Tight Proofs of Space and Replication

Ben Fisch

Abstract

We construct a concretely practical *proof-of-space* (PoS) with arbitrarily *tight* security based on stacked depth robust graphs and constant-degree expander graphs. A *proof-of-space* (PoS) is an interactive proof system where a prover demonstrates that it is persistently using space to store information. A PoS is arbitrarily tight if the honest prover uses exactly N space and for any $\epsilon > 0$ the construction can be tuned such that no adversary can pass verification using less than $(1 - \epsilon)N$ space. Most notably, the degree of the graphs in our construction are independent of ϵ, and the number of layers is only $O(\log(1/\epsilon))$. The proof size is $O(d/\epsilon)$. The degree d depends on the depth robust graphs, which are only required to maintain $\Omega(N)$ depth in subgraphs on 80% of the nodes. Our tight PoS is also secure against parallel attacks.

Tight proofs of space are necessary for *proof-of-replication* (PoRep), which is a publicly verifiable proof that the prover is dedicating unique resources to storing one or more retrievable replicas of a file. Our main PoS construction can be used as a PoRep, but data extraction is as inefficient as replica generation. We present a second variant of our construction called ZigZag PoRep that has fast/parallelizable data extraction compared to replica generation and maintains the same space tightness while only increasing the number of levels by roughly a factor two.

1 Introduction

Proof-of-space (PoS) has been proposed as an alternative to proof-of-work (PoW) for applications such as SPAM prevention, DOS attacks, and Sybil resistance in blockchain-based consensus mechanisms [18, 21, 31]. Several industry projects[1] are underway to deploy cryptocurrencies similar to Bitcoin that use proof-of-space instead of proof-of-work. Proof-of-space is promoted as more egalitarian and eco-friendly that proof-of-work because it is ASIC-resistant and does not consume its resource (space instead of energy), but rather reuses it.

A PoS is an interactive protocol between a prover and verifier in which the prover use a minimum specified amount of space in order to pass verification. The protocol must have compact communication relative to the prover's space requirements and efficient verification. A PoS is *persistent* if repeated audits force the prover to utilize this space over a period of time. More precisely, there is an "offline" phase in which the prover obtains challenges from a verifier (or simulates them non-interactively via Fiat-Shamir), generates an advice string S that it stores, and outputs a compact tag τ to the verifier. This is followed by an "online" challenge-response protocol in which the prover uses its advice S to efficiently compute responses to the verifier's challenges. The soundness of the PoS relies on timing bounds on the online prover's runtime enforced by frequent verifier audits. Timing bounds are necessary as otherwise the prover could

[1] `https://chia.net/`,`https://spacemesh.io/`, `https://filecoin.io/`

store its compact transcript and simulate the setup to re-derive the advice whenever it needs to pass an online proof. If the PoS resists parallelization attacks then it is unconditionally secure in this audit model because the prover must use the minimum amount of space to pass challenges within the wall-clock time allotted. Alternatively, soundness is reasoned through a cost benefit analysis; it is assumed that a *rational* prover will not trade significant computation for a relatively small reduction in its space utilization.

In an (S, T)-sound PoS protocol where the prover commits to persistently utilize N blocks of space, the honest prover uses $S = O(N)$ persistent space and any adversary who passes audits in less than time T provably uses $S = \Omega(N)$ space. There is generally a gap between the honest space utilization and the lower bound on the adversary's space. If the honest prover uses N space and some adversary might be able to use ϵN space then this PoS has an ϵ *space gap*. A *tight PoS* makes ϵ arbitrarily small depending on a construction parameter that will typically impact concrete efficiency. All else equal, a tighter PoS is obviously more desirable as it has tighter provable security. Nearly all existing PoS constructions have enormous space gaps [2,18], including those that are currently being used in practice. The one exception is a recent PoS protocol by Pietrzak [33] based on depth robust graphs, although it does not achieve a tight space gap for concretely practical parameters.

Proof-of-replication (PoRep) [1, 19, 20, 33] is a recently proposed hybrid of PoS with proof-of-retrievability (PoR) [23]. A PoR demonstrates that the prover can retrieve a particular data file of interest, either known to the verifier, committed in a public commitment, or privately preprocessed by a client who produces a verification tag for the file. A PoRep demonstrates that the prover is dedicating unique resources to storing a retrievable copy of the file, and is therefore a *useful proof of space*. It has therefore been proposed as an alternative Sybil resistance mechanism that is not only ASIC resistant and eco-friendly, but also has a useful side-effect: it provides file storage on real data. Furthermore, since the prover may run several independent PoReps for the same file that each require unique resources, PoReps may be used as a publicly verifiable proof of data replication/duplication.

A PoRep is required to be a PoS, and its security as a proof of data replication is closely related to the space gap of the PoS. Formally, the security notion for PoReps is ϵ-*rational replication* [19], which says that an adversary can save at most an ϵ fraction of its space by deviating from storing the data in a replicated format. Storing data in a replicated format is therefore an ϵ-Nash-equilibrium. This is the best possible security notion because a prover can always sabotage its replicated format by scrambling its storage in an efficiently decodable (yet no longer replicated) way. A PoRep that satisfies ϵ-rational replication is also a PoS with an ϵ space gap. Intuitively, if a PoRep is not a tight proof of space then there may be some adversary that would be rationally incentivized to deviate from honest behavior and therefore likely destroy the replication format. This gives a whole new relevance to tight proofs of space, because the security of PoReps is only meaningful when ϵ is reasonably small.

The goal of this work is to construct a practical and provably tight PoS that can also be used as PoRep that satisfies ϵ-rational replication for arbitrarily small ϵ.

1.1 Related work

The original PoS of Dziembowski et. al. [18] was based on hard to pebble directed acyclic graphs (DAGs), using a blend of techniques from superconcentrators, random bipartite expander graphs and depth robust graphs [32]. During the offline initialization the prover computes a certain

labeling of the graph using a collision-resistant hash function where the label e_v on each node $v \in \mathcal{G}$ of the graph is the output of the hash function on the labels of all parent nodes of v. It outputs a commitment to this labeling along with a proof (either interactive or non-interactive) that the committed labeling was "mostly" correct. This offline proof consists of randomly sampled labels and their parent labels, which the verifier checks for consistency. During the online challenge-response phase the verifier simply asks for random labels that the prover must produce along with a standard proof that these labels are consistent with the commitment. Their construction leaves a space gap of at least $1 - \frac{1}{512}$.

The construction of Ren and Devadas from stacked bipartite expander graphs dramatically improved on the space gap, although it is not secure against parallel attacks. Their construction involves λ levels $V_1, ..., V_\lambda$ consisting of n nodes each, with edges between the layers defined by the edges of a constant-degree bipartite expander. The prover computes a labeling of the graph just as in the Dziembowski et. al. PoS, however it only stores the labels on the final level. Otherwise, the protocol is roughly the same. Their construction still leaves a space gap of at least $1/2$ (and much larger with practical parameters, e.g. their construction requires at least degree 40 graphs to achieve a space gap of less than $2/3$).

Recently, Abusalah et. al. [2] revived the simple PoS approach based on storing tables of random functions. The basic idea is for the prover to compute and store the function table of a random function $f : [N] \to [N]$ where f is chosen by the verifier or a random public challenge. During the online challenge-response the verifier simply asks the prover to invert f on a randomly sampled point $x \in [n]$. The intuition for why this should be secure is that a prover who has not stored most of the function table will likely have to brute force $f^{-1}(x)$, performing $\Omega(N)$ work. Unfortunately, this simple approach fails to be a PoS due to Hellman's time/space tradeoffs which enable a prover to succeed with S space and T computation for any $ST = O(N)$. However, Abusalah et. al. build on this approach to achieve a provable time/space tradeoff of $S^k T = \Omega(\epsilon^k T^k)$. This PoS is not secure against parallel attacks, and also has a very large (even asymptotic) space gap of $1 - \frac{1}{64 \log N}$.

Pietrzak [33] and Fisch et. al [20] independently proposed a simpler variant of the graph labeling PoS by Dziembowski et. al. based solely on pebbling a depth robust graph (DRG). A degree d DAG on n nodes is (α, β)-depth robust if any subgraph on αn nodes contains a path of at least length βn. It is trivial to construct DRGs of large degree (a complete DAG is depth robust), but much harder to construct DRGs with small (constant or poly-logarithmic) degree. Achieving constant α, β is only possible asymptotically with degree $\Omega(\log N)$. Simply instantiating the graph labeling PoS on a DRG results in a PoS with a $1 - \alpha$ space gap that is also secure against parallel attacks. Fisch et. al. also suggested combining this labeling PoS with a *verifiable delay function* (VDF) [11] to increase the expense of labeling the graph without increasing the proof verification complexity. The delay on the VDF can be tuned depending on the value of n. Both of these constructions were proposed in the context of designing PoReps. In this variant of the PoS protocol, the prover uses the labeling of the graph to encode a data file on n blocks $D = d_1, ..., d_n$. The ith label e_i is computed by first deriving a key k_i by hashing the labels on the parents of the ith node, and then setting $e_i = k_i \oplus d_i$. If all the labels are stored then any data block can be quickly extracted from e_i by recomputing k_i. We'll call this a *DAG encoding* of the data input. The VDF approach uses an additional encoding scheme $(\mathsf{enc}, \mathsf{dec})$ inside the DAG encoding, where enc is sequentially slow and dec is fast, to derive $e_i = \mathsf{enc}(k_i \oplus d_i)$. The data is decoded by computing $d_i = \mathsf{dec}(e_i \oplus k_i)$.

The labeling PoS on a DRG is not technically a tight PoS because decreasing α also decreases

the time bound βn on the prover's required computation to defeat the PoS. Moreover, while there exist constructions of (α, β) DRGs for arbitrarily small α, these constructions are purely theoretical and are not viable in practice. Pietrzak [33] improved on the basic construction by relying on a stronger property of special DRGs [6, 32] that are $(\alpha, \beta, O(\log n/\epsilon))$-depth robust for all (α, β) such that $1 - \alpha + \beta \geq 1 - \epsilon$. In Pietrzak's modified construction, the prover builds a DRG on $4n$ nodes and only stores the labels on the topologically last n nodes. This can similarly be used as a PoRep where the data is encoded only on the last level and the labels on previous levels are just used as keys. This sacrifices on data extraction time because extracting the data requires recomputing most of the keys from scratch, which is as expensive as the PoS initialization.

Pietrzak shows that a prover who deletes an ϵ' fraction of the labels on the last n nodes will not be able to re-derive them in fewer than n sequential steps. The value ϵ' can be made arbitrarily small, but at the expense of increasing the degree of the graph proportionally to $1/\epsilon'$. Moreover, although these special DRGs achieve asymptotic efficiency and are incredibly intriguing from a theoretical perspective, they still do not have concretely practical degree. According to the analysis in [6], achieving just a 1/2 space gap with a PoS on N 32 byte data blocks in Pietrzak's construction would require instantiating these graphs with degree $2,760 \log N$. The proof size is at least $O(\lambda d)$, so even for $\lambda = 10$ (i.e. 10 bit security) and $N = 2^{30}$ the proof size would be at least 26MB. Furthermore, as ϵ decreases $\lambda = O(1/\epsilon)$ to maintain the same security level. To achieve a space gap of 1/10 with 10-bit security would require $d = 19,310 \log N$ and $\lambda = 60$ for a proof size of at least 1GB.

Boneh et. al. [11] describe a simple PoRep (also a PoS) just based on storing the output of a decodable VDF on N randomly sampled points, which generalizes an earlier proposal by Sergio Demian Lerner [25]. This is in fact an arbitrarily tight PoS with very practical proof sizes (essentially optimal). However, the time complexity of initializing the prover's $O(N)$ storage is $O(N^2)$, and therefore is not practically feasible for large N. This construction is similar to the PoS based on storing function tables [2], but uses the VDF as a moderately hard (non-parallelizable) function on a much larger domain (exponential in the security parameter) and stores a random subset of its function table. The reason for the large initialization complexity is that the prover cannot amortize its cost of evaluating the VDF on the entire subset of points.

In an independent concurrent work, Cecchetti et. al [14] developed a similar idea to our stacked DRGs, using butterfly networks instead of expander graphs, to construct a primitive they called publicly incompressible encodings (PIEs). We elaborate on a comparison in the Appendix.

1.2 Summary of Contributions

We construct a new tight PoS based on graph labeling with asymptotic proof size $O(\log N/\epsilon)$ where ϵ is the achieved space gap. We can instantiate this construction with relatively weak depth robust graphs that do not require any special properties other than retaining $\Omega(N)$ depth in subgraphs on some constant fraction of the nodes bounded away from 1 (e.g. our concrete analysis is for 80% subgraphs).

PoS from Stacked DRGs Our basic approach is a combination of the stacked bipartite expanders of Ren and Devadas [35] with depth robust graphs. Instead of stacking λ line graphs we stack $O(\log(1/\epsilon))$ levels of fixed-degree DRGs where ϵ is a construction parameter. We refer

to this graph construction as Stacked DRGs. We are able to show that this results in a PoS that has only an ϵ space gap. Intuitively, the expander edges between layers amplify the dependence of nodes on the last layer and nodes on earlier layers so that deletion of a small ϵ fraction of node labels on the last level will require re-derivation of nearly all the node labels on the first several layers. Thus, since every layer is a DRG, recomputing the missing ϵ fraction of labels requires $\Omega(N)$ sequential computation. It is easy to see that this would be the case if the prover were only storing $(1-\epsilon)n$ labels on the last level and none of the labels on earlier levels, however the analysis becomes much more difficult when the prover is allowed to store any arbitrary $(1-\epsilon)n$ labels. This analysis is the main technical contribution of this work. Concretely, we analyze the construction with an $(n, 0.80n, \Omega(n))$ DRG, i.e. deletion of 20% of nodes leaves a high depth graph on the 80% remaining nodes, regardless of the value of ϵ.

Our construction is efficient compared to prior constructions of tight PoS primarily because we can keep the degree of the graphs fixed for arbitrary ϵ while only increasing the number of levels *logarithmically* in $1/\epsilon$. In a graph labeling PoS, the offline PoS proofs sample $O(1/\epsilon)$ labels along with their parent labels, which the verifier checks for consistency. Thus, any construction based on this approach that requires scaling the degree of graphs by $1/\epsilon$ also scales the proof size by $1/\epsilon$, resulting in a proof complexity of at least $O(1/\epsilon^2)$. In our stacked DRG PoS construction the offline proof must include queries from each level to prove that each level of computed labels are "mostly" correct. If done naively, $O(1/\epsilon)$ challenge labels are sampled from each level, resulting in a proof complexity $O(d/\epsilon \cdot \log(1/\epsilon))$ where d is the degree of the level graphs. This is already an improvement, however with a more delicate analysis we are able to go even further and show that the total number of queries over all layers can be kept at $O(1/\epsilon)$, achieving an overall proof complexity $O(d/\epsilon)$.[2]

The PoS on Stacked DRGs can also be used as the basis for a PoRep that satisfies ϵ-rational replication for arbitrarily small ϵ.

The PoRep variant on this PoS simply uses the labels on the $\ell - 1$st level as keys to encode the n-block data input $D = d_1, ..., d_n$ on the ℓth (last) level, using the same method described earlier for encoding data into the labels of a PoS (see Related Work, [19, 20, 33]). However, extracting data from this PoRep is as expensive as initializing the PoRep space because it requires recomputing the keys on the $\ell - 1$st level.

PoRep from ZigZag Expander DRGs Our second contribution is a variant of the PoS on Stacked DRGs that compromises slightly on efficiency (requires double the number of levels for the same security guarantee) but improves the efficiency of extracting data when this is used as a PoRep. Instead of adding edge dependencies between the layers, every layer is the union of a DRG and a constant degree non-bipartite expander graph. The only edges between layers are between nodes at the same indices. Since the graph is a DAG this means that the union of a subset with its dependencies and targets is a constant fraction larger than the subset itself. By alternating the direction of the edges between layers, forming a "zig-zag", the dependencies of a subset in one layer become targets of the same subset in the adjacent layer, and the dependencies between layers expands. We refer to this graph construction as ZigZag DRGs. The PoRep on ZigZag DRGs encodes in the labels of each layer the labels of the previous levels. The edges within a layer enforce dependencies between labels by deriving a key for each encoding using a

[2] Asymptotically, this is close to the optimal proof complexity achievable for any PoS based on graph labeling that has an ϵ space gap. If the prover claims to be storing n labels and the proof queries less than $1/\epsilon$ then a random deletion of an ϵ fraction of these labels evades detection with probability at least $(1-\epsilon)^{1/\epsilon} \approx 1/e$.

cryptographic hash function. A special key is derived for the encoding on each ith node from the labels on the parents of the ith node within the same layer. Essentially, this construction iterates the basic DAG encoding of the data inputs ℓ times (treating each layer as an independent DAG) rather than performing a long key derivation. The labels in any layer can be used to recover the labels in the preceding layer. Furthermore, the decoding step can be done in parallel.

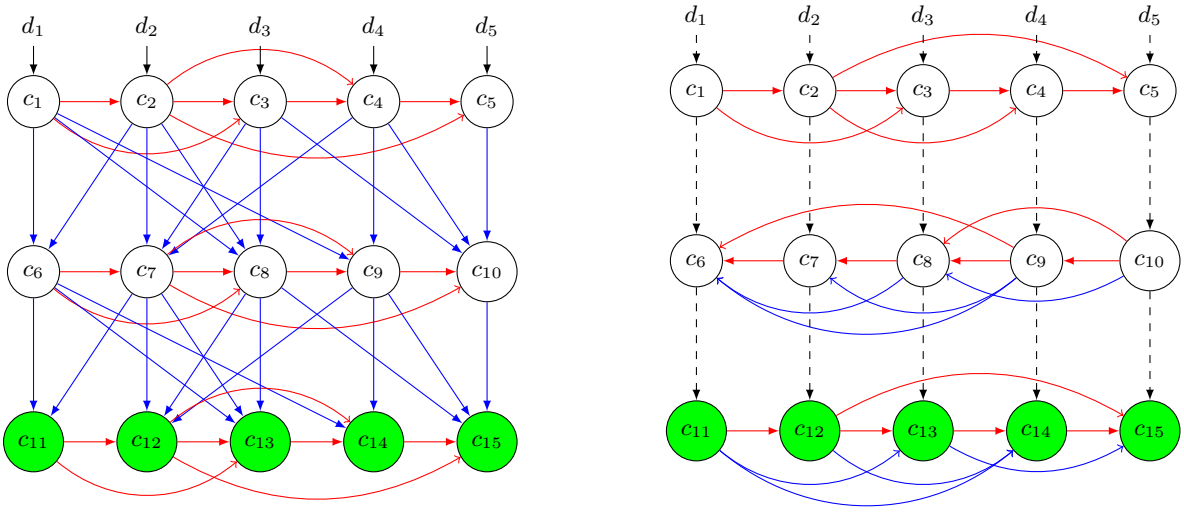

Figure 1.1: The topologies of the stacked DRGs and ZigZag DRGs are depicted with 3 layers and 5 nodes per layer. Red edges are the DRG edges and blue edges are expander edges. The blue edges in ZigZag DRGs are the same as in Stacked DRGs but projected into the layers. Blue edges in ZigZag DRGs are reversed every other layer while red edges are redefined by reversing the order of the nodes. Dashed edges correspond to encoding instead of hashing dependencies. In the PoS on Stacked DRGs the prover computes a labeling of the graph and stores the labels on the nodes in green. In the PoRep on ZigZag DRGs each labeling on a layer encodes the previous layer and the prover stores only the encoding labels of the green nodes.

Contents

2 Preliminaries

2.1 Proof of Retrievable Commitment

A proof-of-retrievability (PoR) is an interactive proof system in which a verifier sends a file F to a prover, retains a compact verification key, and later obtains a compact proof that prover can retrieve F intact. The compactness requirement excludes trivial solutions, such as sending the full file F back to the verifier or requiring the verifier to retain F. They were introduced in [8,23] and further developed in [12,17,37]. An important security property of PoR is that the verifier can extract and recover the file F through sufficiently many successful challenge-response queries to the prover.

A proof-of-retrievability (PoR) [12,17,23,37], or the related proof-of-data-possession (PDP) [8], is an interactive protocol that enables a prover to convince a verifier that it can retrieve the correct contents of a prespecified file without incurring costly communication. The file is first preprocessed by a client who publishes a data tag. The verifier does not need to store the file and only retains the short data tag for verification. In a private-key PoR the verifier needs to also know the private-key used to generate the data tag whereas in a public-key PoR the verification can be performed by anyone without access the private-key. Crucially, the prover cannot learn the private-key as otherwise this compromises the PoR/PDP security. PoR security is distinct from PDPs because it requires that there is a public extraction algorithm that can actually extract the contents of the file through sufficiently many repeated interactions with the prover.

A simple public PoR can be constructed from a Merkle commitment (i.e. a Merkle tree over the blocks of the file). The verifier need only retain the Merkle commitment root. To verify that the prover is still storing a $(1-\epsilon)$ fraction of the committed file blocks it queries for a randomly selected constant number of blocks. The prover then responds with the blocks and Merkle inclusion proofs for each. This public PoR is distinct from both public-key and private-key PoRs in that it is keyless and thus there is no secret key for the prover to potentially compromise. This is a stronger form of public verifiability.

Another way to describe this protocol is as a proof-of-retrievable-commitment (PoRC) [19], i.e. a publicly verifiable proof that the prover can retrieve the contents of a committed file. The

security of a PoRC does not rely on any client preprocessing. In fact, this technique is used ubiquitously in interactive proofs, including proofs of space, CS proofs [29], and more generally interactive oracle proofs (IOPs) [10]. Note that without erasure codes this is only a proof that a $(1-\epsilon)$ fraction of the file blocks can be retrieved. This is a special case of a $(1-\epsilon, \mathcal{C})$-PoRC, where $\mathcal{C}$ is a set cover of the committed data, and the protocol guarantees that a $(1-\epsilon)$ fraction of the sets (in this case blocks) can be retrieved.

The PoRC based on a Merkle commitment can be more generally constructed from any *vector commitment* (VC) [13, 27], which is a compact commitment to a vector of m values $(x_1, ..., x_m)$ that can be opened at any index with a succinct opening proof. A VC is *position binding* in the sense that each ith position can only be opened to a unique value x_i. This makes VCs distinct from set commitments (e.g cryptographic accumulators), which only guarantee that membership in the set can be verified. Merkle trees have $O(\log m)$ size opening proofs and there exist VCs that trade larger public parameters for constant size opening proofs [13, 26].

Similar to a PoR, the soundness of a PoRC is defined in terms of a public extraction algorithm. An important distinction is that the public extraction algorithm does not require a key to extract the data. Without going into the details of the security definition, a PoRC scheme is a μ-sound $(1-\epsilon)$-PoRC if for any adversary passing the PoRC protocol with probability μ there is a public extraction algorithm that can rewind the online adversary on challenges and ultimately extract a $(1-\epsilon)$ fraction of the blocks of the file.

2.2 Proofs of Space

A (persistent) proof of space (PoS) [18] is an interactive proof between and prover and verifier in which the prover can only succeed if it persistently stores some advice of a minimum size. There is an "offline" phase in which the prover generates this advice and outputs a compact tag to the verifier. This is followed by an "online" phase where the verifier challenges the prover and the prover uses its advice to generate a response.

Formally, the PoS interactive protocol involves three protocols:

1. Setup The setup runs on security parameters λ and outputs public parameters pp for the scheme. The public parameters are implicit inputs to the next two protocols.

2. Initialization is an interactive protocol between a prover P and verifier V that run on shared input (id, N). P outputs Φ and S, where S is its storage advice and Φ is a compact $O(\text{polylog}(N))$ string given to the verifier.

3. Execution is an interactive protocol between P and V where P runs on input S and V runs on input Φ. V sends challenges to P, obtains back a proof π, and outputs accept or reject.

