The Pebbling Complexity of Depth-Robust Graphs

Joël Alwen — IST Austria
Jeremiah Blocki — Purdue University
Krzysztof Pietrzak — IST Austria

September 2, 2016

Abstract

There is growing interest in the security community in functions which are moderately hard to compute on a normal single-processor machine, but which cannot be computed at a significantly lower (hardware/energy) cost on dedicated hardware. Such functions are required for password-hashing to prevent brute-force attacks implemented on custom circuits or in proofs-of-work for decentralized cryptocurrencies.

Towards this goal memory-hard functions (MHF) have been suggested, leveraging the fact that – unlike computation – memory cost on custom hardware is not cheaper than on general architectures. MHFs come in two flavors, data-dependent and data independent MHFs, the former are potentially easier to construct, but they succumb to side-channel attacks.

Data independent MHFs can be specified by a directed acyclic graph (DAG) G_n on n nodes of constant indegree and a single sink, the function is then defined as the label of the sink, where a label of a node is computed as the hash of the labels of its parents and the function input. The quality of the memory-hard function is captured by two parameters on the pebbling complexity of the underlying graph:

- The parallel cumulative pebbling complexity $\Pi^{\parallel}_{cc}(G_n)$ must be as high as possible (to ensure that the amortized cost of computing the function on dedicated hardware is dominated by the cost of memory).
- The sequential space-time pebbling complexity $\Pi_{st}(G_n)$ should be as close as possible to $\Pi^{\parallel}_{cc}(G_n)$ (to ensure that using many cores in parallel and amortizing over many instances does not give much of an advantage).

In this paper we construct a family of DAGs with best possible parameters, i.e., where $\Pi^{\parallel}_{cc}(G_n) = \Omega(n^2/\log(n))$ (which matches a known upper bound) and $\Pi_{st}(G_n)$ is within a constant factor of $\Pi^{\parallel}_{cc}(G_n)$.

Our analysis relies on a new connection between depth-robustness (DR) of DAGs – a well studied combinatorial property – and their pebbling complexities. We show that high DR is *sufficient* for high $\Pi^{\parallel}_{cc}$. This contrasts the recent results of Alwen and Blocki (ePrint/2016/115) showing that high DR is *necessary*. Together these results fully characterizes DAGs with high $\Pi^{\parallel}_{cc}$ in terms of DR. Along the way we also give a new tool for reducing the indegree of a DAG without loosing too much in $\Pi^{\parallel}_{cc}$ which may be of interest beyond this work.

Previous to our work the graphs with best asymptotic complexity were due to Alwen and Serbinenko (STOC'15) achieve $\Pi^{\parallel}_{cc}(G_n) \in \Omega(n^2/\log^{10}(n))$. Most of the constructions used in practice have recently been broken in strong sense, the graph underlying Arogon2i, the winner of the recent password hashing competition, has $\Pi^{\parallel}_{cc}$ complexity $\tilde{O}(n^{1.75})$, for the recently proposed Balloon-Hashing the upper bound is $O(n^{1.67})$ and the same bound holds for Catena. We prove a $\tilde{\Omega}(n^{5/3})$ lower bound on $\Pi^{\parallel}_{cc}$ for Argon2i, and a $\tilde{\Omega}(n^{1.5})$ lower bound for Catena and Balloon hashing. The lower bound for Argon2i leverages our connection between $\Pi^{\parallel}_{cc}$ and depth-robustness. To prove the lower bounds for Catena and Balloon hashing we identify a graph property dubbed "dispersity", and show how it also lower bounds $\Pi^{\parallel}_{cc}$. We also present an improved recursive version of the pebbling attack of Alwen and Blocki (ePrint/2016/115) and show that it implies upper bounds of $O\left(n^{1.71}\right)$ and $O(n^{1.62})$ for Argon2i and Catena respectively.

JoëlredCurrently the abstract does mention our new attack. IMO this is at least as interesting as our lowerbounds since the lowerbounds are weak but the attack is strong relative to the n^2 complexity

people were aiming for with those constructions. So if possible it would be good to include this. Maybe under the theme: results for known iMHFs. New lower and upper bounds. (Not to mention that most likely one can apply this attack to many other candidate iMHFs to get the best known attacks for them too.)

1 Introduction

The Black Pebbling Game First introduced by Hewitt and Paterson [HP70] and Cook [Coo73] the (sequential) black pebbling game (and its relatives) have been used to great effect in the theoretical computer science. An early area of investigation for which they proved to be particularly well suited focused on space/time trade-offs for various computational tasks such as matrix multiplication [Tom78], the FFT [SS78, Tom78], integer multiplication [SS79b] and solving linear recursions [Cha73, SS79a]. More recently, pebbling has been extensively used for various cryptographic applications including proofs of space [DFKP15, RD16], proofs of work [DNW05, MMV13], leakage-resilient cryptography [DKW11], memory-bound functions [DGN03] and memory-hard functions [AS15]. It's also an active research topic in proof complexity (cf. The survey on `http://www.csc.kth.se/~jakobn/research/PebblingSurveyTMP.pdf`).

The black pebbling game is played over a fixed directed acyclic graph (DAG) $G = (V, E)$ in rounds. The goal of the game is to pebble all sink nodes of G (not necessarily simultaneously). Each round $i \geq 1$ is characterized by its pebbling configuration $P_i \subseteq V$ which denotes the set of currently pebbled nodes. Initially all nodes are unpebbled $P_0 = \emptyset$. In each round $i \geq 1$ initially we set $P_i := P_{i-1}$. Then P_i can modified arbitrarily according to two simple rules. (1) A node v may be pebbled (added to P_i) if, in the previous configuration all of its parents were pebbled $\mathsf{parents}(v) \subseteq P_{i-1}$. (2) Any pebble can always be removed from P_i. In the sequential version rule (1) may be applied at most once per round while in the parallel version no such restriction applies. A sequence of configurations $P = (P_0, P_1, \ldots)$ is a (sequential) pebbling of G if it adheres to these rules and each sink node of G is contained in at least one configuration.

In this paper we investigate upper and lower bounds on various pebbling complexities of graphs, as they can be related to the cost of evaluating the "labeling function" f_G (to be defined below) in various computational models. In particular, let $\mathcal{P}_G$ and $\mathcal{P}_G^{\parallel}$ denote all valid sequential and parallel pebblings of G, respectively. We are interested in the *parallel* cumulative pebbling complexity of G, denoted $\Pi_{cc}^{\parallel}(G)$, and the sequential space-time complexity of G, denoted $\Pi_{st}(G)$, which are defined as

$$\Pi_{cc}^{\parallel}(G) = \min_{(P_1,\ldots,P_t)\in\mathcal{P}_G^{\parallel}} \sum_{i=1}^{t} |P_i| \qquad \Pi_{st}(G) = \min_{(P_1,\ldots,P_t)\in\mathcal{P}_G} t \cdot \max_i(|P_i|)$$

The main result of this paper is a family of graphs with high (in fact, as we'll discuss below, maximum possible) $\Pi_{cc}^{\parallel}$ complexity, and where the Π_{st} complexity is not much higher than the $\Pi_{cc}^{\parallel}$ complexity.[1] Throughout, we'll denote with $\mathbb{G}_n$ the set of all DAGs on n vertices and with $\mathbb{G}_{n,d} \subseteq \mathbb{G}_n$ the DAGs where each vertex has indegree at most d.

Theorem 1.1 *There exists a family of DAGs $\{G_n \in \mathbb{G}_{n,2}\}_{n\in\mathbb{N}}$ where*

1. *parallel cumulative pebbling complexity*

$$\Pi_{cc}^{\parallel}(G_n) \in \Omega(n^2/\log(n))$$

2. *and where the sequential space-time complexity matches the parallel cumulative pebbling complexity up to a constant*

$$\Pi_{st}(G_n) \in O(n^2/\log(n))$$

[1]Note that $\Pi_{cc}^{\parallel}(G) \leq \Pi_{st}(G)$ as parallelism can only help, and space-time complexity (i.e., number of rounds times the size of the largest state) is always higher than cumulative complexity (the sum of the sizes of all states).

The lower bound on $\Pi^{\|}_{cc}$ in item *1.* above is basically optimal due to the following bound from [AB16]. [2]

Theorem 1.2 ([AB16, Thm. 6.3]) *For any constant $\epsilon > 0$ and sequence of DAGs $\{G_n \in \mathbb{G}_{n,\delta_n}\}_{n\in\mathbb{N}}$ it holds that*

$$\Pi^{\|}_{cc}(G_n) = o\left(\frac{\delta_n n^2}{\log^{1-\epsilon}}\right).$$

In particular if $\delta_n = O(\log^{1-\epsilon})$ then $\Pi^{\|}_{cc}(G_n) = o(n^2)$, and

$$\text{if } \delta_n = \Theta(1) \text{ then } \Pi^{\|}_{cc}(G_n) = o(n^2/\log^{1-\epsilon}(n)) \tag{1}$$

JoëlredGiven the importance of Argon2i I consider the new recursive attack on Argon2i also to be a main result. As for the lowerbounds I think it's fair to say these are only secondary but mainly just because they show a much weaker bound then people had hoped for. Still the community has been looking for quite some time now for security proofs for those (or any really) algorithms too so its fair to say we solve a well known open problem with them (or at least make a lot of progress in a known direction). After all finding these type of results was a major goal of having the password hashing competition. Its just that the result is not what people had hoped for. (Also I bet the new attack will eventually be used to attack various other iMHF candidates from the PHC. At the very least in its most high level form. But maybe even for the class of functions f in Theorem 6.10 given that many of them seem to have the same distribution of lengths of back edges as Argon2i (namely uniform).)

Moderately hard functions. Functions which are "moderately" hard to compute have found many applications including password hashing, key-derivation and for proofs of work. In the context of password hashing, the goal is to minimize the damage done by a security breach where an adversary learns the password file; Instead of storing $(login, password)$ tuples in the clear, one picks a random salt and stores a tuple $(login, f(password, salt), salt)$, where $f(.)$ is a moderately hard function $f(.)$. This comes at a price, the server verifying a password must evaluate $f(.)$, which thus cannot be too hard. On the other hand, if a tuple $(login, y, salt)$ is leaked, an adversary who tries to find the password by a dictionary attack must evaluate $f(.)$ for every attempt. A popular moderately hard function is PBKDF2 (Password Based Key Derivation Function 2) [Kal00], which basically just iterates a cryptographic hash function H several times (1024 is a typical value).

Unfortunately a moderately hard function like PBKDF2 is insufficient to protect against adversaries who can build customized hardware to evaluate the underlying hash function. In particular, the cost of computing a hash function H like SHA256 or MD5 on an ASIC (Application Specific Integrated Circuit) is orders of magnitude smaller than the cost of computing H on traditional hardware [DGN03, NB+15].

JoëlredGiven the recent debate about making Crypto research "more moral" by focusing on privacy it might be nice to slip in a small sentence or even just some key phrase somewhere hinting that our work fits very well in that direction. If we're lucky that could by us points with a more activist / privacy minded reviewer. Indeed in Phil's Asiacrypt invited lecture MHFs were one of the examples he used to show what he feels crypto work be focused on.

Memory-Bound and Memory-Hard Functions. [ABW03] recognized that cache-misses are more egalitarian than computation, in the sense that they cost about the same on different architectures. They propose "memory-bound" functions, which are functions that will incur many expensive cache-misses, this idea was further developed by [DGN03].

Along similar lines, Percival [Per09] observes that unlike computation, memory costs tend to be relatively stable across different architectures, and suggests to use memory-hard functions (MHF) for password hashing. MHFs come in two flavours, data-dependent MHFs (dMHF) and data independent MHFs (iMHF), the former are potentially easier to construct, but they leave open the possibility of side-channel attacks, thus iMHFs are preferable.

[2] The statement bellow is obtained from the result in [AB16] by treating the core-memory ratio as a constant and observing that, trivially, at most n pebbles are on G during a balloon phase and at most n pebbles are placed in one step during a balloon phase.

iMHF as Graphs. An iMHF comes with an algorithm that computes the function using a fixed memory access pattern. In particular the pattern is independent of the input. Such functions can thus be described by a directed acyclic graph (DAG) G, where each node v of the graph corresponds to some intermediate value ℓ_v that appears during the computation of the function, and the edges capture the computation: if ℓ_v is a function of previously computed values $\ell_{i_1}, \ldots, \ell_{i_\delta}$, then the nodes $i_1, \ldots, i_\delta$ are parents of v in G. For an iMHF F, we'll denote with $G(F)$ the underlying graph. For example $G(\mathsf{PBKDF2})$ is simply a path.

The labeling function. Not only can an iMHF be captured by a graph as just outlined, we will actually construct iMHFs by first specifying a graph, and then defining a "labeling function" on top of it: Given a graph G with vertex set $V = [n]$, a hash function $H : \{0,1\}^* \to \{0,1\}^w$ and some input x define the labeling of the vertices of G as follows: every vertex i with parents $i_1 < i_2 < \cdots < i_\delta$ has label $\ell_I(x) = H(i, \ell_{i_1}(x), \ldots, \ell_{i_\delta}(x))$, in particular, if i is a source then $\ell_i = H(i, x)$. For a DAG G with a unique sink s we define the labeling function of G as $f_G(x) = \ell_s(x)$. Note that using the convention from the previous paragraph, we have $G(f_G) = G$.

Pebbling vs. Memory-hardness. The reason to focus on the graph $G = G(F)$ underlying an iMHF F is that clean combinatorial properties of G – i.e., bounds on the pebbling complexities – imply both upper and lower bounds on the cost of evaluating F in various computational settings. For upper bounds (i.e., attacks), no further assumption on F are required to make the transition from pebbling to computation cost. For lower bounds, we have to assume that there's no "shortcut" in computing F, and the only way is to follow the evaluation sequence as given by G. Given the current state of complexity theory, where not even superlinear lower bounds on evaluating any function in $\mathcal{NP}$ are known, we cannot hope to exclude such shortcuts unconditionally. Instead, we assume that the underlying hash function H is a random oracle and circuits are charged unit cost for queries to the random oracle.

For our lower bounds, we must insist on G having constant indegree. The reason is that in reality H must be instantiated with some cryptographic hash function like SHA1, and the indegree corresponds to the input length on which H is invoked. To evaluate H on long inputs, one would typically use some iterated construction like Merkle-Damgard, and the assumption that H behaves like a black-box that can only be queried once the entire input is known would be simply wrong in this case.

As advocated in [AS15], bounds on $\Pi^{\parallel}_{cc}(G)$ are a good approximation for the cost of evaluating f_G in dedicated hardware, whereas a bound on $\Pi_{st}(G)$ gives an upper bound on the cost of evaluating f_G on a single processor machine. The reason [AS15] consider cumulative complexity for lower and space-time complexity for the upper bound is that when lower bounding the cost of evaluating f_G we do want to allow for amortization of the cost over arbitrary many instances,[3] whereas for our upper bound we don't want to make such an assumption. The reason we consider parallel complexity for the lower and only sequential for the upper bound is due to the fact that an adversary can put many (cheap) cores computing H on dedicated hardware, whereas for the upper bound we only want to consider a single processor machine.

If $\Pi^{\parallel}_{cc}(G)$ is sufficiently larger than $|V(G)|$ (in Theorem 1.1 it's almost quadratic), then the cost of evaluating f_G in dedicated hardware is dominated by the memory cost. As memory cost about the same on dedicated hardware and general purpose processors, if our G additionally satisfies $\Pi^{\parallel}_{cc}(G) \approx \Pi_{st}(C)$, then we get a function f_G whose evaluation on dedicated hardware is not much cheaper than evaluating it on an off the shelf machine (like a single core x86 architecture). This is exactly what the family from Theorem 1.1 achieves. We elaborate on these computational models and how they are related to pebbling in Appendix A.

Previous to this work, the construction with the best asymptotic bounds was due to [AS15] and achieved $\Pi^{\parallel}_{cc}(G_n) \in \Omega(n^2/\log^{10}(n))$, the exponent 10 makes this bound uninteresting for practical parameters. [BK15] have broken the most important iMHFs in a rather strong sense: the graph underlying Argon2i [BDK16], the winner of the recent password hashing competition, has $\Pi^{\parallel}_{cc}$ complexity $O(n^{1.75})$, for the recently proposed Balloon-Hashing [CGBS16] the upper bound is $O(n^{1.67})$ and the same bound holds for Catena [FLW13]. We'll

[3]$\Pi^{\parallel}_{cc}$ satisfies a direct product property: pebbling k copies of G cost k times as much as pebbling G, i.e., $\Pi^{\parallel}_{cc}(G^k) = k \cdot \Pi^{\parallel}_{cc}(G)$, but this is not true for Π_{st} complexity.

discuss this in more detail in Section 6, where we further improve previous attacks [BK15, AS15, AB16] to $O(n^{1.708})$ (resp. $O(n^{1.62})$) for Argon2i (resp. Catena) and also provide the first lower bounds $\tilde{\Omega}(n^{1.5})$ for Balloon-Hashing and Catena and a slightly better $\tilde{\Omega}(n^{5/3})$ for Argon2i.

Unfortunately, these bounds are insufficient for practical iMHFs for two reasons.

1. The best known sequential algorithms for these functions have much higher complexity than their parallel counterparts, i.e., whereas the lower bounds give $\Pi^{\parallel}_{cc} \in \tilde{\Omega}(n^{5/3})$, for the upper bounds in the sequential case we can't say anything beyond the trivial $\Pi_{st} \in O(n^2)$ bound.[4] Thus, parallelism seems to help a lot in computing these functions.

2. But even if it turns out that a surprising sequential algorithm exists where $\Pi^{\parallel}_{cc} \approx \Pi_{st}$, our improved upper bounds imply that the best we can hope for is $\Pi_{st} \approx n^{1.708}$. This means that for every invocation of the underlying hash function H, we have to store at least $n^{1.708}/n = n^{0.708}$ values (for the time required to evaluate the underlying hash function). For the function to be memory hard, the area on a chip required to store $n^{0.708}$ hash values should (at minimum) be at least as large as the area required to implement H , a typical value for his ratio is 3000 (i.e., storing 3000 values is as costly as computing H once). For $n^{0.708}$ to be ≥ 3000 we need $n \geq 80000$, which is often much larger than what is used for applications like password hashing [Kal00].

