Stacked-DRGs & ZigZag Security parameters (work in progress!)

irene@protocol.ai, luca@protocol.ai, nicolas.gailly@protocol.ai

August 18, 2019

Contents

1 Parameters Recap

1.1 Stacked DRGs (no tapering)

With the following parameters

- conditions: any δ and ϵ such that $\epsilon \leq 0.24$ and $\delta < \epsilon/2$
- number of layers: $\ell = \max\left\{\frac{0.68-\epsilon+\delta}{0.12-\delta}, \log_2\left(\frac{1}{3(\epsilon-2\delta)}\right) + \frac{0.12}{0.12-\delta} + 1\right\} + 1$
- number of offline challenges: $c = \frac{-\lambda}{\log_2(1-\delta)}$
- number of online challenges: $k = \frac{-\lambda}{\log_2(2-\epsilon-2\delta)-1}$

we get a PoS with

- space gap: $\epsilon + 2\delta$,
- time: $\beta n - 1$,
- soundness: $2^{-\lambda}$.

1.2 ZigZag (no tapering)

With the following parameters

- conditions: any δ and ϵ such that $\epsilon + \delta \leq 0.24$ and $\delta < \epsilon/3$ and
- number of layers: $\ell = 2\log_2\left(\frac{1}{3(\epsilon-2\delta)}\right) + 2\frac{0.8-\epsilon+\delta}{0.12-2\delta} + 2$
- number of offline challenges: $c = \frac{-\lambda}{\log_2(1-\delta)}$
- number of online challenges: $k = \frac{-\lambda}{\log_2(2-\epsilon-3\delta)-1}$

we get a PoS with

to be rewritten after correcting Lemma 6...

- space gap: $\epsilon + 3\delta$,
- time: $\beta n - 1$,
- soundness: $2^{-\lambda}$.

2 Notation and definitions

ToDo: Adapt this notation to the Spec/implementat (*e.g.*, change the numbering of layers)!

1. Numbering always starts from 0. For example, if we have n nodes, the first on the left is node 0, the last on the right is node $n-1$. If we have graph with ℓ layer, then the first layer on the top is layer 0 and the last one is layer $\ell - 1$.

2. Length of a path in a graph = number of edges contained on the path.

3. A directed acyclic graph (DAG) with n nodes is a $(n, 0.80, \beta)$ **DRG** if any set of $0.8n$ nodes contains a path of length $\geq \beta n$ (β is a constant < 0.8).

   Notice, for efficiency we also require small in-degree (*e.g.*, $d = O(\log n)$).

   Question: In our implementation we have $d = 5$, is this secure?

4. Definition of **Stacked DRGs graph**: $\mathcal{G}_{\ell,n}$ is a graph with ℓ layers where each layer V_i is a $(n, 0.80, \beta)$ DRG and we add edges in each pair of layers (V_i, V_{i+1}) following the randomized Chung's construction for regular bipartite graphs with degree 8 (edges from layer i to layer $i+1$).

   Note: The number of nodes n has to be large enough in order to give negligible probability of failure for Chung's construction and the DRG construction.

   Question: Is $n = 2^{30}$ enough?

5. Definition of **ZigZag graph**: $\mathcal{Z}_{\ell,n}$ is a graph with ℓ layers where each even layer V_{2i} is a $(n, 0.80, \beta)$ DRG with edges from lower index nodes to higher index nodes and each odd layer V_{2i+1} is a $(n, 0.80, \beta)$ DRG with edges from higher index nodes to lower index nodes $(i = 1, 2, \dots)$.

   Moreover, for each each pair of layers (V_i, V_{i+1}) add edges following Chung's construction for degree 8 and then project these edges on layer V_{i+1}. Change the direction of these edges following this rule: if i is even then any edge in V_{i+1} has direction from higher to lower indices, if i is odd then any edge in V_{i+1} has direction from lower to higher indices.

   More detail in Section 2.1

6. Let $\mathcal{G}_{\ell,n}[\epsilon, \delta]$ indicate the Stacked DRG graph $\mathcal{G}_{\ell,n}$ with the following pebble configuration: $(1-\epsilon)$ black pebbles overall and δ red pebbles in each layer. We say that $\mathcal{Z}_{\ell,n}[\epsilon, \delta]$ is (t, μ)-hard if t rounds (parallel moves) are required to pebble a fraction μ of nodes in the last layer.

7. Let $\mathcal{Z}_{\ell,n}[\epsilon, \delta]$ indicate the ZigZag graph $\mathcal{Z}_{\ell,n}$ with the following pebble configuration: $(1-\epsilon)$ black pebbles overall and δ red pebbles in each layer. We say that $\mathcal{Z}_{\ell,n}[\epsilon, \delta]$ is (t, μ)-hard if t rounds of moves that only use "forward steps" are required to pebble a fraction μ of nodes in the last layer.

   Further notations for the pebble configuration:

   - ρ_i is the fraction of nodes with black pebbles in layer V_i;
   - $\gamma_i = \sum_{j=0}^{i-1} \rho_j$ (*i.e.*, $\gamma_i n$ is the number of black pebbles from before level i.

2.1 Graph constructions

Chung's construction for degree d.

Take a bipartite graph with two layers, each with n node and sample a random permutation

$$P : \{0, \dots, d-1\} \times \{0, \dots, n-1\} \to \{0, \dots, d-1\} \times \{0, \dots, n-1\}$$

We add an edge from j in the upper level to i in the lower level if there are k, k' such that $(j, k') = P(k, i)$. Equivalently, the parents of nodes in the lower layer are defined by the following procedure:

procedure C.PARENTS(i)
  $Parents(i) = \emptyset$
  for $k = 0, \dots, d-1$ **do**
    evaluate P on the pair (k, i): $(j, k') = P(k, i)$
    $Parents(i) = Parents(i) \cup \{j\}$
  return $Parents(i)$

Chung's construction gives a bipartite graph whee the in-degree and the out-degree are less or equal to d. For large n, this construction give a d-regular graph.

