Data-Independent Memory Hard Functions: New Attacks and Stronger Constructions

Jeremiah Blocki[1], Ben Harsha[1], Siteng Kang[2], Seunghoon Lee[1], Lu Xing[1], and Samson Zhou[3]

[1] Purdue University
[2] Penn State University
[3] Indiana University

Abstract. Data-Independent Memory-hard functions (iMHFs) are a key cryptographic primitive underlying the design of moderately expensive password hashing algorithms and egalitarian proofs of work that are resistant to side-channel attacks. Several goals for MHFs have been proposed including bandwidth hardness, space-time (ST) complexity, amortized area-time (aAT) complexity and sustained space complexity. An iMHF can be specified using a directed acyclic graph (DAG) G with $N=2^n$ nodes and low indegree, and the cost (aAT, ST etc...) to evaluate the iMHF can be analyzed using pebbling games. In particular, given a parameter N (e.g., maximum acceptable running time) we would like to design the DAG G to have maximum possible pebbling cost i.e., to ensure that the iMHF is as expensive as possible for an attacker to compute. Recently, Alwen et al. [ABH17] gave a randomized DAG construction called DRSample and proved that the aAT cost to pebble the graph was $\Omega\left(N^2/\log N\right)$. In an asymptotic sense the DRSample outperformed all prior constructions including Argon2i, the winner of the password hashing competition, which can be pebbled with aAT cost at most $\mathcal{O}\left(N^{1.767}\right)$. In this work we first prove a matching *upper bound* on the pebbling cost of DRSample by analyzing the greedy pebbling attack of Boneh et al. [BCS16]. This sequential attack on DRSample is simple, easy to implement and has good concrete performance. In fact, our results show that, for practical values of $N \le 2^{24}$, Argon2i provides *stronger* resistance to known pebbling attacks than DRSample reversing a finding of Alwen et al. [ABH17]. We then develop a new iMHF candidate by extending DRSample with the bit-reversal graph, and show that the iMHF resists *all known attacks* in practice and has *optimal* asymptotic performance under every MHF metric. In particular, we prove that (1) *any* (nearly) sequential pebbling attack (including the greedy pebbling attack) has aAT cost $\Omega\left(N^2\right)$, (2) *any* parallel attacker has aAT cost at least $\Omega\left(N^2/\log N\right)$ and *at least* $\Omega\left(N^2\log\log N/\log N\right)$ unless one can find new depth-reducing attacks against DRSample which significantly improve upon the state of the art, (3) the graph has high bandwidth-complexity, and (4) any pebbling *either* has aAT cost $\omega(N^2)$ or *requires* at least $\Omega(N)$ steps with $\Omega(N/\log N)$ pebbles on the DAG. This makes our construction the first practical iMHF with strong guarantees on the sustained space-complexity. We also observe that the Argon2i round function can (trivially) be evaluated in parallel, which would allow an attacker to reduce aAT costs by (nearly) an order of magnitude, and we develop an *inherently* sequential version of the Argon2i round function that prevents this attack. We implement our new iMHF candidate (with and without the sequential round function) and show that evaluation speed is nearly identical to Argon2i. Finally, we provide a pebbling reduction which proves that in the parallel random oracle model (PROM) the cost of evaluating an iMHF like Argon2i or DRSample+BRG is given by the pebbling cost of the underlying DAG.

1 Introduction

In the context of password hashing and key-derivation functions it is *desirable* to construct a moderately hard function that can be computed relatively quickly on a standard personal computer, but for which is it economically infeasible for an attacker to evaluate the function millions or billions of times. The former property is important so that user can authenticate reasonably quickly without testing user patience. The purpose of the latter goal is to protect low-entropy secrets (e.g., passwords, PINs, biometrics) against brute-force guessing attacks [BHZ18]. One of the challenges is that the attacker might attempt to reduce computation costs by employing customized hardware such as a Field Programmable Gate Array (FPGA) or an Application Specific Integrated Circuit (ASIC). By employing customized hardware it is often possible to reduce computation costs by several orders of magnitude e.g., commercially available Bitcoin mining ASICs are capable of evaluating the SHA-256 hash function over 10^{13} times per second.

Memory Hard Functions (MHFs) are a crucial cryptographic primitive that can be used to develop a moderately hard password hashing algorithm that satisfies both of our goals and to develop egalitarian proof of work puzzles [Lee11]. Blocki et al. [BHZ18] recently argued that a strong MHF can provide meaningful protection against a brute-force password attacker, while traditional key-stretching algorithms like bcrypt and PBKDF2 are no longer sufficient to protect low-entropy secrets such as passwords. A strong memory hard function should have high amortized space time (aAT) complexity to ensure that an attacker who computes this function many times will have to lock up a large amount of memory for the entire duration of *each individual* computation. In particular, if an MHF takes N *sequential* time steps to evaluate on a PC we can fill up to N blocks of memory in RAM that time. Ideally, we hope that *any* (possibly parallel) attacker will also have aAT complexity $\Omega\left(N^2\right)$ e.g., the attacker needs to keep $\Omega(N)$ blocks in memory for N time steps each time the function is computed.

The scrypt MHF, introduced by Percival in 2009 [Per09], was proven to have aAT complexity $\Omega\left(N^2\right)$ [ACP$^+$17]. However, scrypt is potentially vulnerable to side-channel attacks such as cache-timing [Ber05] as the memory access pattern in a particular execution is *dependent* on the input (e.g., password). Thus, in the context of password hashing data-independent memory hard functions (iMHFs) are of special interest due to their natural resistance to side-channel attacks — as the name suggests the induced memory access pattern is *independent* of the sensitive user input. Arguably, two of the most significant iMHFs candidates are Argon2i [BDK16] and DRSample [ABH17]. Argon2i was the winner of the recently completed password hashing competition [Pin14] and DRSample [ABH17] was the first *practical* construction of an iMHF with aAT complexity *at least* $\Omega\left(N^2/\log N\right)$.

A recent line of research [AB16, AB17, ABP17, BZ17] has developed theoretical depth-reducing attacks on Argon2i showing that the iMHF has aAT complexity *at most* $\mathcal{O}\big(n^{1.767}\big)$[4]. The DRSample [ABH17] iMHF modifies the edge distribution of the Argon2i graph to ensure that the underlying directed acyclic graph (DAG) satisfies a combinatorial property called depth-robustness which is known to be *necessary* [AB16] and *sufficient* [ABP17] for developing a MHF with high aAT complexity.

Alwen et al. [ABH17] proved that the aAT complexity of DRSample was at least $\Omega\big(N^2/\log N\big)$. In an asymptotic sense this upper bound almost matches the upper bound of Alwen and Blocki [AB16] who showed that *any* iMHF has aAT $\mathcal{O}(N^2 \log\log N/\log N)$. However, from a practical standpoint the *constants* in these upper and lower bounds are still poorly understood. On the positive side the upper bounds do not rule out the existence of a practical construction of a DAG G with aAT complexity at least $N^2/2$ (the best possible case) for *practical* values of $N \leq 2^{24}$ — $N = 2^{24}$ corresponds to 16GB of memory in Argon2i. On the negative side the asymptotic lower bounds do not *absolutely* rule out the possibility of an attack that reduces aAT complexity by several orders of magnitude. Alwen et al. [ABH17] also presented an empirical analysis of the security of DRSample by running a battery of depth-reducing pebbling attacks against their construction and against Argon2i [AB16, ABP17, AB17]. The results of this analysis indicated that the new construction was not only stronger from a theoretical standpoint, but that it also provided greater resistance to other pebbling attacks than other iMHF candidates like Argon2i in practice. However, their empirical analysis did not consider the greedy pebbling attack of Boneh et al. [BCS16], perhaps because this attack only yielded a modest constant factor reduction in aAT complexity for Argon2i in comparison to the (asymptotically) more substantial $\Omega\big(N^{0.2+}\big)$-factor reductions guaranteed by depth-reducing attacks [AB16, AB17, ABH17].

1.1 Contributions

Stronger Attack on DRSample We present a theoretical and empirical analysis of the greedy pebbling attack [BCS16] finding that DRSample has aAT complexity at most $\lessapprox N^2/\log N$. The greedy pebbling attack that achieves this bound is *sequential*, easy to implement and achieves high attack quality for practical values of N. In fact, for *practical* values of $N \leq 2^{24}$ we show that DRSample is *more* vulnerable to *known pebbling attacks* than Argon2i, which *reverses* previous conclusions about the *practical* security of Argon2i and DRSample [ABH17]. We next consider a defense proposed by Biryukov et al. [BDK16] against the greedy pebbling attack which we call the XOR-extension gadget. While this defense defeats the *original* greedy pebbling attack [BCS16], we found a simple generalization of the greedy pebbling attack that thwarts this defense.

Improved Heuristic Algorithm for Constructing Depth-Reducing Sets We also develop a *novel* greedy algorithm for constructing depth-reducing sets, which is the first step in the depth-reducing attacks of Alwen and Blocki [AB16, AB17]. Empirical

[4] This latest attack almost matches the *lower bound* of $\tilde{\Omega}\big(n^{1.75}\big)$ on the aAT complexity of Argon2i.

analysis demonstrates that this greedy algorithm constructs *significantly smaller* depth-reducing sets than previous state of the art techniques [AB16, AB17, ABH17], which leads to improved depth-reducing attacks [AB16]. This also leaves us in an uncomfortable situation where *all known* iMHF candidates admit reasonably high quality attacks even for *practical* values of N e.g., DRSample is susceptible to the greedy pebbling attack while Argon2i is susceptible to depth-reducing attacks [AB16, AB17, ABH17].

New iMHF Candidate with Optimal Security We next develop a new iMHF candidate DRSample+BRG by overlaying a bit-reversal graph [LT82, FLW14] on top of DRSample, and analyze the new DAG empirically and theoretically. We prove that *any* sequential pebbling of the bit-reversal graph has cumulative memory cost (cmc) and aAT cost at least $\Omega(N^2)$, which implies that neither the greedy pebbling attack nor its extension will be effective — this result generalizes a well known result that the bit-reversal graph has sequential space-time cost $\Omega(N^2)$ and may be of independent interest e.g. it demonstrates that Password Hashing Competition Finalist Catena-BRG [FLW14] is secure against *all* sequential attacks. Our empirical analysis indicates that DRSample+BRG offers strong resistance to *all known* attacks, including the greedy pebbling attack, depth-reducing attacks and several other novel attacks introduced in this paper. In particular, even when $N=2^{24}$ the *best* attack had aAT cost over $\frac{N^2}{11}$. In contrast, we show that *any* DAG with indegree two has a sequential pebbling with aAT cost $\lessapprox \frac{N^2}{4}$. On the theoretical side we show that DRSample+BRG is asymptotically optimal with respect to *all* proposed MHF metrics. In particular, the iMHF is maximally bandwidth hard (inherited individually from *both* BRG and DRSample), has aAT cost at least $\Omega(N^2/\log N)$ (inherited from DRSample) and *any* parallel pebbling either has maximal sustained space complexity (meaning that there are at least $\Omega(N)$ steps with $\Omega(N/\log N)$ pebbles on the DAG) or has aAT cost at least $\omega(N^2)$. This makes our construction the first practical construction with strong guarantees on the sustained space-complexity — prior constructions of Alwen et al. [ABP18] were theoretical.

We also show that the cumulative memory complexity of DRSample+BRG is *at least* $\Omega\left(N^2 \log\log N/\log N\right)$ under a plausible conjecture about the depth-robustness of DRSample that the graph is (e,d,b)-block depth robust with $e=d=\Omega\left(\frac{N\log\log N}{\log N}\right)$ and $b=\Omega(\log N/\log\log N)$. As evidence for our conjecture we analyze three state-of-the-art approaches for constructing a depth-reducing set, including the layered attack [AB16], Valiant's Lemma [AB16, Val77] and the reduction of Alwen et al. [ABP17], which can transform any lower cost pebbling (e.g., the Greedy Pebbling Attack) into a depth-reducing set. We find that in every-case one would need to obtain significantly stronger attacks to refute our conjecture. In particular, we show that even to reduce the depth to just $d=\mathcal{O}\left(\frac{N}{\sqrt{\log N}}\right)$ each attack still *requires* the removal of *at least* $e=\Omega\left(\frac{N\log\log N}{\log N}\right)$ nodes.

Black Pebbling Reduction for XOR Labeling Rule Alwen and Serbinenko showed that any algorithm evaluating the function $f_{G,H}$ in the parallel random oracle model *must* have cumulative memory cost at least $\Omega\left(w \times \Pi_{cc}^{\parallel}(G)\right)$, where $\Pi_{cc}^{\parallel}(G)$ is the cost of the best (parallel) black pebbling of an N node DAG G, $H : \{0,1\}^* \to \{0,1\}^w$ is a random oracle and $f_{G_H}(x) = \mathsf{lab}_{G,H,x}(N)$ is the label of the last node. In their reduction, Alwen and Serbinenko assume that the labeling function is defined recursively i.e., for each node v with $\mathsf{parents}(v) = v_1,\ldots,v_\delta$ we have

$$\mathsf{lab}_{G,H,x}(v) = H(v, \mathsf{lab}_{G,H,x}(v_1),\ldots,\mathsf{lab}_{G,H,x}(v_\delta)) \ .$$

Similar, pebbling reductions have been given for bandwidth hardness [BRZ18] and sustained space complexity [ABP18] using the same labeling rule.