The correctness and security requirements are as follows.

Efficiency. The commitment Φ is $O(\text{polylog}(N))$ size and the verifier runs in time $O(\text{polylog}(N))$.

Completeness. The prover succeeds with probability 1 (causes verifier to accept) if it follows the protocol honestly.

Soundness. The PoS is (s, t, μ)-sound if any for all adversaries P^* running in time t and storing advice of size s during $\mathsf{Execution}$ passes verification with probability at most $\mu = \mathsf{negl}(\lambda)$. The PoS is parallel (s, t, μ)-sound if P^* may run in parallel time t.

2.3 Proofs of Replication

We review the syntax of a PoRep scheme from [19]. PoRep operates on arbitrary data $D \in \{0,1\}^*$ of up to $O(\text{poly}(\lambda))$ size for a given security parameter λ.

1. $\mathsf{PoRep.Setup}(\lambda, T) \to pp$ is a one-time setup that takes in a security parameter λ, time parameter T, and outputs public parameters pp. T determines the challenge-response period.

2. $\mathsf{PoRep.Preproc}(sk, D) \to \tilde{D}, \tau_D$ is a preprocessing algorithm that may take a secret key sk along with the data input D and outputs preprocessed data $\tilde{D}$ along with its data tag τ_D, which at least includes the size $N = |D|$ of the data. The preprocessor operates in *keyless mode* when $sk = \perp$.

3. $\mathsf{PoRep.Replicate}(id, \tau_D, \tilde{D}) \to R, \mathsf{aux}$ takes a replica identifier id and the preprocessed data $\tilde{D}$ along with its tag τ_D. It outputs a replica R and (compact) auxilliary information aux which will be an input for the Prove and Verify procedures. (For example, aux could contain a proof about the replication output or a commitment).

4. $\mathsf{PoRep.Extract}(pp, id, R) \to \tilde{D}$ on input replica R and identifier id outputs the data $\tilde{D}$.

5. $\mathsf{PoRep.Prove}(R, \mathsf{aux}, id, r) \to \pi$ on input replica R, auxilliary information aux, replica identifier id, and challenge r, outputs a proof π_{id}.

6. $\mathsf{PoRep.Poll}(\mathsf{aux}) \to r$: This takes as input the auxiliary replica information aux and outputs a public challenge r.

7. $\mathsf{PoRep.Verify}(id, \tau_D, r, \mathsf{aux}, \pi) \to \{0,1\}$ on input replica identifier id, data tag τ_D, public challenge r, auxilliary replication information aux, and proof π it outputs a decision to accept (1) or reject (0) the proof.

PoRep interactive protocol The PoRep interactive protocol is illustrated in Figure 1. The setup (whether a deterministic, trusted, or transparent public setup) is run externally and pp is given as an input to all parties. For each file D, a preprocessor (a special party or the prover when operating in keyless mode, but not the verifier) runs $(\tilde{D}, \tau_D) \leftarrow \mathsf{PoRep.Preproc}(sk, D)$. The outputs $\tilde{D}, \tau_D$ are inputs to the prover and τ_D to the verifier.

ϵ-Rational Replication An ideal security goal for PoRep protocols would be to guarantee that any prover who simultaneously passes verification in k distinct PoRep protocols (under k distinct identities) where the input to $\mathsf{PoRep.Replicate}$ is a file D_i in the ith protocol must be storing k independent replicas, one for each D_i, even if several of the files are identical.

Formally, "storing k independent replicas" means that the file can be partitioned into k independent blocks (or more generally substrings) such that each block encodes the file and there is a universal decode algorithm that can decode the file independently from each block.

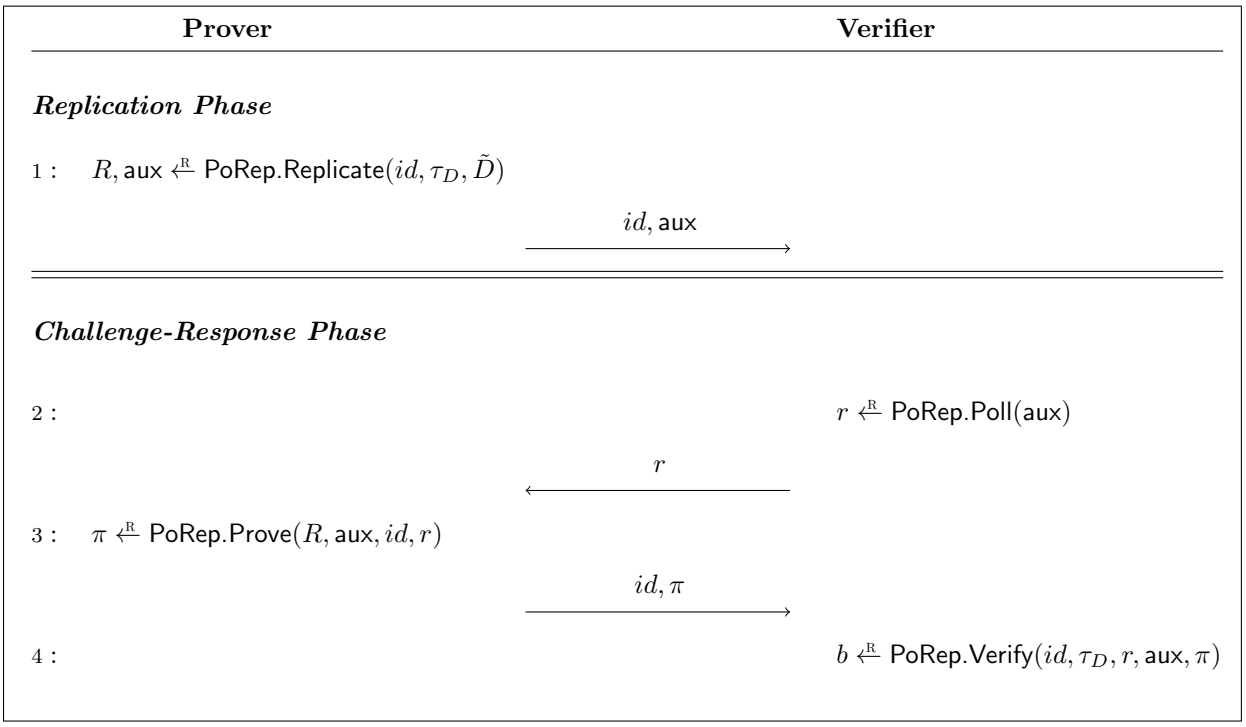

Figure 2.1: The PoRep interactive protocol is depicted above. The setup and data preprocessing is run externally generating pp and $\tilde{D}, \tau_D$. The challenge-response protocol is timed, and the verifier rejects any response that is received more than T time steps after sending the challenge. This is formally captured by requiring PoRep.Prove to run in parallel time at most T. A PoRep protocol is a special case of a PoS protocol where Initialization is the Replication Phase and Execution is the challenge-response phase.

We call this a *k-replication* of a file. Unfortunately, this security property is impossible to achieve. An adversary can easily "sabotage" its replication storage, e.g. by encrypting it with a key and storing the key separately. This still allows the adversary to decode the original string quickly and otherwise interact with the verifier in exactly the same way.

Instead, a rational model of security is defined in [19]. This security model, called ϵ-rational replication roughly says that a PoRep is a μ-sound ϵ-rational replication if for any adversary passing the protocol with probability μ there is an "equivalent" adversary who passes with at least the same probability and store a k-replication of the file without incurring more than a $1/(1-\epsilon)$ storage overhead (i.e. the adversary saves at most an ϵ fraction of its storage by deviating from a k-replication strategy). The security model also accounts for auxiliary information that the adversary might be storing and ensures that the "equivalent" adversary can still extract all the same information from its storage including data unrelated to the PoRep protocol.

Proof of space and PoRC A PoRep is implicitly a publicly verifiable proof of space (Section 2.2). A prover that passes verification in the interactive challenge-reponse protocol for a file $\tilde{D}$ of claimed size $|\tilde{D}| = N$ must be using $\Omega(N)$ persistent storage. Moreover, a prover that passes verification in k instances of this protocol with distinct ids $id_1, ..., id_k$ and claimed file sizes $N_1, ..., N_k$ must be using $\Omega(M)$ space where $M = \sum_{i=1}^{k} N_i$. In fact, it is a tight PoS with

only an ϵ space gap provided that it satisfies ϵ-rational replication, and therefore tight proofs of space are necessary for PoReps. A PoRep is also a proof of retrievability of the file D. In fact, sufficient conditions for a (correct) PoRep scheme to satisfy ϵ-rational replication are that it is a tight PoS and a PoRC of the data and replica with respect to commitments τ_D and aux. This was proven (c.f. Lemma 2, [19]) using a *knowledge of compression assumption*, which roughly says that any prover who knows an algorithm to compress auxiliary data together with the incompressible replica must know an algorithm to compress the auxiliary data independently by the same amount and store the incompressible part separately.

2.4 Graph pebbling games

Pebbling games are the main analytical tool used in graph-based proofs of space and memory hard functions.

Black pebbling game The black pebbling game is a single-player game on a DAG $\mathcal{G} = (V, E)$. At the start of the game the player chooses a starting configuration of $P_0 \subseteq V$ of vertices that contain black pebbles. The game then proceeds in rounds where in each round the player may place a black pebble on a vertex only if all of its parent vertices currently contain pebbles placed in some prior round. In this case we say that the vertex is *available*. Placing a pebble constitutes a *move*, whereas placing pebbles on all simultaneously available vertices consumes a *round*. The adversary may also remove any black pebble at any point. The game stops once the adversary has placed pebbles on all vertices in some target/challenge set $V_C \subseteq V$.

Pebbling complexity The pebbling game on graph G with vertex set V and target set $V_C \subseteq V$ is (s,t)-*hard* if no player can pebble the set V_C in t moves (or fewer) starting from s initial pebbles, and is (s,t)-*parallel-hard* if no player can complete the pebbling in t rounds (or fewer) starting from an initial configuration of at most s pebbles. If the hardness holds for any subset containing an α fraction of V_C then the pebbling game on (G, V_C) is (s,t,α)-(parallel)-hard.

In a random pebbling game a challenge node is sampled randomly from V_C after the player commits to the initial configuration P_0 of s vertices, and the hardness measure includes the adversary's probability of success. The random pebbling game is (s,t,ϵ)-(parallel)-hard if from any s fixed initial pebbles the probability that a uniformly sampled challenge node can be pebbled in t or fewer moves (resp. t or fewer rounds) is less than ϵ.

Fact 1. *The random pebbling game on a DAG G on n nodes with target set V_C is (s,t,α)-parallel-hard if and only if the deterministic pebbling game on G with target set V_C is (s,t,α)-parallel-hard.*

Proof. Fix any P_0 of size s. If the random pebbling game on G with target set V_C is (s,t,α)-parallel-hard then less than an α fraction of the nodes in V_C can be pebbled individually in t rounds starting from P_0. Therefore, every subset U in V_C of size $\alpha|V_C|$ contains at least one node that cannot be pebbled individually in t rounds, hence the (deterministic) pebbling game is (s,t,α)-parallel-hard. Conversely, if G is (s,t,α)-parallel-hard then less than an α fraction of nodes in V_C can be pebbled individually in r rounds. Otherwise, these nodes form a subset U of size $\alpha|V_C|$ and they can all be simultaneously pebbled in parallel in t rounds. This implies

that the probability a randomly sampled node from V_C can be pebbled in t rounds is less than α. □

Fact 2. *A random pebbling game with a single challenge is (s, t, α)-parallel-hard if and only if the the random pebbling with κ challenges is (s, t, α^k)-parallel-hard.*

Proof. If the random pebbling game is (s, t, α) hard then by Fact 1 the deterministic pebbling game is (s, r, α) hard, hence there are at most an α fraction of the nodes in V_C that can be pebbled in r rounds from s initial pebbles. The probability that κ independent random challenges are all nodes from this α fraction is at most α^κ. Conversely, if the random pebbling game is not (s, t, α) hard then the adversary can pebble all the κ challenges simultaneously in parallel time t succeeding on each challenge individually with probability greater than α, hence succeeding on all the challenges with probability greater that α^κ. □

DAG labeling game A labeling game on a degree d DAG $\mathcal{G}$ is analogous to the pebbling game, but involves a cryptographic hash function $H : \{0,1\}^{dm} \rightarrow \{0,1\}^m$, often modeled as a random oracle. The vertices of the graph are indexed in $[n]$ and each ith vertex associated with the label c_i where $c_i = H(i)$ if i is a source vertex, or otherwise $c_i = H(i||c_{\mathsf{parents}}(i))$ where $c_{\mathsf{parents}}(i) = \{c_{v_1}, ..., c_{v_d}\}$ if $v_1, ..., v_d$ are the parents of the ith vertex, i.e. the vertices with a directed edge to vertex i. The game ends when the player has computed all the labels on a target/challenge set of vertices V_C. A "fresh" labeling of $\mathcal{G}$ could be derived by choosing a salt id for the hash function so that $H_{id}(x) = H(id||x)$, and the labeling may be associated with the identifier id.

The complexity of the labeling game (on a fresh identifier id) is measured in queries to the hash function instead of pebbles. This includes the number of labels initially stored, the total number of queries, and the total rounds of sequential queries, etc. The labeling game is $(s, r, q, \epsilon, \delta)$-labeling-hard if no algorithm that stores initial advice of size s and after receiving a uniform random challenge node $v \in [n]$ makes a total of q queries to H in r sequential rounds can output the correct label on v with probability greater than ϵ over the challenge v and δ over the random oracle H.

Random oracle query complexity A general correspondence between the complexity of the black pebbling game on the underlying graph $\mathcal{G}$ and the random oracle labeling game is not yet known. However, Pietrzak [33] recently proved an equivalence between the parallel hardness of the randomized pebbling game and the parallel hardness of the random oracle labeling game for arbitrary initial configurations S_0 adapting the "ex post facto" technique from [16].

Theorem 1 (Pietrzak [33])**.** *If the random pebbling game on a DAG G with n nodes and in-degree d is (s, r, ϵ)-parallel-hard then the labeling game on G with a random oracle $H : \{0,1\}^{md} \rightarrow \{0,1\}^m$ is $(s', r, \epsilon, \delta, q)$-labeling-hard with $s' = s(m - 2(\log n + \log q)) - \log(1/\delta)$.*

Generic PoS from graph labeling game Many PoS constructions are based on the graph labeling game [18, 33, 35]. Let $\mathcal{G}(\cdot)$ be a family of d-in-regular DAGs such that $G_n \leftarrow \mathcal{G}(n)$ is a d-in-regular DAG on $N > n$ nodes and $V_C(n)$ is a subset of n nodes from G_n. Let $H : \{0,1\}^{dm} \rightarrow \{0,1\}^m$ be a collision-resistant hash function (or random oracle). Let $\mathsf{Chal}(n, \Lambda)$ denote a distribution over challenge vectors in $[N]^\lambda$. For each $n \in \mathbb{N}$, the generic PoS based on

the labeling game with G_n and target set $V_C(n)$ is as follows:

Initialization: The prover plays the labeling game on G_n using a hash function $H_{id} = H(id||\cdot)$. The prover does the following:

1. Computes the labels $c_1, ..., c_N$ on all nodes of $\mathcal{G}$ and commits to them in com using any vector commitment scheme.

2. Obtains vector of λ challenges $\vec{r} \xleftarrow{R} \mathsf{Chal}(n)$ from the verifier (or non-interactively derives them using as a seed $H_{id}(com)$).

3. For challenges $r_1, ..., r_\lambda$, the prover opens the label on the r_ith node of G_n, which was committed in com, as well as the labels $c_{\mathsf{parents}}(r_i)$ of all its parent nodes. The labels are added to a list L with corresponding opening proofs in a list Λ and the prover outputs the proof $\Phi = (com, L, \Lambda)$.

The verifier checks the openings Λ with respect to com. It also checks for each challenge specifying an index $v \in [N]$, the label c_v in L label and its parent labels $c_{\mathsf{parents}}(c_v)$, that $c_v = H_{id}(v||c_{\mathsf{parents}}(c_v))$. Finally, the prover stores as S only the n labels in V_C.

Execution: The verifier selects κ challenge nodes $v_1, ..., v_\kappa$ uniformly at random from V_C. The online prover uses its input S to respond with the label on v and an opening of com at the appropriate index. The verifier can repeat this sequentially, or ask for a randomly sampled vector of challenge vertices to amplify soundness.

Given the correspondence between the hardness of the random oracle labeling game and black pebbling game in the parallel pebbling/computation models, we will focus on parallel pebbling complexity as this is easier to analyze directly.

Red-black pebbling game The soundness of the generic labeling PoS is captured through the red-black pebbling game. An adversary places both black and red pebbles on the graph initially. The red pebbles correspond to incorrect labels that the adversary computes during Initialization and the black pebbles correspond to labels the adversary stores in its advice S. Without loss of generality, an adversary that cheats generates some label that does not require any space to store, which is why red pebbles will be "free" pebbles and counted separately from black pebbles. The adversary's choice of red pebble placements (specifically how many to place in different regions of the graph) is constrained by the λ non-interactive challenges, which may catch these red pebbles and reveal them to the verifier. The formal description of the red-black pebbling security game for a graph labeling PoS construction with $\mathcal{G}(n)$, $V_C(n)$, and $\mathsf{Chal}(n)$ is as follows.

$\mathsf{Red\text{-}Black\text{-}Pebbles}^{\mathcal{A}}(\mathcal{G}, V_C, \mathsf{Chal}, t)$:

1. $\mathcal{A}$ outputs a set $R \subseteq [N]$ (of red pebble indices) and $S \subseteq [N]$ (of black pebble indices).

2. The challenger samples $c_1, ..., c_\lambda \xleftarrow{R} \mathsf{Chal}(n)$. If $c_i \in R$ for some i then $\mathcal{A}$ immediately loses. The challenger additionally samples $v_1, ..., v_\kappa$ uniformly at random from indices in $V_C(n)$ and sends these to $\mathcal{A}$.

3. $\mathcal{A}$ plays the random (black) pebbling game on $\mathcal{G}(n)$ with the challenges $v_1, ..., v_\kappa$ and initial pebble configuration $P_0 = R \cup S$. It runs for t parallel rounds and outputs its final pebble configuration P_t. $\mathcal{A}$ wins if P_t contains pebbles on all of $v_1, ..., v_\kappa$.

We formally define PoS soundness for the special case of any graph labeling PoS in terms of complexity of $\mathsf{Red\text{-}Black\text{-}Pebbles}^{\mathcal{A}}(\mathcal{G}, V_C, t)$. Let $c : \mathbb{N} \to N$ denote a cost function $c : \mathbb{N} \to \mathbb{N}$ representing the parallel time cost (e.g. in sequential steps on a PRAM machine) of computing a label on a node of $\mathcal{G}(n)$ for each $n \in \mathbb{N}$.

Definition 1. *A graph labeling PoS with $\mathcal{G}(n), V_C(n), \mathsf{Chal}(n)$ and cost function $c(n)$ is parallel $(s, c(n)\cdot t, \mu)$-sound if and only if the probability that any $\mathcal{A}$ wins $\mathsf{Red\text{-}Black\text{-}Pebbles}^{\mathcal{A}}(\mathcal{G}, V_C, \mathsf{Chal}, t)$ is bounded by μ where $|S| = s$.*

2.5 Verifiable Delay Encodings

A *verifiable delay encoding* (VDE) is a non-parallelizable encoding that has high parallel time complexity to compute, but with a fast decoding operation. A VDE may be a special kind of *verifiable delay function* [11]. It is more restrictive than a VDF because it must be decodable, but it is less restrictive because the encoding does not need to be unique. Existing pactical (heuristic) examples of VDEs include the Pohlig-Hellmann cipher, Sloth [24], MiMC [3], and a special class of permutation polynomials [11].

Formally, a VDE is a tuple of three algorithms $\mathsf{VDE} = \mathsf{VDE.Setup}, \mathsf{VDE.Enc}, \mathsf{VDE.Dec}$ defined as follows (c.f. [19]).

1. $\mathsf{VDE.Setup}(t, \lambda) \to pp$ is given security parameter λ and delay parameter t produce public parameters $\mathbf{pp}$. By convention, the public parameters also specify an input space $\mathcal{X}$ and a code space $\mathcal{Y}$. We assume that $\mathcal{X}$ is efficiently samplable. $\mathsf{VDE.Setup}$ might need secret randomness, leading to a scheme requiring a trusted setup.
2. $\mathsf{VDE.Enc}(pp, x) \to y$ takes an input $x \in \mathcal{X}$ and produces an output $y \in \mathcal{Y}$.
3. $\mathsf{VDE.Dec}(pp, y) \to x$ takes an input $y \in \mathcal{Y}$ and produces an output $x \in \mathcal{X}$.

Correctness For all pp generated by $\mathsf{VDE.Setup}(\lambda, t)$ and all $x \in \mathcal{X}$, algorithm $\mathsf{VDE.Enc}(pp, x)$ must run in parallel time t with $\mathrm{poly}(\log(t), \lambda)$ processors, and $\mathsf{VDE.Dec}(pp, \mathsf{VDE.Enc}(pp, x)) = x$ with probability 1.

Definition 2 (sequentiality, c.f. [11])**.** *For a function $\sigma(t)$ a VDE is σ-sequential if for any pair of randomized algorithms $\mathcal{A}_0$, which runs in total time $O(poly(t, \lambda))$, and $\mathcal{A}_1$, which runs in parallel time $t - \sigma(t)$ on at most $O(poly(t))$ processors, the following probability distribution over $pp \leftarrow \mathsf{VDE.Setup}(t, \lambda)$ is negligible:*

$$Pr\left[\begin{array}{l} y \leftarrow \mathcal{A}_1(\alpha, \mathbf{pp}, x) \\ \wedge\ y = \mathsf{VDE.Enc}(x) \end{array}\middle|\begin{array}{c} x \xleftarrow{R} \mathcal{X} \\ \alpha \leftarrow \mathcal{A}_0(\lambda, \mathbf{pp}, t) \end{array}\right] < \mathsf{negl}(\lambda)$$

A stronger security property is for the VDE to behave like an ideal cipher [19]. This is more than what is needed for most applications of VDFs/VDEs, but is relevant to their application

to proofs of space in order to facilitate incompressibility arguments. We can argue that an algorithm that can use an advice string to succeed with high probability in computing the $\mathsf{Encode}(pp, \cdot)$ function in time less than t would be able to compress the function table of a random permutation, similar to how advice string lower bounds are proven in the random oracle model.

Definition 3 (c.f. [19])**.** *An ideal delay permutation (IDP) is a family of oracles* $\{\mathcal{O}_{\mathsf{IDP}}^{(t)}\}$ *that implement a random permutation* Π *and respond to two types of queries. On a query* $(q, 0)$ *the oracle* $\mathcal{O}_{\mathsf{IDP}}^{(t)}$ *internally simulates* t *sequential queries to* Π^{-1} *and then outputs* $\Pi(q)$*. On a query* $(q, 1)$ *it outputs* $\Pi^{-1}(q)$*.*

2.6 Depth Robust Graphs

A directed acyclic graph (DAG) on n nodes with d-indegree is (n, α, β, d) *depth robust graph* (DRG) if every subgraph of αn nodes contains a path of length at least βn.

Depth robust graph constructions DRGs were first noted by Erdős, Graham, and Szemeredi [32], who constructed a family of $(n, \alpha, \beta, c \log n)$-depth robust graphs using extreme constant-degree bipartite expander graphs, for some constants α, β, c that satisfy specific constraints. Mahmoody, Moran, and Vadhan [28] constructed a more flexible family of DRGs that are $(n, \alpha, \alpha - \epsilon, c \log^2 n)$ depth robust for all $\alpha < 1$. Alwen et. al. [6] recently improved the EGS construction to obtain DRGs for arbitrary α, β as well and $O(\log n)$ degree (i.e. with asymptotically better degree than MMV). All these constructions still rely on extreme constant-degree expanders (also called local expanders). Explicit constructions of local expanders exist [30], however they are complicated to implement and their concrete practicality is hindered by very large hidden constants. The most efficient way to instantiate these extreme expander graphs is probabilistically. We discuss probabilistic constructions of bipartite expander graphs in Section 2.7.