ZigZag construction.

Assume that DRG(i) is the procedure that gives the parents of the node i in a one-layer DRG graph.

procedure ZZ.PARENTS(i, j)
  $Parents(i) = \emptyset$
Layer 0
  if $j = 0$ **then** $Parents(i) =$ DRG(i)
  return $Parents(i)$
Layer $2j + 1$
  if $j \mod 2 = 1$ **then** $Temp.Parents =$ DRG($n - i + 1$)
    for $i' \in Temp.Parents$ **do** $Parents(i) = Parents(i) \cup \{n - i' + 1\}$
    for $k = 0, \dots, d-1$ **do**
      $(i', k') = P(k, i)$
      $(i'', k'') = P^{-1}(k, i)$
      if $i' > i$ **then** $Parents(i) = Parents(i) \cup \{i'\}$
      if $i'' > i$ **then** $Parents(i) = Parents(i) \cup \{i''\}$
  return $Parents(i)$
Layer $2j$
  if $j > 0$ and $j \mod 2 = 0$ **then** $Parents(i) =$ DRG(i)
    for $k = 0, \dots, d-1$ **do**
      $(i', k') = P(k, i)$
      $(i'', k'') = P^{-1}(k, i)$
      if $i' < i$ **then** $Parents(i) = Parents(i) \cup \{i'\}$
      if $i'' < i$ **then** $Parents(i) = Parents(i) \cup \{i''\}$
  return $Parents(i)$

3 Proof overview (Stacked DRGs, no tapering)

3.1 Stacked DRGs, no tapering

- Claim 6 says that $\mathcal{G}_\ell[\epsilon, \delta]$ with

$$\ell = \max\left\{\frac{0.68 - \epsilon + \delta}{0.12 - \delta}, \log_2\left(\frac{1}{3(\epsilon - 2\delta)}\right) + \frac{0.12}{0.12 - \delta} + 1\right\} \tag{1}$$

$$\delta < \epsilon/2 \tag{2}$$

  is $(\beta n - 1, 1)$ hard.

- Then using Claim 4 and we can say that $\mathcal{G}_{\ell+1}[\epsilon + 2\delta, \delta]$ with

$$\epsilon \leq 0.24 \tag{3}$$

  is $(\beta n - 1, 1 - \frac{\epsilon+2\delta}{2})$ hard.

- Finally, use Claim 2. Assume that we ask for c independent random challenges in the offline phase (in each layer we open the same c nodes) and k independent random challenges in the online phase (last layer only).

  Using the Stacked DRGs graph with $\ell + 1$ layers with conditions (1), (2), (3) and with

$$c = \frac{-\lambda}{\log_2(1 - \delta)} \tag{4}$$

$$k = \frac{-\lambda}{\log_2(2 - \epsilon - 2\delta) - 1} \tag{5}$$

  gives a PoS with space gap: $\epsilon + 2\delta$, time: $\beta n - 1$ and soundness: $2^{-\lambda}$.

3.2 ZigZag, no tapering

- Claim 11 says that $\mathcal{Z}_\ell[\epsilon, \delta]$ with

$$\ell = 2\log_2\left(\frac{1}{3(\epsilon - 2\delta)}\right) + 2\frac{0.8 - \epsilon + \delta}{0.12 - 2\delta} \tag{6}$$

$$\delta < \min\{0.06, \epsilon/3\} \tag{7}$$

  is $(\beta n - 1, 1)$ hard.

- Then using Claim 9 and we can say that $\mathcal{Z}_{\ell+2}[\epsilon + 3\delta, \delta]$ with

$$\epsilon + \delta \leq 0.24 \tag{8}$$

  is $(\beta n - 1, 1 - \frac{\epsilon+3\delta}{2})$ hard.

- Finally, use Claim 2. Assume that we ask for c independent random challenges in the offline phase (in each layer we open the same c nodes) and k independent random challenges in the online phase (last layer only).

  Using the ZigZag graph with $\ell + 2$ layers with conditions (6), (7), (8) and with

$$c = \frac{-\lambda}{\log_2(1-\delta)} \tag{9}$$

$$k = \frac{-\lambda}{\log_2(2-\epsilon-3\delta)-1} \tag{10}$$

  gives a PoS with space gap: $\epsilon + 3\delta$, time: $\beta n - 1$ and soundness: $2^{-\lambda}$.

4 Notes about the security analysis

Claim 4.1 (Correct Claim 4)**.** If $\mathcal{G}_{\ell-1}[\epsilon-2\delta,\delta]$ is $(t,1)$-hard and $0<\epsilon-2\delta\leq 0.24$, then $\mathcal{G}_{\ell}[\epsilon,\delta]$ is $(t^*,1-\epsilon/2)$-hard with $t^*=\min(\beta n-1,t+1)$.

Claim 4.2 (Alternative Claim 4)**.** If $\delta<\epsilon/2$ and $\mathcal{G}_{\ell-1}[\epsilon/2-\delta,\delta]$ is $(t,1)$-hard, then $\mathcal{G}_{\ell}[\epsilon,\delta]$ is $(t^*,1-\epsilon/2)$-hard with $t^*=\min(\beta n-1,t+1)$.