While these results are useful, most *practical* iMHF implementations do not follow this labeling rule. In particular, Argon2i/DRSample and our implementation of DRSample+BRG all use the following labeling rule

$$\mathsf{lab}_{G,H,x}(v) = H(\mathsf{lab}_{G,H,x}(v_1) \oplus \ldots \oplus \mathsf{lab}_{G,H,x}(v_\delta)) \ ,$$

where $v_1,\ldots,v_\delta = \mathsf{parents}(v)$ and the DAGs have indegree $\delta = 2$. The XOR labeling rule allows one to work with a faster round function $H : \{0,1\}^w \to \{0,1\}^w$ e.g., Argon2i builds $H : \{0,1\}^{8192} \to \{0,1\}^{8192}$ using the Blake2b permutation function and DRSample(+BRG) uses the same labeling rule as Argon2i.

We would like to show that *any* algorithm evaluating the function $f_{G,H}$ in the parallel random oracle model *must* have cumulative memory cost at least $\Omega\left(w \times \Pi_{cc}^{\parallel}(G)\right)$. This task becomes more difficult to show as the function $H'(x,y) = H(x \oplus y)$ is no longer collision resistant e.g., $H'(y,x) = H'(x,y)$. In fact, the claim simply is not true for arbitrary graphs. In particular, there exist graphs G on N nodes where the function $f_{G,H}$ is a constant function! Suppose we start with a DAG $G' = (V' = [N-3], E')$ on $N-3$ nodes that has high pebbling cost $\Pi_{cc}^{\parallel}(G')$ and define $G = (V = [N], E = E' \cup \{(N-3,N-2),(N-3,N-1),(N-4,N-2),(N-4,N-1),(N-2,N),(N-1,N)\})$ by adding directed edges from node $N-3$ and $N-4$ to nodes $N-2, N-1$ and then adding directed edges from $N-2$ and $N-1$ to node N. Note that for any input x we have

$$\mathsf{lab}_{G,H,x}(N-2) = H(\mathsf{lab}_{G,H,x}(N-3) \oplus \mathsf{lab}_{G,H,x}(N-3)) = \mathsf{lab}_{G,H,x}(N-1) \ .$$

Therefore,

$$f_{G,H}(x) = \mathsf{lab}_{G,H,x}(N) = H(\mathsf{lab}_{G,H,x}(N-2) \oplus \mathsf{lab}_{G,H,x}(N-1)) = H(0^w) \ .$$

The above example exploited the absence of the explicit term v in $\mathsf{lab}_{G,H,x}(v)$ to produce two nodes that always have the same label.

What we can prove is that if the DAG $G = (V = [N], E)$ contains all edges of the form $(i, i+1)$ for $i < N$ then *any* algorithm evaluating the function $f_{G,H}$ in the parallel random oracle model *must* have cumulative memory cost at least $\Omega\left(w \times \Pi_{cc}^{\parallel}(G)\right)$. Furthermore, the cumulative memory cost of an algorithm computing $f_{G,H}$ on m distinct inputs must be at least $\Omega\left(mw \times \Pi_{cc}^{\parallel}(G)\right)$. We stress that all of the practical iMHFs we consider including Argon2i and DRSample(+BRG) satisfy this condition. Intuitively, as long as we can view $\mathsf{lab}_{G,H,x}(v)$ as a uniformly random string then we can also view $\mathsf{lab}_{G,H,x}(v+1)$ as a uniformly random string. This follows because $\mathsf{lab}_{G,H,x}(v+1) = H(\mathsf{lab}_{G,H,x}(v) \oplus Y)$ for some fixed string Y that can only depend on labels $1,...,v-1$. Since we view $\mathsf{lab}_{G,H,x}(v)$ as random, it follows that the *input* to the random oracle $\mathsf{lab}_{G,H,x}(v) \oplus Y$ will be distinct from all prior queries with high probability in which case the output $\mathsf{lab}_{G,H,x}(v+1)$ can be viewed as a uniformly random string.

Sequential Round Function We show how a parallel attacker could reduce aAT costs by nearly an order of magnitude by computation of the Argon2i round function in parallel. For example, the first step to evaluate the Argon2 round function $H(X,Y)$ is to divide the input $R = X \oplus Y \in \{0,1\}^{8092}$ into 64 groups of 16-byte values $R_0,...,R_{63} \in \{0,1\}^{128}$ and then compute $(Q_0,Q_1,...,Q_7) \leftarrow \mathcal{BP}(R_0,...,R_7),...,$ $(Q_{56},Q_{56},...,Q_{63}) \leftarrow \mathcal{BP}(R_{56},...,R_{63})$. Each call to the Blake2b permutation $\mathcal{BP}$ can be trivially evaluated in parallel, which means that the attacker can easily reduce the depth of the circuit by a factor of 8. The issue affects *all* Argon2 modes of operation (including data-dependent modes like Argon2d and Argon2id) and could potentially be used in combination with other pebbling attacks [AB16, AB17] for an even more dramatic decrease in aAT complexity. We also stress that this gain is independent of any other optimizations that an ASIC attacker might make to speed up computation of $\mathcal{BP}$ e.g., if the attacker can evaluate $\mathcal{BP}$ four-times faster than the honest party then the attacker will be able to evaluate the round function H $8 \times 4 = 32$-times faster than the honest party.

Motivated by this observation, we introduce an *inherently sequential* version of the Argon2 round function by injecting a few extra data-dependencies e.g., $(Q_8,Q_9,...,Q_{15}) \leftarrow \mathcal{BP}(R_8,R_9 \oplus Q_0,R_{10},...,R_{15})$ and $(Q_{56},Q_{56},...,Q_{63}) \leftarrow \mathcal{BP}(R_{56},...,R_{62},R_{63} \oplus Q_{48})$. By injecting these extra data-dependencies, we can ensure that the attacker must wait until $(Q_0,Q_1,...,Q_7)$ have been computed before it is possible to compute $(Q_8,Q_9,...,Q_{15})$. This increases a parallel attacker's aAT costs by nearly an order of magnitude. Empirical analysis indicates that our modifications have *negligible* impact on the running time performance of Argon2 for the honest party.

Implementation of our iMHF We develop an implementation of our new iMHF candidate DRSample+BRG, which also uses the improved sequential Argon2 round function. The source code is available on an anonymous Github repository https://github.com/antiparallel-drsbrg-argon/Antiparallel-DRS-BRG. Empirical tests indicate that the running time of DRSample+BRG is equivalent to that of Argon2 for the honest party, while our prior analysis indicates the aAT costs, energy costs and sustained space complexity are all higher for DRSample+BRG.

2 Preliminaries

In this section we will lay out notation and important definitions required for the following sections.

2.1 Graph Notation and Definitions

We use $G = (V,E)$ to denote a directed acyclic graph and we use $N = 2^n$ to denote the number of nodes in $V = \{1,\ldots,N\}$. Given a node $v \in V$, we use

$\mathsf{parents}(v) = \{u \ : \ (u,v) \in E\}$ to denote the *immediate parents* of node v in G. 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 $\mathsf{parents}_G^{i+1}(v) = \mathsf{parents}_G\left(\mathsf{parents}_G^i(v)\right)$. When G is clear from context we will simply write $\mathsf{parents}$ ($\mathsf{ancestors}$). We use $\mathsf{indeg}(G) = \max_v |\mathsf{parents}(v)|$ to denote the maximum indegree of any node in G. All of the practical graphs we consider will contain each of the edges $(i,i+1)$ for $i < N$. Thus, there is a single source node 1 and a single sink node N. Most of the graphs we consider will have $\mathsf{indeg}(G) = 2$ and in this case we will use $r(i) < i$ to denote the *other* parent of node i besides $i-1$.

Block depth-robustness: Block depth-robustness is a stronger variant of depth-robustness. First, we define $N(v,b) = \{v-b+1, v-b+2, \ldots v\}$ to be the set of b contiguous nodes ending at node v. For a set of vertices $S \subseteq V$, we also define $N(S,b) = \bigcup_{v \in S} N(v,b)$. We say that a graph is (e,d,b) block depth robust if, for every set $S \subseteq V$ of size $|S| \leq e$, $\text{depth}(G - N(S,b)) \geq d$. When $b = 1$ we simply say that the graph is (e,d) depth robust. It is known that any (e,d) depth robust DAG G has $\mathsf{aAT}_R(G) \geq ed + RN$ [ABP17]. Block depth-robustness can be used to prove *stronger* lower bounds on the pebbling complexity of a graph in certain cases [ABH17,BZ17]. Intuitively, if a DAG is block-depth robust then it often possible to argue that an attacker must *either* (1) keep at least one pebble on most blocks of b contiguous nodes, or (2) frequently re-pebble the depth-robust graph which is also expensive [ABP17]. DRSample is (e,d,b) block depth robust with $e = \Omega(N/\log N)$, $d = \Omega(N)$ and $b = \Omega(\log N)$ [ABH17].

Graph labeling functions: As mentioned in the introduction, an iMHF $f_{G,H}$ can be described as a mode of operation over a directed acyclic graph using a round function H. Intuitively, the graph represents data dependencies between the memory blocks that are generated as computation progresses and each vertex represents a value being computed based on some dependencies. The function $f_{G,H}(x)$ can typically be defined as a labeling function i.e., given a set of vertices $V = [N] = \{1,2,3,\ldots,N\}$, a compression function $H = \{0,1\}^* \rightarrow \{0,1\}^m$ (often modeled as a Random Oracle in security analysis), and an input x, we "label" the nodes in V as follows. All source vertices (those with no parents) are labeled as $\ell_v(x) = H(v,x)$ and all other nodes with parents $v_1, v_2, \ldots, v_\delta$ are labeled $\ell_v(x) = F_{v,H}(\ell_{v_1}(x), \ell_{v_2}(x), \ldots, \ell_{v_\delta}(x))$ for a function $F_v(\cdot)$that can vary. To ensure that the function $f_{G,H}$ can be computed in $\mathcal{O}(N)$ steps, we require that G is an N node DAG with constant indegree δ.

2.2 iMHFs and the Parallel Black Pebbling Game

Results from Alwen and Serbinenko [AS15] and [AT17] imply that in the parallel random oracle model (PROM) the amortized area time complexity of the function $f_{G,H}$ is completely captured by the (parallel) black pebbling game on the DAG G when we define when $F_{v,H}(\ell_{v_1}(x), \ell_{v_2}(x), \ldots, \ell_{v_\delta}(x)) = H(v, \ell_{v_1}(x), \ell_{v_2}(x), \ldots, \ell_{v_\delta}(x))$. We extend this result when $F_{v,H}(\ell_{v_1}(x), \ell_{v_2}(x), \ldots, \ell_{v_\delta}(x)) = H\left(\bigoplus_{j=1}^{\delta} \ell_{v_j}(x)\right)$ — the compression function that ends up being used in *practical constructions* such as Argon2i. The output $f_{G,H}(x)$ is then defined to be the label(s) of the sink node(s) in G.

Intuitively, placing a pebble on a node represents computing the corresponding memory block and storing it in memory. The rules of the black pebbling game state that we cannot place a pebble on a node v until we have pebbles on the parents of node v i.e., we cannot compute the memory block until we have access to all of the necessary dependent memory blocks. More formally, in the black pebbling game on a directed graph $G=(V,E)$, we place pebbles on certain vertices of G over a series of t rounds. A valid pebbling P is a sequence $P_0,P_1,...,P_t$ of sets of vertices satisfying the following properties: (1) $P_0=\emptyset$, (2) $\forall v \in P_i \setminus P_{i-1}$ we have $\mathsf{parents}(v) \subseteq P_{i-1}$, and (3) $\forall v \in V, \exists i$ s.t. $v \in P_i$.

Intuitively, P_i denotes the *subset* of data-labels stored in memory at time i and $P_i \setminus P_{i-1}$ denotes the new data-labels that are computed during round i — the second constraint states that we can only compute these new data-labels if all of the necessary dependent data values were already in memory. The final constraint says that we must eventually pebble all nodes (otherwise we would never compute the output labels for $f_{G,H}$). We say that a pebbling is *sequential* if $\forall i>0$ we have $|P_i \setminus P_{i-1}| \leq 1$ i.e., in every round at most *one* new pebble is placed on the graph. We use $\mathcal{P}^{\parallel}(G)$ (resp. $\mathcal{P}(G)$) to denote the set of all valid parallel (resp. sequential) black pebblings of the DAG G. We define the space-time cost of a pebbling $P=(P_1,...,P_t) \in \mathcal{P}^{\parallel}_G$ to be $\mathsf{st}(P)=t \times \max_{1 \leq i \leq t}|P_i|$ and the sequential space-time pebbling cost, denoted $\Pi_{st}(G)=\min_{P \in \mathcal{P}_G}\mathsf{st}(P)$, is the space-time cost of the best legal pebbling of G.

There are many other pebbling games one can define on a DAG including the red-blue pebbling game [JWK81] and the black-white pebbling [Len81]. Red-blue pebbling games can be used to analyze the bandwidth-hardness of an iMHF [RD17, BRZ18]. In this work, we primarily focus on the (parallel) black pebbling game to analyze the amortized Area-Time complexity and the sustained space complexity of a memory-hard function.