A probabilistic DRG construction outputs a graph that is a DRG with overwhelming probability. Instantiating any of the above DRG constructions with probabilistic bipartite expanders results in a probabilistic DRG. However, even probabilistic versions of the above constructions are still not concretely efficient due to their use of local expanders. Alwen et. al. [5] proposed and analyzed the most efficient probabilistic DRG construction to date. The analysis still leaves large gaps between security and efficiency although was shown to resist depth-reducing attacks empirically. Their construction is also *locally navigatable*, meaning that it comes with an efficient parent function to derive the parents of any node in the graph using polylogarithmic time and space.

Definition 4. *An* (n, α, β, d)*-DRG sampling algorithm runs in time* $\tilde{O}(n \log n)$ *and a function* $\mathsf{DRG.Sample}(n, \sigma) \to G$ *that takes an* s*-bit seed and outputs a graph on* n *nodes indexed in* $[n]$ *such that* G *is* (n, α, β, d) *depth robust graph with probability* $1 - \mathsf{negl}(n)$ *over* $\sigma \xleftarrow{R} \{0,1\}^s$*. The sampling algorithm is locally navigatable if there is an algorithm* $\mathsf{DRG.Parents}(n, \sigma, i)$ *that runs in time* $O(polylogn)$ *and outputs a list* $\mathcal{P} \subseteq [n]$ *of the parents of the node at index* $i \in [n]$ *in the graph* $G_\sigma \leftarrow DRG.Sample(n, \sigma)$*.*

2.7 Expander graphs

The vertex expansion of a graph $\mathcal{G}$ on vertex set V characterizes the size of the boundary of vertex subsets $S \subseteq V$ (i.e. the number of vertices in $V \setminus S$ that are neighbors with vertices in S).

Definition 5. *Let $\mathcal{G}$ be an undirected graph on a vertex set V of size $n \in \mathbb{N}$ and for any subset $S \subseteq V$ define $\Gamma(S)$ to be the set of vertices in $V \setminus S$ that have an edge to some vertex in S. For any constants $0 < \alpha < \beta < 1$, $\mathcal{G}$ is an (n, α, β) expander graph if and only if $|\Gamma(S) \cup S| \geq \beta n$ for all S of size $|S| \geq \alpha n$. For any $\delta > 0$, the set S is called $(1+\delta)$-expanding if $|\Gamma(S) \cup S| \geq (1+\delta)|S|$.*

In the case of directed bipartite graphs, vertex expansion is defined by the minimum number of sources connected to any given number of sinks.

Definition 6. *For any constants α, β where $0 < \alpha < \beta < 1$ and integer $n \in \mathbb{N}$, an (n, α, β) bipartite expander is a directed bipartite graph with n sources and n sinks such that any subset of αn sinks are connected to at least βn sources. For any $\delta > 0$, a subset S of sinks is called $(1 + \delta)$-expanding if it is connected to at least $(1 + \delta)|S|$ sources.*

It is easy to construct an undirected expander graph given a bipartite expander as defined above.

Claim 1. *Let $\mathcal{H}$ be a (n, α, β) bipartite expander graph with bounded degree d, sources and sinks labeled by indices in $[n]$ and edge set $E \subseteq [n] \times [n]$. Define the undirected graph $\mathcal{G}$ with vertices labeled in $[n]$ and edge set E' such that $(i, j) \in E'$ (for $i \neq j$) if and only if $(i, j) \in E$ or $(j, i) \in E$. Then $\mathcal{G}$ has bounded degree $2d$ and is an (n, α, β) expander.*

Proof. Every vertex in $\mathcal{G}$ has at most degree $2d$ because $\mathcal{H}$ has bounded degree d, and the number of edges added to any node indexed with label i in $\mathcal{G}$ is at most the sum of number of edges incident to the ith source and ith sink in $\mathcal{H}$ respectively. Now consider any subset $S \subseteq [n]$ of αn vertices in $\mathcal{G}$. S also corresponds to a subset of the sources in $\mathcal{H}$ with the same index labels and is connected to at least βn sinks $T \subseteq [n]$. For each $s \in S$ and $t \in T$ where $s \neq t$ there is an edges (s, t) in $\mathcal{G}$. Therefore, $\Gamma(S) = T \setminus S$ and $|\Gamma(S)| \cup S = |T| \geq \beta n$. □

Given this equivalence, for the remainder of this section we focus on constructions of constant degree bipartite expanders.

Constructing bipartite expanders There is a rich literature on constructions of bipartite expanders, and includes both explicit and randomized constructions of constant degree bipartite graphs with very good expansion properties. The randomized construction of Chung [15] simply defines the edges of a d-regular bipartite expander on $2n$ vertices by connecting the dn outgoing edges of the sources to the dn incoming edges of the sinks via a random permutation $\Pi : [d] \times [n] \to [d] \times [n]$. More precisely, the ith source is connected to the jth sink if there is some $k_1, k_2 \in [d]$ such that $\Pi(k_1, i) = (k_2, j)$. A general relationship between the degree and expansion is known, which holds with overwhelming probability over the choice of the permutation. The following lemma was first proven by Bassalygo [9] and Schöning [36], and a much simpler proof was given by Ren and Devadas.

Lemma 1 (RD [35])**.** *The Chung random bipartite graph construction is a d-regular* (n, α, β) *expander with probability* $1 - \mathsf{negl}(nH_b(\alpha))$ *for all* d, α, β *satisfying:*

$$H_b(\alpha) + H_b(\beta) + d(\beta H_b(\alpha/\beta) - H_b(\alpha)) < 0 \tag{2.1}$$

where $H_b(x) = -x \log_2 x - (1-x)\log_2(1-x)$ *is the binary entropy function.*

For example, the above formula shows that for $\alpha = 1/2$ and $\beta = 0.80$ Chung's construction gives an $(n, 0.5, 0.80)$ expander for $d \geq 8$, meaning any subset of 50% of the sinks are connected to at least 80% of the sources when the degree is at least 8.

Expansion vs subset size In general, the expansion factor (the ratio β/α) improves in smaller subsets: if a given expansion factor holds with overwhelming probability in large subsets then it also holds with overwhelming probability in smaller subsets. On the other hand, the absolute expansion β is monotonically non-decreasing as a function of α because the expansion of any set is at least the expansion of its subsets.

Lemma 2. *For any* $k > 1$ *and* $d > 2$*, if the output of Chung's construction is a d-regular* $(n, \alpha, k\alpha)$ *bipartite expander for some* $\alpha < \frac{d-k-1}{k(d-2)}$ *with probability* $1 - \mathsf{negl}(nH_b(\alpha))$ *then* $\beta_G(\alpha') \geq k\alpha'$ *for every subset of size* $\alpha' < \alpha$ *with probability* $1 - \mathsf{negl}(nH_b(\alpha'))$*.*

Proof. For fixed d and k define $\phi(\alpha) = H_b(\alpha) + H_b(k\alpha) + dk\alpha H_b(1/k) - dH_b(\alpha)$. By Lemma 1, Chung's construction outputs a graph that is a d-regular $(n, \alpha, k\alpha)$ bipartite expander with probability $1 - \mathsf{negl}(nH_b(\alpha))$ as long as $\phi(\alpha) < 0$. We will show that if $\phi(\alpha) < 0$ for some α within the domain $X = (0, \frac{d-k-1}{k(d-2)})$, then $\phi(\alpha') < 0$ for all $\alpha' < \alpha$. We will first prove two subclaims:

1. ϕ is smooth on $\alpha \in (0, 1/k)$ (i.e. twice differentiable), and $\lim_{\alpha \to 0} \phi = 0$.
2. ϕ is decreasing at $\alpha = 0$ (i.e. increasing in the limit $\alpha \to 0$) and convex on X.

Together these imply that if $\phi(\alpha) < 0$ for $\alpha \in X$ then $\phi(\alpha') < 0$ for all $\alpha' < \alpha$. Suppose not, then there exists some point $\alpha' < \alpha$ such that $\phi(\alpha') \geq 0 > \phi(\alpha)$. Since ϕ is continuous on X and initially negative and decreasing it must increase on some subinterval of $(0, \alpha')$ and then decrease again on some subinterval of (α', α). However, this contradicts the fact that ϕ is convex on X, and hence once it starts increasing at any point in X it will not decrease again in any higher interval.

Proof of subclaim 1: ϕ is a linear combination of $H_b(\alpha)$, $H_b(k\alpha)$, and α, which are all real and twice differentiable on $(0, 1/k)$. In particular, $H'(\alpha) = \log_2(1-\alpha) - \log_2(\alpha)$ and $H''(\alpha) = \frac{-1}{\log_2(e)\alpha(1-\alpha)}$. Finally, $\lim_{\alpha \to 0} H_b(\alpha) = \lim_{\alpha \to 0} H_b(k\alpha) = 0$, hence the limit $\alpha \to 0$ of any linear combination of $H_b(\alpha)$, $H_b(k\alpha)$, and α is 0.

Proof of subclaim 2: We first show that $\lim_{\alpha \to 0} \phi' = -\infty$. Note that $\phi'(\alpha) = H_b'(\alpha) + kH_b'(\alpha k) + dkH_b(1/k) - dH_b'(\alpha)$. As $\alpha \to 0$ the limit is determined by the terms involving $H_b'(\alpha)$ and $H'(\alpha k)$, which go to $-\infty$ while $dkH_b(1/k)$ is constant. Since $kH_b'(\alpha k) - (d-1)H_b'(\alpha) = k(\log_2(1-\alpha k) - \log_2(\alpha k)) < -k\log_2(\alpha k) - (d-1)H_b'(\alpha)$ we get:

$$\lim_{\alpha \to 0} \phi' < \lim_{\alpha \to 0} -k\log_2(\alpha k) + (d-1)\log_2(\alpha) = \lim_{\alpha \to 0} (d-k-1)\log_2(\alpha) = -\infty$$

Now looking at the second derivative, $\phi''(\alpha) = k^2 H_b''(\alpha k) - (d-1)H''(\alpha)$:

$$\phi''(\alpha) = \frac{-1}{\log_2(e)\alpha}\left(\frac{k}{1-\alpha k} + \frac{d-1}{1-\alpha}\right)$$

Hence $\phi'' > 0$ if and only if $(d-1)(1-\alpha k) > k(1-\alpha)$. Rearranging the equation, we get $\alpha < \frac{d-k-1}{k(d-2)}$. □

Corollary 1. *For $d = 8$ Chung's construction is an 8-regular bipartite graph such that every subset of at most 1/3 of the nodes is 2-expanding, i.e. it is an $(n, \alpha, 2\alpha)$-bipartite expander for every $\alpha \leq 1/3$ with overwhelming probability.*

Proof. Plugging $\alpha = 1/3$ and $\beta = 2/3$ into the formula for degree (Equation 2.1) gives $d = 7.21 < 8$. With $d = 8$ and $k = 2$ the condition in Lemma 2 is satisfied: $\alpha = 1/3 < (d-k-1)/k(d-2) = 5/12$.

□

For fixed d the expansion improves further as α decreases. One can easily verify (by repeated application of L'Hopital's rule) that the limit as $\alpha \to 0$ of the expression $(H_b(\alpha) + H_b(k\alpha))/(H_b(\alpha) - k\alpha H_b(1/k))$ goes to $k+1$. Of course, when α becomes too small the expansion no longer holds with overwhelming probability. Figure 2.2 provides a table of expansion factors over a range of α with fixed degree $d = 8$. Figure 2.3 plots the expansion as a function of subset size.

Expansion boundary In the analysis of our constructions we will also look at the *boundary* of subsets in an expander graph. In an undirected expander, this is simply $\Gamma(S)$ as we defined already. In the case of bipartite expanders, the analogous "boundary" of a set of sources is the set of sinks connected to these sources that have distinct index labels from the sources. We can lower bound the size of the boundary by $\beta - \alpha$. Lemma 1 gives a smooth lower bound on $\beta - \alpha$ for Chung's bipartite expander graphs that we can show has a unique local maximum in $(0, 1)$. Conveniently, this allows us to lower bound the value of $\alpha - \beta$ in any given range by examining just the end points. In particular, when $d = 8$ this achieves a local max at approximately $\alpha = 1/3$ (where the expansion factor reaches 2), which can be seen in Figure 2.3. We prove the claim analytically for arbitrary d.

To simplify the analysis, we look at the function defined by the zeros of $\phi(\alpha, \beta) = d(\beta H_b(\alpha/\beta) - H_b(\alpha)) + 2 = 0$. Any α, β satisfying this relation also satisfies the relation in Lemma 1 because $H_b(\alpha) + H_b(\beta) < 2$ when $\beta > \alpha$. (More generally we can substitute c if it is known that $H_b(\alpha) + H_b(\beta) < c$). This implicitly defines β as a function of α, as well as the function $\hat{\beta} = \beta - \alpha$. More precisely, The implicit function $\hat{\beta}(\alpha)$ defined by pairs of points $(\alpha, \hat{\beta}(\alpha))$ such that $\phi(\alpha, \alpha + \hat{\beta}(\alpha)) = 0$ is a lower bound to the boundary of subsets of size α, which holds at any point α with probability at least $1 - \mathsf{negl}(nH_b(\alpha))$.

Lemma 3. *Define $\phi(x, y) = d(yH_b(x/y) - H_b(x)) + c$ where c is any constant and let $\hat{\beta}$ be the function on $(0, 1)$ defined by pairs of points $(\alpha, \beta - \alpha)$ such that $\phi(\alpha, \beta) = 0$ and $0 < \alpha < \beta < 1$. The function $\hat{\beta}$ is continuously differentiable on $(0, 1)$ and has a unique local maximum.*

Proof. Define a change of variables $z = x - y$ to get $\phi(x, z) = d(x+z)H_b(x/(x+z)) - H_b(x)) + c$ so that $\hat{\beta}$ is the function defined implicitly by pairs of points satisfying $\phi(\alpha, \hat{\beta}) = 0$. The function

$\phi(x,z)$ is continuously differentiable in both variables on the set $(0,1)\times(0,1)$, i.e. its partial derivates $\phi_z = \frac{\partial\phi}{\partial z}$ and $\phi_x = \frac{\partial\phi}{\partial x}$ are each continuous on $(0,1)\times(0,1)$. By the Implicit Function Theorem, $\hat{\beta} : (0,1) \to \mathbb{R}$ is continuously differentiable with derivative $-\phi_x/\phi_z$ defined in an open interval around every point α where $\phi_z \neq 0$. Expanding the binary entropy function $H_b(x) = -x\log_2(x) - (1-x)\log_2(1-x)$ simplifies the inner expression:

$$\begin{aligned}\frac{1}{d}(\phi(x,z)-c) &= (z+x)H_b(x/(x+z)) - H_b(x)\\ &= -x\log_2(x/(z+x)) - z\log_2(z/(x+z)) + x\log_2(x) - (1-x)\log_2(1-x)\\ &= (x+z)\log_2(x+z) - z\log_2(z) - (1-x)\log_2(1-x)\end{aligned}$$

The partial derivatives are:

$$\phi_x = \frac{d}{\ln(2)}((x+z)/(x+z) + \ln(x+z) - (1-\alpha)/(1-\alpha) - \ln(1-\alpha) = d\log_2\left(\frac{x+z}{1-\alpha}\right)$$
$$\phi_z = \frac{d}{\ln(2)}((x+z)(x+z) + \ln(x+z) - z/z - \ln(z) = d\log_2\left(\frac{x+z}{z}\right)$$

$\phi_z(\alpha,\hat{\beta}(\alpha))$ is always positive because $(x+z)/z > 1$ for $x,z \in (0,1)$. On the other hand, $\phi_x(\alpha,\hat{\beta}(\alpha) < 0$ if and only if $\alpha + \hat{\beta}(\alpha) > 1-\alpha$. Setting $\hat{\beta}(\alpha) = \beta - \alpha$ this is the case when $\beta + \alpha > 1$. $\hat{\beta}(\alpha)$ is increasing when $\alpha+\beta < 1$ and decreasing when $\alpha+\beta > 1$. It has a local maximum where $\alpha+\beta = 1$. Since $\beta = \hat{\beta}(\alpha) - \alpha$ is a monotonically non-decreasing function of α it follows that if $\alpha+\beta < 1$ then $\alpha'+\beta' < 1$ at every point $\alpha' < \alpha$. Likewise, if $\alpha+\beta > 1$ then $\alpha'+\beta' > 1$ at every point $\alpha' > \alpha$. We therefore conclude that $\hat{\beta}(\alpha)$ is initially increasing and achieves a unique local maximum on $(0,1)$.

□

Corollary 2. *With overwhelming probability in n, Chung's construction (with $d = 8$) is an 8-regular bipartite graph on n sinks and n sources each indexed in $[n]$ such that for all $\alpha \in (0.10, 0.80)$ every αn sinks are connected to at least $0.12n$ sources with distinct indices.*

Proof. The function $\hat{\beta}(\alpha)$ from Lemma 3 the implicit function $\hat{\beta}_c$ defined by pairs of points $(\alpha, \beta-\alpha)$ satisfying $\phi_c(\alpha,\beta) = d(\beta H_b(\alpha/\beta) - H_b(\alpha)) + c = 0$ is a smooth and has a unique maximum in any interval $L \subseteq (0,1)$. Furthermore, if the points α and β satisfy $H_b(\alpha)+H_b(\beta) < c \leq 2$ then $\hat{\beta}(\alpha)$ is a lower bound on the "boundary" of sinks of size αn in Chung's construction with overwhelming probability (Lemma 1). We will split the interval $(0.10, 0.80)$ into subintervals $(0.10, 0.33]$ and $[0.33, 0.80)$ and analyze them seperately.

For all $\alpha \in [0.33, 0.80)$ the formula in Lemma 1 shows that the expansion for each α is non-decreasing and is at least 0.80 (see Figure 2.3). Thus we can set $c = 1.64 > H_b(0.33) + H_b(0.80)$ and examine the lower bound $\hat{\beta}_c$. It has a unique local maximum in $(0.33, 0.80)$ therefore $\hat{\beta}_c \geq min(\hat{\beta}_c(0.33), \hat{\beta}_c(0.80)) \geq 0.12$. (Observe that $\phi_c(0.33, 0.45) < 0$ and $\phi_c(0.80, 0.92) < 0$ with $d = 8$.)

For $\alpha \in (0.10, 0.33]$ we lazily set $c = 2$. The implicit function $\hat{\beta}_2$ has a unique local max in $(0.10, 0.33]$, so $\hat{\beta}_2 \geq min(\hat{\beta}_2(0.33), \hat{\beta}_2(0.10)) > 0.12$. □

Size (α)	0.01	0.10	0.20	0.30	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80
Expansion (β)	0.04	0.33	0.53	0.65	0.75	0.78	0.81	0.84	0.88	0.89	0.91	0.93	0.94
Factor (β/α)	4	3.3	2.65	2.1	1.8	1.73	1.62	1.53	1.47	1.37	1.3	1.24	1.17

Figure 2.2: A table of the maximum expansion (β) satisfying the condition from Lemma 1 for Chung's construction with fixed degree $d = 8$ over a range of subset sizes (α).

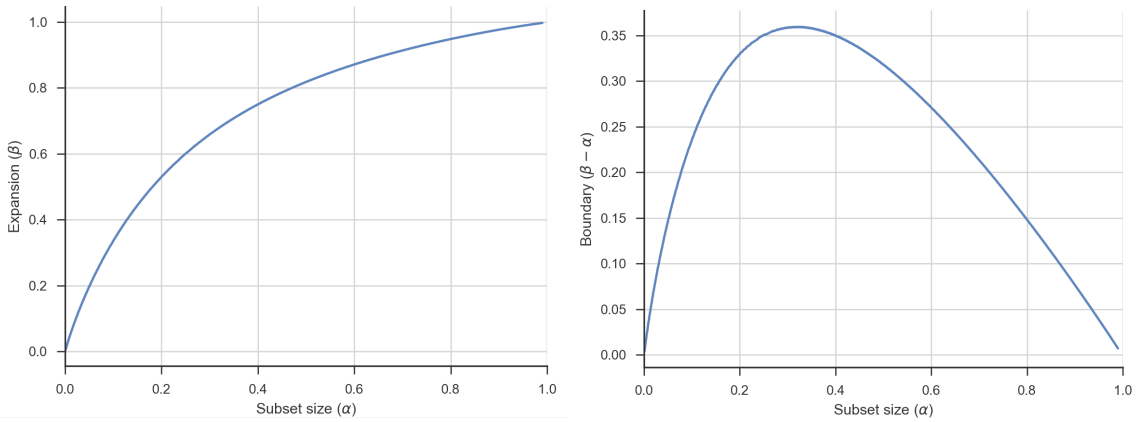

Figure 2.3: The graph on the left plots the lower bound from Lemma 1 on the expansion β as a function of the subset size α (in fractions of the sources/sinks) for Chung's construction with fixed $d = 8$. The graph on the right plots the corresponding lower bound on $\beta - \alpha$, which is the analog of the subgraph boundary in non-bipartite expanders. Specifically, this is a lower bound on the fraction of sinks connected to an α fraction of sources that have distinct index labels from the sources.

Expansion vs degree We can also characterize the expansion of subsets of the sinks more generally as a function of the graph's degree. We can show that for any $d \geq 4$ Chung's construction yields a d-regular graph such that every subset of sinks of size αn is at least $(d/3)$-expanding for *every* $\alpha < \frac{3}{2d}$.

Lemma 4. *For any $d \geq 4$, Chung's construction yields a d-regular bipartite graph that is an $(n, \alpha, (d/3)\alpha)$ bipartite expander for every $\alpha \leq \frac{3}{2d}$ with probability $1 - \mathsf{negl}(nH_b(\alpha))$.*

Proof. First set $\alpha = 3/(2d)$ and $\beta = 1/2$. By Lemma 1, we obtain a graph $\mathcal{G}$ that is an (n, α, β) expander with probability $1 - \mathsf{negl}(nH_b(\alpha))$ as long as $d - \frac{d}{2}H_b(\frac{3}{d}) - H_b(\frac{3}{2d}) - 1 > 0$. Define $g(x) = 1/x - \frac{1}{2x}H_b(3x) - H_b(3x/2) - 1$ so that the condition is equivalent to $g(1/d) > 0$. $H_b(x)$ is real and continuous on $(0, 1)$. Its derivative is $H_b'(x) = \log_2(1 - x) - \log_2(x)$ which is positive on $(0, 1/2)$. Differentiating $g(x)$ with respect to x on $(0, 1/3)$ gives $g'(x) = -1/x^2 - \frac{3}{2x}H_b'(3x) + \frac{1}{2x^2}H_b(3x) - \frac{3}{2}H_b'(\frac{3x}{2})$. Observe that on $(0, 1/3)$ both $H_b'(3x) > 0$ and $\frac{1}{2}H_b(3x) - 1 < 0$, therefore $g'(x) < 0$. This means that $g(1/d)$ is increasing as d increases for $d > 3$. Furthermore $\lim_{d\to 3} g(1/d) = 3 - H_b(1/2) - 1 = 1 > 0$.

Finally, it follows as a special case of Lemma 2 that if $\mathcal{G}$ is a d-regular $(n, \frac{3}{2d}, \frac{1}{2})$ bipartite expander for some fixed d then it is an $(n, \alpha, (d/3)\alpha)$ bipartite expander for every $\alpha \leq \frac{3}{2d}$. Setting $\epsilon = 1/3$, we see that $\frac{3}{2d} < \frac{2-1/d}{d-2} = \frac{1/\epsilon - 1/d - 1}{d-2}$. The inequality holds because $4d - 2 > 3d - 6$ for all $d > 0$.

