in \mathbb{N}}$ consists of a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *with the following syntax*

$\mathsf{Setup}(1^\lambda)$**:** *The setup algorithm takes as imput the security paramater λ and outputs a CRS ω.*

$\mathsf{Prove}(\omega, x, w)$**:** *The prover algorithm takes as input a CRS ω, a statement x and a witness w. The output is a proof π.*

$\mathsf{Verify}(\omega, x, \pi)$**:** *The verification algorithm takes as input a CRS ω, a statement x and proof π. The output is a bit b, which represents whether the statement has been accepted or rejected.*

Definition 16 (Correctness). *A NIZK* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *for an* NP *relation $\mathcal{R} = (\mathcal{R}_\lambda)_{\lambda\in\mathbb{N}}$ is correct if, for every $\lambda \in \mathbb{N}$ and every $(x, w) \in \mathcal{R}_\lambda$, we have*

$$\Pr\left[\mathsf{Verify}(\omega, x, \pi) = 1 \middle| \begin{matrix}\omega \xleftarrow{\$} \mathsf{Setup}(1^\lambda) \\ \pi \xleftarrow{\$} \mathsf{Prove}(\omega, x, w)\end{matrix}\right] = 1.$$

The zero-knowledge property states that a PPT adversary cannot recognise whether it is given access to an oracle that provides honestly generated proofs or one that provides simulated ones, generated without using the witness.

Definition 17 (Zero-knowledge). *A NIZK* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *for an* NP *relation $\mathcal{R} = (\mathcal{R}_\lambda)_{\lambda\in\mathbb{N}}$ is zero-knowledge if there exists PPT algorithms Sim_0 and Sim_1 such that, for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ for which, for every $\lambda \in \mathbb{N}$, we have*

$$\left|\Pr\left[\mathcal{A}^{\mathcal{O}^{\mathsf{real}}_\omega}(1^\lambda, \omega) = 1 \middle| \omega \xleftarrow{\$} \mathsf{Setup}(1^\lambda)\right] - \Pr\left[\mathcal{A}^{\mathcal{O}^{\mathsf{ideal}}_\tau}(1^\lambda, \omega) = 1 \middle| (\omega, \tau) \xleftarrow{\$} \mathsf{Sim}_0(1^\lambda)\right]\right| \leq \mathsf{negl}(\lambda).$$

Above, $\mathcal{O}^{\mathsf{real}}_\omega$ denotes the oracle that on input any pair $(x, w) \in \mathcal{R}_\lambda$, responds with $\pi \xleftarrow{\$} \mathsf{Prove}(\omega, x, w)$. The oracle $\mathcal{O}^{\mathsf{ideal}}_\tau$ answers instead with $\pi \xleftarrow{\$} \mathsf{Sim}_1(\tau, x)$. Both oracles can be queried by the adversary an unbounded number of times.

The simulation-extractability property states that, by relying on a trapdoor hidden in the CRS, we can always extract a valid witness from all adversarially generated proofs. This holds even if the adversary is given access to an oracle that produces simulated proofs. Notice that simulation-extractability implies soundness.

Definition 18 (Simulation extractability). *Let* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *be a NIZK for the* NP *relation $\mathcal{R} = (\mathcal{R}_\lambda)_{\lambda\in\mathbb{N}}$. Let Sim_0 and Sim_1 be the zero-knowledge simulators. We say that the construction is simulation extractable if there exists a PPT algorithm* $\mathsf{Extract}$ *such that, for every PPT adversary $\mathcal{A}$, there exists a neglgible function $\mathsf{negl}(\lambda)$ for which, for every $\lambda \in \mathbb{N}$, we have*

$$\Pr\left[\begin{matrix}\mathsf{Verify}(\omega, x, \pi) = 1 \\ (x, w) \notin \mathcal{R}_\lambda\end{matrix} \middle| \begin{matrix}(\omega, \tau) \xleftarrow{\$} \mathsf{Sim}_0(1^\lambda) \\ (x, \pi) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^{\mathsf{ideal}}_\tau}(1^\lambda, \omega) \\ w \xleftarrow{\$} \mathsf{Extract}(\tau, x, \pi)\end{matrix}\right] \leq \mathsf{negl}(\lambda).$$

Above, $\mathcal{O}^{\mathsf{ideal}}_\tau$ is the oracle that answers any query x with $\pi \xleftarrow{\$} \mathsf{Sim}_1(\tau, x)$. The adversary can query $\mathcal{O}^{\mathsf{ideal}}_\tau$ an unbounded number of times.