Definition 1 (Time/Space/Cumulative Pebbling Complexity). *The time, space, space-time and cumulative complexity of a pebbling* $P=\{P_0,\ldots,P_t\} \in \mathcal{P}^{\parallel}_G$ *are defined to be:*

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

For $\alpha \in \{s,t,st,cc\}$ *the sequential and parallel pebbling complexities of* G *are defined as*

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

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 $\Pi^{\parallel}_{st}(G) \geq \Pi^{\parallel}_{cc}(G)$. Thus, we have $\Pi_{st}(G) \leq \Pi_{cc}(G) \leq \Pi^{\parallel}_{cc}(G)$ and $\Pi_{st}(G) \leq \Pi^{\parallel}_{st}(G) \leq \Pi^{\parallel}_{cc}(G)$. However, the relationship between $\Pi^{\parallel}_{st}(G)$ and $\Pi_{cc}(G)$ is less clear. It is easy to provide examples of graphs for which $\Pi_{cc}(G) \ll \Pi^{\parallel}_{st}(G)$ [5]. Alwen and Serbinenko showed that for the bit-reversal graph $G=\mathsf{BRG}_n$ with

[5] One example of such a graph G would be to start the pyramid graph $\triangle_k$, which has $\mathcal{O}(k^2)$ nodes, a single sink node t and append a path W of length k^3 starting at this sink

$\mathcal{O}(N=2^n)$ nodes we have $\Pi^{\parallel}_{st}(G)=\Omega(n\sqrt{n})$. We show that $\Pi_{cc}(G)=\Omega\left(N^2\right)$, thus, for some DAGs we have $\Pi_{cc}(G)\gg\Pi^{\parallel}_{st}(G)$.

Definition 2 (Sustained Space Complexity [ABP18]). *For* $s\in\mathbb{N}$ *the* s-sustained-space *(s-ss) complexity of a pebbling* $P=\{P_0,...,P_t\}\in\mathcal{P}^{\parallel}_G$ *is:* $\Pi_{ss}(P,s)=|\{i\in[t]:|P_i|\geq s\}|$. *More generally, the sequential and parallel s-sustained space complexities of G are defined as*

$$\Pi_{ss}(G,s)=\min_{P\in\mathcal{P}_G}\Pi_{ss}(P,s)\qquad \textit{and} \qquad \Pi^{\parallel}_{ss}(G,s)=\min_{P\in\mathcal{P}^{\parallel}_G}\Pi_{ss}(P,s)\ .$$

We remark that for any s we have $\Pi_{cc}(G)\geq\Pi_{ss}(G,s)\times s$ and $\Pi^{\parallel}_{cc}(G)\geq\Pi^{\parallel}_{ss}(G,s)\times s$.

2.3 Amortized Area-Time Cost (aAT)

Amortized Area-Time (aAT) cost is a way of viewing the cost to compute an iMHF, and it is closely related to the cost of pebbling a graph. Essentially, aAT cost represents the cost to keep pebbles in memory and adds in a factor representing the cost to compute the pebble. Here we require an additional factor, the core-memory ratio R, a multiplicative factor representing the ratio between computation cost vs memory cost. In this paper we are mainly focused on analysis of Argon2, which has previous calculations showing $R=3000$ [BK15]. It can be assumed that this value is being used for R unless otherwise specified. The formal definition of the aAT complexity of a pebbling $P=(P_0,...,P_T)$ of the graph G is as follows:

$$\mathsf{aAT}_R(P)=\sum_{i=1}^{T}|P_i|+R\sum_{i=1}^{T}|P_i/P_{i-1}|$$

The (sequential) aAT complexity of a graph G is defined to be the aAT complexity of the optimal (sequential) pebbling strategy. Formally,

$$\mathsf{aAT}_R(G)=\min_{P\in\mathcal{P}(G)}\mathsf{aAT}_R(G)\ \text{, and}\quad \mathsf{aAT}^{\parallel}{}_R(G)=\min_{P\in\mathcal{P}^{\parallel}(G)}\mathsf{aAT}_R(P)\ .$$

One of the nice properties of aAT and $\Pi^{\parallel}_{cc}$ complexity is that both cost metrics amortizes nicely i.e., if G^m consists of m independent copies of the DAG G then $\mathsf{aAT}^{\parallel}{}_R(G^m)=m\times\mathsf{aAT}^{\parallel}{}_R(G)$. We remark that $\mathsf{aAT}^{\parallel}{}_R(G)\geq\Pi^{\parallel}_{cc}(G)$, but that in most cases we will have $\mathsf{aAT}^{\parallel}{}_R(G)\approx\Pi^{\parallel}_{cc}(G)$ since the number of queries to the random oracle is typically $o\left(\Pi^{\parallel}_{cc}(G)\right)$. We will work with $\Pi^{\parallel}_{cc}(G)$ when conducting theoretical

node t. The pyramid graph requires $\Pi^{\parallel}_{s}\left(\Delta_k\right)=\Theta(k)$ space to pebble and has $\Pi_{cc}\left(\Delta_k\right)\leq\Pi_{st}\left(\Delta_k\right)\leq k^3$. Similarly, the path W requires at least $\Pi^{\parallel}_{t}(W)=\Pi_t(W)=k^3$ steps to pebble the path (even in parallel). Thus, $\Pi^{\parallel}_{st}(G)\geq k^4$. By contrast, we have $\Pi_{cc}(G)\leq\Pi_{cc}\left(\Delta_k\right)+k^3\leq k^3+k^3\ll k^4$ since we can place a pebble on node t with cost $\Pi_{cc}\left(\Delta_k\right)$, discard all other pebbles from the graph and then walk this pebble across the path.

analysis and we will use $\mathsf{aAT}^{\parallel}{}_R(G)$ when conducting empirical experiments, as the constant factor R is important in practice. This also makes it easier to compare our empirical results with prior work [AB17, ABH17].

2.4 Attack Quality

In many cases we will care about how efficient certain pebbling strategies are compared to others. When we work with an iMHF, we have a naive sequential algorithm $\mathcal{N}$ for evaluation e.g. the algorithm described in the Argon2 specifications [BDK16]. Typically, the naive algorithm $\mathcal{N}$ is relatively expensive e.g., $\mathsf{aAT}_R(\mathcal{N}) = N^2/2 + RN$. We say that an attacker $\mathcal{A}$ is *successful* at reducing evaluation costs if $\mathsf{aAT}_R(\mathcal{A}) < \mathsf{aAT}_R(\mathcal{N})$. Following [AB16] we define the quality of the attack as

$$\mathsf{AT\text{-}quality}(\mathcal{A}) = \frac{\mathsf{aAT}_R(\mathcal{N})}{\mathsf{aAT}_R(\mathcal{A})},$$

which describes how much more efficiently $\mathcal{A}$ is able to evaluate the function compared to $\mathcal{N}$.

3 Related Work

MHF Goals. Dwork et al. and Abadi et al. [DGN03, ABMW05] introduced the notion of a memory-bound function where we require that *any* evaluation algorithm results in a large number of cache-misses. Ren and Devadas recently introduced a refinement to this notion called bandwidth-hardness [RD17]. To the best of our knowledge Percival was the first to propose the goal that a MHF should have high space-time complexity [Per09]. Similarly, the Amortized Area-Time Complexity [AS15, ABH17] metric aims to *capture* the cost of the hardware (e.g., DRAM chips) the attacker must purchase to compute an MHF — amortized by the number of MHF instances computed over the lifetime of that hardware. aAT complexity addresses a concern raised by Alwen and Serbinenko [AS15], who observed that space-time costs do not always *amortize* nicely e.g., the space-time cost to compute a function 100 times might not be any larger than the space-time cost to compute the function a single time. By contrast, bandwidth hardness [RD17] aims to capture the *energy cost* of the electricity required to compute the MHF once. If the attacker uses an ASIC to compute the function then the *energy* expended during computation will typically be small in comparison with the *energy* expended during a cache-miss. Thus, a bandwidth hard function aims to ensure that *any* evaluation strategy incurs a large number $\Omega(n)$ of cache-misses during computation.

In Appendix A we argue that, in the context of password hashing, aAT complexity is more relevant than bandwidth hardness because costs can scale quadratically in the running time parameter N. However, one would ideally want to design a MHF that has high aAT complexity and is also maximally bandwidth hard. Blocki et al. [BRZ18] recently showed that any MHF with high aAT complexity is at least somewhat bandwidth hard. Furthermore, all practical iMHFs (including Catena-Bit Reversal [FLW14], Argon2i and DRSample) are maximally bandwidth hard [RD17, BRZ18], including our new construction DRS+BRG.

Password Hashing Competition. MHFs were of particular interest in the 2015 Password Hashing Competition [Pin14], where the winner, Argon2 [BDK16], and all but one finalists [FLW14, SJAA$^+$15, Pes14] claimed some form of memory hardness. A second area where memory-hard functions have garnered interest is in cryptocurrencies, with both Litecoin [Lee11] and Ethereum [But13] using MHFs.
dMHFs vs iMHFs. Certain dMHFs such as SCRYPT [PJ12] achieve a higher aAT complexity [ACP$^+$17] than any iMHF can possibly have [AB16]. There are some constructions that include hybrid independent-dependent modes, such as Argon2id [KDBJ17]. In this instance, both dependent and independent modes are run for half of the computation time, providing the benefits from both types of MHF. We remark that a side-channel attack on a hybrid "id" mode of operation will reduce security to that of the underlying iMHF at best. Thus, even if one adopts a hybrid mode of operation (e.g., the Argon2 specs have been updated to list Argon2id as the recommended mode of operation for password hashing), it is crucial to ensure that the original iMHF is as strong as possible.

4 Analysis of the Greedy Pebbling Algorithm

In this section we present a theoretical and empirical analysis of the greedy pebbling attack [BCS16] that *reverses* previous results about the *practical* security of Argon2i vs DRSample [ABH17]. We prove two main results using the greedy algorithm. First, we show that for any N node DAG G with a unique topological ordering, we have $\mathsf{aAT}_R(G) \leq \frac{N^2+2N}{4} + RN$ — see Theorem 1. Second, we prove that for any constant $\delta > 0$ and a random DRSample DAG G on N nodes, we have $\Pi_{st}(G) \leq (1+\delta)2N^2/\log N$ with high probability — see Theorem 2. We stress that in both cases the bounds are *explicit* not *asymptotic*, and that the pebbling attacks are simple and sequential.

Alwen and Blocki [AB16] previously had shown that any DAG G with constant indegree has $\mathsf{aAT}^{\|}{}_R(G) \in \Theta(N^2 \log\log N/\log N)$, but the constants from this bound were not well understood and did not rule out the existence of an N node DAG G with $\mathsf{aAT}^{\|}{}_R(G) \geq N^2/2 + RN$ for *practical* values of $N \leq 2^{24}$. By contrast, Theorem 1 immediately implies that $\mathsf{aAT}^{\|}{}_R(G) \leq \frac{N^2+2N}{4} + RN$. Similarly, Alwen et al. [ABH17] previously showed that with high probability a DRSample DAG G has $\mathsf{aAT}^{\|}{}_R(G) \in \Omega(N^2/\log N)$, but the constants in this lower bound were not well understood. On a theoretical side our analysis shows that this bound is tight i.e., $\mathsf{aAT}^{\|}{}_R(G) \in \Theta(N^2/\log N)$. It also proves that DRSample does not match the generic upper bound of Alwen and Blocki [AB16], who showed that any DAG G with constant indegree has $\mathsf{aAT}^{\|}{}_R(G) \in \Theta(N^2 \log\log N/\log N)$.
Argon2i vs. DRSample. It is known that $\mathsf{aAT}^{\|}(G) \in \mathcal{O}(N^{1.767})$ for an Argon2i DAG, while we have $\mathsf{aAT}^{\|}{}_R(G) \in \Theta(N^2/\log N)$. Prior empirical analysis of Alwen et al. [ABH17] indicated that, for practical values of $N \leq 2^{24}$, DRSample also provided greater resistance to pebbling attacks than other iMHF candidates like Argon2i. However, in their empirical analysis they did not consider the greedy pebbling attack of Boneh et al. [BCS16], perhaps because the attack on Argon2i only reduced memory

usage by a small constant factor (2.71–4) while the depth-reducing attacks of Alwen and Blocki [AB16, AB17] led to asymptotic improvements. Our analysis (theoretical and empirical) demonstrates that the greedy pebbling attack leads to sufficiently high quality attacks against DRSample to reverse the conclusion of Alwen et al. [ABH17]. In particular, for practical values of $N \leq 2^{24}$ Argon2i provides better resistance to *known* pebbling attacks despite the fact that DRSample provides much stronger asymptotic guarantees. Furthermore, the best attack on DRSample (greedy pebble) is sequential and potentially simple to implement, while the best pebbling attacks on Argon2i require higher parallelism and only reduce the amortized cost of evaluating *multiple* instances of the function. This has led to some questions about the feasibility of these depth-reducing attacks in practice. For example, Alwen and Blocki [AB17] sought to explicitly quantify and control the required parallelism/bandwidth in their empirical analysis of their depth-reducing attacks.

Extension of the Greedy Pebbling Attack Our analysis leaves us in an uncomfortable position where *every practical* iMHF candidate has high quality pebbling attacks i.e., greedy pebble for DRSample and depth-reducing attacks for Argon2i. We would like to develop a practical iMHF candidate that provides strong resistance against all known pebbling attacks for all practical values of $N \leq 2^{24}$. We first consider a defense proposed by Biryukov et al. [BDK16] against the greedy pebbling attack. While this defense provides optimal protection against the greedy pebbling attack, we introduce an extension of the greedy pebbling attack which we call the *staggered* greedy pebbling attack and show that the trick of Biryukov et al. [BDK16] fails to protect against the extended attack.

4.1 The greedy pebbling algorithm

We first review the greedy pebbling algorithm, shown here as Algorithm 1.We first introduce some notation

$\mathsf{gc}(v)$**:** For each node $v < N$ we let $\mathsf{gc}(v) = \max\{w | (v,w) \in E\}$ denote the maximum child of node v — if $v < N$ then the set $\{w | (v,w) \in E\}$ is non-empty as it contains the node $v+1$. If node v has no children then set $\mathsf{gc}(v) := v$.