□

Ramanujan graphs The explicit bipartite expander construction of Lubotzky, Phillips, and Sarnak [4] achieves similar expansion to Chung's construction for similarly good parameters. It is more complicated to implement as it involves generating a certain Cayley graph of the group $PGL(2, \mathbb{F}_q)$, the 2-dimensional projective general linear group on $\mathbb{F}_q$. This construction yields a bipartite exapnder of degree $(p+1)$ on $q(q^2-1)$ vertices for any primes p, q such that $p, q \equiv 1 \bmod 4$ and p is a quadratic non-residue modulo q. The resulting graph is called a *Ramanujan graph* because it is an optimal *spectral expander*, meaning the absolute value of all the non-trivial eigenvalues of its adjacency matrix (i.e. except the eigenvalue -d) are bounded by $2\sqrt{d-1}$. Other explicit Ramanujan graphs are known, for instance the isogeny graph of supersingular elliptic curves is also Ramanujan [34]. Due to a theorem of Tanner [38] relating spectral and vertex expansion, any d-regular bipartite Ramanujan graph on n vertices is an $(n, \alpha, \frac{d}{4(1-\alpha)+\alpha d})$ bipartite expander for all $\alpha < 1$. In particular, for $d \geq 4$ every fraction of $\alpha < 1/d$ sinks is at least $d/4$-expanding.

Lemma 5 (Tanner [38])**.** *For any $d \geq 4$ and $n \in \mathbb{N}$ a d-regular bipartite Ramanujan graph on n vertices is an $(n, \alpha, (d/4)\alpha)$ bipartite expander for every $\alpha < 1/d$.*

3 Stacked DRG Proof of Space

In this section we show that stacking DRGs with bipartite expander edges between layers yields an arbitrarily tight proof of space with the number of layers increasing as $O(\log_2(1/\epsilon))$ where ϵ is the desired space gap. Moreover, the proof size is also $O(\log_2(1/\epsilon))$, which is asymptotically optimal. Our proofs attempt a tight analysis as well, e.g. showing that just 10 layers achieve a PoS with a 1% space gap, degree $8+d$ graphs where d is the degree of the DRG, and relies only on a DRG that retains depth in 80% subgraphs.

3.1 Review of the Stacked-Expander PoS

In this section we review the PoS construction by Ren and Devadas [35] based on stacked bipartite expander graphs as it is a building block towards our Layered-DRG construction. Their construction uses a layered graph where each layer is a directed line on n nodes and the directed edges of a bipartite expander graph are placed between layers. This was shown to be an $(\epsilon\gamma n, (1-2\epsilon)\gamma n)$-sound PoS for parameters $\epsilon < 1/2$ and $\gamma < 1$.[3] (Achieving practical proof sizes actually requires γ to be rather small as otherwise the required degree of the expander graphs blows up, e.g. $\gamma > 0.6$ requires at least degree 40 graphs hence practically $\epsilon < 1/3$). The PoS is not parallel sound (for meaningful s, t) as a prover running in $O(\lambda)$ parallel time can pass verification using very little space. We will show that by replacing each line graph with a depth robust graph this results in a much tighter proof of space, as well as security against parallelism. Interestingly, the security against parallel attacks seems intimately connected to why the Ren-Devadas construction fails to be tight whereas ours succeeds. Furthermore, the Ren-Devadas security is only proven against an adversary who implements a pebbling attack and is not yet known to be secure more generally against graph labeling in the random oracle model. As our

[3] Under a slightly different definition of a PoS that assumes the prover uses less than $(1-\epsilon)\gamma n$ additional space during execution, Ren and Devadas showed that the stacked-expander construction is an $(\epsilon\gamma n, 2^\lambda)$-sound PoS where λ is a security parameter determining the number of layers in their construction.

construction is secure against parallel pebbling attacks it is also secure in the random oracle model.

The graph $\mathcal{G}_{SE}$ The stacked-expander PoS uses the same underlying graph as the Balloon Hash memory hard function [22]. The graph $\mathcal{G}_{SE}$ consists of $\ell = O(\lambda)$ layers $V_1, ..., V_\ell$ consisting each of n vertices indexed in each level by the integers $[n]$, and where λ is a security parameter. First directed edges are placed from each kth vertex to the $k+1$st vertex in each level, i.e. forming a directed line. Next directed edges are placed from V_{i-1} to V_i according to the edges of an (n, α, β) bipartite expander on (V_{i-1}, V_i). Finally a "localization" operation is applied so that each kth vertex u_k in V_{i-1} is connected to the kth vertex v_k in V_i and any directed edge from the kth vertex of V_{i-1} to some jth vertex of V_i where $j > k$ is replaced with a directed edge from the kth vertex of V_i to the jth vertex of V_i. $\mathcal{G}_{SE}$ can be pebbled in $n\ell$ steps using a total of n pebbles.

Stacked-expander PoS The PoS follows the generic PoS based on graph labeling. We remark only on several nuances. Due to the topology of $\mathcal{G}_{SE}$ after localization, the prover only needs to use a buffer of size n and deletes the labels of V_{i-1} as it derives the labels of V_i. After completing the labels C_i in the ith level it computes a vector commitment (e.g. Merkle) to the labels in C_i denoted com_i. Once it has derived the labels C_ℓ of the final level V_ℓ it computes $com = H_{id}(com_1||\cdots||com_\ell)$ and uses $H_{id}(com||j)$ to derive λ non-interactive challenges for each jth level.[4]

Let $\gamma = \beta - 2\alpha$ for constants $\beta, \alpha \in (0, 1)$ where $\beta > 2\alpha$. For the remainder of the analysis we assume the bipartite expander graph used in the stack-expander PoS construction is an (n, α, β) expander.

Theorem 2 (RD [35])**.** *Let ℓ be the number of layers in G_{SE} and α, β the expansion parameters. Let $\gamma = \beta - 2\alpha$. Every αn subset of initially unpebbled sinks in the layer V_ℓ cannot be pebbled in less than $2^\ell \alpha n$ moves from any initial configuration of γn and using at most γn pebbles overall.*

Corollary 3 (RD [35])**.** *For every $\epsilon < 1/2$ and $0 < \delta < \epsilon\gamma$ arbitrarily small, the stacked-expander PoS construction is an $((\epsilon\gamma - \delta)nm, (1 - 2\epsilon)\gamma n)$-sound PoS (against a pebbling adversary) with probability $1 - \mathsf{negl}(\delta\lambda)$.*

Space-hardness gap 1/2 The stacked-expander PoS leaves at least a $\frac{1}{2}\gamma$ gap between the honest prover's storage and the adversary's storage. In fact, the space gap itself is actually what the analysis exploits in order to argue security: due to the fact that the adversary will need to refill a large fraction of its deleted space in order to pass the verifier's challenges it will end up performing a significant amount of computation as well to refill that space. Furthermore,

[4] This is a slight deviation from the protocol as described by Ren and Devadas, which sampled every label challenge randomly over all vertices in $\mathcal{G}_{SE}$ rather than separately within each layer. The end result is the same because for security they set their parameters to ensure that if at least ϵn labels *in any level* are incorrect (which comprise an ϵ/ℓ fraction of all the vertices) then the challenges will sample one of these incorrect labels with overwhelming probability. This requires sampling a factor ℓ more labels overall than the number of labels one would sample from a given level to achieve the same probability of detection within that specific layer, as for any λ it holds that $(1 - \frac{\epsilon}{\ell})^{\ell\lambda} \approx e^{-\lambda/\epsilon} \approx (1 - \epsilon)^\lambda$. Since their protocol also require $\ell = O(\lambda)$ the number of queries is actually $O(\lambda^2)$. Sampling λ challenges within each layer, as we do, involves the same number of queries with even a slightly tighter guarantee of the desired property within each level.

Corollary 3 proves that nodes on the last level are hard to pebble individually by leveraging the fact that the adversary is storing less than half the pebbles required to pebble a subset of nodes (as required by Theorem 2) and derives a contradiction by considering a lazy adversarial strategy that keeps these pebbles fixed on the initial set and still uses less space than needed. This analysis would be void if the adversary were storing more than half the required online pebbles.

It seems implicitly that the reason this bound on additional online space used is so fundamental to the analysis is that the protocol is not secure against parallel attacks. If instead pebbling any αn sinks was secure against parallel attacks (with no bound on the number of online pebbles used in parallel) then we immediately get that less than αn sinks can be pebbled efficiently individually as otherwise these could all be pebbled in parallel starting from the same initial configuration.

$\mathbf{O(\lambda^2)}$ proof size The proof size in the stacked-expander PoS is quite large as it requires λ challenges in each of the λ levels for total complexity $O(\lambda^2)$. The number of levels needs to be λ as we are only able to prove that the complexity of pebbling a single node out of αn sinks is at least $1/(\alpha n)$ the complexity of pebbling all αn, hence the complexity of pebbling αn nodes needs to be on the order of $2^{\lambda}\alpha n$, i.e. much greater than $O(n)$. If the complexity were only $O(n)$ the adversary might be able to pebble each challenge in less that $1/\alpha$ moves. Again, this is connected to the lack of security against parallel attacks. If the protocol were secure against parallel attacks then individual nodes inherit the pebbling complexity of αn nodes without a factor αn loss.

3.2 A tight PoS from stacked DRGs

The new result that we will next show is that simply replacing each of the line graphs V_i in the stacked-expander PoS construction with a depth robust graph results in an arbitrarily tight PoS. Specifically, only $O(\log 1/\epsilon)$ layers are needed to achieve a $((1-\epsilon)n, \Omega(n))$-parallel-sound PoS. We demand only very basic properties from the DRG, e.g. that any subgraph on 80% of the nodes contains a long path of $\Omega(n)$ length.

Construction of $\mathcal{G}_{SDR}[\ell]$ The graph $\mathcal{G}_{SDR}[\ell]$ will be exactly like $\mathcal{G}_{SE}$ only each of the ℓ layers $V_1, ..., V_\ell$ contains a copy of an $(n, 0.80n, \beta n)$-depth-robust graph for some constant β. For concreteness, we define the directed edges between the layers using the degree 8 Chung random bipartite graph construction.

For simplicity we will first analyze the construction without applying localization to the expander edges between layers. Even without localization this is already a valid PoS, only the initialization requires a buffer of size $2n$ rather than n. The PoS is still "tight" with respect to the persistent space storage.

Vector commitment storage If the vector commitment storage overhead required for the PoS is significant then this somewhat defeats the point of a tight PoS. Luckily this is not the case. Most vector commitment protocols, including the standard Merkle tree, offer smooth time/space tradeoffs. With a Merkle tree the honest prover can delete the hashes on nodes on the first k levels of the tree to save a factor 2^k space and re-derive all hashes along a Merkle path by reading at most 2^k nodes and computing at most 2^k hashes. If $k = 7$ this is less than a

1% overhead in space, and requires at most 128 additional hashes and reads. Furthermore, as remarked in [33] these 2^k reads are sequential memory reads, which in practice are inexpensive compared to the random reads for challenge labels.

Proof size We show that $\ell = O(\log(\frac{1}{3(\epsilon-2\delta)}))$ suffices to achieve $\mathsf{negl}(\lambda)$ soundness against any prover running in parallel time less than βn rounds of queries where Chal samples λ/δ nodes in each layer. This would result in a proof size of $O((1/\epsilon)\log(1/\epsilon))$, which is already a major improvement on any PoS involving a graph of degree $O(1/\epsilon)$ (recall that the only previously known tight PoS construction relied on very special DRGs whose degree must scale with $1/\epsilon$, which results in a total proof size of $O(1/\epsilon^2)$). However, we are able to improve the result even further and show that only $O(1/\delta)$ challenge queries are required overall, achieving proof complexity $O(1/\epsilon)$. This is the optimal proof complexity for the generic pebbling-based PoS with at most an ϵ space gap. If the prover claims to be storing n pebbles and the proof queries less than $1/\epsilon$ then a random deletion of an ϵ fraction of these pebbles evades detection with probability at least $(1-\epsilon)^{1/\epsilon} \approx 1/e$. The same applies if a random ϵ fraction of the pebbles the prover claims to be storing are red (i.e. errors).

Analysis outline We prove the hardness of the game $\mathsf{Red\text{-}Black\text{-}Pebbles}^{\mathcal{A}}(\mathcal{G}_{SDR}[\ell], V_\ell, \mathsf{Chal})$ where Chal samples λ_i uniform challenges over V_i. We first show that it suffices to consider the parallel complexity of pebbling the set $U_\ell \subseteq V_\ell$ of *all* unpebbled nodes on V_ℓ from an initial configuration of γn black pebbles overall and $\delta_i n$ red pebbles in each layer where $\delta_\ell < \epsilon/2$. As a shorthand notation, we will say that $\mathcal{G}_{SDR}[\ell]$ is $(\gamma, \vec{\delta}, t, \mu)$-hard if t rounds are required to pebble a μ fraction of V_ℓ. When $\mu = 1$ this is equivalent to pebbling all of U_ℓ, the set of nodes with missing pebbles in the initial configuration (since there is no constraint on number of pebbles used). This very similar to standard parallel pebbling complexity in black pebbling games as we defined earlier, however, it takes into account the restriction to δ_i red pebbles on each layer that it counts separately[5] from black pebbles.

In Claim 4 we show that if $\mathcal{G}_{SDR}[\ell-1]$ is $(1-\epsilon+2\delta_{\ell-1}, \vec{\delta}, t, 1)$-hard then $\mathcal{G}_{SDR}[\ell]$ is $(1-\epsilon, \vec{\delta}^*, t, 1-\epsilon/2)$-hard where $\vec{\delta}^*$ is equal to $\vec{\delta}$ on all common indices and $\delta_\ell = \delta_{\ell-1}$. This in turn implies that with probability at least $1-\epsilon/2$ a randomly sampled challenge node from V_ℓ requires more than t rounds to pebble. Together with the random challenges bounding δ_i we show (Claim 2) that the labeling PoS on $\mathcal{G}_{SDR}$ is $(\gamma n, t, max\{p^*, \mu^\kappa\})$-sound where $p* = max_i(1-\delta_i)^{\lambda_i}$.

Next we look at the complexity of pebbling all of V_ℓ. We show in Claim 6 that when the adversary uses at most $\gamma < 1-\epsilon$ black pebbles and δ red pebbles in each layer then pebbling all the unpebbled nodes in layer V_ℓ (for ℓ dependent on ϵ and δ) requires pebbling $0.80n$ unpebbled nodes (including both red and black pebbles) in some layer V_i. Since the layer V_i contains a $(n, 0.80, \beta n)$-depth-robust graph, this takes at least βn rounds. We then generalize this analysis (Claim 7) to apply when δ_i is allowed to increase from level ℓ to 1 by a multiplicative factor such that $\sum_i \delta_i = O(\delta_\ell)$.

Theorem 3 ties everything together, taking into account the constraints of each claim to derive the PoS soundness of the labeling PoS on $\mathcal{G}_{SDR}[\ell]$.

[5]In prior uses of the red-black pebbling game (e.g. to analyze the Ren and Devadas or Dziembowski et. al. proofs of space) it sufficed to consider parallel black pebbling complexity as red pebbles were treated just like extra free black pebbles. However, that is not allowed here because the construction places different constraints on red vs black pebbles that are allowed on each level by issuing separate challenge queries for each level.

Theorem 3. *The labeling PoS on $\mathcal{G}_{SDR}[\ell]$ with* Chal *sampling λ_i challenges in each level V_i and κ online challenges in V_ℓ is $((1-\epsilon-\delta)n, \beta n-1, e^{-\lambda})$-sound with $\kappa = 2\lambda/\epsilon$ if either of the following conditions are met for $\epsilon \leq 0.24$:*

(a) $\ell = max(8, \log_2(\frac{1}{3(\epsilon-2\delta)}) + 4)$ and each $\lambda_i = \lambda/\delta$ and $\delta < min(0.01, \epsilon/3)$

(b) $\ell = max(14, \log_2(\frac{1}{3(\epsilon-3\delta)}) + 5)$ and each $\lambda_i = \lambda/\delta_i$ where $\delta_\ell = \delta < min(0.01, \epsilon/2)$ and $\delta_i = min(0.05, \frac{2}{3}\delta_{i-1})$

Proof. For any $\mathcal{G}_{SDR}[\ell]$, if the set of unpebbled nodes in V_ℓ are connected via unpebbled paths to at least $0.80n$ unpebbled nodes (including red and black) in some prior level V_i, then pebbling all of V_ℓ requires pebbling all these $0.80n$ unpebbled nodes, which requires $\beta n-1$ rounds due to the fact that V_i is $(n, 0.80n, \beta n)$ depth robust. Claim 7 thus implies that $\mathcal{G}_{SDR}[\ell]$ is $(1-\epsilon, \vec{\delta}, \beta n-1, 1)$-hard for $\vec{\delta}$ such that $\delta_\ell = \delta < \epsilon/3$ and $\delta_i = min(0.05, \frac{2}{3}\delta_{i-1})$ for $\ell = max(13, \log_2(\frac{1}{3(\epsilon-3\delta)}) + 4)$. Claim 7 gives a different tradeoff between ℓ and $\vec{\delta}$, and shows that this hardness holds for $\vec{\delta}$ and ℓ such that $\delta_i = \delta < \epsilon/2$ for all i and $\ell = max(7, \log_2(\frac{1}{3(\epsilon-2\delta)} + 3))$.

Assuming $\epsilon \leq 0.24$, Claim 4 implies that $\mathcal{G}_{SDR}[\ell+1]$ is $(1-\epsilon-\delta, \vec{\delta}, \beta n-1, 1-\epsilon/2)$-hard extending $\vec{\delta}$ so that $\delta_{\ell+1} = \delta_\ell = \delta$. Finally, by Claim 2, the labeling PoS on $\mathcal{G}_{SDR}[\ell+1]$ with challenge set V_ℓ and Chal sampling λ_i in each level V_i is $((1-\epsilon-\delta)n, \beta n-1, max\{p^*, (1-\epsilon/2)^\kappa\})$-sound where $p* = max_i(1-\delta_i)^{\lambda_i}$. Setting $\lambda_i = \lambda/\delta_i$ and $\kappa = 2\lambda/\epsilon$, the PoS is $((1-\epsilon-\delta)n, \beta n-1, e^{-\lambda})$-sound.

□

Notation 1 (Common analysis notations)**.** *Let U_i denote the entire index set of nodes that are unpebbled in V_i and P_i the set that are pebbled. The total number of pebbles placed in the initial configuration is γn. Each level initially has $\rho_i n$ black pebbles and $\delta_i n$ red pebbles. Finally, $\gamma_i n = \sum_{j<i} \rho_i$ is the number of black pebbles placed before level i.*

Claim 2. *If $\mathcal{G}_{SDR}[\ell]$ is $(\gamma, \vec{\delta}, t, \mu)$-hard then the labeling PoS on $\mathcal{G}_{SDR}[\ell]$ is $(\gamma n, t, max\{p^*, \mu^\kappa\})$-sound where $p* = max_i(1-\delta_i)^{\lambda_i}$.*

Proof. Fix $\gamma = 1-\epsilon$ and δ_i for each i. The λ_i challenges during Initialization in each level ensure that $\mathcal{A}$ wins with at most probability $(1-\delta_i)^{\lambda_i}$ if it places more than δ_i red pebbles on V_i. If $\mathcal{A}$ has exceeded the δ_i bound in more than one level this only increases its probability of failure. Thus, in case 1 ($\mathcal{A}$ places more than δ_i red pebbles on some level i), $\mathcal{A}$'s success probability is bounded by maximum value of $(1-\delta_i)^{\lambda_i}$ over all i. In case 2 ($\mathcal{A}$ places fewer than δ_i on each ith level), the fact that $\mathcal{G}_{SDR}[\ell]$ is $(\gamma, \vec{\delta}, t, \mu)$-hard implies at most a μ fraction of the nodes on V_ℓ can individually be pebbled from the starting configuration in t rounds (recall Fact 1), hence $\mathcal{A}$'s success probability of answering κ independent challenges is μ^κ. Therefore, the overall success probability is the maximum of these two cases. □

The next claim is a very simple observation that the hardness of pebbling all the unpebbled nodes in V_ℓ with s initial pebbles overall can be equivalently stated as the hardness of pebbling any set of u unpebbled nodes on V_ℓ with $s-u$ initial pebbles placed on lower levels (i.e. $V_1, .., V_{\ell-1}$).

Claim 3 (trivial)**.** *$\mathcal{G}_{SDR}[\ell]$ is $(\gamma, \vec{\delta}, t, 1)$-hard if and only if given any initial configuration P_0 of $\gamma_\ell n$ black pebbles placed on layers $V_1, ..., V_{\ell-1}$ (i.e. excluding V_ℓ), at most $\delta_i n$ red pebbles in each layer, and any set $U \subseteq V_\ell$ of αn unpebbled nodes in V_ℓ such that $\alpha - \gamma_\ell \geq 1 - \gamma - \delta$, no adversary can pebble U in t or fewer rounds.*

Proof. Any placement of $\delta_i n$ red pebbles on each level and γn black pebbles overall with γ_ℓ pebbles on $V_1, ..., V_{\ell-1}$ has $\rho_\ell n = (\gamma - \gamma_\ell)n$ black pebbles on V_ℓ and exactly $(1 - \rho_\ell - \delta_\ell)n$ unpebbled nodes in V_ℓ (assuming without loss of generality that red and black pebbles never share the same node). If for any γ_ℓ pebbles placed on $V_1, ..., V_{\ell-1}$ and any set U of $(1 - \rho_\ell - \delta_\ell)n$ of unpebbled nodes in V_ℓ no adversary can pebble U in t or fewer rounds, then no adversary can pebble the set of unpebbled nodes in V_ℓ (i.e. because U is precisely this set).

Conversely, suppose that $\mathcal{G}_{SDR}[\ell]$ is $(\gamma, \vec{\delta}, t, 1)$-hard, i.e. pebbling the entire set of unpebbled nodes in V_ℓ requires t rounds if at most γn black pebbles are placed overall with at most $\delta_i n$ red pebbles in each level. Then for any placement of γ_ℓ pebbles on $V_1, ..., V_{\ell-1}$ with at most $\delta_i n$ red in each level, and for any $U \subseteq V_\ell$ of size $u \geq (1 - \rho_\ell - \delta_\ell)n$, the remaining nodes in $V_\ell \setminus U$ can be pebbled with just $(\gamma - \gamma_\ell + \delta_\ell)n$ pebbles. This means there exists a pebbling of the entire graph with γn pebbles overall and the same configuration on the first $\ell - 1$ levels that leaves only the set U unpebbled in V_ℓ. Thus, by hypothesis, this unpebbled set U requires more than t rounds to pebble. Any other initial pebbling configuration that has fewer pebbles on $V_\ell \setminus U$ would only make it harder to pebble U. □

Claim 4. *For any $\epsilon \leq 0.24$, if $\mathcal{G}_{SDR}[\ell - 1]$ is $(1 - \epsilon + \delta_{\ell-1}, \vec{\delta}, t - 1, 1)$-hard then $\mathcal{G}_{SDR}[\ell]$ is $(1 - \epsilon, \vec{\delta}^*, min(\beta n - 1, t), 1 - \epsilon/2)$-hard where $\vec{\delta}^*$ is identical to $\vec{v}$ on all common indices and $\delta_\ell = \delta_{\ell-1} \leq \epsilon/2$.*

Proof. Refer to Notations 1. Consider the graph $\mathcal{G}_{SDR}[\ell]$ with $\gamma = 1 - \epsilon$ black initially placed pebbled. Let $\delta = \delta_\ell = \delta_{\ell-1} \leq \epsilon/2$. Let $\alpha_\ell n$ denote the size of U_ℓ, i.e. the number of unpebbled nodes in V_ℓ. Every subset of V_ℓ of size $(1 - \epsilon/2)n$ contains at least $\alpha^* n = (\alpha_\ell - \epsilon/2)n$ unpebbled nodes (otherwise there would be fewer than $\alpha_\ell n$ unpebbled nodes as there are only an $\epsilon/2$ fraction remaining outside the selected set). If $\alpha^* \geq 0.80n$ we are done because V_ℓ is a copy of a $(n, 0.80n, \beta n)$-depth robust graph and so pebbling all of α^* requires at least βn rounds to pebble a path of length βn.