Proof. Let S be a subset of nodes from the last layer in $\mathcal{G}_{\ell}[\epsilon,\delta]$ with size $(1-\epsilon/2)n$, we need to show that t^* rounds are required to pebble S. Let X be the subset of S of unpebbled nodes, it is enough to show that X requires t^* rounds to be pebbled. Let $|X|=\alpha^* n$ and notice that $\alpha^*\geq\alpha_\ell-\epsilon/2\geq\epsilon/2-\delta>0$.

Now, consider two cases:

1. If $\alpha^*\geq 0.8$, then $\beta n-1$ rounds are required to pebble X (this is because V_ℓ is a $(n,0.8,\beta)$ DRG, so X contains a path of length βn).

2. If $\epsilon/2-\delta<\alpha^*<0.8$, then we have that the nodes in X are connected to β^* nodes in layer $\ell-1$ with $\beta^*>1.17\alpha^*$ (because of the table in Figure 2.2 in ePrint 2018/702).

   ToDo: Check this!

   Among these nodes, at least $\alpha'\geq\beta^*-\rho_{\ell-1}-\delta$ are unpebbled. From this,

   $$\alpha'\geq\beta^*-\rho_{\ell-1}-\delta=\beta^*+(\gamma_{\ell-1}-\gamma+\rho_\ell)-\delta$$

   And therefore

   $$\begin{aligned}\alpha'-\gamma_{\ell-1}&\geq\beta^*-\gamma+(1-\alpha_\ell-\delta)-\delta\\&\geq\beta^*-\gamma+(1-\alpha^*-\epsilon/2)-2\delta\\&>(1.17-1)\alpha^*+\epsilon/2-2\delta\\&>0.17(\epsilon/2-\delta)+\epsilon/2-2\delta=1.17(\epsilon/2-\delta)-\delta\\&>(\epsilon/2-\delta)-\delta\end{aligned}\tag{11}$$

   The last inequality is because $\delta<\epsilon/2$ implies $1.17(\epsilon/2-\delta)>\epsilon/2-\delta$. Now, consider the graph $\mathcal{G}_{\ell-1}[\epsilon',\delta]$ with $\epsilon'=\epsilon/2-\delta$ and the following constrain in its pebble configuration: the number of black pebbles from layer 1 to layer $\ell-2$ is $\gamma_{\ell-1}n$ (the same number as in $\mathcal{G}_{\ell}[\epsilon,\delta]$). Then, (11) says that we can apply Claim 3 from ePrint 2018/702, and therefore the fact that $\mathcal{G}_{\ell-1}[\epsilon',\delta]$ is $(t,1)$-hard implies that at least t rounds are required to pebble the unpebbled nodes among the $\beta^* n$ dependency of X. Finally, X needs $t+1$ rounds to be pebbled in in $\mathcal{G}_{\ell}[\epsilon,\delta]$.

□

Claim 4.3 (Correct Claim 4)**.** If $\mathcal{Z}_{\ell-2}[\epsilon-3\delta,\delta]$ is $(t,1)$-hard and $\epsilon-2\delta\leq 0.24$, then $\mathcal{Z}_{\ell}[\epsilon,\delta]$ is $(t^*,1-\epsilon/2)$-hard with $t^*=\min(\beta n-1,t+2)$.

Proof. Let S be a subset of nodes from the last layer in $\mathcal{Z}_{\ell}[\epsilon,\delta]$ with size $(1-\epsilon/2)n$, we need to show that t^* rounds are required to pebble S (assuming we have $(1-\epsilon)n$ black pebbles overall and δ red pebbles in each layer).

Let X be the subset of S of unpebbled nodes, it is enough to show that X requires t^* rounds to be pebbled starting from the same configuration of pebbles stated before. Notice that $|X|\geq(\alpha_\ell-\epsilon/2)n$,

and if $(\alpha_\ell - \epsilon/2)n > 0.8$ then $\beta n - 1$ rounds are required to pebble X because the last layer is a DRG.

Define $\alpha^* = (\alpha_\ell - \epsilon/2) - \rho_{i-1} - \rho_{i-2}$ ($\alpha_\ell n$ defined as the number of unpebbled nodes that in the last layer of $\mathcal{Z}_\ell[\epsilon, \delta]$, $\rho_j n$ defined as the number of black pebbles in layer j in $\mathcal{Z}_\ell[\epsilon, \delta]$), define Z as the set of *forward dependencies* of X in layer V_{i-2}, and $\alpha' = |Z|/n$. Because of Lemma 6 we split the proof in two cases:

1. $\alpha^* \leq 1/3$: in this case we have that $\alpha' \geq 2\alpha^* - 2\delta$. This implies that

$$\alpha' - \gamma_{\ell-2} \geq \epsilon - 4\delta \tag{12}$$

   ($\gamma_{\ell-2} n$ defined as the number of black pebbles from layer 1 to layer $\ell - 3$ in $\mathcal{Z}_\ell[\epsilon, \delta]$).

   Now, consider the graph $\mathcal{Z}_{\ell-2}[\epsilon - 3\delta, \delta]$ with the following specific constrain in its pebble configuration: the number of black pebbles from layer 1 to layer $\ell - 3$ is $\gamma_{\ell-2} n$ (the same number as in $\mathcal{Z}_\ell[\epsilon, \delta]$). Then, (12) says that we can apply Claim 3, and therefore the fact that $\mathcal{Z}_{\ell-2}[\epsilon - 3\delta, \delta]$ is $(t, 1)$-hard implies that Z needs at least t rounds in order to be pebbled in $\mathcal{Z}_{\ell-2}[\epsilon - 3\delta, \delta]$ (note that this implies that Z needs at least t round in $\mathcal{Z}_\ell[\epsilon, \delta]$ too). Therefore, X needs $t + 2$ rounds to be pebbled in in $\mathcal{Z}_\ell[\epsilon, \delta]$.