$\chi(i)$**:** This represents what we call the crossing set of the ith node. It is defined as $\chi(i) = \{v | v \leq i \wedge \mathsf{gc}(v) > i\}$. Intuitively this represents the number of edges that cross from nodes before i to nodes after i.

Greedy Pebbling Strategy: Set $\mathsf{GP}(G) = P = (P_1,...,P_N)$ where $P_i = \chi(i)$ for each $i \leq N$. Intuitively, the pebbling strategy can be described follows: In round i we place a pebble on node i and we then discard *any* black pebbles on all nodes v that are no longer needed in any future round i.e., for all future nodes $w > i$ we have $v \notin \mathsf{parents}(w)$ (equivalently, the greatest-child of node v is $\mathsf{gc}(v) \leq i$). We refer the reader to Algorithm 1 in the appendix for a formal algorithmic description.

We first prove the following *general* lower bound for *any* N node DAG with $\mathsf{indeg}(G) \leq 2$ that has a unique topological ordering i.e., G contains each of the edges $(i,i+1)$. In particular, Theorem 1 shows that for any such DAG G we have

$\Pi_{st}(G) \lessapprox \frac{N^2}{2}$ and $\Pi_{cc}(G) \lessapprox N^2/4$. We stress that this is *twice* as efficient as the naive pebbling algorithm $\mathcal{N}$ which set $P_i = \{1,...,i\}$ for each $i \le N$ and has cumulative cost $\Pi_{cc}^{\parallel}(\mathcal{N}) = \frac{N^2}{2}$. Previously, the gold standard was to find constructions of DAGs G with N nodes such that $\Pi_{cc}^{\parallel}(G) \gtrapprox \frac{N^2}{2}$ for *practical* values of N — asymptotic results did not rule out this possibility even for $N \le 2^{40}$. Theorem 1 demonstrates that the best we could hope for is to ensure $\Pi_{cc}^{\parallel}(G) \gtrapprox \frac{N^2}{4}$ for *practical* values of N.

Theorem 1. *Let $r : \mathbb{N}_{>0} \rightarrow \mathbb{N}$ be* any *function with the property that $r(i) < i-1$ for all $i \in \mathbb{N}_{>0}$. Then the DAG $G = (V,E)$ with N nodes $V = \{1,...,N\}$ and edges $E = \{(i-1,i) \ : \ 1 < i \le N\} \cup \{(r(i),i) \ : \ 2 < i \le N\}$ has $\Pi_{st}(G) \le \frac{N^2+2N}{2}$ and $\Pi_{cc}(G) \le \frac{N^2+2N}{4}$ and $\mathsf{aAT}_R(G) \le \frac{N^2+2N}{4} + RN$.*

The full proof of Theorem 1 is in Appendix C. Intuitively, Theorem 1 follows from the observation that in any pebbling we have $|P_i| \le i$, and in the greedy pebbling we also have $|P_i| \le N-i$ since there can be at most $N-i$ nodes w such that $w = r(v)$ for some $v > i$ and other pebbles on any other node would have been discarded by the greedy pebbling algorithm.

4.2 Analysis of the Greedy Pebble Attack on DRSample

We now turn our attention to the specific case of the iMHF DRSample. The DRSample distribution is defined formally in Algorithm 3 in the appendix. A DAG G sampled from this distribution has edges of the form $(i,i+1)$ and $(r(i),i)$ where each $r(i) < i$ is independently selected from some distribution. It is not necessary to understand all of the details of this distribution to follow our analysis in this section as the crucial property that we require is given in Claim 1 which is proved in Appendix C. Intuitively, Claim 1 follows because we have $\Pr[r(j) = i] \sim \frac{1}{\log j} \times \frac{1}{|j-i|}$ for each node $i < j$ in DRSample.

Claim 1 *Let G be a randomly sampled DRSample DAG with N nodes and let $Y_{i,j}$ be an indicator random variable for the event that $r(j) < i$ for nodes $i < j \le N$ then we have $\mathbf{E}[Y_{i,j}] = \Pr[r(j) < i] \le 1 - \frac{\log(j-i-1)}{\log j}$.*

If $P = (P_1,...,P_N) = \mathsf{GP}(G)$, then we remark that $\chi(i)$ can be viewed as an alternate characterization of the set $P_i = \chi(i)$ of pebbles on the graph at time i. Lemma 1 now implies that with high probability, we will have $|P_i| \le (1+\delta)N/n$ during *all pebbling rounds.*

Lemma 1. *Given a DAG G on $N = 2^n$ nodes sampled using the randomized DRSample algorithm for any $\delta > 0$ we have*

$$\Pr\left[\max_i |\chi(i)| > (1+\delta)\left(\frac{2N}{n}\right)\right] \le \exp\left(\frac{-2\delta^2 N}{3n} + n\ln 2\right) .$$

Lemma 1, which bounds the size of $\max_i |\chi(i)|$, is proved in Appendix C. Intuitively, the proof uses the observation that $\chi(i) \le \sum_{j=i+1}^{N} Y_{i,j}$ where $Y_{i,j}$ is an indicator random variable for the event that $r(j) \le i$. This is because $\chi(i)$ is upper bounded by the number of edges that "cross" over the node i. We can then use Claim 1 and standard concentration bounds to obtain Lemma 1.

Theorem 2, our main result in this section, now follows immediately from Lemma 1. Theorem 2 states that, except with negligibly small probability, the sequential pebbling cost of a DRSample DAG is at most $(1+\delta)\left(\frac{2N^2}{n}\right)+RN$.

Theorem 2. *Let G be a randomly sampled DRSample DAG with $N=2^n$ nodes then for all $\delta>0$ we have*

$$\Pr\left[\Pi_{st}(GP(G))>(1+\delta)\left(\frac{2N^2}{n}\right)\right]\leq\exp\left(\frac{-2\delta^2N}{3n}+n\ln 2\right) .$$

Proof. Fix $\delta>0$ and consider a randomly sampled N-node DRSample DAG G. We let $P=\mathsf{GP}(G)$. We observe that $|P_i|=\chi(i)$. By Lemma 1, except with probability $\exp\left(\frac{-\delta^2N/n}{3}\right)$, we have

$$N\times\max_{i\in[N]}\chi(i)\leq(1+\delta)\left(\frac{2N^2}{n}\right) . \tag{1}$$

Assuming that Equation 1 holds, we have $\mathsf{aAT}^{\parallel}{}_R(GP(G))\leq(1+\delta)\left(\frac{2N^2}{n}\right)+RN$ as claimed. □

Discussion. Theorem 2 implies that the (sequential) aAT complexity of DRSample is $\mathsf{aAT}_R(G)\lessapprox 2N^2/\log N\in\mathcal{O}(N^2/\log N)$, which asymptotically matches the lower bound of $\Omega(N^2/\log N)$ [ABH17]. More significant from a practical standpoint is that the constant factors in the upper bound are given explicitly. Theorem 2 implies attack quality at least $\gtrapprox\frac{\log N}{4}$ since the cost of the naive pebbling algorithm is $N^2/2$. Thus, for *practical* values of $N\leq 2^{24}$ we will get high quality attacks and our empirical analysis suggests that attack quality actually scales with $\log N$. On a positive note the pebbling attack is *sequential*, which means that we could adjust the naive (honest) evaluation algorithm to simply use $\mathcal{N}$ to use $\mathsf{GP}(G)$ instead because the greedy pebbling strategy is *sequential.* While this would lead to an *egalitarian* function, the outcome is still undesirable from the standpoint of password hashing where we want to ensure that the attacker's absolute aAT costs are as high as possible given a fixed running time N.

4.3 Empirical analysis of the GP attack

We ran the greedy pebbling attack against several iMHF DAGs, including Argon2iA, Argon2iB, and DRSample, focusing on the notion of Attack Quality from Section 2.4. The results, seen in Figure 2, showed that the GP attack was especially effective against the DRSample DAG, improving attack quality by a factor of up to 7 (at $n=24$) when compared to previous depth-reducing attacks (Valiant, Layered, and various hybrid approaches) [Val77, AB16].

When testing the GP attack against Argon2i we found that while the Greedy Pebbling attack does sometimes outperform depth-reducing attacks at smaller values of n, the depth-reducing attacks appear to be superior once we reach graph sizes that would likely be used in practice. As an example, when $n=20$ we find that the attack quality of the greedy pebbling attack is just 2.99, while the best depth-reducing attack achieved attack quality 6.25 [ABH17].

The most important observation about Figure 2a is simply how effective the greedy pebbling attack is against this graph. We remark that attack quality for DRSample with $N=2^n$ nodes seems to be approximately n — slightly better than the theoretical guarantees from Theorem 2. While DRSample may have the strongest asymptotic guarantees i.e. $\mathsf{aAT}^{\parallel}(G)=\Omega(N^2/\log N)$, Argon2i seems to provide better resistance to known pebbling attacks for *practical* parameter ranges.

4.4 Defense Against Greedy Pebbling Attack: Attempt 1 XOR extension

Biryukov et al. [BDK16] introduced a simple defense against the greedy pebbling attack of Boneh et al. [BCS16] for iMHFs that make two passes over memory. Normally during computation the block $B_{i+N/2}$ would be stored at memory location i overwriting block B_i. The idea of the defense is to XOR the two blocks $B_{i+N/2}$ and B_i before overwriting block B_i in memory. Biryukov et al. [BDK16] observed that this defense does not *significantly* slow down computation because block B_i would have been loaded into cache before it is overwritten in either case. The effect of performing this extra computation is effectively to add each edge of the form $(i-\frac{N}{2},i)$ to the DAG G. In particular, this means that the greedy pebbling algorithm will not discard the pebble on node $i-\frac{N}{2}$ until round i, which is when the honest pebbling algorithm would have discarded the pebble anyway. Given a graph $G=(V,E)$ we use $G^{\oplus}=(V,E^{\oplus})$ to denote the XOR-extension graph of G where $E^{\oplus}=E\cup\{(i-\frac{N}{2},i) \mid i>\frac{N}{2}\}$. It is easy to see that $\Pi_{cc}^{\parallel}(\mathsf{GP}(G^{\oplus}))\geq\frac{N^2+2N}{4}$, which would make it tempting to conclude that the XOR-extension defeats the greedy pebbling attack.
Greedy Pebble Extension: Given a graph G on N nodes let $P=(P_1,...,P_N)=\mathsf{GP}(G)$. Define $\mathsf{GPE}(G^{\oplus})=\left(P_1^{\oplus},...,P_N^{\oplus}\right)$ where $P_{i+N/2-1}^{\oplus}=P_i\cup P_{i+N/2-1}$. See Algorithm 2 in the appendix for a formal algorithm presentation. Intuitively, the attack exploits the fact that *always* ensure that we have a pebble on the extra node $v\in\mathsf{parents}(N/2+v)$ at time $N/2+v-1$ by using the greedy pebble algorithm to synchronously re-pebble the nodes $1,...,N/2$ a second time.

Theorem 3 demonstrates that the new generalized greedy pebble algorithm is effective against the XOR-extension gadget. In particular, Corollary 2 states that we still obtain high quality attacks against $DRSample^{\oplus}$ so the XOR-gadget does significantly improve the aAT cost of DRSample.

Theorem 3. *Let $r:\mathbb{N}_{>0}\rightarrow\mathbb{N}$ be* any *function with the property that $r(i)<i$ for all $i\in\mathbb{N}_{>0}$ and let $G=(V,E)$ be a graph with N nodes $V=\{1,...,N\}$ and directed edges $E=\{(i,i+1) \mid i<N\}\cup\{r(i),i \mid 1<i\leq N\}$. If $P=\mathsf{GP}(G)\in\mathcal{P}(G)$ then the XOR-extension graph $G^{\oplus}$ of G has amortized Area-Time complexity at most*

$$\mathsf{aAT}^{\parallel}{}_R\left(G^{\oplus}\right)\leq\sum_{i=1}^{N/2}|P_i|+\sum_{i=1}^{N}|P_i|+\frac{3RN}{2}\ .$$

Corollary 1. *Let $r:\mathbb{N}_{>0}\to\mathbb{N}$ be* any *function with the property that $r(i)<i$ for all $i\in\mathbb{N}_{>0}$ and let $G=(V,E)$ be a graph with N nodes $V=\{1,\dots,N\}$ and directed edges $E=\{(i,i+1)\mid i<N\}\cup\{r(i),i\mid 1<i\leq N\}$. Then for the XOR-extension graph $G^{\oplus}$ we have $\mathsf{aAT}^{\parallel}{}_R(G^{\oplus})\leq\frac{5N^2+12N}{16}+\frac{3RN}{2}$.*

The proof of Theorem 3 can be found in the appendix. One consequence of Theorem 3 is that the XOR-extension gadget does not rescue DRSample from the greedy pebble attack — see Corollary 2.