Depth-Robust Graphs. The results in this work rely on a new connection between the depth-robustness of a DAG and its $\Pi^{\parallel}_{cc}$ complexity. A DAG G is (e, d)-depth-robust if, after removing any subset of at most e nodes their remains a directed path of length at least d. First investigated by Erdös, Graham and Szemerédi [EGS75], several such graphs enjoying low indegree and increasingly extreme depth-robustness have been constructed in the past [EGS75, PR80, Sch82, Sch83, MMV13] mainly in the context of proving lower-bounds on circuit complexity and turing machine time. Depth-robustness has been used as a key tool in the construction of cryptographic objects like proofs of sequential work [MMV13] and memory-hard functions [AS15].

Depth-Robustness and $\Pi^{\parallel}_{cc}$. While the flavour of the results in this work are related to those of [AS15] the techniques are rather different. As mentioned above already, they stem from a new tight connection between depth-robustness and $\Pi^{\parallel}_{cc}$. A special case of this connection shows that if G is (e, d)-depth-robust, then its $\Pi^{\parallel}_{cc}$ can be lower bounded as

$$\Pi^{\parallel}_{cc}(G) \geq e \cdot d \ .$$

This complements a result from [AB16], who give a pebbling strategy which is efficient for graphs of low depth-robustness (we give the exact statement in Theorem 2.5 below), thus a DAG has high $\Pi^{\parallel}_{cc}$ if and only if it is very depth-robust.

Moreover, we give a new tool for reducing the indegree of a DAG while not reducing the $\Pi^{\parallel}_{cc}$ of the resulting graph (in terms of its size). Together these results directly have some interesting consequences

- The family of DAGs $\{G_n \in \mathbb{G}_{n,\log(n)}\}_{n\in\mathbb{N}}$ from Erdös et al. [EGS75] have optimally high $\Pi^{\parallel}_{cc}(G_n) \in \Omega(n^2)$.
- Using our indegree reduction we can turn the above family of $\log(n)$ indegree into a family of indegree 2 DAGs $\{G'_n \in \mathbb{G}_{(n,2)}\}_{n\in\mathbb{N}}$ with $\Pi^{\parallel}_{cc}(G'_n) \in \Omega(n^2/\log(n))$, which by Theorem 1.2 is optimal for constant indegree graphs.

The last item above almost gives a family as in our Theorem 1.1, except that we can't prove any nontrivial $o(n^2)$ bounds on the sequential space-time pebbling complexity $\Pi_{st}(G'_n)$ for these graphs,[5] whereas for

[4]And this holds even if we allow amortization but no parallelism, i.e., the best we can say is $\Pi_{cc} \leq \Pi_{st} \in O(n^2)$

[5]The problem here is the parallel vs. sequential issue, not space-time versus cumulative. We can't even prove nontrivial bounds $o(n^2)$ on the sequential cumulative pebbling complexity $\Pi_{cc}(G'_n)$.

constructing memory hard functions we want DAGs where $\Pi_{st}(G) \approx \Pi_{cc}^{\parallel}(G)$, ideally $\Pi_{st}(G) \in O(\Pi_{cc}^{\parallel}(G))$. We achieve such a family by stacking the Erdös et al. graphs and carefully interconnecting the layers.

2 Pebbling Complexities and Depth-Robustness of Graphs

We begin by fixing some common notation. We use the sets $\mathbb{N} = \{0, 1, 2, \ldots\}$, $\mathbb{N}^+ = \{1, 2, \ldots\}$, and $\mathbb{N}_{\geq c} = \{c, c+1, c+2, \ldots\}$ for $c \in \mathbb{N}$. Further, we also use the sets $[c] := \{1, 2, \ldots, c\}$ and $[b, c] = \{b, b+1, \ldots, c\}$ where $b \in \mathbb{N}$ with $b \leq c$. For a set of sets $A = \{B_1, B_2, \ldots, B_z\}$ we use the notation $||A|| := \sum_i |B_i|$.

2.1 Depth-Robust Graphs

We say that a directed acyclic graph (DAG) $G = (V, E)$ has *size* n if $|V| = n$. A node $v \in V$ has indegree $\delta = \mathsf{indeg}(v)$ if there exist δ incoming edges $\delta = |(V \times \{v\}) \cap E|$. More generally, we say that G has indegree $\delta = \mathsf{indeg}(G)$ if the maximum indegree of any node of G is δ. A node with indegree 0 is called a source node and one with no outgoing edges is called a sink. We use $\mathsf{parents}_G(v) = \{u \in V : (u, v) \in E\}$ to denote the parents of a node $v \in V$. In general, we use $\mathsf{ancestors}_G(v) = \bigcup_{i \geq 1} \mathsf{parents}_G^i(v)$ to denote the set of all ancestors of v — here, $\mathsf{parents}_G^2(v) = \mathsf{parents}_G(\mathsf{parents}_G(v))$ denotes the grandparents of v and $\mathsf{parents}^{i+1}(v) = \mathsf{parents}_G\left(\mathsf{parents}^i(G)\right)$. When G is clear from context we will simply write $\mathsf{parents}$ ($\mathsf{ancestors}$). We denote the set of all sinks of G with $\mathsf{sinks}(G) = \{v \in V : \nexists(v, u) \in E\}$ — note that $\mathsf{ancestors}(\mathsf{sinks}(G)) = V$. We often consider the set of all DAGs of equal size $\mathbb{G}_n = \{G = (V, E) \ : \ |V| = n\}$ and often will bound the maximum indegree $\mathbb{G}_{n,\delta} = \{G \in \mathbb{G}_n : \mathsf{indeg}(G) \leq \delta\}$. For directed path $p = (v_1, v_2, \ldots, v_z)$ in G its length is the number of nodes it traverses $\mathsf{length}(p) := z$. The depth $d = \mathsf{depth}(G)$ of DAG G is the length of the longest directed path in G.

We will often consider graphs obtained from other graphs by removing subsets of nodes. Therefor if $S \subset V$ then we denote by $G - S$ the DAG obtained from G by removing nodes S and incident edges. The following is a central definition to our work.

Definition 2.1 (Depth-Robustness) *For $n \in \mathbb{N}$ and $e, d \in [n]$ a DAG $G = (V, E)$ is (e, d)-depth-robust if*

$$\forall S \subset V \quad |S| \leq e \Rightarrow \mathsf{depth}(G - S) \geq d.$$

We will make use of the following lemma due to Erdös, Graham and Szemerédi [EGS75], who showed how to construct a family of log indegree DAGs with extreme depth-robustness.

Theorem 2.2 ([EGS75]) *For some fixed constants $c_1, c_2, c_3 > 0$ there exists an infinite family of DAGs $\{G_n \in \mathbb{G}_{n, c_3 \log(n)}\}_{n=1}^{\infty}$ such that G_n is $(c_1 n, c_2 n)$-depth-robust.*

2.2 Graph Pebbling

We fix our notation for the parallel graph pebbling game following [AS15].

Definition 2.3 (Parallel/Sequential Graph Pebbling) *Let $G = (V, E)$ be a DAG and let $T \subseteq V$ be a target set of nodes to be pebbled. A* pebbling configuration *(of G) is a subset $P_i \subseteq V$. A legal* parallel *pebbling of T is a sequence $P = (P_0, \ldots, P_t)$ of pebbling configurations of G where $P_0 = \emptyset$ and which satisfies conditions 1 & 2 below. A* sequential *pebbling additionally must satisfy condition 3.*

1. *At some step every target node is pebbled (though not necessarily simultaneously).*

$$\forall x \in T \ \exists z \leq t \quad : \quad x \in P_z.$$

2. *Pebbles are added only when their predecessors already have a pebble at the end of the previous step.*

$$\forall i \in [t] \quad : \quad x \in (P_i \setminus P_{i-1}) \ \Rightarrow \ \mathsf{parents}(x) \subseteq P_{i-1}.$$

3. At most one pebble placed per step.

$$\forall i \in [t] \quad : \quad |P_i \setminus P_{i-1}| \leq 1 \ .$$

We denote with $\mathcal{P}_{G,T}$ and $\mathcal{P}^{\parallel}_{G,T}$ the set of all legal sequential and parallel pebblings of G with target set T, respectively. Note that $\mathcal{P}_{G,T} \subseteq \mathcal{P}^{\parallel}_{G,T}$. We will be mostly interested in the case where $T = \mathsf{sinks}(G)$ and then will simply write $\mathcal{P}_G$ and $\mathcal{P}^{\parallel}_G$.

Definition 2.4 (Time/Space/Cumulative Pebbling Complexity) *The space, time, space-time and cumulative complexity of a pebbling $P = \{P_0, \ldots, P_t\} \in \mathcal{P}^{\parallel}_G$ are defined as*

$$\Pi_t(P) = t \ , \qquad \Pi_s(P) = \max_{i \in [t]} |P_i| \ , \qquad \Pi_{st}(P) = \Pi_t(P) \cdot \Pi_s(P) \qquad \text{and} \qquad \Pi_{cc}(P) = \sum_{i \in [t]} |P_i| \ .$$

For $\alpha \in \{s, t, st, cc\}$ and a target set $T \subseteq V$, the sequential and parallel pebbling complexities of G are defined as

$$\Pi_\alpha(G,T) = \min_{P \in \mathcal{P}_{G,T}} \Pi_\alpha(P) \qquad \text{and} \qquad \Pi^{\parallel}_\alpha(G,T) = \min_{P \in \mathcal{P}^{\parallel}_{G,T}} \Pi_\alpha(P) \ .$$

When $T = \mathsf{sinks}(G)$ we simplify notation and write $\Pi_\alpha(G)$ and $\Pi^{\parallel}_\alpha(G)$.

It follows from the definition that for $\alpha \in \{s, t, st, cc\}$ and any G the parallel pebbling complexity is always at most as high as the sequential, i.e., $\Pi_\alpha(G) \geq \Pi^{\parallel}_\alpha(G)$, and cumulative complexity is at most as high as space-time complexity, i.e., $\Pi_{st}(G) \geq \Pi_{cc}(G)$ and $\Pi^{\parallel}_{st}(G) \geq \Pi^{\parallel}_{cc}(G)$.

In this work we will consider constant in-degree DAGs $\{G_n \in \mathbb{G}_{n,\Theta(1)}\}_{n \in \mathbb{N}}$, and will be interested in the complexities $\Pi_{st}(G_n)$ and $\Pi^{\parallel}_{cc}(G_n)$ as these will capture the cost of evaluating the labelling function derived from G_n on a single processor machine (e.g. a x86 processor on password server) and amortized AT complexity (which is a good measure for the cost of evaluating the function on dedicated hardware), respectively.

Before we state our main theorem let us observe some simple facts. Every n-vertex graph can be pebbled in n steps, and we cannot have more than n pebbles on a n vertex graph, thus

$$\forall G_n \in \mathbb{G}_n \ : \ \Pi^{\parallel}_{cc}(G_n) \ \leq \ \Pi_{st}(G_n) \ \leq \ n^2$$

This upper bound is basically matched for the complete graph $K_n = (V = [n], E = \{(i,j) \ : 1 \leq i < j \leq n\})$

$$n(n-1)/2 \ \leq \ \Pi^{\parallel}_{cc}(K_n) \ \leq \ \Pi_{st}(K_n) \ \leq \ n^2$$

K_n has the desirable properties that its Π_{st} is within a constant factor to its $\Pi^{\parallel}_{cc}$ complexity and moreover its $\Pi^{\parallel}_{cc}$ complexity is maximally high. Unfortunately, K_n has very high indegree, which makes it useless for our purpose to construct memory-hard functions. The path $Q_n = (V = [n], E = \{(i, i+1) : 1 \leq i \leq n-1\})$ on the other hand has indegree 1 and its Π_{st} is even exactly as large as its $\Pi^{\parallel}_{cc}$ complexity. Unfortunately it has very low pebbling complexity

$$\Pi^{\parallel}_{cc}(Q_n) \ = \ \Pi_{st}(Q_n) \ = \ n$$

which means that in the labelling function we get from Q_n (which is basically PBKDF2 discussed in the introduction) the evaluation cost will not be dominated by the memory cost even for large n. As stated in Theorem 1.1, in this paper we construct a family of graphs $\{G_n \in \mathbb{G}_{n,2}\}_{n \in \mathbb{N}}$ which satisfies all three properties at once: (1) the graphs have indegree 2 (2) the parallel cumulative pebbling complexity is $\Pi^{\parallel}_{cc}(G_n) \in \Omega(n^2/\log(n))$, which by by Theorem 1.2 is optimal for constant indegree graphs, and (3) $\Pi_{st}(G_n)$ is within a constant factor of $\Pi^{\parallel}_{cc}(G_n)$.

We will use the following result from [AB16] which shows that DAGs that are not depth-robust also have low $\Pi^{\parallel}_{cc}$.

Theorem 2.5 ([AB16, Thm. 3.5]) *Let $G_n \in \mathbb{G}_{n,\delta}$ such that G_n is not (e,d)-depth-robust. Then*

$$\Pi^{\|}_{cc}(G_n) = O\left(\min_{g\in[d,n]}\left\{n\left(\frac{dn}{g}+\delta g + e\right)\right\}\right)$$

setting $g = \sqrt{\frac{dn}{\delta}}$ this simplifies to

$$\Pi^{\|}_{cc}(G) = O\left(n(\sqrt{dn\delta}+e)\right)$$

3 Depth-Robustness Implies High $\Pi^{\|}_{cc}$

In this section we state and prove a theorem which lowerbounds the $\Pi^{\|}_{cc}$ of a given DAG G in terms of its depth robustness. An special interesting case of the general theorem is the following corollary

Corollary 3.1 (of Theorem 3.2) *Let G be an (e,d)-depth-robust DAG, then $\Pi^{\|}_{cc}(G) > ed$.*

We will give a proof of this corollary because it's simpler than for the general case but already contains the main ideas of the proof. See Appendix C for the proof of Theorem 3.2.
Proof of Corollary 3.1. Let $(P_1,\ldots,P_m)$ be a parallel pebbling of minimum complexity, i.e., $\sum_{i=1}^m |P_i| = \Pi^{\|}_{cc}(G)$. For any d, we'll show that there exists a set B of size $|B| \le \Pi^{\|}_{cc}(G)/d$ such that there's no path of length d in $G-B$, or equivalently, G is not $(\Pi^{\|}_{cc}(G)/d, d)$-depth-robust, note that this implies the corollary.

For $i \in [d]$ define $B_i = P_i \cup P_{i+d} \cup P_{i+2d}\ldots$. We observe that by construction $\sum_{i=0}^{d-1}|B_i| = \Pi^{\|}_{cc}(G)$, so the size of the B_i's is $\Pi^{\|}_{cc}(G)/d$ on average, and the smallest B_i has size at most this. Let B be the smallest B_i, as just outlined $|B| \le \Pi^{\|}_{cc}(G)/d$.

It remains to show that $G-B$ has no path of length d. For this consider any path $v_1,\ldots,v_d$ of length d in G. Let j be minimal such that $v_d \in P_j$ (so v_d is pebbled for the first time in round j of the pebbling). It then must be the case that $v_{d-1} \in P_{j-1}$ (as to pebble v_d is round j there must have been a pebble on v_{d-1} in round $j-1$). In round $j-2$ either the pebble on v_{d-1} was already there, or there was a pebble on v_{d-2}. This argument shows that each of the pebbling configurations $\{P_{j-d+1},\ldots,P_j\}$ must contain at least one node from $v_1,\ldots,v_d$. As B contains each dth Pebbling configuration, it has nonempty intersection with $\{P_{j-d+1},\ldots,P_j\}$, thus the path $v_1,\ldots,v_d$ is not contained entirely in $G-B$. □

Our general Theorem 3.2 states that it remains expensive to pebble any large enough set of remaining nodes in a depth-robust graph even if we are permitted to first remove an arbitrary node set of limited size. Recall that $\Pi^{\|}_{cc}(G)$ is the parallel pebbling of minimal cumulative cost when pebbling all sinks of G, this requires pebbling all nodes of G at least once. Thus Corollary 3.1 follows from Theorem 3.2 below by setting $S=\emptyset$ letting $T = \mathsf{sinks}(G)$ in which case $\sigma = 0$ and $\tau = 1$.

One application of this general theorem involves analyzing the cost of pebbling stacks of depth-robust graphs. For example if there are not enough pebbles on the graph at some point in time then there must be some layers with few pebbles. If we can then show that many of the nodes on those layers will eventually need to be (re)pebbled then we can use this lemma to show that the remaining pebbling cost incurred by these layers is large.

Theorem 3.2 *Let DAG $G=(V,E)$ be (e,d)-depth-robust and let $S,T \subset V$ such that*

$$|S| \le \sigma n \quad , \quad |\mathsf{ancestors}_{G-S}(T)| \ge \tau n \quad \textit{and} \quad T \cap S = \emptyset$$

Then the cost of pebbling $G-S$ with target set T is $\Pi^{\|}_{cc}(G-S,T) > (e-\sigma n)(d-(1-\tau)n)$.

Another immediate implication of Theorem 3.2 and Theorem 2.2 is that there is an infinite family of DAGs with maximal $\Pi^{\|}_{cc}(G) = \Omega(n^2)$ whose indegree scales with $\log n$. Note that this means that allowing indegree as small as $O(\log(n))$ is sufficient to get DAGs whose $\Pi^{\|}_{cc}$ is within a constant factor of the n^2 upper bond on $\Pi^{\|}_{cc}$ for any n vertex DAG.

Corollary 3.3 (of Theorem 3.2 and Theorem 2.2) *For some constants $c_1, c_2 > 0$ there exists an infinite family of DAGs $\{G_{n,\delta} \in \mathbb{G}_{n,\delta}\}_{n=1}^{\infty}$ with $\delta \leq c_1 \log(n)$ and $\Pi^{\parallel}_{cc}(G) \geq c_2 n^2$.*

This is optimal in the sense that for any family $\{\delta_n \in [n]\}_{n=1}^{\infty}$ and $\{J_n \in \mathbb{G}_{n,\delta_n}\}_{n=1}^{\infty}$ it holds that $\Pi^{\parallel}_{cc}(J_n) \in O(n^2)$. Moreover if $\delta_n = o(\log(n)/\log\log(n))$ then $\Pi^{\parallel}_{cc}(J_n) = o(n^2) = o(\Pi^{\parallel}_{cc}(G_n))$.

4 Reducing The Indegree and Constant Indegree Graphs with Maximal $\Pi^{\parallel}_{cc}$

In this section we use the result from the previous section to show a new, more efficient, degree-reduction lemma.