2. $\alpha^* > 1/3$: in this case we have that $\alpha' \geq 0.12 + \alpha_\ell - \epsilon/2 - 2\delta - \rho_{\ell-1} - \rho_{\ell-2}$. This implies that

$$\alpha' - \gamma_{\ell-2} \geq 0.12 - 3\delta + \epsilon/2 \tag{13}$$

   If $\epsilon - 2\delta \leq 0.24$, then $0.12 - 3\delta + \epsilon/2 \geq \epsilon - 4\delta$ and we can conclude as before.

□

Definition 4.1. Forward dependencies: a node u is a forward dependency of the unpebbled node v if we must first pebble u using the encoding functionality in order to pebble v (in other words, we can't use decoding to pebble u)

Claim 4.4 (Claim 8)**.** Let X be a set of unpebbled nodes in V_ℓ of $\mathcal{G}_{ZZ}[\ell]$. We want to track forward dependenciees of X. To this aim, set $Y = X$ and proceed recursively layer by layer, starting from $V_{\ell-1}$. For each layer V_i, add a node to Y if

1) It is unpebbled and it has the same index of a node in $Y \cap V_{i+1}$ (i.e. the node indexed with the same index in V_{i+1} is in Y).

2) It is unpebbled and had a two-hop directed path to $Y \cap V_{i+1}$, via an expader white edge followed by a green edge (i.e. it is unpebbled and it is a parent of a node of type 1.)

All nodes in Y are forward depeendencies of X.

Lemma 4.5 (Lemma 6)**.** Let X be a set of αn nodes in layer V_i of $\mathcal{Z}_{\ell,n}[\epsilon, \delta]$ and let Z be the set of the forward dependencies of X in layer V_{i-2}. It holds that:

1. If $\alpha \leq 1/3$ ....
2. If

to be rewritten after correcting the proof

Proof. Consider a set $X \subset U_i$ (i.e. a set of unpebbled nodes in the i-th layer) of size αn. We define the following sets:

- $X' = \Pi^{i-1}(X) \cap U_{i-1}$ is the subset of the projection of X in layer $i-1$ that is unpebbled. (take all the indeces of all the nodes in $X \subset V_i$, consider the set of nodes $\Pi^{i-1}(X)$ with correspondent indeces at level $i-1$ and take only the unpebbled ones).
- $X'' = \Pi^{i-2}(X') \cap U_{i-2}$ is the subset of the projection of X' in layer $i-2$ that is unpebbled
- $\Gamma_{out}(X')$ are the nodes which are on the boundary of X' in V_{i-1} and are children of nodes in X' (i.e. with edges *from* X'). Note that $\Gamma_{out}(X')$ and X' are disjoint. $\Gamma_{out}(X'')$ is defined equivalently in layer V_{i-2}.
- $\Gamma_{in}(X')$ are the nodes which are on the boundary of X' in V_{i-1} and are parents of nodes in X' (i.e. with edges *to* X'). Note that $\Gamma_{in}(X')$ and X' are disjoint. $\Gamma_{in}(X'')$ is defined equivalently in layer V_{i-2}.

As a first step, we partition the set $P_{i-1} = \{$ pebbles in layer $i-1\}$, $|P_{i-1}| = \rho_{i-1} + \delta_{i-1}$, in 3 subsets:

- $P_{i-1} \cap \Gamma_{in}(X')$, the subset of pebbles in $\Gamma_{in}(X')$. We set $p^D_{i-1} = |P_{i-1} \cap \Gamma_{in}(X')|$
- $P_{i-1} \cap \Gamma_{out}(X')$, the subset of pebbles in $\Gamma_{out}(X')$. We set $p^T_{i-1} = |P_{i-1} \cap \Gamma_{out}(X')|$
- $P_{i-1} \cup ((P_{i-1} \cap \Gamma_{in}(X')) \cup (P_{i-1} \cap \Gamma_{out}(X')))^{\mathsf{c}}$, the subset of the other pebbles, with cardinality $p_{i-1}n$

In the same way, we partition the set $P_{i-2} = \{$ pebbles in layer $i-2\}$, $|P_{i-2}| = \rho_{i-2} + \delta_{i-2}$, in 3 subsets:

- $P_{i-2} \cap \Gamma_{in}(X'')$, the subset of pebbles in $\Gamma_{in}(X'')$. We set $p^T_{i-2} = |P_{i-2} \cap \Gamma_{in}(X'')|$
- $P_{i-2} \cap \Gamma_{out}(X'')$, the subset of pebbles in $\Gamma_{out}(X'')$. We set $p^D_{i-2} = |P_{i-2} \cap \Gamma_{out}(X'')|$
- $P_{i-2} \cup ((P_{i-2} \cap \Gamma_{in}(X'')) \cup (P_{i-1} \cap \Gamma_{out}(X')))^{\mathsf{c}}$, the subset of the other pebbles, with cardinality $p_{i-2}n$

Now we define the set $Z =: X'' \cup (\Gamma_{in}(X'') \cap U_{i-2}) \cup (\Pi^{i-2}(\Gamma_{in}(X') \cap U_{i-1}) \cap U_{i-2})$.
Observe that each node $z \in Z$ is a forward dependency of X at layer $i-2$. Here is why:

- X'': all nodes here are unpebbled nodes in layer $i-2$ that are connected to nodes in X via green edges (by construction), and so for 4.4, case 1), are forward dependencies of X.
- $(\Gamma_{in}(X'') \cap U_{i-2})$: all nodes here are unpebbled parents of nodes in X'', and so for 4.4, case 2) are forward dependencies of X.
- $(\Pi^{i-2}(\Gamma_{in}(X') \cap U_{i-1}) \cap U_{i-2})$: here we have to bee more careful. First, consider $\Gamma_{in}(X') \cap U_{i-1}$. Each node in $\Gamma_{in}(X') \cap U_{i-1}$ is an unpebbled parent of a node in X'. Since all nodes in X' here are unpebbled nodes in layer $i-1$ that are connected to nodes in X via green edges (by construction), and so for 4.4, case 1), are forward dependencies of X. It follows that, for 4.4, case 1) each node in $\Gamma_{in}(X') \cap U_{i-1}$ is a forward dependency for X. Now, if we take the projection $\Pi^{i-2}(\Gamma_{in}(X') \cap U_{i-1})$ to layer $i-2$ and we take the intersection with U_{i-2} (the

unpebbled nodes in layer $i-2$), it is easy to observe that each node in $(\Pi^{i-2}(\Gamma_{in}(X') \cap U_{i-1}) \cap U_{i-2})$ is an unpeppled node in layer $i-2$ which is connected with a green edge to a node in $\Gamma_{in}(X') \cap U_{i-1}$. Therefore, for 4.4, case 1), each node in $(\Pi^{i-2}(\Gamma_{in}(X') \cap U_{i-1}) \cap U_{i-2})$ is a forward dependency for X.

Now, we recall that $\Gamma_{in}(X'') \cup \Gamma_{out}(X'') \cup X'' = \Gamma(X'') \cup X''$. We observe that $\alpha \geq |X''| = \alpha'' \geq \alpha - \rho_{i-1} - \rho_{i-2} - \delta_{i-1} - \delta_{i-2}$. If $\alpha < \frac{1}{3}$, we know (*from Corollary 1, Ben's paper*, Chung's degree 8 bipartite expansion) that $|\Gamma(X'') \cup X''| = 2|X'| \geq 2(\alpha - \rho_{i-1} - \rho_{i-2} - \delta_{i-1} - \delta_{i-2})n$ with overwhelming probability.

It follows that

$$\begin{aligned}
|Z| &\geq |X''| + |\Gamma_{in}(X'')| - p^T_{i-2}n + |\Gamma_{in}(X')| - p^D_{i-1}n - p^D_{i-2}n \\
&\geq |X''| + |\Gamma_{in}(X'')| + |\Gamma_{out}(X'')| - p^T_{i-2}n - p^D_{i-1}n - p^D_{i-2}n \\
&\geq |X''| + |\Gamma_{in}(X'')| + |\Gamma_{out}(X'')| - \rho_{i-1}n - \delta_{i-1}n - \rho_{i-2}n - \delta_{i-2}n \\
&\geq 2(\alpha - \rho_{i-1} - \delta_{i-1} - \rho_{i-2} - \delta_{i-2})n - \rho_{i-1}n - \delta_{i-1}n - \rho_{i-2}n - \delta_{i-2}n \\
&= 2\alpha n - 3(\rho_{i-1} + \delta_{i-1} + rho_{i-2} + \delta_{i-2})n
\end{aligned}$$

If $\alpha > \frac{1}{3}$ we can use Corollary 2 from Ben's paper. Since we know that $\alpha \geq \alpha'' \geq \alpha - \rho_{i-1} - \rho_{i-2} - \delta_{i-1} - \delta_{i-2}$, assuming $\alpha - \rho_{i-1} - \rho_{i-2} - \delta_{i-1} - \delta_{i-2} > 0.1$

Question: Isn't this granted by the fact that $\alpha > \alpha'' \geq \alpha - \rho_{i-1} - \rho_{i-2} - \delta_{i-1} - \delta_{i-2}$ and $\alpha > \frac{1}{3}$?? I think we do not need to put bounds on $\rho_{i-1}, \rho_{i-2}, \delta_{i-1}, \delta_{i-2}$....i should double check

we know for Corollary 2 that $\Gamma(X'') \geq 0.12n$ with overwhelming probability. Thus:

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

□

5 Different ways to commit to an encoding (ZigZag)

5.1 Encoding of the ZigZag graph

$\mathbb{F}$ is finite field and H hash function

Let $e_i^{(j)}$ be the label of node i in level j (i goes from 0 to $n-1$ and j from 0 to $\ell-1$). Remember that in a sector D we have n data blocks $D_0, D_1, \ldots, D_{n-1}$ (possibly sent to the prover by different users), and that we also have a vector commitment $\tau = \mathsf{Com}_D$ to D (*i.e.*, τ is the root of the Merkle tree constructed on the values $D_0, \ldots, D_{n-1}$).

Note: Com_D is computed by a preprocessor (a special party or the prover). A user can verify that its data block, *e.g.* D_i, is in the sector D asking for the inclusion path of D_i in Com_D.

Define $Parents_j(i)$ the vector of parents of the node i in level j written in topological order. The labels $e_i^{(j)}$ are computed in the following way:

$$e_i^{(0)} = D_i \oplus \mathsf{H}(\tau || Parents_0(i)) \tag{14}$$

$$e_i^{(j)} = \mathsf{Enc}(e_i^{(j-1)}) \oplus \mathsf{H}(\tau || Parents_j(i)) \quad j = 1, \ldots, \ell-1 \tag{15}$$

The vector of the labels of the last layers is called *Replica* and Com_R is a vector commitment to it (*i.e.*, the root of the Merkle tree constructed on the values $e_0^{(\ell-1)}, \ldots, e_{n-1}^{(\ell-1)}$). The value is used in the online phase!