B Proofs Omitted from the Main Body

B.1 Proof of Theorem 1

Proof. Consider any t-admissible adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$. Let $\mathbf{z}_k$ denote the output of the k-distributed sampler execution. We proceed through a series of hybrids.

Hybrid 0: This corresponds to the real world game. In other words, we provide the adversary $\mathcal{A}_0$ with $\sigma \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$ and $U \xleftarrow{\$} \mathsf{Gen}(\sigma)$. We provide $\mathcal{A}_1$ with the actual distributed sampler output. Let $\delta(\lambda)$ be the advantage.

Hybrid 1: In this hybrid, we change the distribution of ω: we generate it by running

$$(\omega, \tau) \xleftarrow{\$} \mathsf{NIZK.Sim}_0(1^\lambda).$$

Hybrid 1 is computationally indistinguishable from the real world by the security of the NIZK.

Hybrid 2: In this hybrid, we change the distribution of $U = (\mathbf{h}_i, \pi_i)_{i\in[d]}$. Specifically, for every $i \in [d]$, we simulate the proof π_i as

$$\pi_i \xleftarrow{\$} \mathsf{NIZK.Sim}_1(\tau, (\mathbf{g}, \mathbf{h}_i)).$$

This hybrid is computationally indistinguishable from Hybrid 1 by the security of the NIZK.

Hybrid 3: Let $U_{j,k} = (\mathbf{h}_{i,j,k}, \pi_{i,j,k})_{i\in[d]}$. In this hybrid, for every $k \in [t]$, $j \in [L_k] \setminus \{\ell_k\}$ and $i \in [d]$, we extract

$$a_{i,j,k} \leftarrow \mathsf{NIZK.Extract}(\tau, (\mathbf{g}, \mathbf{h}_{i,j,k}), \pi_{i,j,k}).$$

If $\mathsf{NIZK.Verify}(\omega, (\mathbf{g}, \mathbf{h}_{i,j,k}), \pi_{i,j,k}) = 1$ and, $\mathbf{g}^{a_{i,j,k}} \neq \mathbf{h}_{i,j,k}$, we provide $\mathcal{A}_1$ with $\perp$. This hybrid is indistinguishable from Hybrid 2 by the simulation extractability of the NIZK.

Hybrid 4: In this hybrid, for every $i \in [d]$, we sample $\mathbf{h}_i \xleftarrow{\$} \mathbb{Z}_q^n$. This is computationally indistinguishable from Hybrid 3 by the hardness of DDH over $\mathcal{G}$. Observe that, for every $k \in [t]$, the output of $\mathsf{Sample}(\sigma, U_{1,k}, \ldots, U_{k,L_k}, r_k)$ is

$$\prod_{i\in[d]} \mathbf{h}_i^{x_k^i} \cdot \mathbf{g}^{\sum_{i\in[d]} \sum_{j\neq\ell_k} a_{i,j,k}\cdot x_k^i}$$

Hybrid 5: In this hybrid, we guess the values $(\hat{x}_k)_{k\in[t]}$ uniformly at random in $\mathbb{Z}_p$ conditioned on being distinct. Then, for every $k \in [t]$, we sample $v_k \xleftarrow{\$} \mathbb{Z}_p$. Finally, we sample $(\hat{\mathbf{z}}_k)_{k\in[t]}$ uniformly at random over $\mathcal{G}$ and (using interpolation over $\mathbb{Z}_p$ in the exponent) we derive $(\mathbf{h}_i)_{i\in[d]}$ such that, for every $k \in [t]$,

$$\prod_{i\in[d]} \mathbf{h}_i^{\hat{x}_k^i} = \hat{\mathbf{z}}_k \cdot \mathbf{g}^{-v_k}.$$

This hybrid is identical to the previous one. Notice also that $(\hat{x}_k, v_k)_{k\in[t]}$ are outside of the view of the adversary, so

$$\Pr\left[\begin{array}{ll} \forall k \in [t]: & \hat{x}_k = x_k \\ \forall k \in [t]: & v_k = \sum_{i\in[d]} \sum_{j\in[L_k]\setminus\{\ell_k\}} a_{i,j,k} \cdot x_k^i \\ & \mathcal{A}_1((\mathbf{z}_k)_{k\in[t]}, \mathsf{aux}) = 1 \end{array}\right] = \frac{1}{p^{2t}} \cdot \Pr\left[\mathcal{A}_1((\mathbf{z}_k)_{k\in[t]}, \mathsf{aux}) = 1\right]$$