Lemma 4.1 *Let DAG G be (e,d)-depth-robust. For $\gamma \in \mathbb{Z}_{\geq 0}$ there exists $(e, d\gamma)$-depth-robust DAG G' with*

$$\mathsf{size}(G') \leq (\mathsf{indeg}(G) + \gamma) \cdot \mathsf{size}(G) \qquad\qquad \mathsf{indeg}(G') = 2.$$

Proof. Fix a $\gamma \in [n]$. We identify each node in V' with an element of the set $V \times [\delta+\gamma]$ and we write $\langle v, j\rangle \in V'$. For every node $v \in V$ with $\alpha_v := \mathsf{indeg}(v) \in [0,\delta]$ we add the path $p_v = (\langle v,1\rangle, \langle v,2\rangle, \ldots, \langle v, \alpha_v + \gamma\rangle)$ of length $\alpha_v + \gamma$. We call v the *genesis node* and p_v its *metanode*. In particular $V' = \cup_{v\in V} p_v$. Thus G has size at most $(\delta+\gamma)n$.

Next we add the remaining edges. Intuitively, for the i^{th} incoming edge (u,v) of v we add an edge to G' connecting the end of the metanode of u to the i^{th} node in the metanode of v. More precisely, for every $v \in V$, $i \in [\mathsf{indeg}(v)]$ and edge $(u_i, v) \in E$ we add edge $(\langle u_i, \mathsf{indeg}(u_i) + \gamma\rangle, \langle v, i\rangle)$ to E'. It follows immediately that G' has indegree (at most) 2.

Fix any node set $S \subset V'$ of size $|S| \leq e$. Then at most e metanodes can share a node with S. For each such metanode remove its genesis node in G. As G is (e,d)-depth-robust we are still left with a path p of length (at least) d in G. But that means that after removing S from G' there must remain a path p' in G' running through all the metanodes of p and $|p'| \geq |p|\gamma \geq d\gamma$. In other words G' is $(e, d\gamma)$-depth-robust. □

Corollary 4.2 *Applying Lemma 4.1 to the family from Theorem 2.2, we get that for some fixed constants $c_1, c_2 > 0$ there exists an infinite family of indegree 2 DAGs $\{G_n \in \mathbb{G}_{n,2}\}_{n=1}^{\infty}$ where G_n is $(c_1 n/\log n, c_2 n)$-depth robust. By Corollary 3.1 then $\Pi^{\parallel}_{cc}(G_n) > (c_1c_2)n^2/\log(n)$, which is basically optimal for constant indegree DAGs by Theorem 1.2.*

5 Constant Indegree Graphs with Maximal $\Pi^{\parallel}_{cc}$ and Constant $\Pi_{st}/\Pi^{\parallel}_{cc}$ Ratio

In this section we construct a simple family of graphs $\{G_n \in \mathbb{G}_{n,2}\}_{n\in\mathbb{N}}$ with maximal parallel cumulative pebbling complexity $\Pi^{\parallel}_{cc}(G_n) \in \Omega(n^2/\log(n))$, and where $\Pi_{st}(G_n) \in O(n^2/\log(n))$, so the pebbling cost increases only by a constant factor if we only allow sequential pebbling and no armotisation (i.e., space-time instead of cumulative).

Construction 5.1 *The construction of DAG $G_n = G(H_n, n) = (V,E)$ is parametrized by $n = 2^m$ for $m \in \mathbb{N}^+$ and a (depth-robust) graph $H_n = (\bar{V}, \bar{E})$ of size n. We identify the nodes of $\bar{V}$ with the set $\{v_j : j \in [n]\}$ such that $(v_1, v_2, \ldots, v_n)$ are sorted according to some topological ordering.*

1. *Initially G_n consists of a stack of m copies of H_n. We identify node $v \in \bar{V}$ at layer $i \in [m]$ with $\langle v, i\rangle$. In particular $V = \{\langle i, v\rangle : i \in [m], v \in \bar{V}\}$ and initially we set*

$$E := \{(\langle i,u\rangle, \langle i,v\rangle) : i \in [m], (u,v) \in H_n\}.$$

2. *Add a path running through the graph in topological order.*

$$\begin{aligned} E := E \quad &\cup \quad \{(\langle i, v_j\rangle, \langle i, v_{j+1}\rangle) : i \in [m], j \in [n-1]\} \\ &\cup \quad \{(\langle i, v_n\rangle, \langle i+1, v_1\rangle) : i \in [m-1]\}. \end{aligned}$$

3. *Connect each node in a layer to its twin in the next layer.*

$$E := E \cup \{(\langle i, v\rangle, \langle i+1, v\rangle) : i \in [m-1], v \in \bar{V}\}.$$

4. *Connect each node in the first half of a layer to a node in the second half of the next layer.*

$$E := E \cup \{(\langle i, v_j\rangle, \langle i+1, v_{j+n/2}\rangle) : i \in [m-1], v_j \in \bar{V}, j \in [n/2]\}.$$

The following theorem summarizes the properties of some important properties of the construction.

Theorem 5.2 *Let $m \in \mathbb{N}_{\geq 2}$ and $n = 2^m$ and let $0 < \alpha, \beta < 1$ be constants. If H_n is $(\frac{\alpha n}{\log(n)}, \beta n)$-depth-robust then the following hold for $G_n = G(H, n)$ of Construction 5.1.*

1. *$G_n \in \mathbb{G}_{N,\delta}$ where $N = n\log(n)$ and $\delta \leq \mathsf{indeg}(H_n) + 3$.*
2. *$\Pi^{\parallel}_{cc}(G_n) \geq \gamma \cdot N^2/\log(N) = \Omega(N^2/\log(N))$ where $\gamma = \max_{0<\alpha_0<\alpha,\beta} \big(\min\{\alpha_0/8, (\alpha-\alpha_0)(\beta-\alpha_0)/2\}\big)$.*
3. *$\Pi_{st}(G_n) \in O(N^2/\log(N))$.*

We can instantiate the H_n in Construction 5.1 with the DAG from Corollary 4.2 where $\mathsf{indeg}(H_n) = 2$. Incidentally, that DAG already contains a path running through every node and thus we can slightly improve the indegree bound to $\mathsf{indeg}\big(G_n\big) \leq 4$, we get the following

Corollary 5.3 *Construction 5.1 where H_n is instantiated with the DAG from Corollary 4.2 gives an infinite family of DAGs $\{G_n \in \mathbb{G}_{n,4}\}_{n=1}^{\infty}$ where*

$$\Pi^{\parallel}_{cc}(G_n) = \Omega(n^2/\log(n)) \quad \textit{and} \quad \Pi_{st}(G_n) = O(n^2/\log(n))$$

This basically proves Theorem 1.1, expect that the indegree is 4 not 2. But can get the indegree down from 4 to 2 using Lemma 4.1, which will make the construction slightly more complicated and worsen the constants hidden in the big-oh notation.

The proof of Theorem 5.2 is in Appendix C. We outline the key ideas here. First, we describe the sequential pebbling strategy $\mathcal{N}$ which witnesses that $\Pi_{st}(G_n) \in O(N^2/\log(N))$. Intuitively it pebbles G_n in topological order removing pebbles from a layer once the subsequent layer has been completely pebbled. Thus, $\Pi_{st}(G_n) \leq 2Nn = O(N^2/\log(N))$ as $\mathcal{N}$ runs for N steps and keeps at most $2n$ pebbles on G_n. Clearly, we must pay cost $\Pi^{\parallel}_{cc}(H_n) = \Omega\left(n^2/\log(n)\right)$ to pebble each layer in G_n, but this only shows that $\Pi^{\parallel}_{cc}(G_n) = \Omega(n^2) = \Omega\left(N^2/\log^2(N)\right)$ (here we use that $\Pi^{\parallel}_{cc}$ satisfies a direct product property, so pebbling the $\log(n)$ H_n graphs is $\log(n)$ times as expensive as pebbling one). A slightly more sophisticated argument shows that we must incur cost at least $\Omega\left(n^2\right)$ while pebbling each of the top $\log(n)/4$ layers of G_n. The key insight is that while pebbling layer $i > 3\log(n)/4$ we need to keep at least γn pebbles on the graph for the first $n/2$ steps (in this case the total cost for pebbling layer i is at least $\Omega(n^2)$). If we don't keep γn pebbles on the graph for the first $n/2$ steps then, at some point in time, we can show that there are unpebbled paths from almost every node in previous layers to the last $n/2$ nodes in layer i (using the directed edges we added in steps 3 and 4 of Construction 5.1). That is, on each layer $j < i$, at least $(1-\gamma)n$ nodes lie on an unpebbled path to one of the last $n/2$ nodes in layer i. This means that at some future point in time we will need to repebble at least $(1-\gamma)n$ nodes on each layer $j < i$ (a total of $\Omega(\log(n))$ layers) before we can finish pebbling layer i. Furthermore, most of these layers $j < i$ will contain fewer than $\alpha n/(2\log(n))$ pebbles and each layer consists of a $(\alpha n/\log(n), \beta n)$-depth robust DAG H_n. By Theorem 3.2 we will need to pay cost $\Omega(n^2/\log(n))$ to repebble each layer that had fewer than $\alpha n/(2\log(n))$ pebbles (i.e., most lower layers $j < i$). In either case, the total cost for pebbling layer $i > 3\log(n)/4$ is at least $\Omega(n^2)$. Thus, $\Pi^{\parallel}_{cc}(G_n) \geq \frac{\log(n)}{4} \times \Omega\left(n^2\right) = \Omega\left(N^2/\log(N)\right)$.

6 Analyzing Candidate iMHFs

On the surface, in this section we give both security proofs and optimal attacks for many of the most prominent iMHF proposals. That is we show both lower and (relatively tight) upperbounds on their memory-hardness. However, more conceptually, we also introduce two new proof techniques for analyzing the depth-robustness of DAGs as well as a new very efficient generic method for pebbling a DAG. Indeed for all candidates considered in this the attack is almost optimal in light of the accompanying security proofs.

More specifically, in the first subsection we prove bounds for a class of random graphs which generalize the Argon2i construction [BDK16] and the Single-Buffer (SB) construction of [CGBS16]. To prove the lowerbound we use a simple and clean new technique for bounding the depth-robustness of a random DAG. Combined with Corollary 3.3 we could immediately obtain a lower bound of $\tilde{\Omega}\left(n^{1.5}\right)$. Using similar analysis as in the proof of Theorem 5.2 we can improve the lower bound to $\tilde{O}\left(n^{5/3}\right)$ by showing that we need to either (1) keep $\tilde{O}\left(n^{2/3}\right)$ pebbles on the first half of the DAG during most pebbling steps, or (2) frequently repebble the first half of the DAG (which also has high cost). Conversely, to show the upperbound we describe a new method for efficiently pebbling arbitrary reducible (i.e. non-depth-robust) DAGs. While we instantiate the method for the case of random DAGs (improving on the hitherto best results of [AB16]) we believe it very likely to also lead to improved attacks on a variety of other iMHFs constructions in the literature.

In the second subsection we prove bounds for a family of layered graphs which generalize both of the Catena constructions [FLW13]. The upperbound for Catena is implied by the same result for the random DAGs. However the lowerbound is a direct proof which uses a new type of pebbling argument without going through the notion of depth-robustness.

JoëlredWe should comment that these results hold also for the newest version of Argon2. Namely v1.3. Also it should be pretty easy to see if the) Catena lowerbound holds for the Double-Buffer and Linear Balloon Hashing constructions. Let σ be the memory cost and τ be the time-cost. Then both DB and Lin. have τ layers each of size σ. Intuitively since $\delta = 20$ back edges of each node are chosen uniform random and independently over the previous layer the probability that a given layer is g dispersed should be grow relatively quickly in g...

6.1 Lowerbounding the CC of Random DAGs.

We begin by defining a (n, δ, w)-random DAG, the underlying DAGs upon which Argon2i and SB are based. The memory window parameter w specifies the intended memory usage and throughput of the iMHF — the cost of the naïve pebbling algorithm is $\Pi^{\parallel}_{cc}(\mathcal{N}) = wn$. In particular, a t-pass Argon2i iMHF is based on a $(n, 2, n/t)$-random DAG. Similarly, a t-pass Single-Buffer (SB) iMHF [CGBS16] is based on a $(n, 20, n/t)$-random DAG. In this section we focus on the $t = 1$-pass variants of the Argon2i and [CGBS16] iMHFs.

Definition 6.1 ((n, δ, w)-random DAG) *Let $n \in \mathbb{N}$, $1 < \delta < n$, and $1 \le w \le n$ such that w divides n. An (n, δ, w)-*random DAG *is a randomly generated directed acyclic (multi)graph with n nodes $v_1, \ldots, v_n$ and with maximum in-degree δ for each vertex. The graph has directed edges (v_i, v_{i+1}) for $1 \le i < n$ and random forward edges $(v_{r(i,1)}, v_i), \ldots, (v_{r(i,\delta-1)}, v_i)$ for each vertex v_i. Here, $r(i, j)$ is independently chosen uniformly at random from the set* $[\max\{0, i - w\}, i - 1]$.

Security Lower Bound. To prove the lower bound we rely on a slightly stricter notion of depth robustness. Given a node v let $N(v, b) = \{v - b + 1, \ldots, v\}$ denote a segment of b consecutive nodes ending at v and given a set $S \subseteq V(G)$ let $N(S, b) = \bigcup_{v \in S} N(v, b)$. We say that a DAG G is (e, d, b)-block depth-robust if for every set $S \subseteq V(G)$ of size $|S| \le e$ we have $\mathsf{depth}(G - N(S, b)) \ge d$. Lemma 6.3 shows that for any $e \ge \sqrt{n}$, with high probability, a $(n, 2, n)$-random DAG G will be (e, d, b)-block depth-robust with $d = \frac{n^2}{e^2 \mathsf{polylog}(n)}$ and $b = n/(10e)$ (in contrast Lemma 6.11 states that G will be (e, d)-reducible with $d = \tilde{O}\left(n^2/e^2\right)$). Taking $e = \sqrt{n}$ in Corollary 3.1 immediately implies that $\Pi^{\parallel}_{cc}(G) = \tilde{\Omega}\left(n^{1.5}\right)$. We can improve this bound with a slightly more sophisticated argument. In particular, we consider the cost incurred while pebbling a set $B \subseteq \{n/2 + 1, \ldots, n\}$ of $e * \mathsf{polylog}(n)$ consecutive nodes in the last half of G. Case 1: We keep $\Omega(e)$ pebbles on G (nodes $\{1,\ldots,n/2\}$) while pebbling B_{first}, the first $|B|/2$ nodes of B (Total Cost: $\Omega(e^2)=\tilde{\Omega}\left(n^{4/3}\right)$). Case 2: At some point in time t we have $|P_t|=o(e)$ pebbles on G while pebbling B_{first}. In this case, we will need to essentially repebble the entire first half of G before we can finish pebbling B_{last}, the last $|B|/2$ nodes of B. In particular, we can prove that (whp) every segment of $b=\tilde{\Omega}\left(n^{1/3}\right)$ consecutive nodes in the first half of G has a directed edge to B_{last}. Now a similar argument to the proof of Theorem 3.2 shows that it will cost $\Omega(ed)=\tilde{\Omega}\left(n^{4/3}\right)$ to repebble the necessary nodes in the first half of G — note that $G-\{n/2+1,\ldots,n\}$ is a $(n/2,\delta,n/2)$-random DAG and is thus block depth robust (whp) by Lemma 6.11. There are $n/(e*\mathsf{polylog}(n))=\tilde{\Omega}\left(n^{1/3}\right)$ disjoint segments of $e*\mathsf{polylog}(n)$ consecutive nodes in $\{n/2+1,\ldots,n\}$. Thus, the total cost is $\tilde{\Omega}\left(n^{5/3}\right)$.

Theorem 6.2 *Let G be a (n,δ,n)-random DAG then, except with probability $o\big(n^{-3}\big)$, we have*

$$\Pi^{\parallel}_{cc}(G)=\tilde{\Omega}\left(n^{5/3}\right) .$$

We make a couple of observations.

1. The lower bound from Theorem 6.2 might be tight. Alwen and Blocki [AB16] gave an attack $\mathcal{A}$ such that $\Pi^{\parallel}_{cc}(\mathcal{A})=O\left(n^{1.75}\delta\log n\right)$ for a (n,δ,t)-random DAG. We reduce the gap of $\tilde{O}\big(n^{1/12}\big)$ by developing an improved recursive version of the attack of Alwen and Blocki [AB16]. In particular, we show that for any $\epsilon>0$ we have $\Pi^{\parallel}_{cc}(\mathcal{A})=o\left(n^{1+\sqrt{1/2}+\epsilon}\right)=o\left(n^{1.708}\right)$. Our modified attack also improves the upper bound for other iMHF candidates like Catena [FLW13].

2. Corollary 3.1 alone will not yield any meaningful lower bounds on the $\Pi^{\parallel}_{cc}$ of the Catena iMHFs [FLW13]. In particular, the results from Alwen and Blocki [AB16] imply that for any t-pass variant of Catena corresponding DAG is not (e,d)-depth robust for $ed\geq nt$ (typically, $t=O(\mathsf{polylog}(n))$). However, we use an alternative techniques below to prove that $\Pi^{\parallel}_{cc}(G)=\Omega\big(n^{1.5}\big)$ for the both Catena iMHFs and the Linear and Double-Buffer iMHFs of [CGBS16].

The proof of Theorem 6.2 relies on the next lemma. The proofs of Lemma 6.3 and Theorem 6.2 are in Appendix C.

Lemma 6.3 *For any $e\geq\sqrt{n}$ any any $\delta\geq 2$ a (n,δ,n)-random DAG will be $\left(e,\Omega\left(\frac{n^2}{e^2\log(n)}\right),\frac{n}{100e}\right)$-block depth robust with probability $1-o\big(n^{-3}\big)$.*

6.2 Lowerbounding Dispersed Graphs

In this section we prove a lowerbound on the CC of a class of DAGs called dispersed graphs (defined bellow). Next we show that several of the iMHF constructions from the literature are based on such graphs. Thus we obtain proofs of security for each of these constructions (albeit for limited levels of security). In the subsequent section we give an upperbound on the CC of these constructions showing that the lowerbounds in this section are relatively tight.

Intuitively a (g,k)-dispersed DAG is a DAG ending with a path ϕ of length k which has widely dispersed dependencies. The following definitions make this concept precises.

Definition 6.4 (Dependencies) *Let $G=(V,E)$ be a DAG and $L\subseteq V$. We say that L has a (z,g)-*dependency *if there exist node disjunct paths $p_1,\ldots,p_z$ each ending in L and with length (at least) g.*

We are interested in graphs with long paths with many sets of such dependencies.