Assumptions:

I assume that the same set of nodes is checked at any layer. Let d be the maximum out-degree in the ZigZag graph. I also assume that in our implementation we pad the nodes such that they all have d parents (*i.e.*, some parents are repeated).

5.2 Basic commitment scheme

Let Com_j be the root of the Merkle tree constructed on the values $e_0^{(j)}, e_1^{(j)}, \ldots, e_{n-1}^{(j)}$, and define the commitment to an encoding as $\mathsf{Com} = \mathsf{H}(\mathsf{Com}_D || \mathsf{Com}_0 || \ldots || \mathsf{Com}_{\ell-1})$. Note that $\mathsf{Com}_{\ell-1} = \mathsf{Com}_R$. To open the node i, reveal:

- the node: the label $e_i^{(j)}$ and its path to Com_j for all $j = 0, 1, \ldots, \ell-1$ and $\mathsf{H}(D_i)$ and its path to Com_D ($\ell+1$ labels, $\ell+1$ paths)
- the "even" parents (*i.e.* parents of the node i in the even layers): for $j = 0, 1, ..., (\ell-1)/2$ and for each $k \in Parents_{2j}(i)$, reveal $e_k^{(2j)}$ and its path in Com_{2j} ($d(\ell+1)/2$ labels, $d(\ell+1)/2$ paths)
- the "odd" parents: for $j = 0, 1, ..., (\ell-3)/2$ and for each $k \in Parents_{2j+1}(i)$, we reveal $e_k^{(2j+1)}$ and its path in Com_{2j+1} ($d(\ell-1)/2$ labels, $d(\ell-1)/2$ paths)

Total (for 1 challenge): $\ell(1+d)+1$ labels, $\ell(1+d)+1$ paths.

Checks:

1. all the inclusions paths
2. $\mathsf{H}(D_i) = \mathsf{Dec}(e_i^{(0)} \oplus \mathsf{H}(\tau||Parents_0(i)))$
3. $e_i^{(j-1)} = \mathsf{Dec}(e_i^{(j)} \oplus \mathsf{H}(\tau||Parents_j(i)))$ for $j = 1, \ldots, \ell - 1$

5.3 Column commitments (2 Merkle trees)

Idea: if we open the same nodes in each layer, we can commit by columns instead of by layers and therefore save inclusion paths that need to be revealed and checked!
Note: we still need Com_D (to prove that a block D_i is in the sector D) and Com_R (to execute the online phase of ZigZag PoRep). So, the column commitments contains values from level 0 to level $\ell - 2$ (second to last level).

Assume ℓ is odd (the other case is similar). For $i = 0, \ldots, n-1$, let $ho_i = \mathsf{H}(e_i^{(1)}||e_i^{(3)}||\ldots||e_i^{(\ell-2)})$ (labels in the odd layers only) and $he_i = \mathsf{H}(e_i^{(0)}||e_i^{(2)}||\ldots||e_i^{(\ell-3)})$ (labels in the even layers only) Define Com_o as the root of the Merkle tree constructed on the values $ho_0, ho_1, \ldots, ho_{n-1}$, and Com_e as the root of the Merkle tree constructed on the values $he_0, he_1, \ldots, he_{n-1}$. We define the commitment to an encoding as $\mathsf{Com} = \mathsf{H}(\mathsf{Com}_D||\mathsf{Com}_0||\mathsf{Com}_e||\mathsf{Com}_R)$. To open the node i, reveal:

- the node: the label $e_i^{(j)}$ for all $j = 0, \ldots, \ell - 1$ and the paths for ho_i to Com_o and for he_i to Com_e; moreover reveal the value $\mathsf{H}(D_i)$ and its path to Com_D and the inclusion path of $e_i^{(\ell-1)}$ to Com_R ($\ell + 1$ labels, 4 path)
- the "even" parents: for $j = 0, 2, ..., \ell - 3$ and for each $k \in Parents_j(i)$ reveal $e_k^{(j)}$ and the path for he_k to Com_e; for $k \in Parents_{\ell-1}(i)$ reveal $e_k^{(\ell-1)}$ and its path to Com_R ($d(\ell + 1)/2$ labels, $2d$ paths)
- the "odd" parents: for $j = 0, 1, ..., (\ell - 3)/2$ and for each $k \in Parents_{2j+1}(i)$ reveal $e_k^{(2j+1)}$ and the path for ho_k in Com_o ($d(\ell - 1)/2$ labels, d paths)

Total (for 1 challenge): $\ell(1 + d) + 1$ labels, $4 + 3d$ paths.

Check (same as before):

1. all the inclusions paths
2. $\mathsf{H}(D_i) = \mathsf{Dec}(e_i^{(0)} \oplus \mathsf{H}(\tau||Parents_0(i)))$
3. $e_i^{(j-1)} = \mathsf{Dec}(e_i^{(j)} \oplus \mathsf{H}(\tau||Parents_j(i)))$ for $j = 1, \ldots, \ell - 1$

5.4 ZigZag commitments (1 Merle tree)

Idea: use the following two facts to further reduce the inclusion paths needed.

Facts:

1. There is an expander edge $(i) \to (j)$ in the even layers if and only if there is an expander edge $(n - j + 1) \to (n - i + 1)$ in the odd layers.

2. There is a DRG edge $(i) \to (j)$ in the even layers if and only if there is a DRG edge $(n-i+1) \to (n-j+1)$ in the odd layers.