Case $\alpha^* < 1/3$: The α^* unpebbled nodes have dependencies on at least a $2\alpha^* \geq 2\alpha_\ell - \epsilon$ fraction of nodes in $V_{\ell-1}$ (Corollary 1). Of these at least $\alpha' n = (2\alpha_\ell - \rho_{\ell-1} - \epsilon - \delta)n$ are unpebbled, and there are $\gamma_{\ell-1} = \gamma - \rho_{\ell-1} - \rho_\ell$ pebbles placed on all prior levels. Substituting $\rho_\ell = 1 - \alpha_\ell - \delta$ and $\gamma = 1 - \epsilon$ shows that $\alpha' - \gamma_{\ell-1} = 2\alpha_\ell - \gamma + \rho_\ell - \epsilon - \delta = \alpha_\ell + 1 - \gamma - \epsilon - 2\delta = \alpha_\ell - 2\delta$. Since $\alpha_\ell \geq 1 - \gamma - \delta$, setting $\gamma' = \delta + \gamma$ satisfies the relation $\alpha' - \gamma_{\ell-1} \geq \alpha_\ell - 2\delta \geq 1 - \gamma - 3\delta = 1 - \gamma' - \delta$. Noting that $\gamma' = 1 - \epsilon + \delta$, by the hypothesis that $\mathcal{G}_{SDR}[\ell - 1]$ is $(1 - \epsilon + \delta_{\ell-1}, \vec{\delta}, t - 1, 1)$-hard along with the bijection in Claim B this shows that the α' pebbles in $V_{\ell-1}$ cannot be pebbled in $t - 1$ rounds. Thus, the α^* unpebbled nodes in V_ℓ cannot be pebbled in t rounds.

Case $\alpha^* \geq 1/3$: In this case $\alpha^* \in (0.33, 0.80)$. It is connected to β^* nodes in $V_{\ell-1}$. Among these at least $\alpha' \geq \beta^* - \rho_{\ell-1} - \delta$ of these are unpebbled. Since $\gamma_{\ell-1} = \gamma - \rho_\ell - \rho_{\ell-1}$ we get $\alpha' - \gamma_{\ell-1} \geq \beta^* - \gamma + \rho_\ell - \delta$. Furthermore, $\rho_\ell = 1 - \alpha_\ell - \delta \geq 1 - \alpha^* - \epsilon/2 - \delta$ so $\alpha' - \gamma_{\ell-1} \geq \beta^* - \gamma + 1 - \alpha^* - \epsilon/2 - 2\delta = \beta^* - \alpha^* + \epsilon/2 - 2\delta \geq 0.12 + \epsilon/2 - 2\delta$ (from Corollary 2 we know that

$\beta^* - \alpha^* \geq 0.12$). As in the previous case, the hypothesis that $\mathcal{G}_{SDR}[\ell-1]$ is $(1-\epsilon+\delta_{\ell-1}, \vec{\delta}, t-1, 1)$-hard and Claim B imply that the α' pebbles cannot be pebbled in $t-1$ rounds as long as $0.12 + \epsilon/2 - 2\delta \geq \epsilon - 2\delta$, which is true for $\epsilon \leq 0.24$.

□

Claim 5. *If $\mathcal{G}_{SDR}$ initially has at most γn black pebbles for $\gamma \leq 1-\epsilon$ and at most $\delta < \epsilon/2$ red pebbles in each layer then for $\ell = \log_2(\frac{1}{3(\epsilon-2\delta)})$ the unpebbled nodes in V_ℓ have unpebbled paths from at least $n/3$ unpebbled nodes in some layer V_i.*

Proof. Refer to Notations 1. Let $\alpha_i n$ denote the number of unpebbled dependencies of U_ℓ in V_i, i.e. the number of nodes in U_i that have unpebbled paths to U_ℓ. Suppose that α_i is bounded by $1/3$ for all levels up to $\ell - k$, i.e. $\alpha_\ell < 1/3, ..., \alpha_{\ell-k} < 1/3$. We will prove the following bound:

$$\alpha_{\ell-k} \geq 2^k(\alpha_\ell - \gamma_\ell/2 - \delta) \geq 2^{k-1}(\alpha_\ell + \epsilon - 3\delta) \geq 2^k(\epsilon - 2\delta) \tag{3.1}$$

For $k = \log_2(\frac{1}{3(\epsilon-2\delta)})$ this implies $\alpha_{\ell-k} \geq 1/3$, which contradicts $\alpha_{\ell-k} < 1/3$. Therefore, it follows that $\alpha_i \geq 1/3$ at some index $i > \ell - \log_2(\frac{1}{3(\epsilon-2\delta)})$. We conclude that if $\ell \geq \log_2(\frac{1}{3(\epsilon-2\delta)})$ then there is some level V_i with at least $n/3$ unpebbled nodes that have unpebbled dependency paths to the set X_ℓ of unpebbled nodes in V_ℓ.

Let $j = \ell - i$. From Corollary 1, as long as $\alpha_j \leq 1/3$ the set X_j is connected to at least $2\alpha_j$ nodes in V_{j-1}, and at most $(\rho_{j-1} + \delta)n$ of these are pebbled. Therefore, $\alpha_{j-1} \geq 2\alpha_j - \rho_{j-1} - \delta$. Now we show by induction that $\alpha_{\ell-k} \geq 2^k\alpha_\ell - 2^{k-1}\rho_{\ell-1} - (2^k - 1)\delta$. The base case $k = 0$ is trivial. Assuming this holds for k:

$$\begin{aligned}\alpha_{\ell-k-1} &\geq 2\alpha_{\ell-k} - \rho_{\ell-k-1} - \delta \geq 2(2^k\alpha_\ell - 2^{k-1}\rho_{\ell-1} - (2^k-1)\delta) - \rho_{\ell-k-1} - \delta \\ &\geq 2^{k+1}\alpha_\ell - 2^k\rho_{\ell-1} - (2^{k+1}-1)\delta\end{aligned}$$

The last inequality used the fact that $\sum_{i=1}^{k} \rho_{\ell-i} \leq \gamma_\ell$ and therefore $\sum_{i=1}^{k} 2^{k-i}\rho_{\ell-i}$ is maximized by setting $\rho_{\ell-1} = \gamma_\ell$ and $\rho_{\ell-i} = 0$ for all $i > 1$.

From the identities $\gamma_\ell = \gamma - \rho_\ell$ and $\alpha_\ell = 1 - \rho_\ell - \delta$ we derive $\gamma_\ell = \gamma + \alpha_\ell - 1 + \delta \leq \alpha_\ell + \delta - \epsilon$. Finally, inserting this into the bound above and using the fact that $\alpha_\ell \geq \epsilon - \delta$ gives:

$$\alpha_{\ell-k} \geq 2^{k-1}(2\alpha_\ell - \gamma_\ell - 2\delta) \geq 2^{k-1}(\alpha_\ell + \epsilon - 3\delta) \geq 2^k(\epsilon - 2\delta)$$

□

We could stop here as we have already shown unpebbled dependency paths from the unpebbled sinks in V_ℓ to a $1/3$ fraction of nodes in some level for $\ell = O(\log(1/(\epsilon - 2\delta))$ and the remainder of our PoS analysis could rely on a graph that is $(n, 0.33n, \Omega(n))$-depth-robust. However, we can tighten the analysis further so that we only need to assume the graph is $(n, 0.80, \Omega(n))$-depth robust.

Claim 6. *If $\mathcal{G}_{SDR}$ initially has at most γn black pebbles for $\gamma \leq 1-\epsilon$ and at most $\delta < \epsilon/2$ red pebbles in each layer then for $\ell = max(\frac{0.68-\epsilon+\delta}{0.12-\delta}, \log_2(\frac{1}{3(\epsilon-2\delta)})) + 3)$ the unpebbled nodes in V_ℓ have unpebbled paths to at least $0.80n$ unpebbled nodes in some layer V_i. In particular, $\ell = max(7, \log_2(\frac{1}{3(\epsilon-2\delta)})) + 3)$ when $\delta \leq 0.01$.*

Proof. In Claim 5 we showed that for $\ell \geq \log_2(\frac{2}{3(\alpha_\ell+\epsilon-3\delta)})$ there exists an index i where $\alpha_i \geq 1/3$ and $\alpha_\ell + \epsilon - 3\delta \geq 2\epsilon - 4\delta$ (Equation 3.1). Picking up from here, we consider what happens once $\alpha_i \geq 1/3$. We break the analysis into two cases: in the first case $\alpha_\ell < 1/3$ and in the second case $\alpha_\ell \geq 1/3$.

In both cases we will use a different bound on α_{i-k} because once $\alpha_i > 1/3$ the unpebbled sets may not be 2-expanding. Define the function $\beta(\alpha)$ to be the minimum bipartite expansion of a set of fractional size α, i.e. every set of αn nodes is connected to at least $\beta(\alpha)n$ nodes in the previous level. Let $\hat{\beta}(\alpha) = \beta(\alpha) - \alpha$. Using the relation $\alpha_{i-1} \geq \beta(\alpha_i) - \rho_{i-1} - \delta$ we derive that $\alpha_{i-2} \geq \hat{\beta}(\alpha_{i-1}) + \beta(\alpha_i) - \rho_{i-1} - \rho_{i-2} - 2\delta$ and more generally, since $\sum_{j=1}^{k} \rho_{i-j} \leq \gamma_i$:

$$\alpha_{i-k} \geq \sum_{j=1}^{k-1} \hat{\beta}(\alpha_{i-j}) + \beta(\alpha_i) - k\delta - \sum_{j=1}^{k} \rho_{i-j} \geq (k-1)(min_{j<k}\hat{\beta}(\alpha_{i-j}) - \delta) + \beta(\alpha_i) - \gamma_i - \delta$$

The final ingredient is the bound $\gamma_i \leq \alpha_i - \epsilon + \delta$ for all i. To see this, first observe that $\alpha_\ell - \gamma_\ell = 1 - \rho_\ell - \delta - (\gamma - \rho_\ell) = \epsilon - \delta$. If $\alpha_i - \gamma_i \geq \epsilon - \delta$ and $\epsilon > 2\delta$, then $0.80 > \alpha_i > \delta$, and so the $\alpha_i n$ dependencies are connected to $\beta(\alpha_i)n > (\alpha_i + \delta)n$ nodes in level V_{i-1}. Therefore $\alpha_{i-1} \geq \alpha_i - \rho_{i-1} = \alpha_i - (\gamma_i - \gamma_{i-1})$. In words, decreasing the number of dependencies requires using black pebbles 1-to-1, so $\alpha_{i-1} - \gamma_{i-1} \geq \alpha_i - \gamma_i$.

$$\alpha_{i-k} \geq (k-1)(min_{j<k}\hat{\beta}(\alpha_{i-j}) - \delta) + \hat{\beta}(\alpha_i) + \epsilon - 2\delta \tag{3.2}$$

By Corollary 2 to Lemma 3, $\hat{\beta}(\alpha) \geq 0.12$ for $\alpha \in (0.10, 0.80)$.

Case $\alpha_\ell \geq 1/3$: First we claim that $\alpha_{\ell-i} \geq 0.12$ for all i. If $\alpha_i \geq 0.12$ then as shown above $\alpha_{i-1} \geq \hat{\beta}(\alpha_i) + \epsilon - 2\delta \geq 0.12$ because $\hat{\beta}(\alpha) \geq 0.12$ for all $\alpha \in (0.10, 0.80)$ and $\epsilon > 2\delta$. Our claim thus follows by induction. Therefore, for all $j \leq k$ we derive that $min_{j\leq k}(\hat{\beta}(\alpha_{\ell-j})) \geq 0.12$. Equation 3.2 then shows that $\alpha_{\ell-k-1} \geq k(0.12 - \delta) + 0.12 + \epsilon - \delta$, or $\alpha_{\ell-k-1} \geq 0.80$ at $k \geq (0.68 - \epsilon + \delta)/(0.12 - \delta)$ (e.g. $k = 7$ when $\delta \leq 0.01$).

Case $\alpha_\ell < 1/3$: From Equation 3.1, $\alpha_i \geq 1/3$ at some index $i \geq \ell - k$ for $k = \log_2(\frac{2}{3(\alpha_\ell+\epsilon-3\delta)})$. At this point $\alpha_i \geq 1/3$ and $\gamma_i < \gamma_\ell \leq \alpha_\ell - \epsilon + \delta$. Combining this with Equation 3.2, we can apply the same analysis as in the previous case to first show by induction that $\alpha_{i-k'} \geq 0.12$ for all k' and then more generally: $\alpha_{i-k'} \geq (k'-1)(0.12-\delta) + \beta(\alpha_i) - \alpha_\ell + \epsilon - 2\delta \geq k'(0.12-\delta) + 0.68 - \alpha_\ell + \epsilon - 2\delta$. We used the fact that $\beta(\alpha_i) \geq \beta(0.33) \geq 0.68$. Therefore, $\alpha_{i-k'-1} \geq 0.80$ when $k' \geq (0.80 - 0.68 + \alpha_\ell)/(0.12 - \delta)$. This shows that the total number of levels where $\alpha_i < 0.80$ is at most:

$$\ell = k + k' + 1 \leq 1 + \log_2\big(\frac{2}{3(\alpha_\ell + \epsilon - 3\delta)}\big) + \frac{0.12 + \alpha_\ell}{0.12 - \delta}$$

The derivative of this expression with respect to α_ℓ is $\frac{1}{0.12-\delta} - \frac{1}{\ln(2)(\alpha_\ell+\epsilon-3\delta)}$, which is initially decreasing when $\ln(2)(\alpha_\ell + \epsilon - 3\delta) < 0.12 - \delta$ and then increasing for larger α_ℓ. Therefore, the maxima are on the endpoints of the interval $\alpha_\ell \in (\epsilon - \delta, 0.33)$. We already considered the case $\alpha_\ell = 0.33$. When $\alpha_\ell = \epsilon - \delta$ then the number of levels is at most $1 - \log_2(3(\epsilon - 2\delta)) + \frac{0.12}{0.12-\delta}$.

In conclusion, the total number of levels before $\alpha_i \geq 0.80$ is at most:

$$\ell \leq max\big((0.68 - \epsilon + \delta)/(0.12 - \delta), 1 - \log_2\big(3(\epsilon - 2\delta)\big) + 1/(1 - \delta/12)\big)$$

In particular, when $\delta \leq 0.01$ this becomes $max(7, -\log_2(3(\epsilon - 2\delta)) + 3)$. □

Relaxing δ Claim 6 improved on Claim 5 to show unpebbled dependency paths to 80% of the subgraph in some layer. The final improvement is to redistribute the δ_i such that $\sum_i \delta_i = O(\delta)$ but security is still maintained. Intuitively, ensuring $\delta < \epsilon$ is necessary on level V_ℓ as otherwise $\gamma + \delta \geq 1$ and there are no unpebbled nodes on level V_ℓ (all the missing black pebbles can be covered with red pebbles). However, as the dependencies expand between levels a larger δ can be tolerated as well. Although the number of black pebbles the prover will place on each level isn't fixed a priori, we show that if $\delta < \epsilon/2$ in level V_ℓ then we can tolerate a factor 3/2 increase between levels as long as $\delta \leq 0.05$ in any layer.[6]

That is, if δ_i denotes the bound on the number of red pebbles in the ith layer then our new analysis requires $\delta_\ell < \epsilon/2$ and $\delta_i = min(0.05, (3/2)\delta_{i+1})$. This means that the total number of queries in the PoS over all levels is $O(1/\epsilon)$ because $\sum_{i=1}^{\ell} 1/\delta_i \leq max(0.10\ell, \frac{3}{2\delta_\ell})$.

Claim 7. *If $\mathcal{G}_{SDR}$ initially has at most γn black pebbles for $\gamma \leq 1 - \epsilon$, at most $\delta_\ell = \delta < \epsilon/3$ red pebbles in layer V_ℓ, and at most $\delta_i = min(0.05, (2/3)\delta_{i-1})$ red pebbles in layer V_i, then for $\ell = max(13, \log_2(\frac{1}{3(\epsilon-3\delta}))+4)$ the unpebbled nodes in V_ℓ have unpebbled paths to at least $0.80n$ unpebbled nodes in some layer V_i.*

Proof. Modifying Equation 3.1 to account for the different values of δ_i gives:

$$\alpha_{\ell-k} \geq 2^k\alpha_\ell - 2^{k-1}\gamma_\ell - (2^{k-1}\delta_{\ell-1} + 2^{k-2}\delta_{\ell-2} + \cdots + \delta_{\ell-k}) \geq 2^k\alpha_\ell - 2^{k-1}\gamma_\ell - \sum_{i=1}^{k} 2^{k-i}(3/2)^{i-1}\delta_\ell$$

Let $\sigma_k = \sum_{i=1}^{k} 2^{k-i}(3/2)^{i-1}$. Then $(4/3)\sigma_k = \sigma_k + 2^{k+1}/3 - (3/2)^{k-1}$. Therefore $\sigma_k = 2^{k+1} - 3^k/2^{k-1} < 2^{k+1}$. Using $\gamma_\ell \leq \alpha_\ell + \delta_\ell - \epsilon$ and $\alpha_\ell \geq \epsilon - \delta_\ell$ we derive the new bound:

$$\alpha_{\ell-k} \geq 2^k\alpha_\ell - 2^{k-1}\gamma_\ell - 2^{k+1}\delta_\ell \geq 2^{k-1}(\alpha_\ell + \epsilon - 5\delta_\ell) \geq 2^k(\epsilon - 3\delta) \tag{3.3}$$

This shows that if $\ell \geq \log_2(\frac{1}{3(\epsilon-3\delta)})$ then there is some level V_i where $\alpha_i \geq 1/3$.

We must also modify Equation 3.2 using $\sum_{j=1}^{k} \rho_{i-j} \leq \gamma_i \leq \alpha_i - \epsilon + \delta_i$:

$$\alpha_{i-k} \geq (k-1)min_{j<k}\hat{\beta}(\alpha_{i-j}) - \sum_{j=0}^{k} \delta_{i-j} + \hat{\beta}(\alpha_i) + \epsilon \tag{3.4}$$

When $i = \ell$ and k is small $\delta_{\ell-k} = (3/2)^k\delta_\ell$ and $\delta_\ell \leq \epsilon/3$ implies:

$$\alpha_{\ell-k} \geq (k-1)min_{j<k}\hat{\beta}(\alpha_{i-j}) + \hat{\beta}(\alpha_\ell) + (\frac{2}{3} - \frac{3^k}{2^{k+1}})\epsilon$$

Otherwise, we can use $\delta_i \leq 0.05$.

$$\alpha_{i-k} \geq (k-1)(min_{j<k}\hat{\beta}(\alpha_{i-j}) - 0.05) + \hat{\beta}(\alpha_i) + \epsilon - 0.10$$

Now we turn back to the two cases for $\alpha_i \geq 1/3$.

[6]The 2/3 factor is somewhat arbitrary. In general we can choose any growth factor for δ that is smaller than the dependency expansion factor we use in our analysis, which for $d = 8$ is at least 2 in subsets smaller than $n/3$. The analysis could also be tightened to allow for a bound larger than 0.05. The maximum allowable δ can also be increased if we decrease the target fraction of unpebbled dependencies, e.g. from 80% to 70%. Our analysis here does not yet optimize this tradeoff.

Case $\alpha_\ell \geq 1/3$*:* We claim that $\alpha_{\ell-k} \geq 0.11$ for all k. This is true for α_ℓ by hypothesis. From the equation above and the bound $\hat{\beta}(\alpha) \geq 0.12$ for all $\alpha \in (0.10, 0.80)$ (Corollary 2), $\alpha_{\ell-1} \geq \hat{\beta}(\alpha_\ell) - \epsilon/12 > 0.11$. Therefore, $\alpha_{\ell-2} \geq (0.12 - 0.05) + 0.12 - (11/24)\epsilon \geq 0.18$. Now assume that $\alpha_{\ell-2-j} \geq 0.12$ for all $j < k$, then $\alpha_{\ell-2-j} \geq (k-1)0.07 + 0.12 - (11/24)\epsilon > 0.11$. The claim follows by induction. This also shows that $\alpha_{\ell-k} \geq (k-3)0.07 + 0.11 > 0.80$ when $k = 13$.

Case $\alpha_\ell < 1/3$*:* From Equation 3.3, $\alpha_i \geq 1/3$ at some index $i \geq \ell - k$ for $k = \log_2(\frac{2}{3(\alpha_\ell + \epsilon - 5\delta_\ell)})$. At this point $\alpha_i \geq 1/3$ and $\gamma_i < \gamma_\ell \leq \alpha_\ell - \epsilon + \delta_\ell$. Combining this with Equation 3.4 gives:

$$\alpha_{i-k'} \geq (k'-1)(min_{j<k'}\hat{\beta}(\alpha_{i-j}) - 0.05) + \beta(\alpha_i) - \alpha_\ell + \epsilon - 0.05 - \delta_\ell$$

We claim that $\alpha_{i-k'} \geq 0.30$ for all k'. Observe that $\alpha i - 1 \geq \beta(\alpha_i) - \alpha_\ell + \epsilon - 0.05 - \delta_\ell \geq 0.68 - 0.38 + \epsilon - \delta_e ll \geq 0.30$ for any value $\alpha_\ell < 0.33$ because $\beta(\alpha_i) \geq \beta(0.33) \geq 0.68$. Assuming this is true for all α_{i-j} where $1 < j \leq k'$ implies $\alpha_{i-k'} \geq (k'-1)0.07 + 0.30 \geq 30$. Therefore, we can state more generally that $\alpha_{i-k'} \geq (k'-1)0.07 + 0.68 - \alpha_\ell$ and $\alpha_{i-k'-1} \geq 0.80$ when $k' = (0.12 + \alpha_\ell)/0.07$. The total number of levels where $\alpha_i < 0.80$ is thus at most:

$$k + k' + 1 \leq 1 - \log_2((3/2)(\alpha_\ell + \epsilon - 5\delta_\ell)) + 2 + \alpha_\ell/0.07$$

Differentiating this expression with respect to α_ℓ shows that the maxima over $\alpha_\ell \in (\epsilon - \delta_\ell, 0.33)$ are on the endpoints. The endpoint $\alpha_\ell = 0.33$ coincides with the case above. At the endpoint $\epsilon - \delta_\ell$ the number of levels is bounded by $3 - \log_2(3(\alpha_\ell + \epsilon - 5\delta)) + \epsilon/0.07$.

In conclusion, considering both cases, the total number of levels before $\alpha_i \geq 0.80$ is at most:

$$\ell \geq max(13, 3 - \log_2(3(\epsilon - 3\delta_\ell)) + \epsilon/0.07)$$

In particular, when $\epsilon \leq 0.07$ and $\delta_\ell = \delta$ then $\ell \leq max(12, 4 - \log_2(3(\epsilon - 3\delta))$.

□

4 "ZigZag" DRG Proof of Replication

The Stacked-DRG PoS can be adapted into a PoRep. The replication algorithm on a data input D with commitment τ_D first derives the labels on level $V_{\ell-1}$ and then uses these as "keys" to encode D on the last level V_ℓ. Each ith label on the final layer encodes d_i and uses a key $k_i = H(\tau_D || c_{\mathsf{parents}}(i))$ where $c_{\mathsf{parents}}(i)$ are the labels on all the nodes with directed edges to the ith node in V_ℓ, including within layer V_ℓ and layer $V_{\ell-1}$. The label e_i then encodes d_i as $\mathsf{Enc}(k_i \oplus d_i)$ (where e.g. Enc is a verifiable delay encoding). The replica R consists of $e_1, ..., e_n$.