Definition 6.5 (Dispersed Graph) *Let $g\leq k$ be positive integers. A DAG G is called (g,k)-*dispersed *if there exists a topological ordering of its nodes such the following holds. Let $[k]$ denote the final k nodes in* *the ordering of G and let $L_j = [jg, (j+1)g-1]$ be the j^{th} subinterval. Then $\forall j \in [\lfloor k/g \rfloor]$ interval L_j has a (g,g)-dependency.*

*More generally, let $\epsilon \in (0,1]$. If each interval L_j only has an $(\epsilon g, g)$-dependency then G is called (ϵ, g, k)-*dispersed.

We show that many graphs in the literature consist of a *stack* of dispersed graphs. Our lowerbound on the CC of a dispersed graph grows in the height of this stack. The next definition precisely captures such stacks.

Definition 6.6 (Stacked Dispersed Graphs) *A DAG $G = (V,E)$ is called $(\lambda, \epsilon, g, k)$-*dispersed *if there exist $\lambda \in \mathbb{N}^+$ disjoint subsets of nodes $\{L_i \subseteq V\}$, each of size k with following two properties.*

1. *For each L_i there is a path running through all nodes of L_i.*
2. *Fix any topological ordering of G. For each $i \in [\lambda]$ let G_i be the subgraph of G containing all nodes of G up to the last node of L_i. Then G_i is an (ϵ, g, k)-dispersed graph. We denote the set of $(\lambda, \epsilon, g, k)$-dispersed graphs by $\mathbb{D}^{\lambda,k}_{\epsilon,g}$.*

We are now ready to state and prove the lowerbound on the CC of stacks of dispersed graphs.

Theorem 6.7

$$G \in \mathbb{D}^{\lambda,k}_{\epsilon,g} \quad \Rightarrow \quad \Pi^{\parallel}_{cc}(G) \geq \epsilon\lambda g\left(\frac{k}{2} - g\right).$$

Intuitively we sum the CC of pebbling the last k nodes L_i of each subgraph G_i. For this we consider any adjacent intervals A of $2g$ nodes in L_i. Let p be a path in the $(\epsilon g, g)$-dependency of the second half of A. Either at least one pebble is always kept on p while pebbling the first half of A (which takes time at least g since a path runs through L_i) or p must be fully pebbled in order to finish pebbling interval A (which also takes time at least g). Either way pebbling A requires an additional CC of g per path in the $(\epsilon g, g)$-dependency of the second half of A. Since there are $k/2g$ such interval pairs each with ϵg incoming paths in their dependencies we get a total cost for that layer of $kg\epsilon/2$. So the cost for all layer of G is at least $\lambda k g \epsilon/2$. The details (for the more general case when g doesn't divide n) are in Appendix C.

Now that we have our lowerbound for stacks of dispersed graphs it remains to analyze for which parameters various graphs in the literature confirm to this notion. The results are summarized in the theorem bellow.

Theorem 6.8 *[iMHF Constructions Based on Dispersed Graphs]*

- *If $\lambda, n \in \mathbb{N}^+$ such that $k = n/(\lambda+1)$ is a power of 2 then it holds that*

$$\mathsf{BRG}^n_\lambda \in \mathbb{G}_{n,2} \qquad \mathsf{BRG}^n_\lambda \in \mathbb{D}^{\lambda,k}_{1,\lceil\sqrt{k}\rceil} \qquad \Pi^{\parallel}_{cc}(\mathsf{BRG}^n_\lambda) = \Omega\left(\frac{n^{1.5}}{\sqrt{\lambda}}\right)$$

- *If $\lambda, n \in \mathbb{N}^+$ such that $n = \bar{n}(\lambda(2c-1)+1)$ where $\bar{n} = 2^c$ for some $c \in \mathbb{N}^+$ then it holds that*

$$\mathsf{DBG}^n_\lambda \in \mathbb{G}_{n,3} \qquad \mathsf{DBG}^n_\lambda \in \mathbb{D}^{\lambda,\bar{n}}_{1,\lceil\sqrt{\bar{n}}\rceil} \qquad \Pi^{\parallel}_{cc}(\mathsf{DBG}^n_\lambda) = \Omega\left(\frac{n^{1.5}}{c\sqrt{c\lambda}}\right)$$

- *If $\sigma, \tau \in \mathbb{N}^+$ such that $n = \sigma * \tau$ then with high probability it holds that*

$$\mathsf{Lin}^\sigma_\tau \in \mathbb{G}_{n,21} \qquad \mathsf{Lin}^\sigma_\tau \in \mathbb{D}^{\tau-1,\sigma}_{0.25,\sqrt{\sigma}/2} \qquad \Pi^{\parallel}_{cc}(\mathsf{Lin}^\sigma_\tau) = \Omega\left(\frac{n^{1.5}}{\sqrt{\tau}}\right).$$

The theorem is proven in three subsections of Appendix B, one per graph.

6.3 Improved Pebbling Attacks

In this section we improve upon the attacks of [AB16] for non-depth robust DAGs G.

Review of [AB16] The general pebbling attack of Alwen and Blocki [AB16] made use of a set $S \subset V$ of size $|S| = e$ such that removing S reduces the depth of the DAG $\mathsf{depth}(G - S) \leq d$. Intuitively keeping pebbles on S compresses the graph in the sense that it can quickly be completely pebbled within d steps as any unpebbled path has length $\leq d$. The attack never removes pebbles from nodes in S and its goal is to always pebble node i at time i so as to finish in $n = \mathsf{size}(G)$ steps. To ensure that parents of node i are all pebbled at time i the attack alternatives between two types of phases: balloon phases (which last d for steps each) followed by a light phases (which last for $g - d$ steps each). Intuitively, during a balloon phase the graph is quickly "decompressed" by greedily re-pebbling everything using the pebbles on nodes in S. Then at the beginning of the subsequent light phase all pebbles are removed from G except those on nodes in S and those needed to pebble the next $g > d$ nodes. Thus parents of node i are always pebbled at time i but G is "compressed" into S during light phases.

The cost of their attack $\mathcal{A}$ was $\Pi_{cc}^{\parallel}(\mathcal{A}) \leq en + \delta gn + \frac{dn^2}{g}$. Intuitively, the $en + \delta gn$ terms upper bound the pebbling costs of light phases, and the third term upper bounds the cost of all balloon phases — each balloon phase costs at most dn and they need at most $\frac{n}{g}$ balloon phases. Setting $g = \sqrt{nd/\delta}$ to balance the gn and $\frac{dn^2}{g}$ terms they obtained $\Pi_{cc}^{\parallel}(\mathcal{A}) \leq O\big(en + n^{1.5}\sqrt{d/\delta}\big)$[6].

Recursive Attack The key insight behind our improved attack is that we may not actually need to pay cost dn for each of the balloon phases. If we could find a second set $S' \supset S$ of $e' > e$ nodes such that $\mathsf{depth}(G - S') \leq d' < d$ then we could potentially reduce the cost of each balloon phase by applying the attack of Alwen and Blocki [AB16] recursively. In general, given a function $f(d)$ we can improve the attack for f-reducible DAGs — a DAG that is not $(f(d), d)$-depth robust for any value of d.

Definition 6.9 *Let $G = (V, E)$ be a DAG with n nodes and let $f : \mathbb{N} \to \mathbb{N}$ be a function. We say that G is f-reducible if for every positive integer $n \geq d > 0$ there exists a set $S \subseteq V$ of $|S| = f(d)$ nodes such that $\mathsf{depth}(G - S) \leq d$.*

We show that any $f(d) = \tilde{O}\big(\frac{n}{\sqrt{d}}\big)$-reducible DAG on n nodes has $\Pi_{cc}^{\parallel}(G) \leq O\left(n^{1.708}\right)$. In particular, this shows that $\Pi_{cc}^{\parallel}(G) \leq O\left(n^{1.708}\right)$ for a (n, δ, n)-random DAGs like Argon2i. We also show that $\Pi_{cc}^{\parallel}(G) \leq O\left(n^{1.6181}\right)$ for a $f(d) = \tilde{O}\big(\frac{n}{d}\big)$-reducible DAG G, which implies that $\Pi_{cc}^{\parallel}(\mathsf{BRG}_{\lambda}^{n}) = O\left(n^{1.6181}\right) = \Pi_{cc}^{\parallel}(\mathsf{DBG}_{\lambda}^{n})$ whenever $\lambda = O(\mathsf{polylog}(n))$. For comparison, the non-recursive attack of Alwen and Blocki [AB16] demonstrated that $\Pi_{cc}^{\parallel}(G) \leq \tilde{O}\big(n^{1.75}\big)$ for (n, δ, n)-random DAGs and that $\Pi_{cc}^{\parallel}(G) \leq O\big(n^{5/3}\big)$ for λ-layered DAGs.

Theorem 6.10 *Let G be a f-reducible DAG on n nodes then if $f(d) = \tilde{O}\big(\frac{n}{d^b}\big)$ for some constant $0 < b \leq 1$ then for any constant $\epsilon > 0$*

$$\Pi_{cc}^{\parallel}(G) \leq O\left(n^{1+a+\epsilon}\right), \text{ where } \qquad a = \frac{1 - 2b + \sqrt{1 + 4b^2}}{2} .$$

Lemma 6.11 states that (n, δ, n)-random DAGs and λ-layered DAGs are f-reducible. The proof of Lemma 6.11, which repeats arguments from Alwen and Blocki [AB16] for different values of d, can be found in the appendix.

Lemma 6.11 *Let $f_b(d) = \tilde{O}\big(\frac{n}{d^b}\big)$ then*

1. *Let $\delta = O(\mathsf{polylog}(n))$ then a (n, δ, n)-random DAG is $f_{0.5}$-reducible with high probability.*

[6]For some DAGs (e.g., $\mathsf{BRG}_{\lambda}^{n}$) Alwen and Blocki [AB16] were able to obtain slightly better bounds on $\Pi_{cc}^{\parallel}(\mathcal{A})$ by exploiting instance-specific structure of the particular DAG in their analysis.

2. *The Catena DAGs* BRG_λ^n *and* DBG_λ^n *are both* f_1*-reducible for* $\lambda = O(\mathsf{polylog}(n))$.

3. *The Balloon Hashing Linear (and the Double Buffer) graph* Lin_τ^σ *is* f_1*-reducible for* $\tau = O(\mathsf{polylog}(n))$.

Corollary 6.12 *Let* $\epsilon > 0$ *be any constant*

1. *Let* $\delta = O(\mathsf{polylog}(n))$ *then an* (n, δ, n)*-random DAG* G *has* $\Pi_{cc}^{\parallel}(G) = O\big(n^{1+\sqrt{1/2}+\epsilon}\big) \approx O\left(n^{1.707+\epsilon}\right)$.

2. *Both* $\Pi_{cc}^{\parallel}\big(\mathsf{BRG}_\lambda^n\big)$ *and* $\Pi_{cc}^{\parallel}\left(\mathsf{DBG}_\lambda^n\right)$ *are in* $O\big(n^{1+\frac{\sqrt{5}-1}{2}+\epsilon}\big) \approx O\left(n^{1.618+\epsilon}\right)$.

3. $\Pi_{cc}^{\parallel}\big(\mathsf{Lin}_\tau^\sigma\big) = O\big(n^{1+\frac{\sqrt{5}-1}{2}+\epsilon}\big) \approx O\left(n^{1.618} + \epsilon\right)$*, where* Lin_τ^σ *has* $n = \tau * \sigma$ *nodes.*

We define a recursive algorithm $\mathsf{RecursiveGenPeb}$, which takes as input a DAG G of depth $d_0 = \mathsf{depth}(G)$, a target set $T \subset V(G)$ of nodes to pebble and sets $S_1, \ldots, S_k$ of sizes $e_1 = f(d_1) < e_2 = f(d_2) \ldots < e_k = f(d_k)$ such that $\mathsf{depth}(G - S_i) \geq d_i$ for each $i \leq k$. $\mathsf{RecursiveGenPeb}$ returns a pebbling $P_1, \ldots, P_{2d_0}$ of G in at most $2d_0$ steps.

We let $G_0 = G$ and $G_{i+1} = G_i - S_{i+1}$. We partition the nodes of G_i according to their depth and further split up sets which are larger than n/d_i. This gives us a partition $D_1^i, \ldots, D_{2d_i}^i$ of $V(G_i)$ which satisfies the following properties:

Sources: $\mathsf{parents}\big(D_1^i\big) = \emptyset$,

Topologically Ordered: $\forall j \leq 2d_i - 1 \quad \mathsf{parents}\big(D_{j+1}^i\big) \subseteq \bigcup_{y \leq j} D_y^i$, and

Maximum Size: $\forall j \leq 2d_i \quad \big|D_j^i\big| \leq \frac{n}{d_i}$.

At top level of recursion we closely follow the attack of Alwen and Blocki [AB16]. Our goal is to pebble G_0 with the target set $T_0 = \mathsf{sinks}(G_0)$ in at most $2d_0$ steps. We ensure that our pebbling maintains the invariant that for pebbling round j we keep pebbles on the set $P_j \supset D_j^0 \cup \mathsf{parents}(\{j+1, \ldots, j+g_0\}) \bigcup \Big(S_1 \bigcap \bigcup_{y \leq j} D_y^0\Big)$. Thus, during pebbling step j $\mathsf{RecursiveGenPeb}$ will pebble all of the nodes in the set D_j^0.

JoëlredAre the superscripts of 1 needed? Same question for the superscripts of 1 bellow for the $P_1, \ldots, P_{2d_1}$ bellow. JoëlredMaybe give super brief intuition why we can find such an order. Something like: partition nodes according to their depth. Then further split up sets which are larger than n/d_0. This gives a partition satisfying the all three properties. To see that this results in at most $2d_0$ sets note that if this were not the case then...

Light Phases. To save space, during a light phase round i we discard most other pebbles except pebbles on nodes in the set $S_1 \cup D_i^0 \cup \mathsf{parents}(\{i+1, \ldots, i+2e_1\})$. We only keep pebbles in the set S_1 and on parents of the next $2e_1$ nodes that we want to pebble. The total cost of all light phases is $2d_0(2\delta + 1)e_1$.

Balloon Phases and Recursion. At some point in time we may find that $D_{i+2d_1+1}^0 \not\subset P_i \cup \Big(\bigcup_{y \leq i+2d_1} D_y^0\Big)$ — in $2d_1 + 1$ steps we will want to pebble a node $v \in D_{i+2d_1+1}^0$ but we have already discarded pebble(s) from some of the parent(s) of this node. Thus, to maintain our invariant we will quickly recover the discarded pebbles for all of parents of the next $2e_1$ nodes. Alwen and Blocki [AB16] used a greedy approach (for d_1 steps pebble everything possible) to accomplish this task (total cost: $d_1 n$). Instead, we call our algorithm $\mathsf{RecursiveGenPeb}$ recursively to find a pebbling $\big(P_1^1, \ldots, P_{2d_1}^1\big)$ of the graph $G_1 = G - S_1$ with the target set $T_1 = \mathsf{parents}(\{i+1, \ldots, i+2e_1\})$ in at most $2d_1$ pebbling steps. The recursive pebbling $P_1^1, \ldots, P_{2d_1}^1$ will also be divided into balloon phases and light phases, and we will maintain the same invariant until we finish pebbling T_1 (i.e., never discard pebbles from S_2, always keep pebbles on the parents of the next e_2 nodes we want to pebble, pebble the set $P_i^1 \supset D_i^1$ in round). Balloon phases involve yet another recursive call to $\mathsf{RecursiveGenPeb}$ with the graph G_2 and target set $T_2 = \{$ parents of the next $2e_2$ nodes we want to pebble in $G_2\}$. At the final level of recursion we use the greedy pebbling strategy (pebble every possible node for $d_k = \mathsf{depth}(G_k)$ steps) which costs at most $d_k n$.

Analysis. We select $n \geq d_0 > d_1 > \ldots > d_k$ such that for each $i > 0$ we can find a set S_i of size $|S_i| = e_i = f(d_i) = \tilde{O}(n^{a_i})$ such that $\mathsf{depth}(G - S_i) \leq d_i$. Here, the sequence $\{a_i\}_{i=1}^{\infty}$ is defined as follows

$$a_1 = a = \frac{1 - 2b + \sqrt{1 + 4b^2}}{2}\ , \quad \text{and} \qquad a_{i+1} = 1 + \frac{(a-1)(1-a_i)}{b}\ .$$

We obtain the following recurrence relationship from our attack

$$\Pi_{cc}^{\parallel}(G_i) \leq (2\delta + 1)e_{i+1}2d_i + \frac{n}{e_{i+1}}\Pi_{cc}^{\parallel}(G_{i+1})\ .$$

In particular, our pebbling of G_i takes at most $2d_i$ steps and we keep pebbles on at most $(2\delta + 1)e_{i+1}$ nodes during light phases. We need to execute at most $\frac{n}{e_{i+1}}$ balloon phases and each balloon phase costs at most $\Pi_{cc}^{\parallel}(G_{i+1})$ to complete. For a sufficiently large constant $k > k_\epsilon$ we have $d_k < n^{\epsilon/2}$ so that $\Pi_{cc}^{\parallel}(G_k) \leq O\left(n^{1+\epsilon/2}\right)$. By exploiting several key properties of the sequence $\{a_i\}_{i=1}^{\infty}$ we can unroll the recurrence to show that

$$\Pi_{cc}^{\parallel}(G_0) \leq O\left(n^{1+a+\epsilon}\right)\ .$$

7 Open Questions

We conclude with several open questions for future research.

- We showed that for some constant $c \geq 0$ we can find a DAG G on n nodes with $\Pi_{cc}^{\parallel}(G) \geq cn^2/\log(n)$ and $\mathsf{indeg}(G) = 2$. While this result is asymptotically optimal the constant terms are relevant for practical applications to iMHFs. How big can this constant c be? Can we find explicit constructions of constant-indegree, $(c_1 n/\log(n), c_2 n)$-depth robust DAGs that match these bounds?

- Another interesting direction concerns understanding the cumulative pebbling complexity of generic graphs. Given a graph G is it computationally tractable to (approximately) compute $\Pi_{cc}^{\parallel}(G)$? An efficient approximation algorithm for $\Pi_{cc}^{\parallel}(G)$ would allow us to quickly analyze candidate iMHF constructions. Conversely, as many existing iMHF constructions are based on fixed random graphs, [BDK16, CGBS16] showing that approximating such a graphs complexity is hard would provide evidence that an adversary will likely not be able to leverage properties of the concrete instance to improve their evaluation strategy for the iMHF. Indeed, it may turn out that the most effective way to construct depth-robust graphs with good constants is via a randomized construction.