Notice that when $\hat{x}_k = x_k$ and $v_k = \sum_{i\in[d]} \sum_{j\in[L_k]\setminus\{\ell_k\}} a_{i,j,k} \cdot x_k^{i-1}$, it holds that $\mathbf{z}_k = \hat{\mathbf{z}}_k$. In conclusion,

$$\begin{aligned} \Pr\left[\mathcal{A}_1((\hat{\mathbf{z}}_k)_{k\in[t]}, \mathsf{aux}) = 1\right] \geq & \Pr\left[\begin{array}{ll} \forall k \in [t]: & \hat{x}_k = x_k \\ \forall k \in [t]: & v_k = \sum_{i\in[d]} \sum_{j\in[L_k]\setminus\{\ell_k\}} a_{i,j,k} \cdot x_k^{i-1} \\ & \mathcal{A}_1((\mathbf{z}_k)_{k\in[t]}, \mathsf{aux}) = 1 \end{array}\right] \\ = & \frac{1}{p^{2t}} \cdot \Pr\left[\mathcal{A}_1((\mathbf{z}_k)_{k\in[t]}, \mathsf{aux}) = 1\right] \\ = & \frac{1}{p^{2t}} \cdot (\delta(\lambda) - \mathsf{negl}(\lambda)). \end{aligned}$$

□

B.2 Proof of Theorem 2

Proof. Consider any t-admissible adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$. Let $\mathbf{z}_h$ denote the output of the h-th distributed sampler execution. We proceed through a series of hybrids.

Hybrid 0: This corresponds to the real world game. In other words, we provide the adversary $\mathcal{A}_0$ with $\sigma \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$ and $U \xleftarrow{\$} \mathsf{Gen}(\sigma)$. We provide $\mathcal{A}_1$ with the actual distributed sampler output. Let $\delta(\lambda)$ be the advantage.

Hybrid 1: In this hybrid, we change the distribution of ω: we generate it by running

$$(\omega, \tau) \xleftarrow{\$} \mathsf{NIZK.Sim}_0(1^\lambda).$$

Hybrid 1 is computationally indistinguishable from the real world by the security of the NIZK.

Hybrid 2: In this hybrid, we change the distribution of $U = (\mathbf{A}_i, \pi_i)_{i\in[d]}$. Specifically, for every $i \in [d]$, we simulate the proof π_i as

$$\pi_i \xleftarrow{\$} \mathsf{NIZK.Sim}_1(\tau, (\mathbf{B}, \mathbf{A}_i)).$$

This hybrid is computationally indistinguishable from Hybrid 1 by the security of the NIZK.

Hybrid 3: Let $U_{j,h} = (\mathbf{A}_{i,j,h}, \pi_{i,j,h})_{i\in[d]}$. In this hybrid, for every $h \in [t]$, $j \in [L_h] \setminus \{\ell_h\}$ and $i \in [d]$, we extract

$$\mathbf{X}_{i,j,h} \leftarrow \mathsf{NIZK.Extract}(\tau, (\mathbf{B}, \mathbf{A}_{i,j,h}), \pi_{i,j,h}).$$

If $\mathsf{NIZK.Verify}(\omega, (\mathbf{B}, \mathbf{A}_{i,j,h}), \pi_{i,j,h}) = 1$ and, $\mathbf{B} \cdot \mathbf{X}_{i,j,h} \neq \mathbf{A}_{i,j,h}$ or $\mathbf{X}_{i,j,h} \notin \mathbb{Z}_2^{m\times \log q}$, we provide $\mathcal{A}_1$ with $\perp$. This hybrid is indistinguishable from Hybrid 2, by the simulation extractability of the NIZK.

Hybrid 4: In this hybrid, for every $i \in [d]$, we sample $\mathbf{A}_i \xleftarrow{\$} \mathbb{Z}_q^{n\times \log q}$. This is statistically indistinguishable from Hybrid 3 by the leftover hash lemma (we are using the fact that $m \geq 2n \cdot \log q$).