However, with this construction, decoding D from R (i.e. data extraction) is nearly as expensive as data replication because it requires re-deriving the keys on level $V_{\ell-1}$ in the same way. Once the labels on level $V_{\ell-1}$ have been derived, then the data can be decoded as $d_i = \mathsf{Dec}(e_i \oplus k_i)$. Once all the labels required for each key derivation are stored in memory (requiring double the memory if done naively) then the last step of the decoding can be completely parallelized.

The ZigZag PoRep construction fixes this with a simple tweak.

4.1 ZigZag PoRep Construction

The basic idea of ZigZag is to layer DRGs so that each layer "encodes" the previous layer. The critical desired property to achieve is: if *all* the labels on a given level are available in memory then the labels can decoded in parallel. To achieve this, instead of adding edge dependencies between the layers, we add the edges of a constant degree expander graph in each layer so that every layer is both depth-robust and has high "expansion". Technically, the graph we construct in each layer is an expander as an undirected graph. As a DAG this means that the union of the dependencies and targets of any subset is large. By alternating the direction of the edges between layers, forming a "zig-zag", we are able to show that the dependencies between layers expand. Now the only edges between layers are between nodes at the same index, and the label on each node encodes the label on the node at the same index in the previous level. The dependencies used for keys are all contained in the same layer. Thus, the labels in any layer are sufficient to recover the labels in the preceding layer. Moreover, the decoding step can be done in parallel.

Without alternating the direction of the edges between layers this construction would fail to be a tight proof of space because the topologically last ϵn nodes in a layer would only depend on the topologically last ϵn nodes in the previous layer. Moreover, if the prover stores the labels on the topologically first $(1-\epsilon)n$ nodes it can quickly recover the labels on the topologically first $(1-\epsilon)n$ nodes in the preceding level, allowing it to recover the missing ϵn labels as well in parallel-time $O(\epsilon n)$.

DAG encodings As a building block the ZigZag PoRep uses the follow DAG encodings scheme, which was used to construct PoReps based on depth robust graphs in [19, 20, 33]. The DAG encoding scheme takes in a data file X on n blocks $x_1, ..., x_n$ and a d-inregular DAG on n nodes together with its parent function $\mathsf{Parents}(i)$ which outputs the parent nodes of the ith node. It also uses a randomized encoding scheme $\mathsf{Enc}, \mathsf{Dec}$. This may be as simple as the identity function, or could use VDE encoding for added delay. The benefit of adding a delay is to increase the time required to regenerate replicas even when n is relatively small. The delay can be tuned appropriately for larger n. Finally, it takes in a seed σ which it uses as a salt for a collision-resistant hash function $H : \{0,1\}^{md} \to \{0,1\}^m$.

$\mathsf{DAGEnc}(\vec{x}, m, n, \sigma)\{$
  for $i = 1$ *to* n :
    $(v_1, ..., v_d) \leftarrow \mathsf{Parents}(i)$
    $k_i \leftarrow H(\sigma || c_{v_1} || \cdots || c_{v_d})$
    $c_i \leftarrow \mathsf{Enc}(k_i \oplus x_i)$
  $e \leftarrow (c_1, ..., c_n)$
  return $e\}$

Construction of $\mathcal{G}_{ZZ}[\ell]$ Similar to $\mathcal{G}_{SDE}$, the graph $\mathcal{G}_{ZZ}[\ell]$ contains a copy of an $(n, 0.80n, \beta n)$-depth-robust graph for some constant β in each of the ℓ layers $V_1, ..., V_\ell$. The nodes in each layer are indexed in $[n]$. Every odd layer overlays the edges of the DRG in the forward direction (edges go from lower to higher indices) while every even layer the edges of the DRG in the reverse direction (edges go from higher indices to lower indices). Note that the even layer graphs

are *not* necessarily the reverse[7] of the odd layer graphs. They are constructed by reversing the numbering of the nodes.

Edges are added between same index nodes in adjacent layers (i.e. the ith node in layer V_k is connected to the ith node in layer V_{k+1} for all i, k). Next, the edges that were between layers in $\mathcal{G}_{SDE}[\ell]$ are projected into each layer of $G_{ZZ}[\ell]$ with the direction of each edge determined by the parity of the layer. We call these *expander edges* to distinguish[8] them from other edges. More precisely, if $G_{SDE}[\ell]$ has an edge from the ith node of a layer V_k to the jth node of layer V_{k+1} then $G_{ZZ}[\ell]$ has an edge between the ith node of V_{k+1} and the jth node of V_{k+1}. The direction of the edges added to V_{k+1} is from lower indices to higher indices when $k+1$ is odd, and from higher indices to lower indices when $k+1$ is even. (For concreteness in the analysis, the edges between layers in the reference graph $\mathcal{G}_{SDE}$ are assumed to be constructed using the degree 8 Chung random bipartite graph construction).

PoRep Replicate The ZigZag PoRep construction uses $\mathcal{G}_{ZZ}[\ell]$ to uniquely encode an input data file D. The PoRep.Replicate algorithm derives the replica encoding of D with data tag τ_D and unique identifier id as a labeling of the final layer V_ℓ. PoRep construction iterates over all the layers from $i = 1$ to $i = \ell$. The labels on the ith layer re-encode the labels on the $i-1$st layer using the basic DAGEncode scheme on the subgraph (V_i, E_i) (where E_i are the edges of $\mathcal{G}_{ZZ}[\ell]$ within the layer V_i). Each ith label $e_i^{(j)}$ in layer V_j encodes the label $e_i^{(j-1)}$ using key $k_{i,j} = H(\tau_D || c_{\mathsf{parents}}(i,j))$ where $c_{\mathsf{parents}}(i,j)$ are the labels on all the nodes with directed edges to the ith node in V_j. The encoding $e_i^{(j)}$ is then $\mathsf{Enc}(k_{i,j} \oplus e_i^{(j-1)})$ (where e.g. Enc is a verifiable delay encoding). Note that the encoding of the next layer can be computed just using a buffer of size n blocks because the new encodings of the blocks are derived in reverse topological order (i.e. the topological order of the next layer) and once a label is replaced with its fresh encoding it is no longer needed. The final replica R consists of the encodings $e_1^{(\ell)}, ..., e_n^{(\ell)}$ on the final layer.

PoRep "offline" proof The compact output aux of PoRep.Replicate is nearly the same as the PoS offline proof in the Initialization of the stacked DRG PoS. First a vector commitment Φ to all the labels in all layers is derived. Chal samples λ_i uniform random challenge labels in each layer V_i and provides their parent labels and inclusion proofs of all these labels in a vector commitment Φ. However, the one difference is that aux also includes for each challenged label in V_i the inclusion proof of the label it encodes in V_{i-1}. The verification of this proof obtains the decoded label in V_{i-1} by first deriving the keys from the parent labels provided and then using Dec to decode each challenge label. For the challenges on layer V_1 the inclusion proofs of the encoded data inputs are with respect to τ_D.

PoReP Prove and Verify Likewise, the basis for the online proofs output by PoRep.Proof in the challenge-response protocol is also the Execution challenge-response protocol of the stacked DRG PoS. There are two variants. The first is exactly the same, containing the labels of κ

[7]The reverse of a graph is a graph on the same nodes with the direction of all edges reversed. While this would still work conceptually for our construction it might not be locally navigatable. If the edges are sampled pseudo-randomly using a random oracle then evaluating the inverse DRG.Parents function would be a linear operation. For efficiency, it would be necessary to separately store a table defining all the edges of the graph.

[8]The distinction between expander edges and all other edges is important in the analysis. In particular, the expander edges are between the same index nodes in every layer and differ only in their directionality.

randomly sampled nodes on the final layer V_ℓ and their inclusion proofs in the vector commitment Φ in aux. The second variant includes in response to a challenge on the ith index of V_ℓ the ith keys in each layer for a total of $\kappa\ell$ keys. This enables the $\mathsf{PoRep.Verify}$ to check that each challenged label correctly encodes a committed data input. It also requires the prover to run $\mathsf{PoRep.Extract}$ to generate each proof. The second variant is more expensive (larger proves and longer proving time), but achieves tighter security for the same parameters.

PoRep Extraction $\mathsf{PoRep.Extract}$ does the reverse operation of $\mathsf{PoRep.Replicate}$ on the replica R, consisting of the n block labels in V_ℓ. Once the labels on layer V_j have been computed and stored in memory, then each key $k_{i,j} = H(\tau_D||c_{\mathsf{parents}}(i,j))$ can be computed and then each label $e_i^{(j)}$ is replaced with $e_i^{(j-1)} = \mathsf{Dec}(k_{i,j} \oplus e_i^{(j)})$. The keys $k_{i,j}$ can be computed in parallel. If they are stored in a buffer in memory then all the labels can be decoded in parallel as well. There is a smooth tradeoff between the additional space required and the parallel time speedup. With an additional buffer of k blocks the labels can decoded in reverse topological order in parallelized groups of $k \leq n$ blocks at a time achieving a factor k parallel speedup. Moreover, the extraction is already faster than replication due to asymmetry in the runtime of Dec vs Enc.

PoRep Security A PoRep construction is secure if it satisfies *ϵ-rational replication* (Section 2.3). It was shown (Lemma 2, [19]) that this security definition is satisfied[9] if the PoRep is both a PoS and a PoRC. Roughly, the PoRep achieves ϵ-rational replication (with soundness error μ) if it is a PoS with a $1 - \epsilon + \delta$ space gap and it is a PoRC that a $1 - \delta$ fraction of the committed data and corresponding replica blocks are retrievable (both with soundness error μ).

While both variants can be tuned to a μ-sound $(1 - \delta)$-PoRC (Section **??**) for any μ and δ, i.e. a $1 - \delta$ fraction of the committed data inputs are retrievable from any prover passing with probability μ, the second variant is tighter (i.e. achieves smaller δ keeping other parameters equal). In the first variant there is a factor ℓ union bound loss because although the proof in aux ensures there are a bounded number of errors (i.e. incorrectly encoded labels) in each layer, if just one label on the ith index in one of the layers is incorrect the ith block of the replica may no longer encode d_i. Here we do not care about the correctness of the keys only how many blocks of D can be successfully decoded from R.

Our analysis will focus on proving that the ZigZag PoRep, similar to the stacked DRG PoS, is an arbitrarily tight PoS with only $\ell = O(\log(1/\epsilon)$ layers.

4.2 Invertible pebbling games

The red-black pebbling game no longer entirely captures the PoS security of the ZigZag PoRep due to the involvement of the encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ in the labeling rather than purely a collision-resistant hash function. Most significantly, the labels are now invertible. In terms of the dependency graph of the labeling computation, the keys in each layer V_i still need to be computed in topological order, however the labels may either be derived by decoding labels in layer V_{i+1} or encoding labels in layer V_{i-1}. We modify the black pebbling game to capture invertibility of labels by coloring edges.

[9]Technically this security reduction requires a "knowledge of compression" assumption in order to remain true under composition with other storage protocols.

White & green colored edges White edges are "one-way streets" corresponding to edge dependencies involved in deriving keys via calls to the random oracle and are treated like normal pebbling game edges. Green edges are "two-way street", but still have a direction and different rules in either direction. If there is a directed green edge from u to v then a pebble can be placed on v if and only if u and all nodes with white edges to v have pebbles. A pebble can be placed on u if and only if v and all nodes *with white edges to* v have pebbles.

PoS soundness We still analyze the soundness of a PoS with invertible labels through the game Red-Black-Pebbles as in Definition 1, however with the modification that the adversary plays the black pebbling game with white/green edges as described above instead of the plain black pebbling game. Unfortunately, it is no longer clear that soundness under this security definition is equivalent to soundness of the underlying labeling game in the random oracle model. Pietrzak [33] proved this equivalence only for parallel soundness of the standard red-black pebbling game and the random-oracle labeling game. If Enc/Dec are modeled as random permutations then it seems reasonable that a similar equivalence could be proven in the ideal cipher model, where a prover who uses too little space could either compress a random oracle's function table (used to derive keys) or a random permutation's function table. However, exploring the details of this analysis further is beyond the scope of this paper.

Labeling games with VDEs If Enc/Dec are instantiated with a VDE scheme in order to slow down the adversary on smaller graphs then this also subtly changes the pebbling analysis. Queries to the VDE are significantly more expensive than queries to H so they cannot be treated equally as pebbling moves. However, as remarked in [19], without loss of generality one can assume a parallel adversary always queries the VDE to derive a label whenever both the data/label input and key are available. Therefore, we can model the VDE and hash function queries as a single pebbling move that can be made only when there are labels on all the dependencies, including the input label and all the labels required for the key derivation.

4.3 PoS analysis of ZigZag PoRep

Our analysis follows the same general outline as the PoS analysis for the stacked DRG PoS on $\mathcal{G}_{SDR}[\ell]$. Here we analyze the hardness of the modified game $\mathsf{Red\text{-}Black\text{-}Pebbles}^{\mathcal{A}}(\mathcal{G}_{ZZ}[\ell], V_\ell, \mathsf{Chal})$ using green/white edges, or equivalently the soundness of the PoS in this model. The directed edges within every layer V_i are white, whereas the directed edges between the same index nodes in adjacent layers are green. (There are no other directed edges between layers).

As before, the bulk of the analysis involves showing that pebbling the entire last level V_ℓ (i.e. all unpebbled nodes) from an initial configuration of only $(1-\epsilon)n$ black pebbles overall and a bounded number $\delta_i n$ of red pebbles in each level requires pebbling (in topological order) an entire 80% unpebbled subgraph in some layer. Pebbling 80% of this subgraph in topological order requires βn parallel rounds due to the depth robustness of each layer. Using the same notation as before, this means that $\mathcal{G}_{ZZ}[\ell]$ is $(\gamma n, \vec{\delta}, \beta n - 1, 1)$-hard. The proof that this translates to PoS soundness needs to be updated as well because Claim 4 did a case-by-case analysis using the expansion properties of $\mathcal{G}_{SDR}[\ell]$. Claim 2 and Claim B are not affected by the new white/green edge rules and the structural differences between $\mathcal{G}_{SDR}[\ell]$ and $\mathcal{G}_{ZZ}[\ell]$ (the only relevant structural property used in these claims is that these graphs are layered).

Index sets and nodes We associate nodes in any layer with indices in $[n]$ and say that an index "is unpebbled in V_i" if the node at that index in V_i is initially unpebbled (including black and red pebbles). We use the same notations in Notation 1 (Section 3) to denote the U_i unpebbled index sets and P_i pebbled index sets in each layer, with ρ_i, δ_i, and γ_i defined the same.

Forward dependencies Bounding the parallel complexity of pebbling a set of nodes in $\mathcal{G}_{ZZ}[\ell]$ is no longer as simple as tracing unpebbled directed paths in the graph due to the fact that pebbling moves can occur in either the "forward" or "reverse" directions. The direction in which a dependency is pebbled is important to pebbling complexity. For instance, suppose all nodes in V_2 except the nth node (call it v_n) were pebbled and no pebbles were placed in V_1. The parallel complexity of completing V_2 is just 3 rounds in this case: all the dependencies of the nth node u_n could be pebbled in the first round, u_n could be pebbled in the second round, and finally v_n in the third. On the other hand, if there were pebbles on the first $n-1$ nodes of V_1 then the complexity would be n rounds because all the nodes in V_2 must be pebbled in the forward direction before v_n. Our analysis will focus on identifying dependencies that must be pebbled in the forward direction, which we call *forward dependencies*. The following claim gives a helpful strategy for tracing forward dependencies in $\mathcal{G}_{ZZ}[\ell]$.

Claim 8. *Let X be a set of unpebbled nodes in V_ℓ of $\mathcal{G}_{ZZ}[\ell]$. Form the set Y as follows. Initialize $Y := X$. Iterate over the layers starting from $V_{\ell-1}$. In each layer V_i add a node to Y if it is unpebbled and has a green edge to a node in $Y \cap V_{i+1}$ (i.e the same index node in V_{i+1} is in Y). Additionally, add to Y any unpebbled node in V_i that has a two-hop directed path to $Y \cap V_{i+1}$ via an expander white edge followed by a green edge. All nodes in Y are unpebbled forward dependencies of X.*

Proof. The proof is by finite strong induction on the nodes added in levels ℓ to i. In the base case Y just consists of X in V_ℓ and the statement is vacuously true (all nodes in X must be pebbled in the forward direction). Now assume that all nodes added to Y in layers $V_\ell, ..., V_{i+1}$ are forward dependencies of X. If an unpebbled node $u \in V_i$ has a green edge to a node v in Y then u must be pebbled before v because by hypothesis v must be pebbled in the forward direction. Clearly, u must also be pebbled in the forward direction (otherwise v must be pebbled before u). This makes u a forward dependency of X. Next consider any unpebbled $u' \in V_i$ with a white directed expander edge to u. This is a dependency of u and therefore a dependency of X. The node u' also has a green edge the same index node v' in V_{i+1}. Since the expander edges are the same in each layer with alternating directions there is a directed white edge from v to v' and so v must have a pebble before u' can be pebbled in the reverse direction via v'. This cannot happen because u' is a dependency of v. Hence, u' must be pebbled in the forward direction. We have shown that both u' and u are *forward dependencies* of X. □

The following lemma provides lower bounds on the dependency expansion of subsets in any given layer and is the main fact needed for the remaining analysis.

Lemma 6. *For any $\alpha \in (0,1)$ and $i \in [\ell]$ and a set X of αn unpebbled nodes in layer V_i of $\mathcal{G}_{ZZ}[\ell]$, if $\alpha - \rho_{i-1} - \rho_{i-2} \leq 1/3$ then X has at least $(2(\alpha - \rho_{i-1} - \rho_{i-2}) - \delta_{i-1} - \delta_{i-2})n$ forward dependencies in layer V_{i-2}. Otherwise, X has at least $(0.12 + \alpha - \rho_{i-1} - \rho_{i-2} - \delta_{i-1} - \delta_{i-2})n$ forward dependencies in V_{i-2}.*

Proof. Fixing any given subset S of V_i of size αn, we define the following index sets:

- $X = S \cap U_i$ is the index set of the αn unpebbled nodes contained in this subset of V_i.
- $X' = X \cap U_{i-1}$ is the index subset of X that is unpebbled in V_{i-1}
- $X'' = X' \cap U_{i-2}$ is the index subset of X' that is unpebbled in V_{i-2}.
- $\Gamma_{\text{out}}(X')$ are the indices of nodes on the boundary of X' in V_i with edges from X'. $\Gamma_{\text{out}}(X')$ is disjoint from X' as it does not include nodes in X' with internal edges to X'. $\Gamma_{\text{out}}(X'')$ is defined the same way for the set X''.
- $\Gamma_{\text{in}}(X')$ are the indices of nodes on the boundary of X'' in V_i with edges to X'. $\Gamma_{\text{in}}(X'')$ is defined the same.

We partition the $(\rho_{i-1} + \delta)n$ pebbles on V_{i-1} as follows:

$$p^T_{i-1}n = |P_{i-1} \cap \Gamma_{\text{in}}(X')| \quad p^D_{i-1}n = |P_{i-1} \cap \Gamma_{\text{out}}(X')| \quad p_{i-1}n \text{ (the remaining pebbles)}$$

Likewise, we partion the $(\rho_{i-2} + \delta)n$ pebbles on V_{i-2} into:

$$p^D_{i-2}n = |P_{i-2} \cap \Gamma_{\text{in}}(X'')| \quad p^T_{i-2} = |P_{i-2} \cap \Gamma_{\text{out}}(X'')| \quad p_{i-2}n \text{ (the remaining pebbles)}$$

Define the index set $Z = X'' \cup (\Gamma_{\text{out}}(X') \cap U_{i-1} \cap U_{i-2}) \cup (\Gamma_{\text{out}}(X'') \cap U_{i-2})$. X'' are unpebbled nodes connected to X via green edges. $\Gamma_{\text{out}}(X') \cap U_{i-1}$ are unpebbled direct dependencies of X' in V_{i-2} due to reversal of edges, and $\Gamma_{\text{out}}(X') \cap U_{i-1} \cap U_{i-1}$ are connected to these via green edges. $\Gamma_{\text{in}}(X')$ are targets of X' in V_{i-2} and $\Gamma_{\text{in}}(X') \cap U_{i-2}$ are unpebbled direct dependencies of X''. Thus, by Claim 8, all nodes at the indices Z in V_{i-2} are all unpebbled forward dependencies of X.

Finally, recall from Claim that $\Gamma_{\text{in}}(X'') \cup \Gamma_{\text{out}}(X'') \cup X'' = \Gamma(X'') \cup X''$ inherits the expansion of the bipartite expander used in the construction. Since $|X''| = \alpha'' \geq \alpha - \rho_{i-1} - \rho_{i-2}$, in the case that $\alpha - \rho_{i-1} - \rho_{i-2} \leq 1/3$ then $|\Gamma_{\text{out}}(X'') \cup X''| \geq 2|\alpha - \rho_{i-1} - \rho_{i-2}|$ with overwhelming probability, assuming the expander construction uses Chung's degree 8 bipartite expanders (by Corollary 1, Section 2.7). Thus:

$$\begin{aligned}
|Z| &\geq |X''| + (|\Gamma_{\text{out}}(X')| - p^D_{i-1}n - p^T_{i-2}n) + (|\Gamma_{\text{in}}(X'')| - p^D_{i-2}n \\
&\geq |X''| + |\Gamma_{\text{out}}(X'')| + |\Gamma_{\text{in}}(X'')| - p^D_{i-1}n - p^T_{i-2}n - p^D_{i-2}n \\
&\geq (2(\alpha - p_{i-1} - p_{i-2}) - p^D_{i-1} - p^T_{i-2} - p^D_{i-2})n \\
&\geq (2(\alpha - \rho_{i-1} - \rho_{i-2}) - \delta_{i-1} - \delta_{i-2})n
\end{aligned}$$

In the case that $\alpha - \rho_{i-1} - \rho_{i-2} \geq 1/3$ then $|X''| \in (0.33, 0.80)$ and $\Gamma(X'') \geq 0.12$ (by Claim 2, Section 2.7).

$$\begin{aligned}
|Z| &\geq |X''| + |\Gamma(X'')| - (p^D_{i-1} + p^T_{i-2} + p^D_{i-2})n \\
&\geq (0.12 + \alpha - p_{i-1} - p^D_{i-1} - p_{i-2} - p^T_{i-2} - p^D_{i-2})n \\
&\geq (0.12 + \alpha - \rho_{i-1} - \rho_{i-2} - \delta_{i-1} - \delta_{i-2})n
\end{aligned}$$

□

Theorem 4. *The labeling PoS on $\mathcal{G}_{ZZ}[\ell]$ with* Chal *sampling λ/δ challenges in each level V_i and κ online challenges in V_ℓ is $((1-\epsilon-2\delta)n, \beta n-1, e^{-\lambda})$-sound with $\kappa = 2\lambda/\epsilon$, $\delta < min(0.01, \epsilon/3)$, $\epsilon + 2\delta \leq 0.24$, and $\ell = 2\log_2(\frac{1}{3(\epsilon-3\delta)}) + 18$.*

Proof. For any $\mathcal{G}_{SDR}[\ell]$, if the set of unpebbled nodes in V_ℓ are connected via unpebbled paths to at least $0.80n$ unpebbled forward dependencies in some prior level V_i, then pebbling all of V_ℓ requires pebbling all these $0.80n$ unpebbled nodes in topological order, which requires $\beta n - 1$ rounds due to the fact that V_i is $(n, 0.80n, \beta n)$ depth robust. Claim 11 thus implies that $\mathcal{G}_{ZZ}[\ell]$ is $(1-\epsilon, \vec{\delta}, \beta n-1, 1)$-hard where $\delta_i = \delta < \epsilon/3$ for all i and $\ell = 2\log_2(\frac{1}{3(\epsilon-3\delta)}) + 16$ when $\delta < 0.01$.