- Another open challenge is to develop a data-dependent MHF f with parallel cost $\Pi_{cc}^{\parallel}(f) = \Omega(n^2)$. Alwen and Blocki [AB16] show that this goal is impossible for iMHFs, but data-dependent MHFs could potentially satisfy both properties. However, at this time it is not known whether any dMHF candidate (e.g., `scrypt` [Per09, PJ12]) satisfy this property.

References

[AB16] Joël Alwen and Jeremiah Blocki. Efficiently computing data-independent memory-hard functions. Cryptology ePrint Archive, Report 2016/115, 2016. `http://eprint.iacr.org/`.

[ABW03] Martín Abadi, Michael Burrows, and Ted Wobber. Moderately hard, memory-bound functions. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA*, 2003.

[AS15] Joël Alwen and Vladimir Serbinenko. High Parallel Complexity Graphs and Memory-Hard Functions. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '15, 2015. `http://eprint.iacr.org/2014/238`.

[BDK16] Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Argon2 password hash. Version 1.3, 2016. `https://www.cryptolux.org/images/0/0d/Argon2.pdf`.

[BK15] Alex Biryukov and Dmitry Khovratovich. Tradeoff cryptanalysis of memory-hard functions. Cryptology ePrint Archive, Report 2015/227, 2015. `http://eprint.iacr.org/`.

[CGBS16] Henry Corrigan-Gibbs, Dan Boneh, and Stuart Schechter. Balloon hashing: Provably space-hard hash functions with data-independent access patterns. Cryptology ePrint Archive, Report 2016/027, Version: 20160601:225540, 2016. `http://eprint.iacr.org/`.

[Cha73] Ashok K. Chandra. Efficient compilation of linear recursive programs. In *SWAT (FOCS)*, pages 16–25. IEEE Computer Society, 1973.

[Coo73] Stephen A. Cook. An observation on time-storage trade off. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC '73, pages 29–33, New York, NY, USA, 1973. ACM.

[DFKP15] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 585–605. Springer, Heidelberg, August 2015.

[DGN03] Cynthia Dwork, Andrew Goldberg, and Moni Naor. On memory-bound functions for fighting spam. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 426–444. Springer, 2003.

[DKW11] Stefan Dziembowski, Tomasz Kazana, and Daniel Wichs. Key-evolution schemes resilient to space-bounded leakage. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 335–353. Springer, Heidelberg, August 2011.

[DNW05] Cynthia Dwork, Moni Naor, and Hoeteck Wee. Pebbling and proofs of work. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 37–54. Springer, Heidelberg, August 2005.

[EGS75] Paul Erdoes, Ronald L. Graham, and Endre Szemeredi. On sparse graphs with dense long paths. Technical report, Stanford, CA, USA, 1975.

[FLW13] Christian Forler, Stefan Lucks, and Jakob Wenzel. Catena: A memory-consuming password scrambler. *IACR Cryptology ePrint Archive*, 2013:525, 2013.

[HP70] Carl E. Hewitt and Michael S. Paterson. Record of the project mac conference on concurrent systems and parallel computation. chapter Comparative Schematology, pages 119–127. ACM, New York, NY, USA, 1970.

[Kal00] Burt Kaliski. Pkcs# 5: Password-based cryptography specification version 2.0. 2000.

[MMV13] Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Publicly verifiable proofs of sequential work. In Robert D. Kleinberg, editor, *ITCS 2013*, pages 373–388. ACM, January 2013.

[NB$^+$15] Arvind Narayanan, Joseph Bonneau, , Edward W Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technology (manuscript)*. 2015. Retrieved 8/6/2015.

[Per09] C. Percival. Stronger key derivation via sequential memory-hard functions. In *BSDCan 2009*, 2009.

[PJ12] Colin Percival and Simon Josefsson. The scrypt password-based key derivation function. 2012.

[PR80] Wolfgang J. Paul and Rüdiger Reischuk. On alternation II. A graph theoretic approach to determinism versus nondeterminism. *Acta Inf.*, 14:391–403, 1980.

[RD16] Ling Ren and Srinivas Devadas. Proof of space from stacked bipartite graphs. Cryptology ePrint Archive, Report 2016/333, 2016. `http://eprint.iacr.org/`.

[Sch82] Georg Schnitger. A family of graphs with expensive depth reduction. *Theor. Comput. Sci.*, 18:89–93, 1982.

[Sch83] Georg Schnitger. On depth-reduction and grates. In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*, pages 323–328. IEEE Computer Society, 1983.

[SS78] John E. Savage and Sowmitri Swamy. Space-time trade-offs on the fft algorithm. *IEEE Transactions on Information Theory*, 24(5):563–568, 1978.

[SS79a] John E. Savage and Sowmitri Swamy. Space-time tradeoffs for oblivious interger multiplications. In Hermann A. Maurer, editor, *ICALP*, volume 71 of *Lecture Notes in Computer Science*, pages 498–504. Springer, 1979.

[SS79b] Sowmitri Swamy and John E. Savage. Space-time tradeoffs for linear recursion. In Alfred V. Aho, Stephen N. Zilles, and Barry K. Rosen, editors, *POPL*, pages 135–142. ACM Press, 1979.

[Tom78] Martin Tompa. Time-space tradeoffs for computing functions, using connectivity properties of their circuits. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 196–204, New York, NY, USA, 1978. ACM.

[Val77] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In Jozef Gruska, editor, *Mathematical Foundations of Computer Science 1977, 6th Symposium, Tatranska Lomnica, Czechoslovakia, September 5-9, 1977, Proceedings*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 1977.

A Memory-Hard Functions

We define MHFs in the Parallel Random Oracle Model (pROM) of [AS15]. For this we first define the model and associated complexity notions and then fix the exact notion of MHF.

The Parallel Random Oracle Model. We consider an arbitrary repeatedly invoked algorithm $\mathcal{A}$ executing in the parallel random oracle model (pROM) [AS15] of computation where we make states between invocations explicit as follows. At invocation $i \in \{1, 2, \ldots\}$ algorithm $\mathcal{A}$ is given the state (bit-string) σ_{i-1} it produced at the end of the previous invocation. Next $\mathcal{A}$ can make a batch of calls $\mathbf{q}_i = (q_{1,i}, q_{2,i}, \ldots)$ to the *fixed input length* random oracle H (a.k.a. an ideal compression function). Then it receives the response from H and can perform arbitrary computation before finally outputting an updated state σ_i. The initial state σ_0 contains the input to the computation which terminates once a special final state is produced by $\mathcal{A}$. Apart from the explicit states σ the algorithm may keep no other state between invocations. For an input x and coins r we denote by $\mathcal{A}(x; r; H)$ the corresponding (deterministic) execution of $\mathcal{A}$. We say that $\mathcal{A}$ is *sequential* if in no execution does it ever make a batch of queries $\mathbf{q}$ with $|\mathbf{q}| > 1$.

The *cumulative memory complexity* (CMC) is defined to be

$$\mathsf{cmc}(\mathcal{A}) = \mathop{\mathbb{E}}_{H}\left[\max_{x,r} \sum_i |\sigma_i|\right]$$

where $|\sigma|$ is the bit-length of state σ, the expectation is taken over the choice of H and $\max_{x,r}$ denotes the maximum over all inputs and coins of $\mathcal{A}$.

We also need the following two worst-case complexity notions. The *time complexity* (TC) $\mathsf{time}(\mathcal{A})$ is the maximum running time of $\mathcal{A}$ in any execution (over all choices of x, r and H). Similarly, the *space complexity* (SC) is the largest state it ever outputs in any execution.

$$\mathsf{space}(\mathcal{A}) = \max_{x,r,H} \{\max_i |\sigma_i|\}.$$

We remark that SC and TC are somewhat stricter than usual since we maximise over all choices of H. However, this can only help us as these measures are used as worst case estimates of the complexity for the honest party and we ask that an MHF has reasonable upper-bounds on their values.

An oracle function f is a function over strings which depends on the choice of H. Let $\mathbb{A}_{f,m,q}$ be the set of pROM algorithms which compute f on $m \in \mathbb{N}^+$ arbitrary (distinct) inputs making at most q queries to H. Then the *amortized cumulative memory complexity* (aCMC) of f is defined to be

$$\mathsf{cmc}_{m,q}(f) = \min \left\{ \frac{\mathsf{cmc}(\mathcal{A})}{n} \;:\; n \in [m], \mathcal{A} \in \mathbb{A}_{f,n,q} \right\}.$$

We comment on two differences between the above complexity notions and those in [AS15] (and why they do not prevent us from using the results in that work).

- In the definition of CMC above we maximize over all coins of $\mathcal{A}$ instead of including them in the expectation. This is with out loss of generality for the aCMC of a function since hardcoding the coins which minimize the aCMC of any $\mathcal{A}$ has at least as much CMC as the expected value for random coins.
- The aCMC of [AS15] is also parametrized by the minimum success probability ϵ of any algorithm computing f. Instead, for reasons of exposition, in this work we restrict ourselves to the special case when $\epsilon = 1$.

Memory-Hard Functions. As observed in [AS15] the aCMC of a function provides a good lower-bound on the amortized AT-complexity of that function. Thus the following definition captures the intuition of a memory-hard function in the pROM.

Definition A.1 (Memory-Hard Function) *Let $\{f_{\sigma,\tau}\}_{\sigma,\tau\in\mathbb{N}^+}$ be a family of (oracle) functions and $\mathcal{N}$ be a sequential pROM algorithm which, on input (σ, τ, x), outputs $f_{\sigma,\tau}(x)$ in time at most $\tau\sigma$ using space at most σ. Then $F = (\{f_{\sigma,\tau}\}, \mathcal{N})$ is an (h, g, t)-*memory-hard function (for up to m instances and q queries) *if it has* memory-hardness *at least h,* memory-gap *at most g and* throughput *at least t (all functions of σ and τ).*

$$\mathsf{cmc}_{m,q}(f_{\sigma,\tau}) \geq h(\gamma, \tau) \qquad \frac{\mathsf{space}(\mathcal{N}) * \mathsf{time}(\mathcal{N})}{\mathsf{cmc}_{m,q}(f_{\sigma,\tau})} \leq g(\sigma, \tau) \qquad \frac{\mathsf{space}(\mathcal{N})}{\mathsf{time}(\mathcal{N})} \geq t(\sigma, \tau).$$

In practice, we may content ourselves with families of infinite size but which do not contain a member for possible $k \in \mathbb{N}^+$ as long as the family is not too sparse (e.g. we have a function for all powers of 2).

Using the following theorem from [AS15] the results in this work imply MHFs with various desirable properties.

Theorem A.2 ([AS15]) *Let $G \in \mathbb{G}_{n,\delta}$, P be a sequential pebbling of G with $(\Pi_t(P), \Pi_s(P)) = (z_n, s_n - 1)$ and let H be a random oracle with $w > 13$ bits of output. Fix any $m, q \in \mathbb{N}^+$ subject to the following (reasonable) pair of constraints.*

- *Not too many copies of f are computed: $m \leq 2^{w-2}/n$.*
- *Not too many oracle queries are made: $q \leq 2^{w/2}$.*

Then there exists an oracle function f and pROM evaluation algorithm $\mathcal{N}$ for f with

$$\mathsf{time}(\mathcal{N}) = z_n \qquad \mathsf{space}(\mathcal{N}) = w * s_n \qquad \mathsf{cmc}_{m,q}(f) \geq \frac{w * \Pi^{\parallel}_{cc}(G)}{4}.$$

In particular this theorem shows that in order to construct an MHF with both memory-cost σ and time-cost τ parameters it suffices to construct a family of DAGs for every size (with high CC) together with matching sequential pebblings. For simplicity we state the theorem for DAGs with a single source and sink but it can be easily extended to the more general case.[7]

Corollary A.3 (High CC Graphs Imply MHFs) *Let $\{G_n \in G_{n,\delta_n}\}_{n=1}^{\infty}$ be a family of DAGs each with a single source and sink and let H be a random oracle with $w > 13$ bits of output. For each n let P_n be a sequential pebbling of G_n where $(\Pi_t(P_n), \Pi_s(P_n)) = (z_n, s_n - 1)$. Then there exists a (h, g, t)-MHF with*

$$h(\sigma,\tau) \geq \frac{w * \tau * \Pi^{\parallel}_{cc}(G_\sigma)}{4} \qquad g(\sigma,\tau) \leq \frac{4z_\sigma * s_\sigma}{\Pi^{\parallel}_{cc}(G_\sigma)} \qquad t(\sigma,\tau) \geq \frac{w * s_\sigma}{z_\sigma \tau}$$

Proof. The idea is simple. Fix any σ and τ. To obtain oracle function connect τ copies of the DAG G_σ in a chain and let $f_{\sigma,\tau}$ be the MHF given by Theorem A.2. The corresponding sequential pebbling $P_{\sigma,\tau}$ is simply the pebbling P_σ repeated τ times for each copy of G_σ with the following caveat. Whenever a pebble is placed on the source node of any copy of G_σ it is only removed once no more children of that node will be pebbled. Thus $\mathsf{space}(P_{\sigma,\tau}) \leq \mathsf{space}(P_\sigma) + w = w * s_\sigma$ and $\mathsf{time}(P_{\sigma,\tau}) = \tau * \mathsf{time}(P_\sigma) = \tau z_\sigma$. Finally, it is easy to see that $\Pi^{\parallel}_{cc}(G_{\sigma,\tau}) = \tau * \Pi^{\parallel}_{cc}(G_\sigma)$ since any pebbling of $G_{\sigma,\tau}$ with CC less than $\tau * \Pi^{\parallel}_{cc}(G_\sigma)$ would have to pebble at least one copy of G_σ with less than $\Pi^{\parallel}_{cc}(G_\sigma)$ which is a contradiction. $\square$

Two remarks are in order.

- In practice one would probably want a stronger connection between copies of G_σ. For example one could connect the last σ nodes pebbled by P_σ in one copy of G_σ to the first σ nodes pebbled by P_σ in the next copy of G_σ. This would affect neither the time nor space complexity of the honest evaluation algorithm but would potentially increase the concrete (though not asymptotic) memory-hardness of $f_{\sigma,\tau}$ which would also result in a smaller memory-gap. For the purpose of this work though the simple construction in the proof suffices as it has the same asymptotic behaviour.

- Estimating the effect of requiring a pebble on the source of each internal copy of G_σ to potentially require the space complexity to grow by 1 is extremely pesimistic. To the best of our knowledge the $\mathcal{N}$ algorithm for all MHF constructions in the literature as well the sequential pebbling of all DAGs in this work already keep such a pebble there (rather than repebble the source repeatedly). Thus the space complexity of the sequential pebbling when composing those DAGs would not grow by 1. Never the less, rather than make the corollary seem less general then it is (by requiring such a property from P) we have opted for its current form. In particular the difference is both asymptotically, and practically speaking, imaterial.

B Dispersed Graphs

B.1 Catena Bit Reversal

We begin with the Catena Bit Reversal graph. We first briefly recall the properties of the BRG^n_λ construction, relevant to our proof, summarized in the following lemma which follows easily from the definition of BRG^n_λ in [FLW13, Def. 8 & 9].

For this we describe the "bit-reversal" function from which the graph derives its name. Let $k \in \mathbb{N}^+$ such that $c = \log_2 k$ is an integer. On input $x \in [k]$ the *bit-reversal function* $\mathbf{br}(\cdot) : [k] \to [k]$ returns y such that the binary representation of x using c bits is the reverse of the binary representation of y using c bits.

Lemma B.1 (Catena Bit Reversal Graph) *Let $\lambda, n \in \mathbb{N}^+$ be such that $k = n/(\lambda + 1)$ is a power of 2. Let $G = \mathsf{BRG}^n_\lambda$ be the Catena Bit Reversal graph. Then the following holds:*

[7]Moreover any DAG can easily be extended to be of this form with no penalty to the CC as a function of its size.

1. G has n nodes.

2. Number them in topological order with the set $[0, n-1]$ and $\forall i \in [k]$ let node set $L_i = [ik, (i+1)k-1]$. A path runs through all nodes in each set L_i.

3. Node $ki + x \in L_i$ has an incoming edge from $k(i-1) + \mathbf{br}(x) \in L_{i-1}$.

We are ready to state and prove the lowerbound.

Lemma B.2 *It holds that $\mathsf{BRG}_\lambda^n \in \mathbb{D}_{1,\sqrt{k}}^{\lambda,k}$ where $k = \frac{n}{(\lambda+1)}$ and $\Pi_{cc}^{\parallel}(\mathsf{BRG}_\lambda^n) = \Omega\left(\frac{n^{1.5}}{\sqrt{\lambda}}\right)$.*

Proof of Lemma B.2. Let $G = \mathsf{BRG}_\lambda^n$ and set $k = n/(\lambda+1)$, $c = \log_2 k$ and $g = \sqrt{k}$. By construction c is an integer. For simplicity assume c is even and so $g \in \mathbb{N}^+$.[8] Number the nodes of G according to (the unique) topological order with the set $[0, n-1]$. It suffices to show that for all $i \in [\lambda]$ the subgraph G_i consisting of nodes $[(i+1)k-1]$ is (g,k)-dispersed (with probability $\epsilon = 1$). If this holds then Theorem 6.7 immediately implies that $\Pi_{cc}^{\parallel}(\mathsf{BRG}_\lambda^n) = \Omega\left(\frac{n^{1.5}}{\sqrt{\lambda}}\right)$.

Fix any $i \in [\lambda]$ and let the nodes of layer L_i be the nodes $[ik, (i+1)k-1]$. We represent the nodes of L_i with bit strings of length c. For $j \in [k/2g]$ let $L_{i,j} = \{ik + 2gj + 1, \ldots, ik + 2gj + 2g\}$ be the j'th interval of length $2g$ in layer L_i and let $R = \{ik + 2gj + g + 1, \ldots, ik + 2gj + 2g\}$ be the second half of $L_{i,j}$. We will show that there are g node-distinct paths terminating in R with all other nodes in layer L_{i-1}. Define set $S_x = [s + y - (g-2), s + y]$ where $y = \mathbf{br}(x)$ and $s = (i-1)2^{(c+1)}$. The next three properties are follow immediately from Lemma B.1 and they imply the lemma.

- $\forall x \in R$ it holds that $S_x \subset L_{i-1}$.
- $\forall x \in R$ there is a path of length g going through the nodes of S_x and ending in x.
- $\forall$ distinct $x, x' \in R$ sets S_x and $S_{x'}$ are disjoint.