Corollary 2. *Let $G=(V,E)$ be randomly sampled DRSample DAG with $N=2^n$ nodes $V=\{1,\dots,N\}$ and directed edges $E=\{(i,i+1)\mid i<N\}\cup\{r(i),i\mid 1<i\leq N\}$. Then*

$$\Pr\left[\mathsf{aAT}^{\parallel}{}_R\left(G^{\oplus}\right)>(1+\delta)\left(\frac{3N}{n}-\frac{2N}{n(n-1)}\right)+\frac{3RN}{2}\right]\leq\exp\left(\frac{-\delta^2N}{3(n-1)}+1+n\ln2\right)\ .$$

Proof. Fix $\delta>0$ and let $P=\mathsf{GP}(G)$ where G is a randomly sampled DRSample DAG. By Lemma 1, except with probability $\exp\left(\frac{-2\delta^2N}{3n}+n\ln2\right)$, we have $\max_i|P_i|=\max_i\chi(i)\leq(1+\delta)\frac{2N}{n}$ which means that $\sum_{i=1}^{N}|P_i|\leq(1+\delta)\frac{2N}{n}$. Similarly, except with probability $\exp\left(\frac{-\delta^2N}{3(n-1)}+(n-1)\ln2\right)$ we have $\max_{i\leq N/2}|P_i|=\max_i\chi(i)\leq(1+\delta)\frac{N}{n-1}$ since the first $N/2$ nodes of G form a random DRSample DAG $N/2=2^{n-1}$ nodes. This would imply that $\sum_{i=1}^{N/2}|P_i|\leq(1+\delta)\frac{N}{n-1}$. Putting both bounds together Theorem 3 implies that $\mathsf{aAT}^{\parallel}(G^{\oplus})\leq(1+\delta)\left(\frac{3N}{n}-\frac{2N}{n(n-1)}\right)+\frac{3RN}{2}$. $\square$

5 New iMHF Construction with Optimal Security

In this section we introduce a new iMHF construction called DRSample+BRG. The new construction is obtained by overlaying a bit-reversal graph BRG_n [LT82] on top of a random DRSample DAG. If G denotes a random DRSample DAG with $N/2$ nodes then we will use $\mathsf{BRG}(G)$ to denote the bit-reversal overlay with N nodes. Intuitively, the result is a graph that resists both the greedy pebble attack (which is very effective against DRSample alone) and depth-reducing attacks (which DRSample was designed to resist). We show that this new DAG has *many* other desirable properties for an iMHF.

First, our new construction inherits desirable properties from *both* the bit-reversal graph and DRSample. For example,

$$\Pi_{cc}^{\parallel}(\mathsf{BRG}(G))\geq\Pi_{cc}^{\parallel}(G)=\Omega\left(N^2/\log N\right)\ .$$

Similarly, it immediately follows that $\mathsf{BRG}(G)$ is maximally bandwidth hard. In particular, Ren and Devadas [RD17] showed that BRG_n is maximally bandwidth hard, and Blocki et al. [BRZ18] showed that DRSample is maximally bandwidth hard.

Second, $\mathsf{BRG}(G)$ provides optimal resistance to the greedy pebbling attack — $\Pi_{cc}^{\parallel}(\mathsf{GP}(\mathsf{BRG}(G)))\approx N^2/4$. Furthermore, we can show that *any* c-parallel pebbling attack $P=(P_1,\dots,P_t)$ in which $|P_{i+1}\setminus P_i|\leq c$ has cost $\Pi_{cc}^{\parallel}(P)=\Omega\left(N^2\right)$. This rules out any *extension* of the greedy pebble attack i.e., if $c=2$-parallel. In fact, we prove that this property already holds for any parallel pebbling of the bit reversal graph BRG_n. Our proof that $\Pi^{\parallel}_{cc}(\mathsf{BRG}_n)=\Omega(N^2)$ generalizes the well-known result that $\Pi_{st}(\mathsf{BRG}_n)=\Omega(N^2)$ and may be of independent interest.

Third, we can show that *any* parallel pebbling P of $\mathsf{BRG}(G)$ *either* has $\Pi(P)=\Omega\left(N^2\right)$ or has maximal sustained space complexity $\Pi_{ss}(P,s)=\Omega(N)$ for space $s=\Omega(N/\log N)$ i.e., there are at least $\Omega(N)$ steps with at least $\Omega(N/\log N)$ pebbles on the graph. To prove this last property we must rely on properties of *both* graphs G and BRG_n. This makes $\mathsf{BRG}(G)$ the first *practical* construction of a DAG with provably strong sustained space complexity guarantees.

Finally, we can show that $\Pi^{\parallel}_{cc}(G)=\Omega\left(N^2\log\log N/\log N\right)$, matching the general upper bound of Alwen and Blocki [AB16], under a plausible conjecture about the block-depth-robustness of G. In particular, we conjecture that G is (e,d,b)-block depth robust for $e=\Omega\left(\frac{N\log\log N}{\log N}\right)$, $d=\Omega\left(\frac{N\log\log N}{\log N}\right)$ and $b=\Omega\left(\frac{\log N}{\log\log N}\right)$. In the appendix, we also show how to construct a constant indegree DAG G' with $\Pi^{\parallel}_{cc}(G')=\Omega\left(N^2\log\log N/\log N\right)$ from *any* (e,d)-depth robust graph by overlaying a superconcentrator on top of G [Pip77]. However, the resulting construction is not *practically efficient.* For the bit reversal overlay $G'=\mathsf{BRG}(G)$ we require the slightly stronger assumption that G is *block*-depth-robust. As evidence for the conjecture we show that *known* attacks require the removal of a set S of $e=\Omega\left(\frac{N\log\log N}{\log N}\right)$ to achieve $\mathsf{depth}(G-S)\le\frac{N}{\sqrt{\log N}}$. Thus, we would need to find *substantially* improved depth-reducing attacks to refute the conjectures.

Bit-Reversal Graph Background: The bit reversal graph was originally proposed by Lenguer and Tarjan [LT82] who showed that any sequential pebbling has maximal space-time complexity. Forler et al. [FLW14] previously incorporated this graph into the design of their iMHF candidate Catena which received special recognition at the password hashing competition [PHC16]. While we are not focused on sequential space-time complexity the bit reversal graph has several other useful properties that we exploit in our analysis (see Lemma 2).

Local Sample-able. We note that one benefit of DRS+BRG is that it is locally sample-able, a notion mentioned as desirable in [ABH17]. Specifically, we want to be able to compute the parent blocks with time and space $\mathcal{O}(\log|V|)$ with small constants. DRS+BRG meets this requirement. Edges sampled from DRSample were shown to be locally navigable in [ABH17], and each bit-reversal edge a simple operation called requires one bit reversal operation, which can easily be computed in $\mathcal{O}(\log|V|)$. The formal description of the bit-reversal overlay graph $\mathsf{BRG}(G)$ is presented in Definition 4 and is presented in algorithmic form in Algorithm 4 in the appendix.

The Bit-Reversal DAG Given a sequence of bits $X=x_1\circ x_2\circ\cdots x_n$, let $\mathsf{ReverseBits}(X)=x_n\circ x_{n-1}\circ\cdots\circ x_1$. Let $\mathsf{integer}(X)$ be the integer representation of bit-string X starting at 1 so that $\mathsf{integer}(\{0,1\}^n)=[2^n]$ i.e., $\mathsf{integer}(0^n)=0$ and $\mathsf{integer}(1^n)=2^n$. Similarly, let $\mathsf{bits}(v,n)$ be the length n binary encoding of $(v-1) \mod 2^n$ e.g., $\mathsf{bits}(v,n)=0^n$ and $\mathsf{bits}(2^n,n)=1^n$ so that for all $v\in[2^n]$ we have $\mathsf{integer}(\mathsf{bits}(v,n))=v$.

Definition 3. *We use the notation* BRG_n *to denote the bit reversal graph with* 2^{n+1} *nodes. In particular,* $\mathsf{BRG}_n = \left(V = \left[2^{n+1}\right], E = E_1 \cup E_2\right)$ *where* $E_1 := \{(i,i+1) \ : \ 1 \leq i < 2^{n+1}\}$ *and* $E_2 := \{(x, 2^n + y) \ : \ x = \mathsf{integer}(\mathsf{ReverseBits}(\mathsf{bits}(y,n)))\}$*. That is,* E_2 *contains an edge from node* $x \leq 2^n$ *to node* $2^n + y$ *in* BRG_n *if and only if* $x = \mathsf{integer}(\mathsf{ReverseBits}(\mathsf{bits}(y,n)))$*.*

Claim 2 $\Pi(\mathsf{GP}(\mathsf{BRG}_n)) \geq N^2 + N$

Proof. Let $P = (P_1,...,P_{2N}) = \mathsf{GP}(\mathsf{BRG}_n)$. We first note that for all $i \leq N$ we have $P_i = \{1,...,i\}$ since $\mathsf{gc}(i) > N$ — every node on the bottom layer $[N]$ has an edge to some node on the top layer $[N+1,2N]$. Second, observe that for any round $i > N$ we have $|(P_i \setminus P_{i+1}) \cap [N]| \leq 1$ since the only pebble in $[N]$ that might be discarded is the (unique) parent of node i. Thus,

$$\sum_{i=1}^{2N} |P_i| \geq \sum_{i=1}^{N} i + \sum_{i=1}^{N} (N-i+1) = N(N+1) \ . \qquad \square$$

Thus, we now define the bit-reversal overlay of the bit reversal graph on a graph G_1. If the graph G_1 has N nodes then $\mathsf{BRG}(G_1)$ has $2N$ nodes, and the subgraph induced by the first N nodes of $\mathsf{BRG}(G_1)$ is simply G_1.

Definition 4. *Let* $G_1 = (V_1 = [N], E_1)$ *be a fixed DAG with* $N = 2^n$ *nodes and* $\mathsf{BRG}_n = (V = [2N], E)$ *denote the bit-reversal graph. Then we use* $\mathsf{BRG}(G_1) = (V, E \cup E_1)$ *to denote the bit-reversal overlay of* G_1*.*

In our analysis, we will rely heavily on the following key-property of the bit-reversal graph from Lemma 2.

Lemma 2. *Let* $G = \mathsf{BRG}_n$ *and* $N = 2^n$ *so that* G *has* $2N$ *nodes. For a given* b*, partition* $[N]$ *into* $\frac{N}{2^{n-b}} = 2^b$ *intervals*

$$I_k = \left[N + (k-1)2^{n-b} + N + k2^{n-b} - 1\right],$$

each having length 2^{n-b}*, for* $1 \leq k \leq 2^b$*. Then for any interval* I *of length* 2^b*, with* $I \subseteq [N+1,2N]$*, there exists an edge from each* I_k *to* I*, for* $1 \leq k \leq 2^b$*.*

Proof of Lemma 2. Let I be any interval of length 2^b, with $I \subseteq [N+1,2N]$. Note that every 2^b length bitstring appears as a suffix in I. Thus, there exists an edge from each interval containing a unique 2^b length bitstring as a prefix. It follows that there exists an edge from each I_k to I, for $1 \leq k \leq 2^b$. $\square$

As we will see, the consequences of Lemma 2 will have powerful implications for the pebbling complexity of $\mathsf{BRG}(G_1)$ whenever the underlying DAG G_1 is (e,d,b)-block-depth-robust. In particular, Lemma 3 states that if we start with pebbles on a set $|P_i| < e/2$ then for *any initially empty* interval I of $\mathcal{O}(N/b)$ consecutive nodes in the top-half of G_1 we have the property that $H := G - \bigcup_{x \in P_i} [x-b+1, x]$ is an $(e/2,d,b)$-block-depth-robust graph that will need to be *completely re-pebbled* (at cost *at least* $\Pi_{cc}^{\parallel}(H) \geq ed/2$) just to advance a pebble across the interval I. See Appendix C for the proof of Lemma 3.

Lemma 3. *Let $G_1 = (V_1 = [N], E)$ be a (e, d, b)-block depth-robust graph with $N = 2^n$ nodes and let $G = \mathsf{BRG}(G_1)$ denote the bit-reversal extension of G_1 with $2N$ nodes $V(G) = [2N]$. For any interval $I = \left[N+i+1, N+i+1+\frac{4N}{b}\right] \subseteq [2N]$ and any $S \subseteq [1, N+i]$ with $|S| < \frac{e}{2}$, $\mathsf{ancestors}_{G-S}(I)$ is $\left(\frac{e}{2}, d, b\right)$-block depth-robust.*

Lemma 4. *Let G be any (e,d,b)-Block Depth Robust DAG with $N = 2^n$ and let $G' = \mathsf{BRG}(G)$ be the bit reversal overlay of G. Let $P \in \mathcal{P}^{\parallel}(G')$ be a legal pebbling of G' and let t_v be the first time where $v \in P_{t_v}$. Then for all $v \geq 1$ such that $e' := |P_{t_{v+N}}| \leq \frac{e}{4}$ and $v \leq N - \frac{32Ne'}{be}$, we have*

$$\sum_{j=t_{v+N}}^{t_{v+N+\frac{32Ne'}{be}-1}} |P_j| = \Omega(ed).$$

Proof of Lemma 4. For $S = P_{t_{N+v}}$ and $I = [N+v+1, N+v+\frac{8N}{b'}]$, we have

$$H := G - \bigcup_{x \in S} [x-b+1, x] \subseteq \mathsf{ancestor}_{G'-S}(I)$$

because if $v \notin \bigcup_{x \in S}[x-b+1, x]$ then $[v, v+b-1] \cap S = \varnothing$ which implies that there exists an "S-free path" from v to I by Lemma 2. Thus, H will have to be repebbled completely at some point during the time interval $\left[t_{v+N}, t_{v+N+\frac{32Ne'}{be}-1}\right]$.

Set $b' = \frac{eb}{4e'}$ and note that $e'\left(\frac{b'}{b}\right) + e' = \frac{e}{4} + e' \leq \frac{e}{2}$. Hence, Lemma 3 implies that H is still $(e/2, d, b)$-Block Depth Robust and, consequently, we have that $\Pi^{\parallel}_{cc}(H) \geq ed/2$ by [ABP17]. We can conclude that

$$\sum_{j=t_{v+N}}^{t_{v+N+\frac{32Ne'}{be}-1}} |P_j| \geq \Pi^{\parallel}_{cc}(H) \geq ed/2\ . \qquad \square$$

5.1 Sustained Space Complexity (Tradeoff Theorem)

We prove that for any parameter $e = \mathcal{O}\left(\frac{N}{\log N}\right)$ the cumulative pebbling cost of any parallel (legal) pebbling P is has cost least $\Pi(P) = \Omega(N^3/(e\log N))$, or there are at least $\Omega(N)$ steps with at least e pebbles on the graph i.e., $\Pi_{ss,e}(P) = \Omega(N)$. Note that the cumulative pebbling cost rapidly increases as e decreases e.g., if $e = \sqrt{N}/\log N$ then any pebbling P for which $\Pi_{ss}(P,e) = o(N)$ must have $\Pi(P) = \Omega(N^{2.5})$.