Assuming $\epsilon + 2\delta \leq 0.24$, Claim 9 implies that $\mathcal{G}_{ZZ}[\ell+2]$ is $(1-\epsilon-2\delta, \vec{\delta}, \beta n-1, 1-\epsilon/2)$-hard extending $\vec{\delta}$ so that $\delta_{\ell+1} = \delta_\ell = \delta$. Finally, by Claim 2 the labeling PoS on $\mathcal{G}_{ZZ}[\ell+1]$ with challenge set V_ℓ and Chal sampling λ in each level V_i is $((1-\epsilon-2\delta)n, \beta n-1, max\{(1-\delta)^\lambda, (1-\epsilon/2)^\kappa\})$-sound. When $\lambda_i = \lambda/\delta$ and $\kappa = 2\lambda/\epsilon$, the PoS is $((1-\epsilon)n, \beta n-1, e^{-\lambda})$-sound.

□

The next claim proves the same result as Claim 4 but for the graph $G_{ZZ}[\ell]$.

Claim 9. *For any $\epsilon + 2\delta \leq 0.24$, if $\mathcal{G}_{ZZ}[\ell-2]$ is $(1-\epsilon+2\delta, \vec{\delta}, t-2, 1)$-hard then $\mathcal{G}_{ZZ}[\ell]$ is $(1-\epsilon, \vec{\delta}^*, min(\beta n-1, t), 1-\epsilon/2)$-hard where $\vec{\delta}^*$ is identical to $\vec{v}$ on all common indices and $\delta = \delta_\ell = \delta_{\ell-1} = \delta_{\ell-2} \leq \epsilon/2$.*

Proof. Let $\gamma = 1-\epsilon$ denote the black pebbled initially placed on $\mathcal{G}_{SDR}[\ell]$. As noted in Claim 4, every subset of V_ℓ of size $(1-\epsilon/2)n$ contains at least $(\alpha_\ell - \epsilon/2)n$ unpebbled nodes. Without loss of generality we assume $\alpha_\ell - \epsilon/2 \leq 0.80$ (otherwise these nodes already require βn rounds to pebble).

Fixing any given subset S of V_ℓ of size $(1-\epsilon/2)n$, let $X = S \cap U_\ell$, the unpebbled set among the nodes at indices S in V_ℓ. We will show that X requires more than $min(\beta t, t)$ rounds to pebble given the hypothesis that $\mathcal{G}_Z[\ell-2]$ is $(1-\epsilon+2\delta, \vec{\delta}, t-2, 1)$-hard. Since this holds for any choice of S this shows that $\mathcal{G}_{ZZ}[\ell]$ is $(1-\epsilon, \vec{\delta}^*, min(\beta n-1, t), 1-\epsilon/2)$-hard.

As noted, $|X| \geq (\alpha_\ell - \epsilon/2)n$. Let $\alpha^* = \alpha_\ell - \epsilon/2 - \rho_{\ell-1} - \rho_{\ell-2}$. The proof is broken into two cases:

Case $\alpha^* \leq 1/3$: Let Z denote the unpebbled forward dependencies of X inside $V_{\ell-2}$. From Lemma 6 we know that $|Z| \geq (2\alpha^* - \delta)n$. Thus:

$$|Z| \geq 2(\alpha_\ell - \rho_{\ell-1} - \rho_{\ell-2} - \delta)n - \epsilon n = (2(1 - \rho_\ell - \rho_{\ell-1} - \rho_{\ell-2} - 2\delta) - \epsilon)n$$

Let $|Z|/n = \alpha'$. Using the identity $\gamma_{\ell-2} = \gamma - \rho_\ell - \rho_{\ell-1} - \rho_{\ell-2}$ gives:

$$\alpha' - \gamma_{\ell-2} \geq 2(1-2\delta) - \rho_\ell - \rho_{\ell-1} - \rho_{\ell-2} - \gamma - \epsilon \geq 2(1-\gamma-2\delta) - \epsilon = \epsilon - 4\delta$$

Setting $\gamma' = 1-\epsilon+2\delta$, the above relation shows that $\alpha' - \gamma_{\ell-2} \geq 1-\gamma'-2\delta$. Thus, by Claim B, if $\mathcal{G}_{ZZ}[\ell-2]$ is $(\gamma', \vec{\delta}, t-2, 1)$-hard then no adversary can pebble Z in $t-2$ or fewer rounds. Since these are forward dependencies of X at least t rounds are required to pebbled X.

Case $\alpha^* > 1/3$: In this case Lemma 6 showed that $|Z| \geq (0.12 - 2\delta + \alpha - \rho_{\ell-1} - \rho_{\ell-2})n$. Noting that $\alpha \geq \alpha_\ell - \epsilon/2$ and $\gamma_{\ell-2} = \gamma - \rho_\ell - \rho_{\ell-1} - \rho_{\ell-2}$ and $1 - \rho_\ell - \delta = \alpha_\ell$ we derive:

$$\alpha' - \gamma_{\ell-2} \geq 0.12 - 2\delta + \alpha_\ell - \epsilon/2 + \rho_\ell - \gamma = 0.12 - 2\delta + 1 - \gamma - \delta - \epsilon/2 = 0.12 - 3\delta + \epsilon/2$$

Given that $\mathcal{G}_{ZZ}[\ell-2]$ is $(\gamma', \vec{\delta}, t-2, 1)$-hard with $\gamma' = 1-\epsilon+2\delta$ no adversary can pebble Z in $t-2$ rounds as long as $0.12 + \epsilon/2 - 3\delta \geq \epsilon - 2\delta$, which is true for $\epsilon + 2\delta \leq 0.24$.

□

The remaining claims complete the analysis by showing that $\mathcal{G}_{ZZ}[\ell]$ is $(\gamma n, \vec{\delta}, \beta n - 1, 1)$-hard. These claims follow Claim 5, Claim 6, and Claim 6 very closely, substituting the new dependency expansion relations proved in Lemma 6. We assume the reader is familiar with the proofs of these prior claims and we will not repeat all the details.

Claim 10. *If $\mathcal{G}_{ZZ}[\ell]$ for $\ell = \log_2(\frac{1}{3(\epsilon-3\delta)})$ initially has at most γn black pebbles for $\gamma \leq 1-\epsilon$ and at most $\delta < \epsilon/3$ red pebbles in each layer then the unpebbled nodes U_ℓ in V_ℓ have least $n/3$ unpebbled forward dependencies in some layer V_i. Moreover, $\alpha_i - \rho_{i-1} - \rho_{i-2} \geq 1/3$.*

Proof. Let α_i denote the fraction of nodes in V_i that are unpebbled forward dependencies of U_ℓ with $\alpha_\ell n = |U_\ell|$. Suppose $\alpha_{\ell-i} - \rho_{\ell-i-1} - \rho_{\ell-i-2} \leq 1/3$ for $i = 0$ to $2k$. By Lemma 6, $\alpha_{j-2} \geq 2(\alpha_j - \rho_{j-1} - \rho_{j-2} - \delta)$. We prove by induction that $\alpha_{\ell-2k} \geq 2^k\alpha_\ell - 2^k(\rho_{\ell-1}+\rho_{\ell-2}) - (2^{k+1}-2)\delta$. The base case $k=0$ is trivial. Assuming this holds for k, then:

$$\begin{aligned}
\alpha_{\ell-2(k+1)} &\geq 2(\alpha_{\ell-k} - \rho_{\ell-2k-1} - \rho_{\ell-2k-2} - \delta) \\
&\geq 2(2^k\alpha_\ell - 2^k(\rho_{\ell-1}+\rho_{\ell-2}) - \rho_{\ell-2k-1} - \rho_{\ell-2k-2} - (2^{k+1}-2)\delta - \delta \\
&\geq 2(2^k\alpha_\ell - 2^k(\rho_{\ell-1}+\rho_{\ell-2}) - (2^{k+1}-1)\delta) \geq 2^{k+1}\alpha_\ell - 2^{k+1}(\rho_{\ell-1}+\rho_{\ell-2}) - (2^{k+2}-2)\delta
\end{aligned}$$

We used the fact that when $k > 0$ the expression $2^k(\rho_{\ell-1}+\rho_{\ell-2}) - \rho_{\ell-2k-1} - \rho_{\ell-2k-2}$ is maximized when $\rho_{\ell-1} + \rho_{\ell-2} = \gamma_\ell$ and $\rho_{\ell-i} = 0$ for all $i > 2$.

This shows that $\alpha_{\ell-2k} \geq 2^k\alpha_\ell - 2^k\gamma_\ell - 2^{k+1}\delta$. Substituting $\gamma_\ell = \gamma + \alpha_\ell - 1 + \delta \leq \alpha_\ell + \delta - \epsilon$ gives:

$$\alpha_{\ell-2k} \geq 2^k\alpha_\ell - 2^k(\alpha_\ell + \delta - \epsilon) - 2^{k+1}\delta = 2^k(\epsilon - 3\delta)$$

Therefore, $\alpha_i - \rho_{i-1} - \rho_{i-2} \geq 1/3$ at some index $i > \ell - \log_2(\frac{1}{3(\epsilon-3\delta)})$. □

Claim 11. *If $\mathcal{G}_{ZZ}[\ell]$ with $\ell = 2\log_2(\frac{1}{3(\epsilon-2\delta)})) + 2\frac{0.80-\epsilon+\delta}{0.12-2\delta}$ initially has at most γn black pebbles for $\gamma \leq 1-\epsilon$ and at most $\delta < min(0.06, \epsilon/3)$ red pebbles in each layer then the unpebbled nodes in V_ℓ have at least $0.80n$ unpebbled forward dependencies in some layer V_i.*

Proof. From Claim 10 there exists an index i where V_i contains more than $n/3$ forward dependencies of U_ℓ. We break our analysis into two cases depending on the initial size of U_ℓ. First, there are two important subclaims:

Subclaim 1: $\alpha_i - \gamma_i \geq \epsilon - \delta$ for all i.

This is true for the base case $\alpha_\ell - \gamma_\ell \geq 1 - \rho_\ell - \delta - (\gamma - \rho_\ell) \geq \epsilon - \delta$. Now suppose that $\alpha_i - \gamma_i \geq \epsilon - \delta$ and $\epsilon > 4\delta$. By the analysis in Lemma 6 and the identity $\gamma_{i-2} = \gamma_i - \rho_{i-1} - \rho_{i-2}$ we get that $\alpha_{i-2} - \gamma_{i-2} \geq |\Gamma(X'')| - 2\delta + \alpha_i - \gamma_i \geq |\Gamma(X'')| + \epsilon - 3\delta$ where X'' was a set of size at least $\alpha_i - \gamma_i - 2\delta \geq \epsilon - 3\delta \geq \delta$. Thus, $\alpha_{i-2} - \gamma_{i-2} \geq \epsilon - \delta$ as long as $\Gamma(X'') \geq 2\delta$. As long as $\delta < 0.06$, then either $|X''| < 0.10$ and is at least 3-expanding (see Table 2.2, Lemma 2) so $|\Gamma(X'')| \geq 2|X''| \geq 2\delta$, or otherwise $|X''| \in (0.10, 0.80)$ and $\Gamma(|X''|) \geq 0.12$ (Corollary 2).

Subclaim 2: If $\alpha_i - \rho_{i-1} - \rho_{i-2} > 1/3$ *then* $\alpha_{i-k} - \rho_{i-k-1} - \rho_{i-k-2} \geq 0.12$ *for all* k.

This is true for the base case α_i by hypothesis. For the inductive step, suppose that $\alpha_{i-k} - \rho_{i-k-1} - \rho_{i-k-2} \in (0.12, 0.80)$. By the analysis in Lemma 6 and Corollary 2, $\alpha_{i-k-2} \geq 0.12 - 2\delta + \alpha_i - \gamma_i$. Putting this together with $\alpha_{i-k} - \gamma_{i-k} > \epsilon - \delta$ from Subclaim 1 above shows:

$$\begin{aligned}\alpha_{i-k-2} - \rho_{i-k-3} - \rho_{i-k-4} &\geq 0.12 - 2\delta + \alpha_{i-k} - \rho_{i-k-1} - \rho_{i-k-2} - \rho_{i-k-3} - \rho_{i-k-4} \\ &\geq 0.12 - 2\delta + \alpha_{i-k} - \gamma_{i-k} \geq 0.12 + \epsilon - 3\delta\end{aligned}$$

Thus, the inductive step holds true as long as $\epsilon > 3\delta$.

Now we proceed with the two cases. Let $\alpha^* = \alpha_\ell - \rho_{\ell-1} - \rho_{\ell-2}$.

Case $\alpha^* \geq 1/3$*:* It follows from Subclaim 1, Subclaim 2 and Lemma 6 that $\alpha_{\ell-2k} \geq k(0.12 - 2\delta) + \alpha_\ell - \gamma_\ell \geq k(0.12 - 2\delta) + \epsilon - \delta$. Therefore $\alpha_{\ell-2k} \geq 0.80$ for $k \geq \frac{0.80-\epsilon+\delta}{0.12-2\delta}$ (e.g. $k = 8$ for $\delta \leq 0.01$).

Case $\alpha^* < 1/3$*:* By Claim 10 there exists an index $i \geq \ell - 2\log_2(\frac{1}{3(\epsilon-3\delta)})$ where $\alpha_i - \rho_{i-1} - \rho_{i-2} \geq 1/3$. Then just as in the previous case, Subclaim 1 and Subclaim 2 imply that $\alpha_{i-2k} \geq 0.80$ for $k \geq 0.80/(0.12 - 2\delta)$. Therefore, setting:

$$\ell = 2\log_2(\frac{1}{3(\epsilon - 3\delta)}) + 2\frac{0.80 - \epsilon + \delta}{0.12 - 2\delta}$$

there is some index i where $\alpha_i \geq 0.80$. In particular, for $\delta \leq 0.01$, the number of required levels is at most $2\log_2(\frac{1}{3(\epsilon-3\delta)}) + 16$. □

Acknowledgments

Dan Boneh and Nicola Greco contributed to this work through helpful comments and conversations. Nicola and Juan Benet came up with the name "ZigZag PoRep".

References

[1] Proof of replication. Protocol Labs, 2017. `https://filecoin.io/proof-of-replication.pdf`.

[2] Hamza Abusalah, Joël Alwen, Bram Cohen, Danylo Khilko, Krzysztof Pietrzak, and Leonid Reyzin. Beyond hellman's time-memory trade-offs with applications to proofs of space. In *ASIACRYPT*, 2017.

[3] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *ASIACRYPT*, pages 191–219, 2016.

[4] Ralph Phillips Alexander Lubotzky and Peter Sarnak. Ramanujan graphs. In *Combinatorica*, 1988.

[5] Joël Alwen, Jeremiah Blocki, and Benjamin Harsha. Practical graphs for optimal side-channel resistant memory-hard functions. In *CCS*, 2017.

[6] Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Sustained space complexity. In *EUROCRYPT*, 2018.

[7] Giuseppe Ateniese, Ilario Bonacina, Antonio Faonio, and Nicola Galesi. Proofs of space: When space is of the essence. In *SCN 2014*, 2014.

[8] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *ACM Conference on Computer and Communications Security*, 2007.

[9] Leonid Alexandrovich Bassalygo. Asymptotically optimal switching circuits. In *Problemy Peredachi Informatsii*, 1981.

[10] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Theory of Cryptography*, 2016.

[11] Dan Boneh, Joseph Bonneau, Benedikt Bunz, and Ben Fisch. Verifiable delay functions. 2018. To appear in CRYPTO 2018.

[12] Kevin D. Bowers, Ari Juels, and Alina Oprea. Proofs of retrievability: theory and implementation. In *CCSW'09 Proceedings of the 2009 ACM workshop on Cloud computing security)*, 2009.

[13] Dario Catalano and Dario Fiore. Vector commitments and their applications. In *PKC 2013*, 2013.

[14] Ethan Cecchetti, Ian Miers, and Ari Juels. Pies: Public incompressible encodings for decentralized storage. Cryptology ePrint Archive, Report 2018/684, 2018. `https://eprint.iacr.org/2018/684`.

[15] F.R.K. Chung. On concentrators, superconcentrators, generalizers, and nonblocking networks. In *Bell System Technical Journal*, 1979.

[16] Moni Naor Cynthia Dwork and Hoeteck Wee. Pebbling and proofs of work. In *CRYPTO*, 2005.

[17] Yevgeniy Dodis, Salil Vadhan, and Daniel Wichs. Proofs of retrievability via hardness amplification. In *Theory of Cryptography Conference (TCC)*, 2009.

[18] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In *CRYPTO*, 2015.

[19] Ben Fisch. Poreps: Proofs of space on useful data. Cryptology ePrint Archive, Report 2018/678, 2018. `https://eprint.iacr.org/2018/678`.

[20] Ben Fisch, Joseph Bonneau, Juan Benet, and Nicola Greco. Proofs of replication using depth robust graphs. In *Blockchain Protocol Analysis and Security Engineering 2018*, 2018. `https://cyber.stanford.edu/bpase2018`.

[21] Antonio Faonio Giuseppe Ateniese, Ilario Bonacina and Nicola Galesi. Proofs of space: when space is of the essence. In *Security and Cryptography for Networks, 2014.*, 2014.

[22] Dan Boneh Henry Corrigan-Gibbs and Stuart Schechter. Balloon hashing: a provably memory-hard function with a data-independent access pattern. In *Asiacrypt*, 2016.

[23] Ari Juels and Burton S Kaliski Jr. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 584–597. Acm, 2007.

[24] Arjen K Lenstra and Benjamin Wesolowski. A random zoo: sloth, unicorn, and trx. *IACR Cryptology ePrint Archive*, 2015, 2015.

[25] Sergio Demian Lerner. Proof of unique blockchain storage, 2014. `https://bitslog.wordpress.com/2014/11/03/proof-of-local-blockchain-storage/`.

[26] Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based from simple assumptions. In *ICALP*, 2016.

[27] Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent

zero-knowledge sets with short proofs. In *TCC*, 2010.
[28] Mohammad Mahmoody, Tal Moran, and Salil P Vadhan. Time-lock puzzles in the random oracle model. In *CRYPTO*. Springer, 2011.
[29] Silvio Micali. Computationally sound proofs. In *SIAM Journal on Computing.*, 2000. Preliminary version appeared in FOCS 1994.
[30] Salil Vadhan Omer Reingold and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *FOCS*, 2000.
[31] Sunoo Park, Krzysztof Pietrzak, Albert Kwon, Joël Alwen, Georg Fuchsbauer, and Peter Gai. Spacemint: A cryptocurrency based on proofs of space. Cryptology ePrint Archive, Report 2015/528, 2015. `http://eprint.iacr.org/2015/528`.
[32] Ronald L. Graham Paul Erdös and Endre Szemeredi. On sparse graphs with dense long paths. In *Computers & Mathematics with Applications*, 1975.
[33] Krzysztof Pietrzak. Proofs of Catalytic Space. Cryptology ePrint Archive # 2018/194, 2018.
[34] Arnold K. Pizer. Ramanujan graphs and hecke operators. In *Bull. Amer. Math. Soc.*, 1990.
[35] Ling Ren and Srinivas Devadas. Proof of space from stacked expanders. In *TCC*, 2016.
[36] Uwe Schöning. Better expanders and superconcentrators by kolmogorov complexity. In *SIROCCO*, 1997.
[37] Hovav Shacham and Brent Waters. Compact proofs of retrievability. In *Asiacrypt*, 2008.
[38] R. Michael Tanner. Explicit concentrators from generalized n-gons. In *Siam Journal on Algebraic and Discrete Methods*, 1984.
[39] Martin Tompa. Time-space tradeoffs for computing functions, using connectivity properties of their circuits. In *STOC*, 1978.
[40] Leslie G. Valiant. Graph-theoretic properties in computational complexity. In *Journal of Computer and System Sciences*, 1976.

A Concurrent work: PIEs

Independently and concurrently[10] to our work, Cecchetti et. al. [14] proposed a similar idea to our stacked DRGs (using butterfly networks instead of expander graphs) to construct what they called publicly incompressible encodings (PIEs).

PIE vs tight PoS A PIE has a similar soundness security definition to a PoS (c.f. Definition 1 in [14] and Page 7 [18] or Definition 9 [33]). An "n-encoder" PIE is a special case of a graph labeling PoS. An additional correctness requirement of a PIE is that the labeling is decodable (exactly like PoReps based on graph labeling, see definitions in [19,33]). However, the soundness of a PIE is weaker than a PoS because the definition of a PIE assumes that the prover cannot cheat in the computation of the encoding. Equivalently, it assumes the verifier knows the correct labeling of the graph. A PoS addresses the more general setting that the verifier does not know the correct labeling. The authors of PIEs remark that proving to a weaker verifier that the labeling was computed correctly is orthogonal, however this tends to be the most critical component affecting proof size and thus practicality of the construction. (In fact, to convert

[10] A preprint of their paper was first made publicly available on same day as a technical report containing the constructions described in this work, `https://web.stanford.edu/~bfisch/porep_short.pdf`, and the full version of this paper was submitted to Cryptology ePrint Archive a week later.

the construction they propose using stacked Butterfly graphs into a PoS using interactive oracle proofs would result in approximately a factor $\log n$ larger proof size than the construction we propose using stacked expander graphs, or a factor $\log n$ larger SNARK proof complexity if using SNARKs to compress the proof size). Nonetheless, a PIE combined with a suitable proof of correct encoding would result in a tight PoS and PoRep.

Proof size: Butterfly graphs vs expanders The specific construction of a PIE proposed in [14] uses stacked DRGs (exactly like our construction), but connects each adjacent layer with a Butterfly graph instead of expander edges. Each layer has an n-node (n, α, β)-DRG and layers are connected by a Butterfly graph, each consisting of $(n/2) \log n$ nodes and $n \log n$ edges. Instantiating the generic graph-labeling PoS or PoRep on this graph also gives a tight PoS or PoRep provided that the proof involves sufficient queries to bound the number of incorrect labels in each layer and Butterfly connector to δn. This requires the proof to query $(\lambda/\delta) \log n$ labels in each layer/connector, for a total of $(\lambda/\delta)\ell \log n$ overall. We prove in Appendix B that $\ell = 1/(1-\alpha)$ layers is sufficient.[11] For concrete comparison, when $\alpha = 0.80$ and $\epsilon = 0.01$ the number of proof queries required for the stacked DRGs with Butterfly graphs is proportional to $5 \log n\lambda/\epsilon$ and with expander graphs it is proportional to $11\lambda/\epsilon$. The Butterfly graph proofs are already a $5\times$ larger for kilobyte size data, and is $15\times$-$20\times$ larger for gigabyte or terabyte size data.

Compression-robustness vs proof of space Cecchetti et. al. introduce a definition called compression-robustness and prove that the stacked DRGs with Butterfly graph connectors is compression-robust. A $(1, \delta)$-*compression-robust* graph is a *DAG* with n sources, n sinks, balanced "data" input/output edges on each internal node, and all other edges treated as "key" edges such that after removal of fewer than n data edges and all key edges that share an origin node with one of these data edges there remains a path ending in a sink that includes at least δn key edges. Any DAG with n sources/sinks and balanced data input/output edges in which deletion of fewer than n nodes leaves a path of length at least δn (containing δn key edges) ending in a sink satisfies this property. This is because any path that remains after removing these nodes would also remain after removing only outgoing edges of these nodes. In fact, this covers many cases, except the case where a node in the path has multiple data outputs and does not contribute a key edge to the path.