B.2 Catena Double Butterfly

Next we focus on the Catena Double Butterfly graph. We begin with a lemma (which follows immediatly by inspection of the Catena Double Butterfly definition [FLW13, Def. 10 & 11]) summarizing the properties of that construction relevant to our proof.

Lemma B.3 (Catena Double Butterfly Graph) *Let $\lambda, n \in \mathbb{N}^+$ such that $n = \bar{n}(\lambda(2c-1)+1)$ where $\bar{n} = 2^c$ for some $c \in \mathbb{N}^+$. Then the* Catena Double Butterfly Graph *DBG_λ^n consists of a stack of λ subgraphs such that the following holds.*

1. The graph DBG_λ^n has n nodes in total.

2. The graph DBG_λ^n built as a stack of λ subgraphs $\{G_i\}_{i \in [\lambda]}$ each of which is a superconcentrator. In the unique topological ordering of DBG_λ^n denote the first and final $\bar{n}$ nodes of each G_i as $L_{i,0}$ and $L_{i,1}$ respectively. Then there is a path running through all nodes in each $L_{i,1}$.

3. Moreover, for any $i \in [\lambda]$ and subsets $S \subset L_{i,0}$ and $T \subset L_{i,1}$ with $|S| = |T| = h \le \bar{n}$ there exist h node disjoint paths $p_1, \ldots, p_h$ of length $2c$ from S to T.

We are ready to state and prove the lowerbound.

Lemma B.4 *Let $\lambda, n \in \mathbb{N}^+$ such that $n = \bar{n}(\lambda(2c-1)+1)$ with $\bar{n} = 2^c$ for some $c \in \mathbb{N}^+$. It holds that $\mathsf{DBG}_\lambda^n \in \mathbb{D}_{1,g}^{\lambda,n}$ for $g = \lceil\sqrt{\bar{n}}\rceil$ and $\Pi_{cc}^{\parallel}(\mathsf{DBG}_\lambda^n) = O\left(\frac{n^{1.5}}{c\sqrt{c\lambda}}\right)$.*

[8]The odd case is identical but with messy but inconsequential rounding terms.

Proof of Lemma B.4. Let $G = \mathsf{DBG}^n_\lambda$ and let $G_1, G_2, \ldots, G_\lambda$ be the subgraphs of G described in Lemma B.3. We will show that each G_i is $(g, \bar{n})$-dispersed for $g = \lfloor\sqrt{\bar{n}}\rfloor$. Fix arbitrary $i \in [\lambda]$ and L_1 be the last $\bar{n}$ nodes in the (the unique) topological ordering of G_i. We identify the nodes in L_1 with the set $\{1\} \times [\bar{n}]$ such that the second component follows their topological ordering. Let $\bar{g} = \lfloor \bar{n}/g \rfloor$ and for each $j \in [\bar{g}]$ let $L_{1,j} = \{\langle 1, ig + x\rangle : x \in [0, g-1]\}$ ik+2gj+2g We will show that $L_{1,j}$ has a (g, g)-dependency.

Let L_0 be the first $\bar{n}$ nodes of G_i which identify with the set $\{0\} \times [\bar{n}]$ (again with the second compenent respecting their ordering). Notice that for $n > 1$ and $g = \lfloor\sqrt{\bar{n}}\rfloor$ it holds that $g(g - 2c + 1) \leq n$. Thus the set $S = \{\langle 0, i(g - 2c + 1)\rangle : i \in [g]\}$ is fully contained in L_0. Property (3) of Lemma B.3 implies there exist g node disjunct paths from S to $L_{1,j}$ of length $2c$. In particular $L_{1,j}$ has a $(2c, g)$-dependency.

We extend this to a (g, g)-dependency. Let path p, begining at node $\langle 0, v\rangle \in S$, be a path in the $(2c, g)$-dependency of $L_{1,j}$. Prepend to p the path traversing

$$(\langle 0, v - (g - 2c - 1)\rangle, \langle 0, v - (g - 2c - 2)\rangle, \ldots, \langle 0, v\rangle)$$

to optain a new path p^+ of length g. As this is a subinterval of L_0 property (2) of Lemma B.3 implies this prefix path always exists. Moreover since any paths $p \neq q$ in a $(g, 2c)$-dependency of $L_{1,i}$ are node disjunct they must, in particular, also begin at distinct nodes $\langle 0, v_p\rangle \neq \langle 0, v_q\rangle$ in S. But by construction of S any such pair of nodes is seperated by $g - 2c$ nodes. In particular paths p^+ and q^+ are also node disjoint and so by extending all paths in a $(2c, g)$-dependency we obtain a (g, g)-dependency for $L_{1,i}$. This concludes the first part of the lemma.

It remains to lowerbound $\Pi^{\parallel}_{cc}(\mathsf{DBG}^n_\lambda)$ using Theorem 6.7.

$$\Pi^{\parallel}_{cc}(\mathsf{DBG}^n_\lambda) \geq \lambda g\left(\frac{k}{2} - g\right) \geq \lambda\left\lfloor\sqrt{\bar{n}}\right\rfloor\left(\frac{\bar{n}}{2} - \left\lfloor\sqrt{\bar{n}}\right\rfloor\right) = \lambda\sqrt{\bar{n}}\left(\frac{\bar{n}}{2} - \sqrt{\bar{n}}\right) - O(\bar{n}) = \Omega\left(\lambda\bar{n}^{1.5}\right) = \Omega\left(\frac{n^{1.5}}{c\sqrt{c\lambda}}\right).$$

B.3 Balloon Hashing Linear Graph

The graph $G = \mathsf{Lin}^\sigma_\tau$ is a pseudo-randomly constructed τ-layered graph with $\mathsf{indeg}(G) = 21$. It is defined as follows:

$G = (V, E)$ has n nodes $V = \{0, \ldots, n-1\}$, and G contains a path $0, 1, 2, \ldots, n-1$ running through V.

For $0 \leq i < \tau$ let $L_i = [i\sigma, (i+1)\sigma - 1]$ denote the i'th layer. For each node $x \in L_i$, with $i > 0$, we select 20 nodes $y_1, \ldots, y_n \in L_{i-1}$ (uniformly at random) and add the directed edges $(y_1, x), \ldots, (y_{20}, x)$ to E.

Lemma B.5 *If $\sigma, \tau \in \mathbb{N}^+$ such that $n = \sigma * \tau$ then with high probability it holds that*

$$\mathsf{Lin}^\sigma_\tau \in \mathbb{G}_{n,21} \qquad \mathsf{Lin}^\sigma_\tau \in \mathbb{D}^{\tau,\sigma}_{0.25,\sqrt{\sigma}/2} \qquad \Pi^{\parallel}_{cc}(\mathsf{Lin}^\sigma_\tau) = \Omega\left(\frac{n^{1.5}}{\sqrt{\tau}}\right).$$

Proof. (sketch) It suffices to show that $\mathsf{Lin}^\sigma_\tau \in \mathbb{D}^{\tau-1,\sigma}_{0.25,\sqrt{\sigma}/2}$. By Theorem 6.7 it immediately follows that

$$\Pi^{\parallel}_{cc}(\mathsf{Lin}^\sigma_\tau) \geq \frac{(\tau-1)\sqrt{\sigma}/2}{4}\left(\sigma/2 - \sqrt{\sigma}/2\right) = \Omega\left(\frac{n^{1.5}}{\sqrt{\tau}}\right).$$

Consider layer L_i ($\tau \geq i > 0$) and given let $S_x = \{x, \ldots, x + \sqrt{\sigma}/2 - 1\} \subset L_i$ denote a segment of $\sqrt{\sigma}/2$ consecutive nodes in L_i. Without loss of generality we suppose that each node in S_x only has one randomly chosen parent in L_{i-1} — adding additional edges can only improve dispersity. We partition L_{i-1} into $\sqrt{\sigma}$ intervals of length $\sqrt{s}$. We say that an interval $\{u, \ldots, u + \sqrt{s} - 1\} \subset L_{i-1}$ is covered by S_x if there is a directed edge (y, v) from $y \in \{u + \sqrt{s}/2, \ldots, u + \sqrt{s} - 1\}$ (the last half of the interval) to some $v \in S_x$. In this case the path $u, u+1, \ldots, y, v$ has length $\geq \sqrt{s}/2$ and this path will not intersect the corresponding paths from any of the other (disjoint) intervals in L_{i-1} (recall that we are assuming that $v \in S_x$ only has one parent in L_{i-1}). The probability that an interval $\{u, \ldots, u + \sqrt{s} - 1\} \subset L_{i-1}$ is covered by S_x is at least

$$1 - \left(1 - \frac{\sqrt{\sigma}/2}{\sigma}\right)^{\sqrt{\sigma}} \approx 1 - \sqrt{1/e} \ .$$

Thus, in expectation we will have at least $\mu = \sqrt{\sigma}\left(1 - \sqrt{1/e}\right) \geq 0.39 \times \sqrt{\sigma}$ node disjoint paths of length $\sqrt{\sigma}/2$ ending in S_x. Standard concentration bounds imply that we will have at least $\sqrt{\sigma}/4$ such paths with high probability.

□

C Missing Proofs

Reminder of Theorem 3.2. *Let DAG $G = (V, E)$ be (e, d)-depth-robust and let $S, T \subset V$ such that*

$$|S| \leq \sigma n \quad , \quad |\mathsf{ancestors}_{G-S}(T)| \geq \tau n \quad \textit{and} \quad T \cap S = \emptyset$$

Then the cost of pebbling $G - S$ with target set T is $\Pi^{\parallel}_{cc}(G - S, T) > (e - \sigma n)(d - (1 - \tau)n)$.

Proof of Theorem 3.2. Set $\bar{e} = (e - \sigma n)$ and $\bar{d} = d - (1 - \tau)n$. As G is (e, d)-depth-robust the DAG $G - S$ is $(\bar{e}, d)$-depth-robust: assume for the sake of contradiction there exits $B \subset V(G - S)$ of size $|B| \leq \bar{e}$ s.t. $G - (S \cup B)$ has no path of length d, but then $S' = S \cup B$ is a set of size $|S'| = |S| + |B| \leq \sigma n + \bar{e} = e$ where $G - S'$ has no path of length d contradicting G's (e, d)-depth robustness.

Now consider the subgraph G_T of $G - S$ which contains only the vertices $\mathsf{ancestors}_{G-S}(T)$, note that $\Pi^{\parallel}_{cc}(G - S, T) = \Pi^{\parallel}_{cc}(G_T)$. Below we'll show that G_T is $(\bar{e}, \bar{d})$-depth-robust, using this and Corollary 3.1 we further get

$$\Pi^{\parallel}_{cc}(G - S, T) = \Pi^{\parallel}_{cc}(G_T) > \bar{e} \cdot \bar{d} = (e - \sigma n)(d - (1 - \tau)n)$$

which proves the theorem.

We still must show that G_T is $(\bar{e}, \bar{d})$-depth-robust. Assume for contradiction that there exists a set $A \subset V(G_T), |A| \leq \bar{e}$ s.t. $G_T - A$ has no path of length $\bar{d}$. As G_T is a subgraph of $G - S$ and $G - S$ has $(1 - \tau)n$ vertices more than G_T we have no path of length $d = \bar{d} + (1 - \tau)n$ in $G - (S \cup A)$. This contradicts the $(\bar{e}, d)$-depth-robustness of $G - S$.

Reminder of Corollary 3.3. *For some constants $c_1, c_2 > 0$ there exists an infinite family of DAGs $\{G_{n,\delta} \in \mathbb{G}_{n,\delta}\}_{n=1}^{\infty}$ with $\delta \leq c_1 \log(n)$ and $\Pi^{\parallel}_{cc}(G) \geq c_2 n^2$.*

This is optimal in the sense that for any family $\{\delta_n \in [n]\}_{n=1}^{\infty}$ and $\{J_n \in \mathbb{G}_{n,\delta_n}\}_{n=1}^{\infty}$ it holds that $\Pi^{\parallel}_{cc}(J_n) \in O(n^2)$. Moreover if $\delta_n = o(\log(n)/\log\log(n))$ then $\Pi^{\parallel}_{cc}(J_n) = o(n^2) = o(\Pi^{\parallel}_{cc}(G_n))$.

Proof of Corollary 3.3. Take $\{G_n \in \mathbb{G}_{n,c_3 \log(n)}\}_{n=1}^{\infty}$ to be the family of DAGs from Theorem 2.2. Now the first and second statements follow immediately from Theorem 3.2 and the observation that $\Pi^{\parallel}_{cc}(J_n) \leq n^2$ for any n node DAG J_n. The final statement is based on the simple observation that $\Pi^{\parallel}_{cc}(J_n) = o\big(n^2\big)$ whenever $\delta_n = o\big(\log(n)/\log\log(n)\big)$ because any such DAG is $(o(n), o(n/\delta_n))$-reducible. We can see this by applying Lemma C.1, due to Valiant [Val77], $3\log(\delta_n)$ times to obtain a set S of at most

$$e = |S| \leq \frac{3\log(\delta_n) n \delta_n}{\log(n) - 2\log(\delta_n)} = o(n)$$

nodes such that $d = \mathsf{depth}(G) \leq 2^{-3\log(\delta_n)} n = o(n/\delta_n^2)$. Now by Theorem 2.5 we have $\Pi^{\parallel}_{cc}(G) = O(ne + n\sqrt{dn\delta_n}) = o(n^2)$. □

Lemma C.1 ([Val77] Extension) *Given a DAG G with m edges and depth $\mathsf{depth}(G) \leq d = 2^i$ there is a set of m/i edges s.t. by deleting them we obtain a graph of depth at most $d/2$.*

Reminder of Theorem 5.2. *Let $m \in \mathbb{N}_{\geq 2}$ and $n = 2^m$ and let $0 < \alpha, \beta < 1$ be constants. If H_n is $(\frac{\alpha n}{\log(n)}, \beta n)$-depth-robust then the following hold for $G_n = G(H, n)$ of Construction 5.1.*

1. *$G_n \in \mathbb{G}_{N,\delta}$ where $N = n\log(n)$ and $\delta \leq \mathsf{indeg}(H_n) + 3$.*
2. *$\Pi^{\parallel}_{cc}(G_n) \geq \gamma \cdot N^2/\log(N) = \Omega(N^2/\log(N))$ where $\gamma = \max_{0<\alpha_0<\alpha,\beta}\big(\min\{\alpha_0/8, (\alpha-\alpha_0)(\beta-\alpha_0)/2\}\big)$.*
3. *$\Pi_{st}(G_n) \in O(N^2/\log(N))$.*

Proof of Theorem 5.2. The size and (maximal) indegree of G_n follow by inspection.

We describe the sequential algorithm $\mathcal{N}$ for pebbling G_n. Intuitively it pebbles G_n in topological order only removing pebbles from a layer once the subsequent layer has been completely pebbled.

More precisely, for $i \in [m]$ denote layer i of G_n with $L_i = \{\langle i, v\rangle : v \in \bar{V}\}$. First we sort the nodes of G_n according to their layer beginning at L_1 and ending with L_n. We further sort each layer according to the topological ordering $(v_1, v_2, \ldots, v_n)$ of H. Finally we number the resulting sequence of nodes using the set $[N]$. Now algorithm $\mathcal{N}$ can be described as follows.

1. At time t pebble node $t \in [N]$.
2. At time $t \in \{ni : i \in [2, m]\}$ remove all pebbles from $L_{(t/n)-1}$.

Clearly $\mathcal{N}$ will pebble the sink of G_n in N steps. Moreover it never stores more then 2 layers at a time, each of which has size $n = N/m \leq N/\log(N)$. Thus we have $\Pi_{st}(\mathcal{N}(n)) \leq \frac{2N^2}{\log(N)}$. To see that $\mathcal{N}$ always produces a legal pebbling notice that it pebbles G_n in topological order. Moreover, since edges only connect nodes in adjacent layers or in the same layer, $\mathcal{N}$ only removes pebbles from nodes once all of their children have been pebbled.

It remains to lower-bound $\Pi^{\parallel}_{cc}(G_n)$ in terms of n. Fix some $n = 2^m$ with $m \in \mathbb{N}_{\geq 2}$ and let $G = G_n$. To see why the lower-bound on $\Pi^{\parallel}_{cc}(G)$ in terms of γ holds let $P = (P_1, P_2, \ldots)$ be a legal pebbling of G. For all $i \in [m]$ let time interval $(\bar{t}_{i,1}, \bar{t}_{i,2}, \ldots, \bar{t}_{i,n})$ be such that nodes $\langle i, v_1\rangle$ and $\langle i, v_n\rangle$ are pebbled for the first time in P at steps $\bar{t}_{i,1}$ and $\bar{t}_{i,n}$ respectively. Notice that (due to step 2 of the construction) we have $\bar{t}_{i,n} - \bar{t}_{i,1} \geq n$. To prove the theorem it suffices to prove the following inequality which lower-bounds the cost of pebbling each layer of G.

$$\forall i \in [3m/4, m] \quad \sum_{j \in [\bar{t}_{i,1}, \bar{t}_{i,n}]} |P_j| \geq 4\gamma n^2. \tag{2}$$

In particular the theorem follows since

$$\Pi^{\parallel}_{cc}(G) = \sum_{i\in[m]} \sum_{j\in[\bar{t}_{i,1}, \bar{t}_{i,n}]} |P_j| \geq 4\gamma n^2 m/4 = \gamma N^2/log(N).$$

To show Equation 2 fix arbitrary $i \in [3m/4, m]$ and fix α_0^* such that $\gamma = \max_{0<\alpha_0<\alpha,\beta}\big(\min\{\alpha_0/8, (\alpha - \alpha_0)(\beta-\alpha_0)/2\}\big) = \min\{\alpha_0^*/8, (\alpha-\alpha_0^*)(\beta-\alpha_0^*)/2\}$. Let $x = \min\{|P_{j+\bar{t}_{i,1}}| : j \in [n/2]\}$ be the smallest number of pebbles simultaneously on G during the first $n/2$ steps after we start to pebble L_i. If $x \geq \alpha_0^* n/4$ then we are done since that would mean that

$$\sum_{j\in[\bar{t}_{i,1}, \bar{t}_{i,n}]} |P_j| \geq \sum_{j\in[n/2]} |P_{j+\bar{t}_{i,1}}| \geq (n/2) * (\alpha_0^* n/4) = \alpha_0^* n^2/8 \geq \gamma n^2.$$

Assume instead that $x < \alpha_0^* n$ and let $t \in [\bar{t}_{i,1}, \bar{t}_{i,n}]$ be a time step when x pebbles are on G. We call a node $\langle i', v\rangle$ *good* if the following holds:

1. $i' < i$, $v \in \bar{V}$
2. $\exists u \in \{v_{n/2}, v_{(n/2)+1}, \ldots, v_n\}$ such that there is an unpebbled path (at time t) from $\langle i', v\rangle$ to $\langle i, u\rangle$.