To begin we start with the known result that (with high probability) a randomly sampled DRSample DAG G is (e,d,b)-Block Depth Robust with $e = \Omega(N/\log N)$, $b = \Omega(\log N)$, and $d = \Omega(N)$ [ABH17]. Lemma 5 now implies that the DAG is also (e', d, b')-block-depth robust for any suitable parameters e' and b'. Intuitively, if we delete e' intervals of length $b' > b$ then we can cover these deleted intervals with *at most* $e' + \left(\frac{b'}{b} + 1\right)$ intervals of length b, as illustrated in Figure 1. The formal proof of Lemma 5 is in the appendix.

Lemma 5. *Suppose that a DAG G is (e,d,b)-Block Depth Robust and that parameters e' and b' satisfies the condition that $e'\left(\frac{b'}{b}\right)+e' \le \frac{e}{2}$. Then G is (e',d,b')-Block Depth Robust, and for all S with size $|S| \le e'$ the graph $H = G - \bigcup_{x\in S}[x-b'+1,x]$ is $\left(\frac{e}{2},d,b\right)$-Block Depth Robust.*

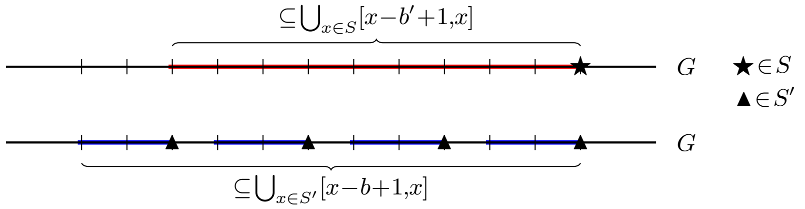

$\subseteq \bigcup_{x\in S}[x-b'+1,x]$
G
$\bigstar \in S$
$\blacktriangle \in S'$
G
$\subseteq \bigcup_{x\in S'}[x-b+1,x]$

Fig. 1: Intervals $\bigcup_{x\in S}[x-b'+1,x]$ and $\bigcup_{x\in S'}[x-b+1,x]$ when $b'=10$ and $b=3$. Observe that $\bigcup_{x\in S'}[x-b+1,x] \supset \bigcup_{x\in S}[x-b'+1,x]$.

Together Lemma 4 and Lemma 5 imply that we must incur pebbling cost $\Omega(ed)$ to pebble *any* interval of $\Omega\left(\frac{Ne'}{be}\right)$ consecutive nodes in the top-half of $\mathsf{BRG}(G)$ starting from *any* configuration with at most $e' \le e/4$ pebbles on the graph.

Theorem 4, our main result in this subsection, now follows because for any pebbling $P \in \Pi^{\parallel}(\mathsf{BRG}(G))$ and any interval I of $\Omega\left(\frac{Ne'}{be}\right)$ nodes in the top-half of G we must either (1) keep at least e' pebbles on the graph while we walk a pebble accross the first half of the interval I, or (2) pay cost $\Omega(ed)$ to re-pebble a depth-robust graph. Since there are $\Omega\left(\frac{eb}{e'}\right)$ such disjoint intervals we must either keep $|P_i| \ge e'$ pebbles on the graph for $\Omega(N)$ rounds, or pay cost $\Pi^{\parallel}_{cc}(P) \ge \frac{e^2db}{64e'}$.

Theorem 4. *Let G be any (e,d,b)-Block Depth Robust DAG on $N=2^n$ nodes, and $G' = \mathsf{BRG}(G)$ be the bit reversal overlay of G. Then for any pebbling $P \in \Pi^{\parallel}(G)$ and all $e' \le \frac{e}{4}$, we have either $\Pi^{\parallel}_{cc}(P) \ge \frac{e^2db}{64e'}$, or $\Pi_{ss}(P,e') \ge \frac{N}{4} - o(N)$ i.e., at least $\frac{N}{4}-o(N)$ rounds i in which $|P_i| \ge e'$.*

Corollary 3 follows immediately from Theorem 4.

Corollary 3. *Let G be any $\left(\frac{c_1N}{\log N}, c_2N, c_3\log N\right)$-Block Depth Robust DAG on $N=2^n$ nodes for some constants $c_1,c_2,c_3>0$ and let $G'=\mathsf{BRG}(G)$ be the bit reversal overlay of G. Then for any $e' < \frac{c_1N}{4\log N}$ and any pebbling $P \in \mathcal{P}^{\parallel}(G')$ we have either $\Pi^{\parallel}_{cc}(P) \ge \frac{c_1^2c_2c_3N^3}{64e'\log N}$, or $\Pi_{ss}(P,e') \ge \frac{N}{4}-o(N)$ i.e., at least $\frac{N}{4}-o(N)$ rounds j in which $|P_j| \ge e'$.*

Remark 1. Note that for some constants c_1, c_2, c_3 a randomly sampled DAG G from DRSample will be $\left(\frac{c_1N}{\log N}, c_2N, c_3\log N\right)$-Block Depth Robust with high probability [ABH17]. Thus, with high probability Corollary 3 can be applied to the bit reversal overlay $\mathsf{BRG}(G)$. Notice also that as e' decreases, the lower bound on $\Pi^{\parallel}_{cc}(P)$ increases rapidly e.g., if a pebbling does not have at least $\Omega(N)$ steps with at least $e' = \Omega\left(\sqrt{N}\right)$ pebbles on the graph, then $\Pi^{\parallel}_{cc}(P) = \tilde{\Omega}\left(N^{2.5}\right)$.

A Conjectured (tight) Lower Bound on $\Pi^{\parallel}_{cc}(\mathsf{BRG}(G))$. The idea behind the proof of Theorem 5 in the appendix is very similar to the proof of Theorem 4 — an attacker must either keep $e/2$ pebbles on the graph most of the time or the attacker must pay $\Omega(edb)$ to repebble an (e,d)-depth $\Omega(b)$ times. In fact, a slightly weaker version (worse constants) of Theorem 5 follows as a corollary of Theorem 4 since $\Pi^{\parallel}_{cc}(P) \geq e' \times \Pi_{ss}(P,e')$. Under our conjecture that DR-Sample DAGs are $(c_1 N\log\log N/\log N, c_2 N\log\log N/\log N, c_3\log N/\log\log N)$-block depth-robust graph, Theorem 5 implies that $\Pi^{\parallel}_{cc}(\mathsf{BRG}(G)) = \Omega(N^2\log\log N/\log N)$. In fact, any pebbling must either keep $\Omega(N\log\log N/\log N)$ pebbles on the graph for $\approx N/4$ steps or the pebbling has cost $\Omega(N^2\log\log N)$.

Theorem 5. *Let G_1 be an (e,d,b)-block depth-robust graph with $N=2^n$ nodes. Then $\Pi^{\parallel}_{cc}(\mathsf{BRG}(G_1)) \geq \min\left(\frac{eN}{2},\frac{edb}{32}\right)$.*

Evidence for Conjecture. In Appendix I we present evidence for our conjecture on the (block)-depth robustness of DRSample. We show that *all* known techniques for constructing depth-reducing sets *fail* to refute our conjecture. Along the way we introduce a general technique for bounding the size of a set S produced by Valiant's Lemma[6]. In this attack we partition the edges into sets $E_1,\ldots,E_n$ where E_i contains the set of all edges (u,v) such that the most significant different bit of (the binary encoding of) u and v is i. By deleting j of these edge sets (e.g., by removing one node incident to each edge) we can reduce the depth of the graph to $N/2^j$. In Corollary 7 we show that for any edge distribution function $r(v) < v$ we have

$$\mathbb{E}[|E_i|] = \frac{N}{2^i} + \sum_{j=0}^{\frac{N}{2^i}-1}\sum_{m=0}^{2^{i-1}-1} \Pr\left[2^{i-1}+m \geq v-r(v) > m\right]$$

where the value of the random variable $|E_i|$ will be tightly concentrated around its mean since for each node v the edge distribution function $r(v)$ is independent.

5.2 (Nearly) Sequential Pebblings of BRG_n have Maximum Cost

In this section, we show that for *any* constant $c \geq 1$ *any* c-parallel pebbling P of BRG_n must have cost $\Pi_{\mathsf{cc}}(P) = \Omega\left(N^2\right)$. A pebbling $P=(P_1,\ldots,P_t)$ is said to be

[6] In the appendix we also analyze the performance of Valiant's Lemma attack against Argon2i. Previously, the best known upper bound was that Valiant's Lemma yields a depth-reducing set of size $e = \mathcal{O}\left(\frac{N\log(N/d)}{\log N}\right)$ for any DAG G with constant indegree. For the specific case of Argon2i this upper bound on e was significantly larger than the upper bound —$e = \tilde{\mathcal{O}}\left(\frac{N}{d^{1/3}}\right)$ — obtained by running the layered attack [AB17, BZ17]. Nevertheless, empirical analysis of both attacks surprisingly indicated that Valiant's Lemma yields *smaller* depth-reducing sets than the layered attack for Argon2i. We show how to customize the analysis of Valiant's Lemma attack to a *specific* DAG such as DRSample or Argon2i. Our theoretical analysis of Valiant's Lemma explain these surprising empirical results. By focusing on Argon2i specifically we can show that, for a target depth d, the attacker yields a depth-reducing set of size $e = \tilde{\mathcal{O}}\left(\frac{N}{d^{1/3}}\right) \ll \mathcal{O}\left(\frac{N\log(N/d)}{\log N}\right)$, which is optimal and *matches* the performance of the layered attack [BZ17].

c-parallel if we have $|P_{i+1} \setminus P_i| \leq c$ for all round $i < t$. We remark that this rules out *any* natural extension of the greedy pebbling attack e.g., the extension from the previous section which defeated the XOR extension graph $G^{\oplus}$ was a $c = 2$-parallel pebbling. We also remark that our proof generalizes a well-known result of [LT82] which implied that $\Pi_{st}(\mathsf{BRG}_n) = \Omega\left(N^2\right)$ for *any* sequential pebbling. For parallel pebblings it is known that $\Pi_{st}(P) = \mathcal{O}\left(N^{1.5}\right)$ [AS15] though this pebbling attack requires parallelism $c = \sqrt{N}$.

It is easy to show (e.g., from Lemma 2) that starting from a configuration with $|P_i| \leq e$ pebbles on the graph, it will take $\Omega(N)$ steps to advance a pebble $\mathcal{O}(e)$ steps on the top of the graph. It follows that $\Pi_{st}(\mathsf{BRG}_n) = \Omega\left(N^2\right)$. The challenge in lower bounding $\Pi_{\mathsf{cc}}(G)$ as in Theorem 6 is that space usage might not remain constant throughout the pebbling. Once we have proved that $\Pi_{\mathsf{cc}}(G) = \Omega(N^2)$ we then note that any c-parallel pebbling P can be transformed into a sequential pebbling Q s.t. $\Pi_{\mathsf{cc}}(Q) \leq c \times \Pi_{\mathsf{cc}}(P)$ by dividing each transition $P_i \rightarrow P_{i+1}$ into c transitions to ensure that $|Q_j \setminus Q_{j-1}| \leq 1$. Thus, it follows that $\Pi_{\mathsf{cc}}(P) = \Omega\left(N^2\right)$ for any c-parallel pebbling.

Theorem 6. *Let $G = \mathsf{BRG}_n$ and $N = 2^n$. Then $\Pi_{\mathsf{cc}}(G) = \Omega\left(N^2\right)$.*

The full proof of Theorem 6 can be found in Appendix G. Briefly, we introduce a potential function Φ and then argue that, beginning with a configuration with at most $\mathcal{O}(e)$ pebbles on the graph, advancing the pebble e steps on the top of the graph either costs $\Omega(Ne)$ (i.e., we keep $\Omega(e)$ pebbles on the graph for the $\Omega(N)$ steps required to advance the pebble e steps) or increases the potential function by $\Omega(Ne)$ i.e., we *significantly* reduce the number of pebbles on the graph during the interval. Note that the cost $\Omega(Ne)$ to advance a pebble e steps on the top of the graph corresponds to an average cost of $\Omega(N)$ per node on the top of the graph. Thus, the total cost is $\Omega(N^2)$. Lemma 6, which states that it is expensive to transition from a configuration with *few* pebbles on the graph to a configuration with *many* well-spread pebbles on the graph, is a core piece of the potential function argument.