Hybrid 5: In this hybrid, we sample $\mathbf{M} \xleftarrow{\$} \mathbb{Z}_q^{k\times m}$, $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n\times k}$ and $\mathbf{E} \xleftarrow{\$} \chi_s^{n\times m}$. Then, we set $\mathbf{B} \leftarrow \mathbf{S}\cdot\mathbf{M}+\mathbf{E}$. This hybrid is computationally indistinguishable from the previous one under the hardness of LWE. Observe that, for every $h \in [t]$, the output of $\mathsf{Sample}(\sigma, U_{1,h}, \ldots, U_{h,L_h}, r_h)$ is

$$\sum_{i\in[d]} \mathbf{A}_i \cdot \mathbf{G}^{-1}(x_h^{i-1}) + \mathbf{S} \cdot \left(\sum_{i\in[d]} \sum_{j\in[L_h]\setminus\{\ell_h\}} \mathbf{M} \cdot \mathbf{X}_{i,j,h} \cdot \mathbf{G}^{-1}(x_h^{i-1}) \right) + \sum_{i\in[d]} \sum_{j\in[L_h]\setminus\{\ell_h\}} \mathbf{E} \cdot \mathbf{X}_{i,j,h} \cdot \mathbf{G}^{-1}(x_h^{i-1})$$

Notice also that, for every $h \in [t]$, we have

$$\left\| \sum_{i\in[d]} \sum_{j\in[L_h]\setminus\{\ell_h\}} \mathbf{E} \cdot \mathbf{X}_{i,j,h} \cdot \mathbf{G}^{-1}(x_h^{i-1}) \right\|_\infty \leq N \cdot m \cdot s \cdot \lambda \cdot \log q \cdot d.$$

Hybrid 6: In this hybrid, we guess the values $(\hat{x}_h)_{h\in[t]}$ uniformly at random in $\mathbb{Z}_q$ conditioned on being distinct. Then, for every $h \in [t]$, we sample $\mathbf{v}_h \xleftarrow{\$} \mathbb{Z}_q^k$ and $\mathbf{e}_h \xleftarrow{\$} \chi^n$ where

$$\Pr[\chi = r] = \begin{cases} \frac{p}{q} & \text{if } r \in \left[-\frac{q}{2p} + 2Nsmd\lambda \log q, \frac{q}{2p} - 2Nsmd\lambda \log q \right[\\ \frac{p}{2q} & \text{if } r \in \left[-\frac{q}{2p} - 2Nsmd\lambda \log q, -\frac{q}{2p} + 2Nsmd\lambda \log q \right[\\ \frac{p}{2q} & \text{if } r \in \left[\frac{q}{2p} - 2Nsmd\lambda \log q, \frac{q}{2p} + 2Nsmd\lambda \log q \right[\end{cases}$$

For every $i \in [d]$, sample also $\mathbf{R}_i \xleftarrow{\$} \mathbb{Z}_q^{n\times \log q}$ and, for every $h \in [t]$, compute

$$\mathbf{u}_h \leftarrow \sum_{i\in[d]} \mathbf{R}_i \cdot \mathbf{G}^{-1}(\hat{x}_h^{i-1}).$$

Finally, we sample $(\hat{\mathbf{z}}_h)_{h\in[t]}$ uniformly at random over $\mathbb{Z}_p^n$ and (using interpolation over $\mathbb{Z}_q$) we derive $(\hat{\mathbf{y}}_i)_{i\in[d]}$ such that, for every $h \in [t]$,

$$\sum_{i\in[d]} \hat{\mathbf{y}}_i \cdot \hat{x}_h^{i-1} = \frac{q}{p} \cdot \hat{\mathbf{z}}_h + \mathbf{e}_h - \mathbf{S} \cdot \mathbf{v}_h - \mathbf{u}_h.$$

For every $i \in [d]$, we set $\mathbf{A}_i \leftarrow \mathbf{R}_i + \hat{\mathbf{y}}_i \cdot \mathbf{g}^\intercal$ where $\mathbf{g}$ is the gadget vector. Notice that

$$\sum_{i\in[d]} \mathbf{A}_i \cdot \mathbf{G}^{-1}(\hat{x}_h^{i-1}) = \frac{q}{p} \cdot \hat{\mathbf{z}}_h + \mathbf{e}_h - \mathbf{S} \cdot \mathbf{v}_h. \tag{1}$$

Notice that $\frac{q}{p} \cdot \hat{\mathbf{z}}_h + \mathbf{e}_h$ is uniformly distributed over $\mathbb{Z}_q$, and $(\mathbf{A}_i)_{i\in[d]}$ are random conditioned on Eq.1 holding. We conclude that this hybrid is identical to the previous one.