Compression-robustness of the underlying encoder graph does not in general imply that the graph encoding is a proof of space, or even a PIE, at least not with the same tightness. Namely, we can provide a simple counterexample of a $(1, \delta)$-*compression-robust* graph for which the adversary can compress its storage to $n - 1$ data outputs and still recompute the nth data output with fewer than δn sequential rounds of KDF/PRP oracle queries. The reason this counterexample exists has to do with the fact that data labels can either be computed from upstream labels (predecessors) using KDF/PRP or also decoded from downstream labels (successors) using the inverse PRP. This means that a long path in the graph that remains after removing data edges (or equivalently unpebbled path after placing pebbles) does not necessarily need to be recomputed in topological order and therefore may not require sequential work.

[11] While it was proved in [14] that removal of fewer than n edges from the entire graph with $1/\alpha + 1$ layers leaves a path of length βn, it was not proven that this missing path is sequentially hard to pebble. Furthermore, the analysis did not show that this construction combined with an imperfect proof of correct encoding is a proof of space (i.e. when the prover is allowed up to δn errors per level).

Compression-robustness is insufficient for PoS/PIE The counterexample graph G' starts with any $(1, \delta)$-compression-robust graph G on $n-1$ sources U and $n-1$ sinks V and builds a new graph G' as follows. Add $n-1$ new sources, connecting each source to a distinct node in U, and $n-1$ new sinks, connecting each node of V to a distinct sink. Next add key edges to U and V turning each into a complete DAG. Finally, add an nth source u, an nth sink v, and a node w that connects u to v via single input/output "data" edges. Add key edges from every node in U and V to w. We claim this DAG is $(1, min(\delta, 1/2))$-compression-robust. Consider any removal of $n-1$ data edges. If any of these edges include any edge incident on u, w, or v, then fewer than $n-1$ data edges are removed from G. By construction, this means that G contains a path containing at least δn key edges and ending in a sink. Otherwise, no edges incident to u, w, or v are removed. Furthermore, fewer than $n/2$ edges have been removed from either U or V, thus one of U or V still contains a path of at least length $n/2$ (along key edges). The endpoint of this path connects via a non-removed key edge to w, which connects to the sink v. Therefore, the graph is $(1, min(\delta, 1/2))$-compression-robust. On the other hand, if an adversary stores only the data outputs to the first $n-1$ sinks then it can decode all the labels of G in parallel. Once it has computed these labels it can recover the labels on the data outputs of w and v in two sequential steps.

This first counterexample showed there exists a $(1, \delta)$-compression-robust graph encoding that the adversary can "compress" to $n-1$ data outputs. We now give another counterexample for which the adversary achieves a larger compression factor. Consider two stacked copies of G', a $(1, \delta)$-compression-robust graph on $n/2$ nodes, connected such that the sinks of the first copy G'_1 are the sources of the second copy G'_2. We then add $n/2$ new sinks V and sources U connected via an intermediary layer of $n/2$ nodes W so that each $u \in U$ is connected to a unique $v \in V$ via a unique intermediary $w \in W$. Finally, add key edges from every non sink/source nodes in $G'' = G'_1 \cup G'_2$ to each node in W. Suppose that k data output edges from W to the sinks V are removed, and $n-k-1$ data edges from G'' are removed for a total of $n-1$ removed edges. If $k \geq n/2$ then there are fewer than $n/2-1$ edges removed from G'', hence there is a path including $\delta n/2$ key edges ending in a sink of G''. Otherwise, if e_1 is the number of data edges removed from G'_1 and e_2 is the number of data edges removed from G'_2 then $min(e_1, e_2) \leq n/2 - k - 1$. If e_2 is the smaller value then there is a path including $\delta n/2$ key edges ending in a sink of G''. If e_1 is the smaller value, then there are at least k distinct paths including $\delta n/2$ key edges in G'_1 ending in k distinct sinks of G'_1. Although these are not sinks of G'', at least one of these paths connect to a sink in V because every node in G'_1 whose data output has not been removed still has key edges to every node in W. At least one data output from some $w \in W$ to some $v \in V$ has not been removed because $k < n/2$, hence the path in G'_1 connects to v. Thus, this graph is $(1, \delta/2)$-compression robust. On the other hand, if the adversary stores only the $n/2$ data outputs of G'', then it can decode in 2ℓ rounds of parallel queries all the data labels in G'', where ℓ is the number of layers in G', and then recover the data outputs to the sinks V in a single additional round of parallel queries. In total this requires $2\ell + 1$ parallel operations, (e.g. $O(\log n)$ when G' is constructed with stacked DRGs and Butterfly graph connectors).

In summary, this counterexample demonstrates the existence of an n-encoder graph in which removing fewer than n data edges leaves a path along at least $\delta n/2$ key edges, yet the PoS/PIE adversary can compress its storage of the encoding by at least a factor two (storing only $n/2$ data labels) and can recover the entire encoding without doing significant sequential work.

In the following section, we provide a different analysis of this construction, which is simplified

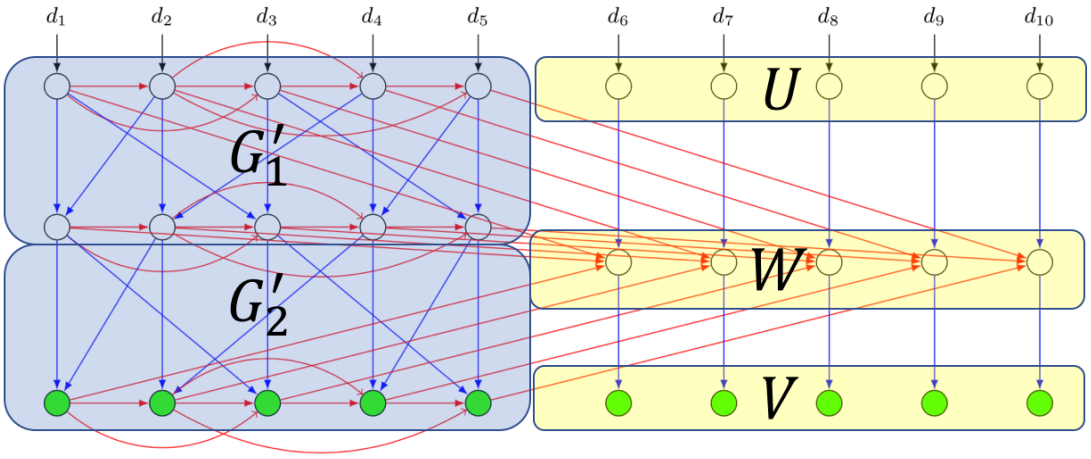

Figure A.1: Illustration of the counterexample graph built from two mock $(1, \delta)$-"compression-robust" graphs G'_1 and G'_2 on $n/2$ nodes. The result is a $(1, \delta)$-"compression-robust" whose labeling game is neither a tight PoS nor a PIE. Red edges are key edges and blue edges are data edges.

by using the fact that Butterfly graphs are superconcentrators [39, 40]. Superconcentrators with n nodes and n sinks have the property that any k sinks are connected to any k sources via k vertex disjoint paths. In this sense they have similar connectivity properties to extreme bipartite expanders. This analysis also accounts for the bounded number of deliberate errors the adversary can make in its computation of the graph labels (using a pebbling game analysis).

B Stacked DRGs with Superconcentrators

We show that instantiating the generic graph-labeling PoS on stacked DRGs connected between layers with superconcentrators instead of bipartite expanders (as in our construction) is also a tight PoS. This is a generalization of the construction described in [14]. Stacked superconcentrators (specifically stacked Butterfly graphs) have also been used before in the literature on proofs of space [7], just like stacked bipartite expanders [35]. They were also used in the PoS construction by Dziembowski et. al. in combination with DRGs [18]. The main difference here is how the DRGs and superconcentrators are interleaved, achieving a tighter PoS, and also how they are used to encode data.

The proof is quite straightforward and in particular much easier than analyzing the stacked DRGs with expanders. An *n-superconcentrator* is a graph with n sources and n nodes such that that any $k \leq n$ sources and k nodes are connected via k vertex disjoint paths. The Butterfly graph is a special superconcentrator on $n \log n$ nodes where all nodes except the sources have in-degree two and all except the nodes have out-degree two [39, 40]. More generally, an n-superconcentrator is d-balanced if every non-source has in-degree d and every non-node has out-degree d. If V_i and V_{i+1} are connected via a superconcentrator with balanced in-degree and out-degree then the graph labeling on V_{i+1} can encode the data labels on V_i (similar to the ZigZag PoRep). Each node assigns a unique component of its data label to each of its outgoing edges, and the label on each node is a PRP of the data inputs associated with each incoming

edge.

Graph construction and labeling The graph $\mathcal{G}_{DRSuper}$ consists of layers $V_1, ..., V_\ell$ where each V_i contains n nodes and the edges of a depth robust graph. A layered set of nodes G_i and edges E_i connect V_i to V_{i+1} such that $(V_i \cup G_i \cup V_{i+1}, E_i)$ is a d-balanced n-superconcentrator with sources V_i and nodes V_{i+1}. Note that the edges E_i of the superconcentrator do not include any edges internal to V_i or V_{i+1}. A labeling is then computed such that the label on each node of G_i encodes components of the labels on each of its parent nodes and each node of V_i encodes components of the labels on each of its parent nodes in G_{i-1}. Every label is broken into d blocks, represented as a length d vector. The jth component of a label c is denoted $c[j]$. Concretely, if a node $v \in G_i$ has d predecessors $v_1, ..., v_d$ and the edge (v_j, v) is the e_jth outgoing edge from v_j, then the label on c_v is $\mathsf{Enc}(c_{v_1}[e_1], ..., c_{v_d}[e_d])$. The label on node $v \in V_i$ that has parents $v_1, ...v_d$ in G_{i-1} and parents $u_1, .., u_{d'}$ in V_i is a keyed encoding $\mathsf{Enc}(k_v, c_{v_1}[e_1], ..., c_{v_d}[e_d])$ where $k_v = H(c_{u_1}, ..., c_{u_{d'}})$. This is exactly like ZigZag PoRep which treats labels on parents in the previous level as data inputs and all other parent labels as key derivation inputs.

Pebbling game vs labeling game Placing/storing a pebble on a node v corresponds to computing/storing its label c_v. A green edge corresponds to a data input and a white edge corresponds to a key input. Recall that the rules of the pebble game with white/green edges are that a pebble can only be placed on v if there are pebbles on all its immediate parent nodes, or alternatively if for each of its children nodes u such that (v, u) is a green edge there is a pebble on u and all nodes with white edges to u.

When the key derivation uses a random oracle then intuitively the key derivation inputs, namely the labels on all the nodes with white edges to v, must be known before the key for v can be derived. As noted in our analysis of ZigZag PoRep, when Enc is a PRP (modeled as an ideal cipher) then intuitively all the components on the parent labels that are input to c_v must be known, along with the key, before c_v can be even partially computed from these inputs (in the forward direction). Additionally c_v can be completely decoded only if all the labels on the nodes that encode components of c_v are known (i.e. nodes with data edges from v). While each ith component of c_v could be individually decoded from the label on the target of the ith edge from v, these components are not useful to derive any new labels that are known.

This only provides heuristic intuition for why the pebbling game captures the security of the labeling game and is not a proof! A formal proof requires an adaptation of Pietrzak's analysis of the random oracle labeling game and black pebbling game to show that any adversary who defeats the labeling game in some number of rounds with s bits of storage can be used to construct an adversary who defeats the pebbling game in the same number of rounds with $\approx s/m$ initial pebbles (m is length of the label), as otherwise it could compress the function table of the random oracle and/or ideal cipher.

Analysis Our first two claims establish that for any pebbling strategy using fewer than n black pebbles and appropriately bounded red pebbles in each level, a constant fraction of the final layer of nodes V_ℓ in $\mathcal{G}_{SuperDR}[\ell]$ are endpoints of long paths, i.e. of length at least βn, on unpebbled nodes. Furthermore, each node in this long path individually connects via an unpebbled path along green (data) edges to an unpebbled node in V_ℓ. This detail is important for showing that the long path must be pebbled in topological order in the black pebbling game with white/green edges.

In rough comparison to the analysis of [14] for the special case of Butterfly graphs, our Claim 12 encompasses their Theorem 2. Claim 13 matches their Proposition 1, but also takes into account the additional red pebbled nodes (i.e corresponding to any incorrectly derived labels not caught by the probabilistic proof of correctness). Finally, Claim 14 shows that the unpebbled paths identified in Claim 13 require sequential work to pebble (i.e. they must be pebbled in topological order) in the game with white/green edges, where pebbling can occur in both forward/reverse directions along green edges (corresponding to encode/decode operations respectively). This was not shown in the previous analysis [14].

Claim 12. *Let $\mathcal{G}_{SuperDR}[\ell]$ be the graph described above with $\ell = \frac{1}{1-\alpha-\delta}$ layers of stacked DRGs where each layer is connected by a balanced n-superconcentrator and each DRG is (n, α, β) depth robust. For any $\epsilon > 0$, if $(1-\epsilon)n$ black pebbles are initially placed on $\mathcal{G}_{DRSuper}[\ell]$ with at most δn additional pebbles on each level $V_i \cup G_i$ where $\delta < \epsilon/2$ then there exists an unpebbled path P of length at least βn ending in an unpebbled node of V_ℓ. Moreover, every node in P individually connects via a path on green edges (i.e. data edges) to an unpebbled node in V_ℓ.*

Proof. Let ρ_i denote the number of black pebbles placed on $V_i \cup G_i$. Let U_i denote a set of unpebbled nodes in V_i of size u_i. Let W be any subset of u_i nodes in V_{i-1}. There are u_i vertex disjoint paths starting from u_i distinct nodes in W, passing through G_{i-1}, and ending in u_i distinct nodes in U_i. Since there are at most $\rho_{i-1} + \delta n$ pebbles on $V_{i-1} \cup G_{i-1}$ at least $u_i - \rho_{i-1} - \delta$ of these paths do not contain any pebbles. Assuming $u_i - \rho_{i-1} - \delta > 0$, there exists at least one unpebbled path from W to U_i starting at a node $w \in W$. All these paths are exclusively along green edges, they do not involve the edges internal to V_{i-1} or V_i.

Form a new $W' \subseteq V_{i-1}$ of the same size by removing w and adding a new w' from $V_{i-1} \setminus W$. Repeating the argument shows the existence of another unpebbled path from W' to U_i. Repeat the process of removing one of the starting nodes of an unpebbled path from V_{i-1} to U_i each time and adding a new node from V_{i-1} that has not yet been included. This iterative process continues $1 - u_i$ times until there are no more fresh nodes to be included, and removes in sequence $1 - u_i$ starting nodes of unpebbled paths to U_i. The remaining nodes in the final set W'' contain an additional $u_i - \rho_i - \delta n$ unpebbled vertices with unpebbled paths to U_i. Thus, the total number of distinct nodes in V_{i-1} with unpebbled paths to U_i is at least $n - u_i + u_i - \rho_{i-1} - \delta n = n - \rho_{i-1} - \delta n$. Denote this set by U_{i-1} and $u_{i-1} = |U_{i-1}|$. We have shown that $u_{i-1} \geq n - \rho_{i-1} - \delta n$ provided that $u_i \geq \rho_{i-1} - \delta n$.

Consider U_ℓ to be the set of all unpebbled nodes in V_ℓ. For each $i < \ell$, let U_i be the set of unpebbled nodes in V_i that are connected via unpebbled paths to the set U_{i+1}, or equivalently by induction they are the set of unpebbled nodes in V_i that are connected to U_ℓ via unpebbled paths. We now prove by induction that $|U_i| = u_i \geq n - \rho_i - \delta n$ using the result of our analysis above. First, $u_\ell = n - \rho_\ell - \delta n$ and $\rho_\ell \leq (1-\epsilon)n - \rho_{\ell-1}$, so $u_\ell \geq \epsilon n + \rho_{\ell-1} - \delta n$. Since $\epsilon > 2\delta$ by hypothesis this also shows $u_\ell > \rho_{\ell-1} + \delta n$. Next, assuming that $u_{i+1} \geq \rho_i + \delta n$, it follows that $u_i \geq n - \rho_i - \delta n$ as shown above. Therefore $u_i - \rho_{i-1} - \delta n \geq n - \rho_{i-1} - \rho_i - 2\delta n \geq (\epsilon - 2\delta)n > 0$. By induction, every $u_i \geq \rho_{i-1} + \delta n$ and $u_i \geq n - \rho_i - \delta n$.

Equivalently, we have shown that for all i, $\rho_i \geq n - u_i - \delta n$. Suppose that $u_1, ..., u_\ell < \alpha n$. Then $\sum_{i=1}^{\ell} \rho_i > (1-\alpha-\delta)n\ell > (1-\epsilon)n$. This contradicts the hypothesis that $\sum_{i=1}^{\ell} \rho_i \leq (1-\epsilon)n$. Therefore, it must be the case that $u_i \geq \alpha n$ for some $i \leq \ell$. Each of these u_i unpebbled nodes in V_i connects via a green edge unpebbled path to U_ℓ. Furthermore, among these u_i unpebbled nodes in V_i there is an unpebbled white edge path of length βn because V_i is (n, α, β) depth

robust. The endpoint of this path is still in U_i and is thus connected via an unpebbled green edge path to U_ℓ. □

Claim 13. *For any $\epsilon > 0$, if $(1-\epsilon)n$ black pebbles are initially placed on $\mathcal{G}_{DRSuper}[\ell]$ with at most δn additional pebbles on each level $V_i \cup G_i$ where $\delta < \epsilon/4$, then there are at least $\epsilon n/2$ unpebbled paths of length at least βn ending in $\epsilon n/2$ distinct nodes in V_ℓ. All nodes along all of these paths individually connect via green edge paths to an unpebbled node of V_ℓ.*

Proof. By the same reasoning as Claim (Section 3), if for any configuration of γn black pebbles on the graph $\mathcal{G}_{SuperDR}[\ell]$ with δn additional red pebbles in each $V_i \cup G_i$ there exists an unpebbled path of length t ending in an unpebbled node of V_ℓ, then for any configuration of $\gamma' n$ black pebbles placed on nodes outside V_ℓ and δn red in each level, every set U of $(1-(\gamma-\gamma')-\delta)n$ unpebbled nodes in V_ℓ contains at least one endpoint of a length t unpebbled path. This is simply because the remaining nodes in $V_\ell \setminus U$ can be entirely pebbled with $(\gamma-\gamma')n$ black and δn red pebbles for a total usage of γn black pebbles overall.

If $(1-\epsilon)n$ black pebbles are placed overall then certainly at most $(1-\epsilon)n$ are placed outside of V_ℓ. Setting $\epsilon' = \epsilon/2$, and further restricting $\delta < \epsilon n/4$, we have already shown in Claim 12 that for any configuration of $\epsilon' n$ black pebbles with δn red in each level leaves an unpebbled path of length βn ending in V_ℓ. Moreover, all nodes along this path are connected via an unpebbled path along green edges to V_ℓ. Therefore, equating $\gamma = 1-\epsilon/2$ and $\gamma' = 1-\epsilon$, every set of $(1-\epsilon/2-\delta)n$ unpebbled nodes in V_ℓ contain an endpoint of an unpebbled path of length βn. There are at least $(1-\epsilon-\delta)n$ unpebbled nodes U in V_ℓ for any configuration of the $(1-\epsilon)n$ black pebbles and red pebbles, therefore at least $\epsilon n/2$ nodes in U are the endpoints of length βn unpebbled paths. To see this, consider subsets of U of size $(1-\epsilon/2-\delta)n$. Each subset $U' \subseteq U$ contains the endpoint of a length βn unpebbled path. We iterate the following procedure. Remove one of these endpoints from U' and place it in a list of used nodes, then form a new subset of the same size by adding a node in $U \setminus U'$ that is not in the used list. After $\epsilon/2$ iterations there are no more unused nodes and $\epsilon/2$ distinct endpoints have been identified and added to the list. □

Finally we show that for any unpebbled directed path P in $\mathcal{G}_{SuperDR}[\ell]$ of length t consisting of nodes that all connect individually via an unpebbled green edge path to a node in V_ℓ, any pebbling strategy requires at least t rounds to pebble P in the black pebbling game with white/green edges.

Claim 14. *Any unpebbled path P in $\mathcal{G}_{SuperDR}[\ell]$ consisting of nodes that each individually connect via unpebbled green edge paths to an unpebbled node in V_ℓ requires t rounds to pebble in the black pebbling game with white/green edges, where edges between two nodes in some V_i are white and all other edges are green.*

Proof. We can break up a path P into green edge sections and white edge sections. A white edge section of a path is a directed path within some V_i, whereas green edge sections pass through some G_i. We separately claim that white edge sections and green edge sections must be pebbled in topological order by any pebbling strategy. Furthermore, if a white edge section in some V_i precedes a green edge section that connects V_i and V_{i+1} then the white edge section must be pebbled in topological order before the green edge section.

First consider a green edge section $P_g \subseteq P$. P_g is also a segment of an unpebbled green edge path to a node in V_ℓ because by hypothesis the endpoint of P_g connects via an unpebbled

green edge path to a node in V_ℓ. We now prove by induction that any unpebbled green edge path ending in a node in V_ℓ must be pebbled in topological order. This implies that P_g must be pebbled in topological order. For the base case, there is only one node (a node in V_ℓ), and the claim is trivial. Now suppose that any unpebbled directed path of length $t-1$ along green edges ending in V_ℓ must be pebbled in topological order. Let P' be a length t unpebbled green edge path to V_ℓ starting at a node u. P' breaks into a path P'' of length $t-1$ starting at v and a node u with a directed green edge to v. The only way to pebble v is either by first pebbling its parents (including u) or by first pebbling the targets of all its green edges (assuming it has at least one green edge target). If v is in V_ℓ then it doesn't have any green edge targets, and therefore can only be pebbled via its parents. If v is not in V_ℓ then it has a directed edge to another vertex w in P'', hence by hypothesis any strategy must pebble v before w. This implies u must be pebbled before v, hence P' must be be pebbled in topological order. We conclude that P_g must be pebbled in topological order.

Second consider a white edge section $P_w \subseteq P$ in some V_i that precedes a green edge section P_g connecting V_i to V_{i+1}. By hypothesis all nodes in this path individually connect via unpebbled green edges paths to V_{i+1}. Consider any node u along the path. We claim that any pebbling strategy which pebbles this path does not pebble u via its green edge targets. Since u connects via an unpebbled green edge path to V_ℓ, at least one of its green edge targets is unpebbled. Furthermore, these two nodes are along an unpebbled green edge path ending in V_ℓ and thus must be pebbled in topological sequence as already shown. Since each node of P_w cannot be pebbled via its green edge targets, they are each pebbled via their parent nodes, which include their predecessors in P_w. Therefore P_w must be pebbled in topological order. Finally, as both P_w and P_g must be pebbled in topological order and the starting node v of P_g is the ending node of P_w, it follows that P_w must be pebbled before P_g.

□

C Mixing Data Labels in Stacked DRGs with Expanders

Cecchetti et. al. introduced a very nice idea, which is to split data across over multiple output edges, effectively mixing the data encoded by different labels and not just the key derivation dependencies. This works as long as the non sink/source nodes have balanced in-degree and out-degree (for data edges). Our stacked construction with bipartite expanders naturally has balanced in-degree and out-degree for edges crossing between layers since Chung's randomized construction is a d-regular bipartite expander. Thus, with degree 8 expanders we can break up the labels into 8 components on every node that are each propagated along a different output edge, just as in the construction using d-balanced n-superconcentrator connectors. This turns the Stacked-DRG PoS into a PoRep that admits parallelized decoding. The advantage over ZigZag PoRep is the number of layers in the stacked DRGs with bipartite expanders, for which we were able to give a tighter bound (by roughly a factor of two). Even the buffer size used for the encoding need not exceed the file size. The data components along all input edges to a node on the next level are consumed all at once and space can be freed for the data components of the newly computed label. The locality of data writes is not preserved and tracking the locations of stored data label components is more complex, as they are no longer simply stored within the buffer at the node's index.