Claim C.2 *$\forall i' \in [i-1]$ at least $(1-\alpha_0^*)n$ nodes in $L_{i'}$ are good.*

PROOF. To see this consider the paths of the form $p_v = (\langle i', v\rangle, \langle i'+1, v\rangle, \ldots, \langle i-1, v\rangle)$. Notice that for any distinct pair $u, v \in \bar{V}$ the paths p_v and p_u are node disjunct. Thus at most $\alpha_0^* n$ of them can contain a pebble. What's more, the last node on each path is connected to a node in the second half of L_i, none of which have yet been pebbled (as $t < \bar{t}_{i,n/2+1}$). Thus all unpebbled paths begin at good nodes. In particular, on $L_{i'}$ we have at least $n - \alpha_0^* n = (1-\alpha_0)n$ good nodes. □

The following lemma lower-bounds the cost of (re)pebbling any layer bellow L_i incurred while pebbling the second half of L_i.

Claim C.3 *For all $l \in [i]$ set $b_i = \frac{i}{\log(n)} - \frac{1}{4}$. Then for $K_i = \{k \in [i-1] : |P_t \cap L_k| \leq \alpha_0^*(n/\log(n))\}$*

1. *$|K_i| \geq b_i * \log(n) \geq 1$*
2. *$\forall k \in K_i \quad \sum_{j \in [\bar{t}_{i,1}, \bar{t}_{i,n}]} |P_j \cap L_k| \geq (\alpha - \alpha_0^*)(\beta - \alpha_0^*) n^2$.*

Equation 2 (and thus the theorem) follows from Claim C.3. Indeed, since $b_i \geq (1/2)$ and $\gamma \leq \big(\alpha - \alpha_0^*\big)\big(\beta - \alpha_0^*\big)/2$ we see that:

$$\begin{aligned}
\sum_{j \in [\bar{t}_{i,1}, \bar{t}_{i,n}]} |P_j| &\geq \sum_{\ell \in [i-1]} \sum_{j \in [\bar{t}_{i,1}, \bar{t}_{i,n}]} |P_j \cap L_\ell| \\
&\geq \sum_{\ell \in K_i} \sum_{j \in [\bar{t}_{i,1}, \bar{t}_{i,n}]} |P_j \cap L_\ell| \\
&\geq b_i \log(n) * (\alpha - \alpha_0^*)(\beta - \alpha_0^*) n^2 \\
&\geq \gamma N^2/\log(N) \ .
\end{aligned}$$

PROOF. (Of Claim C.3) We begin with point 1 of the claim. Suppose, for the sake of contradiction, that $K_i < b_i * \log(n)$. Then the number of pebbles on G at time t is (at least)

$$|P_t| \geq (i - b_i \log(n)) * \frac{\alpha_0^* n}{\log(n)} = n\left(\frac{\alpha_0^* i}{\log(n)} - \alpha_0^* b_i\right) = n\left(\frac{\alpha_0^* i}{\log(n)} - \frac{\alpha_0^* i}{\log(n)} + \frac{\alpha_0^*}{4}\right) = \alpha_0^* n/4$$

which is a contradiction to our assumption that less than $\alpha_0^* n/4$ pebbles are on G at time t.

We turn to the second part of the claim. By definition, at time $t < \bar{t}_{i,z}$ there is an unpebbled path from each good node in L_k and the second half of L_i. Thus, in order to pebble the second half of L_i, at some point during the interval $[t, \bar{t}_{i,n}]$ each of these good nodes must first be pebbled.

Let $S = P_t \cap L_k$ and T be the set of good nodes in L_k. Then Theorem 3.2 implies that the cost of (re)pebbling T is at least $\big(e - |S|\big)\big(d - (n - |T|)\big)$, where $e = \alpha n/\log(n)$ and $d = \beta n$. In particular the cost is at least

$$\begin{aligned}
\sum_{j \in [t, \bar{t}_{i,n}]} |P_j \cap L_k| &\geq \left(\frac{\alpha n}{\log(n)} - \frac{\alpha_0^* n}{\log(n)}\right)\big(\beta n - \big(n - (1-\alpha_0^*)n\big)\big) \\
&= (\alpha - \alpha_0^*)(\beta - \alpha_0^*) \frac{n^2}{\log(n)} \ .
\end{aligned}$$

□

Reminder of Lemma 6.3. *For any $e \geq \sqrt{n}$ any any $\delta \geq 2$ a (n, δ, n)-random DAG will be $\left(e, \Omega\left(\frac{n^2}{e^2 \log(n)}\right), \frac{n}{100e}\right)$-block depth robust with probability $1 - o\big(n^{-3}\big)$.*

Proof of Lemma 6.3. Let G be a random (n, δ, n)-random DAG $G = (V, E)$ with nodes $V = [n]$. Fix an arbitrary integer $m \in [n]$ and set $n' = \lfloor n/m \rfloor$. We will define a DAG G_m called the meta-graph of G. For this we use the following sets. For all $i \in [n']$ let $S_i = [(i-1)m+1, im] \subseteq V$. Moreover we denote the first and last thirds respectively of S_i with

$$S_i^F = [(i-1)m+1, (i-1)*m+\lfloor m/3 \rfloor] \subseteq S_i \qquad\qquad S_i^L = [(i-1)m+\lceil 2m/3 \rceil+1, im].$$

We define the meta-graph $G_m = (V_m, E_m)$ as follows:

Nodes: V_m contains one node v_i per set S_i. We call v_i the *simple node* and S_i its *meta-node.*

Edges: If the end of a meta-node is connected to the begining of another meta-node we connect their simple nodes.

$$V_m = \{v_i : i \in [n']\} \qquad\qquad E_m = \{(v_i, v_j) : E \cap (S_i^L \times S_j^F) \neq \emptyset\}.$$

Lemma 6.3 now follows from the next two claims.

Claim C.4 *If G_m is (e,d)-depth robust then G is $\left(e/\left(1+\lceil \frac{n}{100em} \rceil\right), dm/3, \frac{n}{100e}\right)$-block depth robust.*

Proof. (sketch) Notice meta-nodes are disjunct subsets. Fix any set $S \subseteq V$ of size e. The set

$$N\left(S, \frac{n}{100e}\right) = \bigcup_{v \in S} \left\{v - \frac{n}{100e}+1, \ldots, v\right\}$$

can intersect at most $e\left(1+\lceil \frac{n}{100em} \rceil\right)$ meta-nodes because for each $v \in S$ the set $\left\{v-\frac{n}{100e}+1,\ldots,v\right\}$ intersects at most $\left(1+\lceil \frac{n}{100em} \rceil\right)$ meta-nodes. Remove the simple nodes corresponding to $N\left(S, \frac{n}{100e}\right)$ from G_m and there remains a path ϕ' in G_m of length d. Thus, after removing S from G there must remain a path ϕ going through the middle thirds of the d meta-nodes corresponding to ϕ'. In particular we get that

$$\mathsf{length}(\phi) \geq \frac{\mathsf{length}(\phi')*m}{3} \geq \frac{dm}{3}.$$

□

Claim C.5 *With high probability G_m is $\left(n'/10, m/(200\log(n))\right)$-depth robust.*

Proof. (sketch) Divide G_m into $d = m/(100\log(n))$ layers $L_1, \ldots, L_d$ each containing n'/d meta-nodes. We first observe that (with high probability) each fixed pair of layers L_i and L_j is connected with a γ-bipartite expander graph: Fix any pair of layers L_i and L_j $(i<j)$ and let $S \subset L_i$ and $V \subset L_j$ be such that $|S| > \gamma n'/d$ and $|V| > \gamma n'/d$ then $|\mathsf{parents}(V) \bigcap S| > 0$ — for a small value γ (say $\gamma \leq 0.1$). To see why this expansion property holds (whp) we note that any two meta-nodes are connected with probability at least $\frac{m^2}{3^2 n}$. Thus, a meta-node in L_i is connected to $\frac{n'}{d} \times \frac{m^2}{9n} \geq 10\log(n)$ random meta-nodes in L_j. Thus, except with very small probability, L_i and L_j will be connected with a γ-bipartite expander graph and we can union bound over all $\binom{d}{2}$ pairs $i<j$ to show that this holds for every pair with high probability[9].

Suppose that the above expansion property holds, and suppose that we remove a set S of $n'/10$ meta-nodes from G_m. Call a layer L_i good if $|S \bigcap L_i| \leq \frac{n'}{5d}$. By Markov's inequality we have at least $d/2$ good layers. Let $Y_1, \ldots, Y_{d/2}$ denote $d/2$ good layers. Now we claim that there is a path traversing through every good layer (hence, $\mathsf{depth}(G_m - S) \geq d/2$. Let $R_1 = Y_1 - S$ denote the set of meta-nodes in the first good layer which remain in $G_m - S$. In general, $R_{i+1} = \mathsf{parents}\left(Y_{i+1}-S\right) \bigcap (R_i - S)$ denotes the set of meta-nodes in good layer Y_{i+1} that are reachable by a path in $G_m - S$. Now a simple inductive argument shows that $|R_i| \geq \frac{4n'}{5d}$ for all $i \leq d/2$.

□

[9]It is well known that a random bipartite graph with degree $d = \Omega(1)$ is an expander with good probability (e.g., see Erdös et al. [EGS75]). In our case we have degree $d = \Omega(\log(n))$ so the bipartite graph with be a γ expander with overwhelming probability.

From Claim C.4 and Claim C.5 it follows that G is $\left(\frac{\lfloor n/m \rfloor}{10\left(1+\lceil \frac{n}{100em} \rceil\right)}, m^2/(600\log(n)), \frac{n}{100e}\right)$-depth robust. Setting $m = 20n/e$ we obtain the desired result. □

Reminder of Theorem 6.2. *Let G be a (n, δ, n)-random DAG then, except with probability $o\left(n^{-3}\right)$, we have*

$$\Pi_{cc}^{\|}(G) = \tilde{\Omega}\left(n^{5/3}\right) .$$

Proof of Theorem 6.2. Let G be a (n, δ, n)-random DAG, and let $G_1 = G - \{n/2+1, \ldots, n\}$ denote the subgraph on the first $n/2$ nodes. We observe that G_1 is itself a $(n/2, \delta, n/2)$-random DAG.

By setting $e = n^{2/3}$ Lemma 6.3 we can find some fixed constant $c > 0$ such that our graph G_1 is $\left(n^{2/3}, cn^{2/3}/\log(n), n^{1/3}/100\right)$-block depth robust except with probability $o\left(n^{-3}\right)$.

Now consider any fixed segment $B_v = \{v, v+1, \ldots, v+10\tau n^{2/3}\log(n)-1\} \subset \{n/2+1, \ldots, n\}$. That is, B_v is a segment of $10\tau n^{2/3}\log(n)$ consecutive nodes in the last half of G. Let $B_v^{first} = \{v, v+1, \ldots, v+5n^{2/3}\tau\log(n)-1\}$ and $B_v^{last} = \{v+5n^{2/3}\tau\log(n), \ldots, v+10n^{2/3}\tau\log(n)-1\}$. We first note that we can select $\tau = O(1)$ such that for every segment $B_u = \{u, u+1, \ldots, u+n^{2/3}\} \subset V(G_1)$ of $n^{2/3}/100$ consecutive nodes in G_1 there is an edge from B_u to B_v^{last} with high probability.

Now we consider the cost incurred while pebbling B_v. Let t_1 (resp. t_2, t_3) denote the first time at which we place a pebble on node v (resp. node $v+5n^{2/3}\tau\log(n)-1$, node $v+10n^{2/3}\tau\log(n)-1$).

Claim C.6 *Let $P_0, \ldots, P_t$ denote a pebbling of G then*

$$\sum_{i\in[t_1,t_3]} |P_i| = \tilde{\Omega}\left(n^{4/3}\right) .$$

Proof. There are two cases:

1. For every $i \in [t_1, t_2]$ we have $|P_i| \geq e$ pebbles on G. Total Cost: $\sum_{i\in[t_1,t_2]} |P_i| \geq e(t_2+1-t_1) = \Omega\left(n^{4/3}\right)$.
2. For some $t' \in [t_1, t_2]$ we have at most $|P_{t'}| < e$ pebbles on G.

In the second case we define $S_0, \ldots, S_{d-1}$ where

$$S_i = \bigcup_{j\in[t',t_3]:j\equiv i \mod d} \left(P_i \cap V(G_1)\right) .$$

We can find some i such that $|S_i| \leq \sum_{i\in[t_1,t_3]} |P_i|/d$. Now we observe that, because (whp) there is a backedge from B_v^{last} to every set B_u of $n^{2/3}/100$ consecutive nodes in G_1, every node in

$$V(G_1) - \bigcup_{v\in S_i} \left\{v - \frac{n^{1/3}}{100}+1, \ldots, v\right\}$$

must be pebbled at some point in time in the interval $[t', t_3]$. This means that

$$\mathsf{depth}\left(G - \bigcup_{v\in S_i} \left\{v - \frac{n^{1/3}}{100}+1, \ldots, v\right\}\right) \leq d .$$

By the $\left(n^{2/3}, cn^{2/3}/\log(n), n^{1/3}/100\right)$-block depth robustness of G_1 it follows that $\sum_{i\in[t',t_3]} |P_i| = \tilde{\Omega}\left(n^{4/3}\right)$. □

It follows from Claim C.6 that the total cost of a pebbling is at least $\frac{n}{2|B_v|}\tilde{\Omega}\left(n^{4/3}\right) = \tilde{\Omega}\left(n^{5/3}\right)$. □

Reminder of Theorem 6.7.

$$G \in \mathbb{D}_{\epsilon,g}^{\lambda,k} \quad \Rightarrow \quad \Pi_{cc}^{\|}(G) \geq \epsilon\lambda g\left(\frac{k}{2}-g\right) .$$

Proof of Theorem 6.7. We number the nodes of $G = (V, E)$ in a topological order with the set $[n]$. Let $\{L_i \subseteq V\}$ be the λ disjunct node subsets described in Definition 6.6. For any L_i and $j \in [\lfloor k/2g \rfloor]$ let $L_{i,j}$ be the j^{th} interval of $2g$ consecutive nodes in L_i. That is if $L_i = [a, a+k-1]$ then $L_{i,j} = [a+i2g, a+(i+1)2g-1)]$. Let G_i denote the subgraph of G consisting of the nodes up to the end of L_i. That is G consists exactly of the node set $[a + (i+1)2g - 1]$ and edges of G between those nodes.

Let $P = (P_1, P_2, \ldots)$ be a legal pebbling of G and let $\bar{t}_v$ denote the first time step at which node v is pebbled by P. We use the following shorthand:

$$c_{i,j} := \sum_{i=\bar{t}_a}^{\bar{t}_z} |P_i| \qquad\qquad c_i := \sum_{i=\bar{t}_\alpha}^{\bar{t}_\omega} |P_i|$$

where a and z are the first and last nodes of $L_{i,j}$ and α and ω are the first and last nodes of L_i respectively. In particular $\mathsf{cc}(P) \geq \sum_{i\in[\lambda]} c_j$ and our goal is to lowerbound this sum. The theorem follows from the next claim.

Claim C.7 *For all $i \in [\lambda]$ and $j \in [\lfloor k/2g \rfloor]$ we have $c_{i,j} \geq \epsilon g^2$.*

In particular, from Claim C.7, it immediately follows that

$$c_i \geq \left\lfloor \frac{k}{2g} \right\rfloor (\epsilon g^2) \geq \epsilon g \left(\frac{k}{2} - g\right).$$

Therefore it follows that

$$\Pi^{\parallel}_{cc}(G) \geq \sum_{i\in[\lambda]} c_i \geq \epsilon\lambda g \left(\frac{k}{2} - g\right).$$

It remains only to prove Claim C.7. □

Proof of Claim C.7. Fix any i and j as in the claim. Let L and R be the first and second halves of $L_{i,j}$ and let a and y be the first and last nodes of L respectively and let z denote the last node of R. As G_i is (ϵ, g)-dispersed there exists a set Γ of ϵg node disjunct paths each of length g, terminating R and with no other nodes in L_i. Let $p \in \Gamma$ be such a path.

Now either p contains (at least) one pebble during all time steps in $[\bar{t}_a, \bar{t}_y]$ or not. If so p contributes at least g to $c_{i,j}$ with pebbles not lying on any of other $p' \in \Gamma$. If not then in order to pebble z, the last node of R, all of p must also be pebbled between steps $[\bar{t}_a, \bar{t}_z]$. Thus again p contributes at least g to $c_{i,j}$ with pebbles not lying on any other $p' \in \Gamma$.

Summing over all paths in Γ we get that $c_{i,j} \geq \epsilon g^2$ which concludes the proof of the claim and theorem. □

Reminder of Lemma 6.11. *Let $f_b(d) = \tilde{O}\left(\frac{n}{d^b}\right)$ then*

1. *Let $\delta = O(\mathsf{polylog}(n))$ then a (n, δ, n)-random DAG is $f_{0.5}$-reducible with high probability.*
2. *The Catena DAGs BRG^n_λ and DBG^n_λ are both f_1-reducible for $\lambda = O(\mathsf{polylog}(n))$.*
3. *The Balloon Hashing Linear (and the Double Buffer) graph Lin^σ_τ is f_1-reducible for $\tau = O(\mathsf{polylog}(n))$.*

The proof of Lemma 6.11 closely follows arguments from Alwen and Blocki [AB16], who proved these DAGs were (e, d)-reducible for specific values of e and d. Because the attack of Alwen and Blocki [AB16] was non-recursive they only focused on proving (e, d)-reducible for the values e, d which optimized the quality of their attack.