Lemma 6. *Let $G = \mathsf{BRG}_n$ for some integer $n > 0$ and $N = 2^n$. Let $P = (P_1, \ldots, P_t) \in \mathcal{P}(G)$ be some legal sequential pebbling of G. For a given b, partition $[N]$ into $\frac{N}{2^b} = 2^{n-b}$ intervals $I_x = \left[(x-1)2^b + 1, x \times 2^b\right]$, each having length 2^b, for $1 \leq x \leq 2^{n-b}$. Suppose that at time i, at most $\frac{N}{2^{b'+3}}$ of the intervals contain a pebble with $b' \geq b$ and at time j, at least $\frac{N}{2^{b'+1}}$ of the intervals contain a pebble. Then*

$$|P_i| + \ldots + |P_j| \geq \frac{N^2}{2^{b'+5}} \quad \textit{and} \quad (j - i) \geq \frac{2^{b-b'}N}{4} \ .$$

6 Empirical Analysis

We empirically analyze the quality of DRS+BRG by subjecting it to a variety of known depth-reducing pebbling attacks [AB16, AB17] as well as the "new" greedy pebbling attack. We additionally present a *new heuristic* algorithm for constructing *smaller* depth-reducing sets, which we call greedy depth reduce. We extend the pebbling attack library of Alwen et al. [ABH17] to include the greedy pebbling algorithm [BCS16] as well as our new heuristic algorithm. The source code is available on a (currently anonymous) Github repository `https://github.com/NewAttacksAndStrongerConstructions/EC2019_submission`.

6.1 Greedy Depth Reduce

We introduce a novel greedy algorithm for constructing a depth-reducing set S such that $\mathsf{depth}(G-S) \leq d_{tgt}$. Intuitively, the idea is to repeatedly find the node $v \in V(G) \setminus S$ that is incident to the largest number of paths of length d_{tgt} in $G-S$ and add v to S until $\mathsf{depth}(G-S) \leq d_{tgt}$. While we can compute $\texttt{incident}(v,d_{tgt})$, the number of length d_{tgt} paths incident to v, in polynomial time using dynamic programming, it will $\mathcal{O}(Nd_{tgt})$ time and space to fill in the array. Thus, a naive implementation would run in total time $\mathcal{O}(Nd_{tgt}e)$ since we would need to recompute the array after each iteration. This proves not to be feasible in many instances we encountered e.g. $N=2^{24}$, $d_{tgt}=2^{16}$ and $e \approx 6.4 \times 10^5$ and we would need to run the algorithm multiple times in our experiments. Thus, we adopt two key heuristics to reduce the running time. The first heuristic is to fix some parameter $d' \leq d_{tgt}$ (we used $d'=16$ whenever $d_{tgt} \geq 16$) and repeatedly delete nodes incident to the largest number of paths of length d' *until* $\mathsf{depth}(G-S) \leq d_{tgt}$. The second heuristic is to select a larger set $T \subseteq V(G) \setminus S$ of k nodes (we set $k=400 \times 2^{(18-n)/2}$ in our experiments) to delete in each round so that we can reduce the number of times we need to re-compute $\texttt{incident}(v,d_{tgt})$. We select T in a greedy fashion: repeatedly select a node v (with maximum value $\texttt{incident}(v,d')$) subject to the constraint $\mathbf{dist}(v,T) \leq r$ for some radius r (we used $r=8$ in our experiments) until $|T| \geq k$ or there are no nodes left to add — here $\mathbf{dist}(v,T)$ denotes the length of the *shortest directed path* connecting v to T in $G-S$. In our experiments we also minimized the number of times we need to run the greedy heuristic algorithm for each DAG G by *first* identifying the target depth value $d^*_{tgt}=2^j$ with $j \in [n]$ which resulted in the highest quality attack against G when using *other* algorithms (Valiant's Lemma/Layered Attack) to build the depth-reducing set S. For each DAG G we then ran our heuristic algorithm with target depths $d_{tgt}=2^j \times d^*_{tgt}$ for each $j \in \{-1,0,1\}$. We refer the reader to Appendix E and Algorithm 5 for a more *detailed* discussion of our *heuristic algorithm.*

Our analysis indicates that our greedy heuristic algorithm outperforms all prior state-of-the art algorithms for constructing depth-reducing sets including Valiant's Lemma [Val77] and the layered attack [AB16]. Given a DAG G (either Argon2i, DRSample or DRS+BRG) on $N=2^n$ nodes and a target depth d we run each algorithm algorithm finds a smaller set S such that $\mathsf{depth}(G-S) \leq d$ — see Figure 3 for an explicit comparison with $d_{tgt} \in \{8,16\}$ and $n \in [14,24]$. Our results indicate that greedy reduces the size of the depth-reducing set S by a factor of 2.5 to 5 in comparison to the *best* depth-reducing set found by *any* other approach — the improvement is strongest for DRSample.

6.2 Comparing Attack Quality

We ran each DAG G (either Argon2i, DRSample or DRS+BRG) with $N=2^n$ nodes against a battery of pebbling attacks including both depth-reducing attacks [AB16, AB17] and the greedy pebble attack. In our analysis we focused on graphs of size $N = 2^n$ with n ranging from $n \in [14,24]$, representing memory ranging from 16MB to 16GB. While DRSample provided strong resistance to pebbling results, the greedy pebbling attack yields *a very* high quality attack (for $n \geq 20$ attack quality is $\approx n$) against this graph. Similarly, as we can see in Figure 2a, Argon2i provides reasonably strong resistance to the *greedy pebble* attack, but is vulnerable to depth-reducing attacks. DRS+BRG strikes a *healthy* middle ground as it provides reasonably strong resistance to both attacks. Even if we use our *improved* heuristic algorithm to construct the depth-reducing sets as in Figure 2b, the attack quality never exceeds 6 for DRS+BRG. In summary, DRS+BRG provides the strongest resistance to *known* pebbling attacks for *practical* parameter ranges $n \in [14,24]$.

We remark that for larger instances of Argon2i Valiant's Lemma outperforms our greedy heuristic algorithm when $d^*_{tgt} \gg d'$ (DRSample and DRS+BRG typically have small d^*_{tgt} i.e., for DRSample it was always the case that $d^*_{tgt} \leq d' = 16$). We conjecture that our greedy heuristic algorithm would outperform Valiant's Lemma if we set $d' \sim d^*_{tgt}$ leading to *even higher* attack quality attacks against Argon2i though the attack would then require substantial pre-computation.

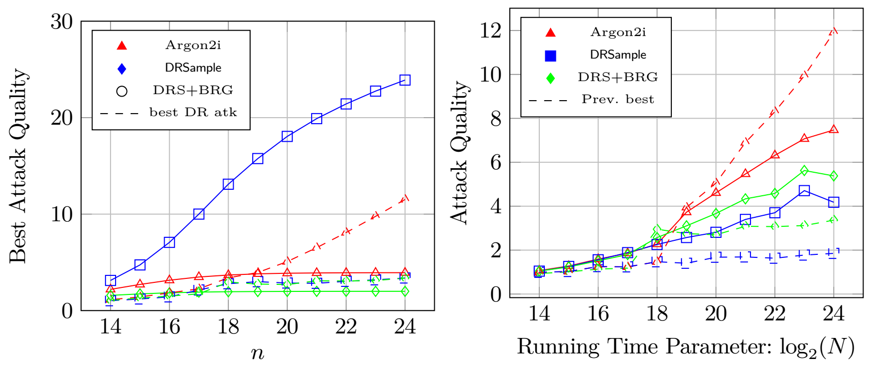


Fig. 2: Attack Quality for Greedy Pebble and Greedy Depth Reduce

7 Pebbling Reduction

Alwen and Serbinenko [AS15] previously showed that, in the parallel random oracle model, the cumulative memory complexity (cmc) of an iMHFs $f_{G,H}$ can be characterized by the black pebbling cost $\Pi^{\parallel}_{cc}(G)$ of the underlying DAG. However, their reduction assumed that the output of $f_{G,H}(x) := \mathsf{lab}_{G,H,x}(N)$ is the label of the last node N of G where labels are defined recursively using the rule $\mathsf{lab}_{G,H,x}(v) := H(v, \mathsf{lab}_{G,H,x}(v_1), \ldots, \mathsf{lab}_{G,H,x}(v_\delta))$ where $v_1, \ldots, v_\delta = \mathsf{parents}_G(v)$. To improve performance, real world implementations of iMHFs such as Argon2i, DRSample and our own implementation of BRG(DRSample) are defined using the XOR labeling rule below so that we can avoid Merkle-Damgard and work with a *faster* round

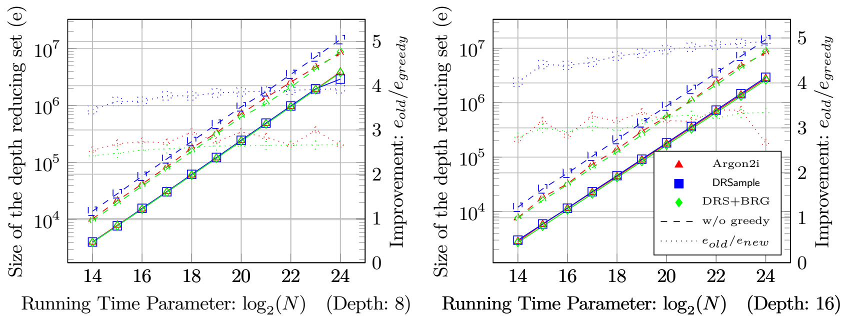


Fig. 3: Size of the depth-reducing set vs Size of the graph

function $H:\{0,1\}^w \to \{0,1\}^w$ instead of requiring $H:\{0,1\}^{(\delta+1)w} \to \{0,1\}^w$.

$$\mathsf{lab}_{G,H,x}(v) := H(\mathsf{lab}_{G,H,x}(v_1) \oplus \mathsf{lab}_{G,H,x}(v_2) \oplus \ldots \oplus \mathsf{lab}_{G,H,x}(v_\delta)),$$

where $v_1,\ldots,v_\delta$ are the parents of node v.

We prove that in the parallel random oracle model, the cumulative memory complexity of $f_{G,H}$ is still captured by $\Pi^{\|}_{cc}(G)$ when using the XOR labeling rule (* under certain restrictions discussed below that will hold for all of the iMHF constructions we consider in this paper). We leave a formal definition of cumulative memory complexity cmc to Appendix H as it is identical to [AS15]. Intuitively, one can consider the *execution trace* $\mathsf{Trace}_{\mathcal{A},R,H}(x) = \{(\sigma_i, Q_i)\}_{i=1}^t$ of an attacker $\mathcal{A}^{H(\cdot)}(x;R)$ on input value x with internal randomness R. Here, Q_i denotes the set of random oracle queries made in *parallel* during round i and σ_i denotes the state of the attacker immediately before the queries Q_i are answered. In this case, $\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}(x)) := \sum_i |\sigma_i|$ sums the memory required during each round in the parallel random oracle model. For a list of distinct inputs $X = (x_1, x_2, \ldots, x_m)$, let $f^{\times m}_{G,H}(X)$ be the ordered pair $f^{\times m}_{G,H}(X) = (f_{G,H}(x_1), f_{G,H}(x_2), \ldots, f_{G,H}(x_m))$. Then the memory cost of a $f^{\times m}_{G,H}$ is defined by

$$\mathsf{cmc}_{q,\epsilon}(f^{\times m}_{G,H}) = \min_{\mathcal{A},x} \mathbb{E}[\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}(x))],$$

where the minimum is taken over all valid inputs $X = (x_1, x_2, \ldots, x_m)$ with $x_i \neq x_j$ for $i < j$ and all algorithms $\mathcal{A}^{H(\cdot)}$ that compute $f^{\times m}_{G,H}(X)$ correctly with probability at least ϵ and make at most q queries for each computation of $f_{G,H}(x_i)$. Let $G^{\times m}$ be a DAG with mN nodes, including m sources and m sinks.

Theorem 7, our main result, states that $\mathsf{cmc}_{q,\epsilon}(f^{\times m}_{G,H}) \geq \frac{\epsilon w m}{8} \cdot \Pi^{\|}_{cc}(G)$. Thus, the cost of computing $f_{G,H}$ on m distinct inputs is at least $\Omega\left(m \times w \times \Pi^{\|}_{cc}(G)\right)$ — here, we assume that $H:\{0,1\}^w \to \{0,1\}^w$.

Theorem 7. *Let G be a DAG with N nodes, indegree $\delta \geq 2$, and* $\mathsf{parents}(u) \neq \mathsf{parents}(v)$ *for all pairs $u \neq v \in V$, and let $f_{G,H}$ be a function that follows the XOR labeling rule, with label size w. Let $\mathcal{H}$ be a family of random oracle functions with* *outputs of label length w and $H=(H_1,H_2)$, where $H_1,H_2\in\mathcal{H}$. Let m be a number of parallel instances such that $mN<2^{w/32}$, $q<2^{w/32}$ be the maximum number of queries to a random oracle, and let $\frac{\epsilon}{4}>2^{-w/2+2}>\frac{qmN+1}{2^w}+\frac{m^2N^2}{2^{w-1}-2mn}$. Then*

$$\mathsf{cmc}_{q,\epsilon}(f_{G,H}^{\times m})\geq\frac{\epsilon mw}{8}\cdot\Pi_{cc}^{\parallel}(G).$$

As in [AS15] the pebbling reduction relies on an extractor argument to show that we can find a black pebbling $P=(P_1,...,P_t)$ s.t. $|P_i|=\mathcal{O}(|\sigma_i|/w)$. The extractor takes a hint h of length $|h|=|\sigma_i|+h_2$ and then extracts ℓ distinct random oracle pairs $(x_1,H(x_1)),...,(x_\ell,H(x_\ell))$ by simulating the attacker. Here, one can show that $\ell\geq h_2/w+\Omega(|P_i|)$, which implies that $|\sigma_i|=\Omega(w|P_i|)$ since a random oracle cannot be compressed.

There are several additional challenges we must handle when using the XOR labeling rule. First, in [AS15] we effectively use an *independent* random oracle $H_v(\dot{})=H(v,\dot{})$ to compute the label of each node v — a property that does not hold for the XOR labeling rule we consider. Second, even if H is a random oracle the labeling rule we consider uses the round function $F(x,y)=H(x\oplus y)$. We remark that F is not even collision resistant e.g., $F(x,y)=F(y,x)$. Because of this, we will not be able to prove a pebbling reduction for *arbitrary* DAGs G.