Notice also that $(\hat{x}_h, \mathbf{v}_h)_{h\in[t]}$ are outside of the view of the adversary, so

$$\Pr\left[\begin{array}{ll} \forall h \in [t]: & \hat{x}_h = x_h \\ \forall h \in [t]: & \mathbf{v}_h = \sum\limits_{i\in[d]} \sum\limits_{j\in[L_h]\setminus\{\ell_h\}} \mathbf{M} \cdot \mathbf{X}_{i,j,h} \cdot \mathbf{G}^{-1}(x_h^{i-1}) \\ & \mathcal{A}_1((\mathbf{z}_h)_{h\in[t]}, \mathsf{aux}) = 1 \end{array}\right] = \frac{1}{q^{t\cdot(k+1)}} \cdot \Pr\left[\mathcal{A}_1((\mathbf{z}_h)_{h\in[t]}, \mathsf{aux}) = 1\right]$$

Notice that when

$$\mathbf{e}_h \in \left[-q/2p + 2N \cdot m \cdot s \cdot d \cdot \lambda \cdot \log q, q/2p - 2N \cdot m \cdot s \cdot d \cdot \lambda \cdot \log q\right[^n,$$

it holds that $\mathbf{z}_h = \hat{\mathbf{z}}_h$. If instead (let's say) r_h entries of $\mathbf{e}_h$ do not belong to the above interval, there is still a $1/2^{r_h}$ probability that $\mathbf{z}_h = \hat{\mathbf{z}}_h$. This is because the adversary is not able to distinguish between

$$\left\{\frac{q}{p} \cdot z + e \middle| \begin{array}{l} z \xleftarrow{\$} \mathbb{Z}_p \\ e \xleftarrow{\$} \left[\frac{q}{2p} - 2N \cdot m \cdot s \cdot d \cdot \lambda \cdot \log q, \frac{q}{2p} + 2N \cdot m \cdot s \cdot d \cdot \lambda \cdot \log q\right[\end{array}\right\}$$
$$\left\{\frac{q}{p} \cdot (z+1) + e \middle| \begin{array}{l} z \xleftarrow{\$} \mathbb{Z}_p \\ e \xleftarrow{\$} \left[-\frac{q}{2p} - 2N \cdot m \cdot s \cdot d \cdot \lambda \cdot \log q, -\frac{q}{2p} + 2N \cdot m \cdot s \cdot d \cdot \lambda \cdot \log q\right[\end{array}\right\}$$

Now, let r be the total number of entries of the vectors $(\mathbf{e}_h)_{h\in[t]}$ whose value does not belong to

$$\left[-\frac{q}{2p} + 2N \cdot m \cdot s \cdot d \cdot \lambda \cdot \log q, \frac{q}{2p} - 2N \cdot m \cdot s \cdot d \cdot \lambda \cdot \log q\right[$$