Proof of Lemma 6.11. (sketch) We first consider an arbitrary $\lambda = O(\mathsf{polylog}(n))$-layered DAG G. This includes Catena DAGs BRG^n_λ and DBG^n_λ as well asBalloon Hashing Linear (and the Double Buffer) graph Lin^σ_τ (with $\tau = \lambda = O(\mathsf{polylog}(n))$). Let d be given and let $e = (\lambda + 1)n/d$. For simplicity assume that

e, $n/(\lambda+1)$ and $d/(\lambda+1)$ are integers[10]. Let $S = \{i \times d/(\lambda+1) : i \leq g\}$ and observe that S has size $|S| = e = \tilde{O}(n/d)$. Thus, to show that G is f_1 reducible it suffices to show that $\mathsf{depth}(G-S) \leq d$. Define $L_i = \{i \times n/(\lambda+1)+1, \ldots, (i+1) \times n/(\lambda+1)\}$ for $0 \leq i \leq \lambda$. We note that any path in $G-S$ can remain on layer L_i for at most $d/(\lambda+1)$ steps before moving to a higher layer and there are $\lambda+1$ layers. Thus, the maximum length of any path is $(\lambda+1)d/(\lambda+1) = d$.

Next we consider with a (n,δ,n)-random DAG G on nodes $\{1,\ldots,n\}$. Let d be given and let $g = n/\sqrt{d}$. For simplicity assume that $\sqrt{d}$ and g are integers[11]. Let $S_1 = \{i \times \sqrt{d} : i \leq g\}$ and observe that S_1 has size $|S_1| = g$. Define $L_i = \{i \times g+1, \ldots, i \times g + g\}$. We call an node $v \in L_i$ good if $\mathsf{parents}(v) \bigcap L_i = \emptyset$ and we let $B_i = \{v \in L_i : \mathsf{parents}(v) \bigcap L_i \neq \emptyset\}$ denote the set of all bad vertices in L_i. Finally, we let $S = S_1 \cup \bigcup_{i=0}^{\sqrt{d}-1} B_i$. It is easy to verify that $\mathsf{depth}(G-S) \leq d$ because any path in $G-S$ can remain on layer L_i for at most $\sqrt{d}$ steps before moving up to a higher layer and there are $\sqrt{d}$ layers. Thus, the maximum length of any path is $\sqrt{d}^2 = d$. It is easy to see that $\mathbb{E}[|B_i|] \leq \frac{|B_i|}{i+1} = \frac{\delta g}{i+1}$ — let x_v denote the indicator random variable for the event $v \in B_i$ then $\Pr[x_v = 1] \leq \frac{1}{i+1}$. Thus, $\mathbb{E}[|S|] \leq g + \delta g \sum_{i=1} i^{-1} = O\left(\frac{\delta n \log n}{\sqrt{d}}\right)$. Furthermore, standard concentration bounds imply that $|S| - 2\mathbb{E}[|S|] \leq 0$ except with very small probability. Thus, a (n,δ,n)-random DAG G is $f_{0.5}$-reducible with high probability. □

Reminder of Theorem 6.10. *Let G be a f-reducible DAG on n nodes then if $f(d) = \tilde{O}\left(\frac{n}{d^b}\right)$ for some constant $0 < b \leq 1$ then for any constant $\epsilon > 0$*

$$\Pi^{\parallel}_{cc}(G) \leq O\left(n^{1+a+\epsilon}\right), \text{ where } \qquad a = \frac{1-2b+\sqrt{1+4b^2}}{2} .$$

Proof of Theorem 6.10. Let c be a constant such that $f(d) \leq \frac{cnlog^c n}{d^b}$. We define a sequence $\{a_i\}_{i=1}^{\infty}$ as follows

$$a_1 = a = \frac{1-2b+\sqrt{1+4b^2}}{2} \quad , \quad \text{and} \qquad a_{i+1} = 1 + \frac{(a-1)(1-a_i)}{b} \ .$$

We remark that a is the (positive) solution to the equation $x^2 + (2b-1)x - b = 0$. When we run our recursive attack $\mathsf{RecursiveGenPeb}$ we will set $d_i = \tilde{O}\left(n^{\frac{1-a_i}{b}}\right)$ such that $e_i = f(d_i) = n^{a_i} = g_{i-1}$. The following claims about the sequence $\{a_i\}_{i=1}^{\infty}$ will also be useful when analyzing the cost of our recursive attack.

Claim C.8 *For $i \geq 1$ we have*

$$0 \leq a_i \leq 1, \qquad a_{i+1} \geq a_i, \ \text{and} \qquad a_{i+1} + \frac{1-a_i}{b} = 1 - a_{i+1} + a_{i+2} + \frac{1-a_{i+1}}{b} \ .$$

Claim C.9

$$\lim_{i\to\infty} a_i = 1, \quad \text{and} \qquad \lim_{i\to\infty} \sum_{j=1}^{i} (1-a_i) = a \ .$$

By Claim C.9 for any $\epsilon > 0$ we can find a value k_ϵ such that $1 - a_k \leq \epsilon$ and for all $k > k_\epsilon$.

We run $\mathsf{RecursiveGenPeb}$, see Algorithm 1, setting $T = \{n\}$, $k = k_{\epsilon/2} + 1$ and defining $\overline{S} = S_1, \ldots, S_k$, $\overline{g} = g_0, g_1, \ldots, g_k$ and $\overline{d} = d_0 = n, d_1, \ldots, d_k$ as follows: for each $i \leq k$ we let S_i denote a set of size $|S_i| = e_i \leq n^{a_i}$ such that $\mathsf{depth}(G - S_i) \leq d_i = f^{-1}(e_i) \leq c\left(\frac{n}{n^{a_i}}\right)^{1/b} \log^c n$. We also set $g_{i-1} = e_i = n^{a_i}$.

Let $T(k)$ denote the cost of a balloon phase at the final level k of the recursion. In this case a balloon phase lasts for at most $2d_k$ steps so the total cost is at most $T(k) \leq 2d_k n \leq 2cn\left(\frac{n}{n^{a_k}}\right)^{1/b} \log^c n \leq 2cn^{1+\epsilon/2} \log^c n$.

[10]This allows us to simply presentation by ignoring insignificant rounding terms.
[11]This allows us to simply presentation by ignoring insignificant rounding terms.

In general, we have the following recurrence

$$\begin{aligned}T(i) &\leq e_{i+1}d_i + 2\delta g_i d_i + \frac{n}{g_i}T(i+1) = (2\delta+1)d_i e_{i+1} + \frac{n}{e_{i+1}}T(i+1)\\ &\leq (2\delta+1)cn^{a_{i+1}+\frac{1-a_i}{b}}\log^c n + n^{1-a_{i+1}}T(i+1)\ .\end{aligned}$$

In particular, a level-i balloon phase lasts for $2d_i$ rounds and in every step we keep $e_{i+1}+2\delta g_i$ pebbles on the graph — corresponding to the nodes in S_{i+1} and the parents of the next $2g_i$ nodes that we want to pebble. We make at most $\frac{n}{g_i}$ recursive calls to level-$i+1$ balloon phases — each of cost $\leq T(i+1)$.

Unrolling the recurrence, and applying Claim C.8, we get that

$$\begin{aligned}T(i) &\leq (2\delta+1)cn^{a_{i+1}+\frac{1-a_i}{b}}\log^c n + n^{1-a_{i+1}}\left((2\delta+1)cn^{a_{i+2}+\frac{1-a_{i+1}}{b}}\log^c n + n^{1-a_{i+2}}T(i+2)\right)\\ &\leq 2(2\delta+1)cn^{1+\frac{a(1-a_i)}{b}}\log^c n + n^{2-a_{i+1}-a_{i+2}}\left(T(i+2)\right)\ .\end{aligned}$$

In particular,

$$\begin{aligned}\Pi^{\parallel}_{cc}(\mathcal{A}) &\leq e_1 n + 2\delta g_0 n + \frac{n}{g_0}T(1)\\ &\leq (2\delta+1)n^{1+a_1}\log^c n + n^{1-a}\left((2\delta+1)n^{a_2+\frac{1-a}{b}}\log^c n\right) + \frac{n^2}{g_0 g_1}T(2)\\ &= 2(2\delta+1)n^{1+a_1}\log^c n + \frac{n^2}{g_0 g_1}T(2)\\ &\leq (k+1)(2\delta+1)cn^{1+\frac{a(1-a)}{b}}\log^c n + T(k)\prod_{j=0}^{k}\frac{n}{g_j}\\ &\leq (k+1)(2\delta+1)cn^{1+a}\log^c n + T(k)n^{k+1-\sum_{i=0}^{k}a_{i+1}}\\ &\leq (k+1)(2\delta+1)cn^{1+a}\log^c n + n^{1+\epsilon/2}n^{\sum_{i=0}^{k}(1-a_{i+1})}\\ &\leq (k+1)(2\delta+1)cn^{1+a}\log^c n + n^{1+\epsilon/2+\sum_{i=1}^{\infty}(1-a_i)}\\ &\leq k(\delta+1)cn^{1+a+\epsilon/2}\log^c n\\ &= O\left(n^{1+a+\epsilon}\right)\ .\end{aligned}$$

□

Reminder of Claim C.8. *For $i \geq 1$ we have*

$$0 \leq a_i \leq 1, \qquad a_{i+1} \geq a_i, \ \textit{and} \qquad a_{i+1} + \frac{1-a_i}{b} = 1 - a_{i+1} + a_{i+2} + \frac{1-a_{i+1}}{b}\ .$$

Proof of Claim C.8. We first two statements by induction. Clearly, $0 \leq a = a_1 \leq 1$. Now for $i \geq 1$ we have

$$a_{i+1} = 1 + \frac{(a-1)(1-a_i)}{b} = 1 - \left(\frac{(1-a)}{b}\right)(1-a_i) \leq 1\ ,$$

because b, $(1-a)$ and $(1-a_i)$ are all positive values. Similarly, we also have

$$a_{i+1} - a_i = 1 - \left(\frac{(1-a)}{b}\right)(1-a_i) - a_i = (1-a_i)\left(1 + \frac{(1-a)}{b}\right) \geq 0\ .$$

Thus, $a_{i+1} \geq a_i \geq 0$.

For the third statement we observe that $a_{i+1} + \frac{1-a_i}{b} = \frac{a\cdot a_{i+1}-1}{a-1}$ and that $1 - a_{i+1} + a_{i+2} + \frac{1-a_{i+1}}{b} = 2 - a_{i+1} + \frac{a(1-a_{i+1})}{b}$. Now we claim that $2 - a_{i+1} + \frac{a(1-a_{i+1})}{b} - \frac{a\cdot a_{i+1}-1}{a-1} = 0$. Multiplying both sides by

$(a-1)b$ we get $2(a-1)b - a_{i+1}(a-1)b + a(a-1)(1-a_{i+1}) - ab \cdot a_{i+1} + b = 0$. Refactoring we get $-b(1-a_{i+1}) + a^2(1-a_{i+1}) + a((2b-1)(1-a_{i+1})) = 0$. Dividing by $(1-a_{i+1})$ we get $a^2 + (2b-1)a - b = 0$, which is true by definition of a. □

Reminder of Claim C.9.

$$\lim_{i\to\infty} a_i = 1, \quad \textit{and} \qquad \lim_{i\to\infty} \sum_{j=1}^{i} (1-a_i) = a \ .$$

Proof of Claim C.9. We first note that $0 < \left(\frac{1-a}{b}\right) < 1$, and that

$$\begin{aligned} 1 - a_{i+1} &= \frac{(1-a)(1-a_i)}{b} \\ &= \left(\frac{(1-a)}{b}\right)(1-a_i) \\ &= \left(\frac{(1-a)}{b}\right)^2 (1-a_{i-1}) \\ &= \ldots \\ &= \left(\frac{(1-a)}{b}\right)^i (1-a_1) \ . \end{aligned}$$

Therefore, $\lim_{i\to\infty} a_i = 1 - \lim_{i\to\infty} \left(\frac{(1-a)}{b}\right)^i (1-a_1) = 1$. Similarly,

$$\begin{aligned} \sum_{j=1}^{i} (1-a_j) &= \sum_{j=0}^{i-1} \left(\frac{(1-a)}{b}\right)^j (1-a_1) \\ &= (1-a) \sum_{j=0}^{i-1} \left(\frac{(1-a)}{b}\right)^j \ . \end{aligned}$$

Thus, $\lim_{i\to\infty} \sum_{j=1}^{i} (1-a_j) = \frac{1-a}{1-\frac{1-a}{b}} = \frac{b-ba}{b-1+a} = a$, where the last equality follows because a was chosen so that $a^2 + (2b-1)a - b = 0$. □

C.1 Tighter Bounds on Memory-Hardness

As an example of the power of our results we can immediately improve on the analysis of the high CC graph of [AS15]. In that work, first a graph $G \in \mathbb{G}_{n,\delta}$ with indegree $\delta \le \log^3(n)$ is given and it is show to have CC approximately $n^2/\log(n)$. Next the indegree is reduced to obtain G' at a cost of δ^3 to the CC. That is G' has size $N = n * \delta$ and CC of roughly $N^2/\log^{10}(N)$.

In contrast Lemma 4.1 only loses a factor of δ in the CC when reducing the indegree. Moreover, the indegree of the depth-robust graphs of [MMV13] used in the construction of [AS15] actually have indegree $\log^2(n)$ times some polyloglog function of n and so the same is true G. Ignoring polyloglog factors and setting $g = n/\log^3(n)$ in Theorem 2.5 we see that G must be at least (e,d)-depth-robust for some $e = \Omega(n/\log(n))$ and $d = \Omega(n/\log^4(n))$. So we can reduce the indegree of G to 2 using Lemma 4.1 with $\gamma = \delta$ and we obtain a graph $H \in G_{n\delta,2}$ of size $N = 2n\delta$ which is $(e, d\delta)$-depth-robust. In particular Corollary 3.1 shows that the CC of H is at least $\Omega(n^2/\log^3(n)) = \Omega(N^2/\log^7(N))$.

The above analysis uses the results from [AS15] as a blackbox. In fact, if we look into the construction the graph $G_{n,\delta}$ we will observe that is consists of a stack of $\log\log(n)$ depth-robust graphs from the construction of [MMV13]. Corollary 3.1 directly implies that $\Pi^{\parallel}_{cc}(G_{n,\delta}) \ge \frac{cn^2}{(\log\log(n))^2}$ for some constant $c > 0$ (in fact this is just the cost of pebbling the top layer). Lemma 4.1 allows us to reduce the indegree $G_{n,\delta}$

at a cost of $\delta \leq \log^2(n)\mathsf{polylog}\log(n)$. Thus, we obtain a a constant indegree graph on N nodes with $\Pi^{\parallel}_{cc}(G) \geq \frac{cN^2}{\log^2(n)\mathsf{polylog}\log(n)}$.

JoëlredWell... now that we've put it like that I guess we don't really improve on [AS15] after all. After all, going from $\log^{10}$ to $\log^7$ could have already be done in [AS15] simply by using the more precise bound on the indegree of the [MMV13] depth-robust graphs. So maybe we should just leave this subsection out?

JeremiahblueI would advocate for moving this section to the appendix. We can get tighter bounds by directly using the fact that the [AS15] construction is a stack of DR DAGs

Algorithm 1: RecursiveGenPeb (G, k, $\overline{S}$, $\overline{g}$, $\overline{d}$, T)

```
Arguments       : $G = (V, E)$, $k$, $S_1 \subseteq \ldots \subseteq S_k \subseteq V$, $g_0, g_1, \ldots, g_k$ s.t $g_{j-1} \in [2 \cdot \mathsf{depth}(G - S_j), |V|]$,
                  $d_0, d_1, \ldots, d_k$ s.t. $d_j \geq \mathsf{depth}(G - S_j)$ and $T \subseteq V$
Local Variables : $n = |V|$, a node-partition $\emptyset = D_0, D_1, \ldots, D_{2d_0} \subseteq V$ s.t. $|D_i| \leq \frac{n}{d_0}$ and
                  $\mathsf{parents}(D_{i+1}) \subseteq \bigcup_{j=0}^{i} D_j$
Output          : $P_1, \ldots, P_{2d_0} \subseteq V$
$d \leftarrow \max\{j : T \bigcap D_j \neq \emptyset\}$            // Need $d \leq 2d_0$ steps to pebble nodes in $T$.
$P_0, \ldots, P_{2d_0 - d} \leftarrow \emptyset$
if $k$=0 then                                                       // Greedy Pebble
    for $i = 1$ to $d$ do
        $P_{2d_0 - d + i} \leftarrow D_i \cup P_{2d_0 - d + i - 1}$
    end
    Return $P_1, \ldots, P_{2d_0}$
else
    $A_1, \ldots, A_{2d_0} \leftarrow \emptyset$
    for $i = 1$ to $d$ do
        $j_g \leftarrow \min\left\{j > i : \left|\bigcup_{r=i+1}^{j} D_r\right| \geq g\right\}$
        $j_{2g} \leftarrow \min\left\{j > i : \left|\bigcup_{r=i+1}^{j} D_r\right| \geq 2g\right\}$
        $N_g \leftarrow \mathsf{parents}\left(\bigcup_{r=i+1}^{j_g} D_r\right) \bigcap \left(\bigcup_{r=0}^{i-1} D_r\right)$
        $N_{2g} \leftarrow \mathsf{parents}\left(\bigcup_{r=i+1}^{j_{2g}} D_r\right) \bigcap \left(\bigcup_{r=0}^{i-1} D_r\right)$
        $K \leftarrow S_1 \cup N_{2g} \cup T$                       // Do not discard pebbles in $K$
        $A \leftarrow \bigcup_{j=i}^{i+2d_1} A_i$    // Pebbles already scheduled to be added in next $2d_1$ rounds
        if $N_g \not\subset D_i \cup P_{i-1} \cup A$ then                         // Balloon Phase
            $\overline{S}' \leftarrow S_2, \ldots, S_k$
            $\overline{g}' \leftarrow g_1, \ldots, g_k$
            $\overline{d}' \leftarrow d_1, \ldots, d_k$
            $T' \leftarrow N_{2g} - P_{i-1} - D_i - A$
            $B_1, \ldots, B_{2d_1} \leftarrow \mathsf{RecursiveGenPeb}(G - S_1, k - 1, \overline{S}', \overline{g}', \overline{d}', N_{2g})$    // Add $B_1, , \ldots, B_{2d_1}$
                                                                    // to pebbling schedule
            for $j = 1$ to $2d_1$ do
                $A_{i+j-1} \leftarrow A_{i+j-1} \cup B_j$
            end
            $P_{2d_0 - d + i} \leftarrow D_i \cup A_i \cup (P_{2d_0 - d + i - 1} \cap K)$
        else                                                        // Light Phase
            $P_{2d_0 - d + i} \leftarrow D_i \cup A_i \cup (P_{2d_0 - d + i - 1} \cap K)$
        end
    end
    Return $P_1, \ldots, P_{2d_0}$
end
```