The ZigZag graph is constructed using a DRG graph and a expander graph. Let d_1 be the maximum out-degree in the DRG graph and d_2 be the maximum out-degree in the expander graph. Notice that $d = d_1 + d_2$.

Using fact 2: For $i = 0, 1, \ldots, n-1$, let DRG_i the vector containing the labels of node i for all even layers and the labels of the node $n-i+1$ (the reverse of node i) for all odd layer, and define $h'_i = \mathsf{H}(DRG_i)$. In other words, $h'_i = \mathsf{H}(e_i^{(0)}||e_{n-i+1}^{(1)}||e_i^{(2)}||e_{n-i+1}^{(3)}||\ldots||e_i^{(\ell-1)})$. Let $h_i = H(e_i^{(0)}||e_i^{(1)}||\ldots||e_i^{(\ell-2)})$, and define the commitment Com to an encoding as the root of the Merkle tree constructed on the values $h_0, h_1, \ldots, h_{n-1}, h'_0, h'_1, \ldots, h'_{n-1}$.

Work in progress!

Now opening the node i actually means open the node i for even layers and node $n-i+1$ for odd layers. For 1 opening, reveal:

- the node: the label $e_i^{(j)}$ for all $j = 0, \ldots, \ell-1$ and the paths for h_i for h'_i to Com; moreover reveal the value $\mathsf{H}(D_i)$ and its path to Com_D and the inclusion path of $e_i^{(\ell-1)}$ to Com_R ($\ell+1$ labels, 4 path)
- opening the DRG parents: for each j for each parent reveal the label $(e_k^{(2j+1)}, e_{n-k+1}^{(2j)})$ and the path for h'_k in com ($d_1\ell$ labels, d_1 paths)
- opening the expander parents: consider the union of even parent of node $n-i+1$ and odd parents of node i. For each one of them: reveal the labels $e_k^{(j)}$ and the inclusion path of h_k in com ($2d_2\ell$ labels, $2d_2$ paths).

Total (for 1 challenge): $\ell(1+d_1+2d_2)$ labels, $(1+d_1+2d_2)$ paths.

Comparison:

	Basic	Columns, 2tree	ZigZag, 1trees
labels	$\ell(1+d_1+d_2)+1$	$\ell(1+d_1+d_2)+1$	$\ell(1+d_1+2d_2)$
paths	$\ell(1+d_1+d_2)+1$	$4+3(d_1+d_2)$	$1+d_1+2d_2$ (??)

6 Attacks to non Interactive PoS

In this section we consider a setting were the initialization phase of a ZigZag proof is made non interactive using Fiat Shamir. In what follows we consider the following parameters

- n is the number of nodes in a single layer of a ZigZag graph
- ℓ is the number of layers in a ZigZag graph
- λ is the security parameter
- δn is the maximum number of incorrect labels that is tolerated in each layer

Moreover, we informally define *correct encoding, incorrect encoding* and λ-*soundness* as follows:

Correct Encoding: an encoding of a graph is said to be correct if each layer contains at most δn incorrect labels.

Incorrect Encoding: there exist at least one layer that has $\delta' n > \delta n$ incorrect labels.

λ-Soundness: If an encoding is incorrect, the probability that the proof is accepted is at most $\frac{1}{2^\lambda}$. This means that on average, an attacker needs 2^λ attempts to have an incorrect encoding which proof is accepted.

An Upper Bound on the Cost of the Attack

As we said above, if we have λ-soundness, a malicious prover must try (on average) 2^λ different incorrect encodings to find one that allows him to get one which proof is accepted. In order to evaluate the cost of this attack we explicitly evaluate the cost $\mathsf{C}_{\mathsf{IncEnc}}$ of computing an incorrect encoding, in terms of *number of hash evaluation* needed.

$$\mathsf{C}_{\mathsf{IncEnc}} = (1-\delta)n(\ell - \mathsf{IncLayers}) + (1-\delta')n \cdot \mathsf{IncLayers}$$

where $\mathsf{IncLayers}$ is the number of layer with $\delta' n > \delta n$ incorrect labels. **Note:** given $\delta' > \delta$, we have that $\mathsf{C}_{\mathsf{IncEnc}} < (1-\delta)n \cdot \ell$.

The total cost of the attack $\mathsf{C}_{\mathsf{Attack}}$ then consists in the product of the cost $\mathsf{C}_{\mathsf{IncEnc}}$ of producing a single incorrect encoding times the number of attempts $\mathsf{N}_{\mathsf{Att}}$ needed to the prover to produce a valid proof. We have

$$\mathsf{C}_{\mathsf{Attack}} = \mathsf{C}_{\mathsf{IncEnc}} \cdot \mathsf{N}_{\mathsf{Att}} < (1-\delta)n \cdot \ell \cdot 2^\lambda$$

What does it mean in Practice? Assuming $n = 2^{30}, \ell < 2^5$ and $\lambda = 8$, the cost of the attack is less that 2^{43} hash evaluations. Moreover, if the attacker re-encode only the last layer (meaning all but the last one are encoded correctly), this cost decreases to less that 2^{38} hash evaluations.