If we take any value $c \in [tn]$, we obtain

$$
\begin{aligned}
&\Pr\left[\mathcal{A}_1((\hat{\mathbf{z}}_h)_{h\in[t]}, \mathsf{aux}) = 1\right] \\
\geq &\Pr\left[\begin{array}{l}\forall h \in [t] : \hat{x}_h = x_h \\ \forall h \in [t] : \mathbf{v}_h = \sum_{i\in[d]} \sum_{j\neq \ell_h} \mathbf{M}\cdot\mathbf{X}_{i,j,h}\cdot\mathbf{G}^{-1}(x_h^{i-1}) \\ \forall h \in [t] : \mathbf{z}_h = \hat{\mathbf{z}}_h \\ \mathcal{A}_1((\mathbf{z}_h)_{h\in[t]}, \mathsf{aux}) = 1\end{array}\right] \\
= &\sum_{j\in[tn]} \frac{1}{2^j} \Pr\left[\begin{array}{l}\forall h \in [t] : \hat{x}_h = x_h \\ \forall h \in [t] : \mathbf{v}_h = \sum_{i\in[d]} \sum_{j\neq \ell_h} \mathbf{M}\cdot\mathbf{X}_{i,j,h}\cdot\mathbf{G}^{-1}(x_h^{i-1}) \\ \mathcal{A}_1((\mathbf{z}_h)_{h\in[t]}, \mathsf{aux}) = 1 \\ r = j\end{array}\right] \\
\geq &\frac{1}{2^c}\cdot\sum_{j\in[c]} \Pr\left[\begin{array}{l}\forall h \in [t] : \hat{x}_h = x_h \\ \forall h \in [t] : \mathbf{v}_h = \sum_{i\in[d]} \sum_{j\neq \ell_h} \mathbf{M}\cdot\mathbf{X}_{i,j,h}\cdot\mathbf{G}^{-1}(x_h^{i-1}) \\ \mathcal{A}_1((\mathbf{z}_h)_{h\in[t]}, \mathsf{aux}) = 1 \\ r = j\end{array}\right] \\
\geq &\frac{1}{2^c}\cdot\Pr\left[\begin{array}{l}\forall h \in [t] : \hat{x}_h = x_h \\ \forall h \in [t] : \mathbf{v}_h = \sum_{i\in[d]} \sum_{j\neq \ell_h} \mathbf{M}\cdot\mathbf{X}_{i,j,h}\cdot\mathbf{G}^{-1}(x_h^{i-1}) \\ \mathcal{A}_1((\mathbf{z}_h)_{h\in[t]}, \mathsf{aux}) = 1\end{array}\right] - \frac{1}{2^c}\cdot\Pr[r \geq c] \\
\geq &\frac{1}{2^c\cdot q^{t\cdot(k+1)}}\cdot\Pr\left[\mathcal{A}_1((\mathbf{z}_h)_{h\in[t]}, \mathsf{aux}) = 1\right] - \frac{1}{2^c}\cdot\Pr[r \geq c].
\end{aligned}
$$

Observe that

$$\Pr[r \geq c] \leq \binom{nt}{c}(4N\cdot m\cdot p\cdot d\cdot\lambda\cdot\log q\cdot\alpha)^c \leq (4N\cdot m\cdot p\cdot t\cdot d\cdot\lambda\cdot\log q\cdot\alpha\cdot e\cdot n/c)^c.$$

Moreover,

$$\log(4N\cdot m\cdot p\cdot t\cdot d\cdot\lambda\cdot\log q\cdot\alpha\cdot e\cdot n/c) = \log N + \log p - k^\delta + O(\log\lambda) = -\omega(\log\lambda)$$

(this is because $N\cdot p\cdot\alpha = \mathsf{negl}(\lambda)$). Now, pick $c = t(k+1)\log q$. We observe that

$$\Pr[r \geq c] \leq 2^{-t(k+1)\log q\cdot\omega(\log\lambda)} = q^{-t\cdot(k+1)}\cdot\mathsf{negl}(\lambda).$$

This ends the proof.

□

B.3 Proof of Theorem 6

Proof. For simplicity we consider a deterministic protocol. Let L be the set of possible identities with $|L| = n$ with $n = 4k$. Consider a set of n parties to be divided into four equal-size sets A, B, C, D with the partial PKI performing an arbitrary assignment of identities from L to $A \cup B \cup C \cup D$. We define the following protocol execution types where parties in A are always honest but have no visibility (i.e., parties in A do not know about their existence) of the corrupted parties D, while parties in B, C are honest and have full visibility of the whole set of parties. Observe that this satisfies the properties of a $(\frac{1}{4}, 0)$-partial PKI.

– In execution type 1, parties in D are corrupted and execute the protocol honestly with input 1, while parties in A, B, C are honest have input 0. Observe that due to Validity, parties in A, B, C must output 0.
– In execution type 2, parties in D are corrupted and execute the protocol honestly with input 0, while A, B, C are honest and with input $0, 0, 1$, respectively. It follows that parties in set B must output 0, as in their view, executions of type 1 and 2 are indistinguishable. By Agreement, this also implies that parties in A must output 0.
– In execution type 3, parties in A, B, C are honest and with inputs $0, 0, 1$, respectively, while parties in D are corrupted and run with input 1. Observe that parties in A do not see parties in D and as a result execution type 3 is indistinguishable from execution type 2 in the view of parties in A, and thus they have to output 0. By Agreement, this also means that parties in B must output 0.

Observe now we can repeat the above steps while flipping $0, 1$. The identical argument will result in B outputting 1 on an indistinguishable configuration to execution type 3, resulting in a violation of Agreement. □