A Lower Bound on the Cost of the Attack, when Re-Encoding one Layer

As explained above, a maliciuos prover can try to find an incorrect encoding allowing him to pass the initialization by tampering only with the last layer. Notably, this seems to be the less expensive strategy, and then the best one on the attacker's side. Similarly to what we explained in the former

paragraph, the cost of re-encoding the last layer in terms of number of hash evaluations, assuming $\delta' n > \delta n$ incorrect labels is

$$\mathsf{C}_{\mathsf{1IncEnc}} = (1-\delta')\frac{n}{k}.$$

Here we are assuming that we set to zero $\delta' n$ nodes (the incorrect ones), and once this is done then the labels for the remaining $(1-\delta')n$ nodes can be computed in groups of size k simultaneously. Note that the worst case in terms of security (that is, the less expensive scenario for an attacker) happens when $k = n$, while the best optimal case consists in $k = 1$ (labels can not be computed in parallel).

In therms of probability of success, the probability that an incorrect encoding which consists in the correct encoding of the first $\ell - 1$ layers and an incorrect encoding of the last layer passes the initialization phase when we have c independent challenges for the last layer is $p = (1-\delta')^c$ (which means that an attacker needs $\frac{1}{p} = (1-\delta')^{-c}$ attempts in order to find an incorrect encoding of the last layer which passes the initialization. Remember that due to λ soundness this probability has to be $\frac{1}{2^\lambda}$). If we call this attack Attack^+, then its cost is

$$\mathsf{C}_{\mathsf{Attack}^+} = \mathsf{C}_{\mathsf{1IncEnc}} \cdot \mathsf{N}_{\mathsf{Att}} = \mathsf{C}_{\mathsf{1IncEnc}} \cdot \frac{1}{p} = (1-\delta') \cdot (1-\delta')^{-c} \cdot \frac{n}{k} = (1-\delta')^{1-c} \cdot \frac{n}{k}$$

Since $\delta' > \delta$, we trivially have that $(1-\delta) > (1-\delta')$ and then $(1-\delta)^{1-c} < (1-\delta')^{1-c}$. It follows that

$$\mathsf{C}_{\mathsf{Attack}^+} > (1-\delta)^{1-c} \cdot \frac{n}{k} = (1-\delta) \cdot (1-\delta)^{-c} \cdot \frac{n}{k} = (1-\delta) \cdot \frac{1}{p} \cdot \frac{n}{k} = 2^\lambda \cdot (1-\delta) \cdot \frac{n}{k}$$

What does it mean in Practice? Assuming $n = 2^{30}$, if we want that the cost of the improved attack $\mathsf{C}_{\mathsf{Attack}^+}$ is at least 2^{80} hash evaluations, then we need

$$\begin{aligned}
&2^\lambda \cdot (1-\delta) \cdot \frac{n}{k} > 2^{80} \\
&2^\lambda \cdot (1-\delta) \cdot \frac{2^{30}}{k} > 2^{80} \\
&(1-\delta) \cdot \frac{1}{k} > 2^{80-30-\lambda} \\
&log_2((1-\delta) \cdot \frac{1}{k}) > log_2(2^{80-30-\lambda}) \\
&\lambda > 50 - log_2(1-\delta) + log_2(k).
\end{aligned}$$

ZigZag Suffers Attack^+ with $k = n$

Consider a ZigZag graph where the last layer is odd and all edges go from lower indices nodes from higher indices nodes. A cheating prover wants to to perform Attack^+. compute the labels first $(1-\delta')n$ nodes (meaning, the $(1-\delta')n$ nodes with lower indices) correctly, and try the grinding attack the last $\delta' n$ labels. Note that in this setting, due to the topology of the ZigZag graph, none of the last $\delta' n$ nodes is parent of any of the first $(1-\delta')n$ ones. This means that for each attempt of the cheating prover, none of the first $(1-\delta')n$ labels needs to be re-computed. Moreover, note that the cheating prover could also decide to set the first $\delta' n - 1$ incorrect labels to zero and try

to win only modifying the last one. This means that in this case each attempt cost only 1 hash evaluation).

$$| - - - (1-\delta')n \textit{ correct } || - - - 000 \textit{ all zeros } 000 || - - - H(\textit{last ones}) - - - |$$

What does it mean in Practice? The probability of success of the attack is still $p = (1-\delta')^c$, so the number of attempts needed is $\frac{1}{p} = (1-\delta')^{-c}$.

As we saw before, since $\delta' > \delta$, we have that $(1-\delta')^{-c} > (1-\delta)^{-c}$. Moreover, due to λ soundness, we have $(1-\delta)^{-c} = 2^\lambda$. This means that the total cost of the attack is

$$\mathsf{C}_{\mathsf{ZigZag}} = (1-\delta')^{-c} > 2^\lambda$$

Moreover, if we want to express $c = c(\lambda, \delta)$ as a functions of λ and δ (assuming no tapering), it is sufficient to observe that

$$(1-\delta)^{-c} = 2^\lambda \Rightarrow log_2((1-\delta)^{-c}) = \lambda \Rightarrow -c \cdot log_2(1-\delta) = \lambda \Rightarrow -c = \frac{\lambda}{log_2(1-\delta)}$$

Which implies

$$\mathsf{C}_{\mathsf{ZigZag}} = (1-\delta')^{\frac{\lambda}{log_2(1-\delta)}}$$

Examples:

- If $\lambda = 80$ and $\delta = 0.01$ we have $\mathsf{C}_{\mathsf{ZigZag}} = (1-\delta')^{-5518}$.
  Considering $\delta' = 2 \cdot \delta$ we have $\mathsf{C}_{\mathsf{ZigZag}} = 2^{160}$ hash evaluations
- If $\lambda = 80$ and $\delta = 0.1$ we have $\mathsf{C}_{\mathsf{ZigZag}} = (1-\delta')^{-527}$
  Considering $\delta' = 2 \cdot \delta$ we have $\mathsf{C}_{\mathsf{ZigZag}} = 2^{170}$ hash evaluations