In fact, one can easily find examples of DAGs G where $\mathsf{cmc}(f_{G,H})\ll\Pi_{cc}^{\parallel}(G)$ i.e., the cumulative memory complexity is much less than the cumulative pebbling cost by exploiting the fact that $\mathsf{lab}_{G,H,x}(u)=\mathsf{lab}_{G,H,x}(v)$ whenever $\mathsf{parents}(u)=\mathsf{parents}(v)$. For example, observe that if $\mathsf{parents}(N)=\{u,v\}$ and $\mathsf{parents}(u)=\mathsf{parents}(v)$ then

$$f_{G,H}(x)=\mathsf{lab}_{G,H,x}(N)=H(\mathsf{lab}_{G,H,x}(u)\oplus\mathsf{lab}_{G,H,x}(v))=H(0^w)\ ,$$

so that $f_{G,H}(x)$ becomes a constant function and any attempt to extract a pebbling from an execution trace computing $f_{G,H}$ would be a fruitless exercise!

For this reason, we only prove that $\mathsf{cmc}(f_{G,H})=\Omega\left(\Pi_{cc}^{\parallel}(G)\times w\right)$ when $G=(V=[N],E)$ satisfies the *unique parents* property i.e., for any $u<v$ we have $\mathsf{parents}(v)\neq\mathsf{parents}(u)$. We remark that any DAG which contains all edges of the form $(i,i+1)$ with $i<N$ will satisfy this property since $v-1\notin\mathsf{parents}(u)$. Thus, Argon2i, DRSample and DRSample+BRG all satisfy the unique parents property.

Extractor: We argue that, except with negligible probability, a successful execution trace must have the property that $|\sigma_i|=\Omega(w|P_i|)$ for each round of some legal pebbling P. Our extractor takes a hint which include σ_i (to simulate the attacker), the set P_i and some (short) additional information e.g., to identify the index of the next random oracle query q_v where the label for node v will appear as input. To address the challenge that the query $q_v=\mathsf{lab}_{G,H,x}(v)\oplus\mathsf{lab}_{G,H,x}(u)$ our hint will additionally include the pair $(u,\mathsf{lab}_{G,H,x}(u))$. Our extractor will attempt to extract labels for each node $v\in P_i$ as well as for a few of these extra sibling nodes u. If G satisfies the *unique parents* property then we can prove that whp our *extractor* will be successful. It follows that $|\sigma_i|=\Omega(w|P_i|)$ since the hint must be as long all of the labels that we extract.

8 An Improved Argon2 Round Function

In this section we show how a parallel attacker could reduce aAT costs by nearly an order of magnitude by computing the Argon2i round function in parallel. We then present a tweaked round function to ensure that the function must be computed *sequentially*. Empirical analysis indicates that our modifications have *negligible* impact on the running time performance of Argon2 for the honest party (sequential), while the modifications will *increase* the attackers aAT costs by nearly an order of magnitude.
Review of the Argon2 Compression Function. We begin by briefly reviewing the Argon2 round function $\mathcal{G}:\{0,1\}^{8092}\rightarrow\{0,1\}^{8092}$ which takes two 1KB blocks X and Y as input and outputs the next block $\mathcal{G}(X,Y)$. $\mathcal{G}$ builds upon a second function $\mathcal{BP}:\{0,1\}^{128}\rightarrow\{0,1\}^{128}$, which is the Blake2b round function [JAA+15]. In our analysis we treat $\mathcal{BP}$ as a blackbox. For a more detailed explanation including the specific definition of $\mathcal{BP}$, we refer the readers to the Argon2 specification [BDK16].

To begin, $\mathcal{G}$ takes the intermediate block R (which is being treated as an 8x8 array of 16 byte values $R_0,...,R_{63}$), and runs $\mathcal{BP}$ on each row to create a second intermediate stage Q. We then apply $\mathcal{BP}$ to Q column-wise to obtain one more intermediate value Z: Specifically:

$$\begin{array}{cc}
(Q_0,Q_1,...,Q_7)\leftarrow\mathcal{BP}(R_0,R_1,...,R_7) & (Z_0,Z_8,...,Z_{56})\leftarrow\mathcal{BP}(Q_0,Q_8,...,Q_{56}) \\
(Q_8,Q_9,...,Q_{15})\leftarrow\mathcal{BP}(R_8,R_9,...,R_{15}) & (Z_1,Z_9,...,Z_{57})\leftarrow\mathcal{BP}(Q_1,Q_9,...,Q_{57}) \\
... & \\
(Q_{56},Q_{57},...,Q_{63})\leftarrow\mathcal{BP}(R_{56},R_{57},...,R_{63}) & (Z_7,Z_{15},...,Z_{63})\leftarrow\mathcal{BP}(Q_7,Q_{15},...,Q_{63})
\end{array}$$

To finish, we have one last XOR, giving the result $\mathcal{G}(X,Y)=R\oplus Z$.

Parallel Attack. From the above description, it is clear that computation of the round function is trivially parallelizble. In particular, the first (resp. last) eight calls to the permutation $\mathcal{BP}$ are all independent and could easily be evaluated in parallel i.e., compute $\mathcal{BP}(R_0,R_1,\ldots,R_7),\ldots,\mathcal{BP}(R_{56},R_{57},\ldots,R_{64})$ then compute $\mathcal{BP}(Q_0,Q_8,...,Q_{56}),...,\mathcal{BP}(Q_7,Q_{15},...,Q_{63})$ in parallel. Similarly, XORing the 1KB blocks in the first $(R=X\oplus Y)$ and last $(\mathcal{G}(X,Y)=R\oplus Z)$ steps can be done in parallel. By contrast, the honest party (sequential) will only be able to evaluate one call to $\mathcal{BP}$ at a time. This implies that the depth of the attacker's circuit to compute $\mathcal{G}$ will be roughly 8 times shallower than the depth of the circuit used by the honest party, allowing the attacker to reduce running time by nearly an order of magnitude. Note that reducing running time by a factor of 8 while holding memory constant will reduce the attacker's area-time product by a factor of 8 — nearly an order of magnitude! We stress that the attack applies to *all* modes of Argon2 both data-dependent (Argon2d,Argon2id) and data-independent (Argon2i), and that the attack could potentially be combined with other pebbling attacks on Argon2i [AB16, BCS16].

Inherently Sequential Round Function. We present a small modification to the Argon2 compression function which prevents the above attack. The idea is simply to inject extra data-dependencies between calls to $\mathcal{BP}$ to ensure that an attacker must evaluate each call to $\mathcal{BP}$ sequentially just like the honest party would. In short, we require the first output byte from the $i-1^{th}$ call to $\mathcal{BP}$ to be XORed with the i^{th} input byte for the current (i^{th}) call, as shown in Figure 4.

In particular, we now compute $\mathcal{G}(X,Y)$ as:

$$\begin{array}{cc}
(Q_0,Q_1,...,Q_7)\leftarrow\mathcal{BP}(R_0,R_1,...,R_7) & (Z_0,Z_8,...,Z_{56})\leftarrow\mathcal{BP}(Q_0,Q_8,...,Q_{56}) \\
(Q_8,Q_9,...,Q_{15})\leftarrow\mathcal{BP}(R_8,R_9\oplus Q_0,...,R_{15}) & (Z_1,Z_9,...,Z_{57})\leftarrow\mathcal{BP}(Q_1,Q_9\oplus Z_0,...,Q_{57}) \\
... & ... \\
(Q_{56},Q_{57},...,Q_{63})\leftarrow\mathcal{BP}(R_{56},R_{57},...,R_{64}\oplus Q_{48}) & (Z_7,Z_{15},...,Z_{63})\leftarrow\mathcal{BP}(Q_7,Q_{15},...,Q_{63}\oplus Z_6)
\end{array}$$

where, as before, $R=X\oplus Y$ and the output is $\mathcal{G}(X,Y)=Z\oplus R$.

We welcome cryptanalysis of both this round function and the original Argon2 round function. We do stress that the primary threat to passwords is brute-force attacks (not hash inversions/collisions etc...) so increasing the attacker's cost is arguably the primary goal.

Remark 2. We remark that the implementation of $\mathcal{BP}$ in Argon2 is heavily optimized using SIMD instructions so that the function $\mathcal{BP}$ would be computed in parallel on *most* computer architectures. Thus, we avoid trying to make $\mathcal{BP}$ sequential as this would slow down *both* the attacker *and* the honest party.

Implementation and Empirical Evaluation To determine the performance impact this would have on Argon2, we modified the publicly available code to include this new compression function. The source code is available on an anonymous Github repository https://github.com/antiparallel-drsbrg-argon/Antiparallel-DRS-BRG. We then ran experiments using both the Argon2 and DRS+BRG edge distributions, and further split these groupings to include/exclude the new round function for a total of four conditions. For each condition, we evaluated 1000 instances of the memory hard function in single-pass mode with memory parameter $N=2^{20}$ blocks (i.e., 1GB=$N\times$1KB). In our experiments, we interleave instances from different conditions to ensure that any incidental interference from system processes affects each condition equally. The experiments were run on a desktop with an Intel Core 15-6600K CPU capable of running at 3.5GHz with 4 cores. After 1000 runs of each instance, we observed only small differences in runtimes, (3%) at most. The exact results can be seen in Table 1 along with 99% confidence intervals. The evidence suggests that there is no large difference between any of these versions, and that the anti-parallel modification would not cause a large increase in running time for legitimate users.

	Argon2i	DRS+BRG
Current	1405.541±1.036 ms	1445.275±1.076 ms
Anti-parallel	1405.278±1.121 ms	1445.017±0.895 ms

Table 1: Anti-parallel runtimes with 99% confidence

References

AB16. Joël Alwen and Jeremiah Blocki. Efficiently computing data-independent memory-hard functions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 241–271. Springer, Heidelberg, August 2016.

AB17. Joël Alwen and Jeremiah Blocki. Towards practical attacks on argon2i and balloon hashing. In *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on*, pages 142–157. IEEE, 2017.

ABH17. Joël Alwen, Jeremiah Blocki, and Ben Harsha. Practical graphs for optimal side-channel resistant memory-hard functions. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 17*, pages 1001–1017. ACM Press, October / November 2017.

ABMW05. Martin Abadi, Mike Burrows, Mark Manasse, and Ted Wobber. Moderately Hard, Memory-bound Functions. *ACM Trans. Internet Technol.*, 5(2):299–327, May 2005.

ABP17. Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Depth-robust graphs and their cumulative memory complexity. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 3–32. Springer, Heidelberg, April / May 2017.

ABP18. Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Sustained space complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 99–130. Springer, Heidelberg, April / May 2018.

ACP$^+$17. Joël Alwen, Binyi Chen, Krzysztof Pietrzak, Leonid Reyzin, and Stefano Tessaro. Scrypt is maximally memory-hard. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 33–62. Springer, Heidelberg, April / May 2017.

AS15. Joël Alwen and Vladimir Serbinenko. High parallel complexity graphs and memory-hard functions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 595–603. ACM Press, June 2015.

AT17. Joël Alwen and Björn Tackmann. Moderately hard functions: Definition, instantiations, and applications. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 493–526. Springer, Heidelberg, November 2017.

BCS16. Dan Boneh, Henry Corrigan-Gibbs, and Stuart E. Schechter. Balloon hashing: A memory-hard function providing provable protection against sequential attacks. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 220–248. Springer, Heidelberg, December 2016.

BDK16. Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Argon2: new generation of memory-hard functions for password hashing and other applications. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 292–302. IEEE, 2016.

Ber05. Daniel J Bernstein. Cache-timing attacks on aes. 2005.

BHZ18. Jeremiah Blocki, Benjamin Harsha, and Samson Zhou. On the economics of offline password cracking. In *2018 IEEE Symposium on Security and Privacy*, pages 853–871. IEEE Computer Society Press, May 2018.

BK15. Alex Biryukov and Dmitry Khovratovich. Tradeoff cryptanalysis of memory-hard functions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 633–657. Springer, Heidelberg, November / December 2015.

BRZ18. Jeremiah Blocki, Ling Ren, and Samson Zhou. Bandwidth-hard functions: Reductions and lower bounds. Cryptology ePrint Archive, Report 2018/221, 2018. `https://eprint.iacr.org/2018/221`.

But13. Vitalik Buterin. Ethereum, 2013.

BZ17. Jeremiah Blocki and Samson Zhou. On the depth-robustness and cumulative pebbling cost of Argon2i. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 445–465. Springer, Heidelberg, November 2017.

DGN03. Cynthia Dwork, Andrew Goldberg, and Moni Naor. On memory-bound functions for fighting spam. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 426–444. Springer, 2003.

DKW11. Stefan Dziembowski, Tomasz Kazana, and Daniel Wichs. One-time computable self-erasing functions. In *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC Proceedings*, pages 125–143, 2011.

FLW14. Christian Forler, Stefan Lucks, and Jakob Wenzel. Memory-demanding password scrambling. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 289–305. Springer, Heidelberg, December 2014.

JAA$^+$15. Marcos A. Simplicio Jr., Leonardo C. Almeida, Ewerton R. Andrade, Paulo C. F. dos Santos, and Paulo S. L. M. Barreto. Lyra2: Password hashing scheme with improved security against time-memory trade-offs. Cryptology ePrint Archive, Report 2015/136, 2015. `http://eprint.iacr.org/2015/136`.

JWK81. Hong Jia-Wei and H. T. Kung. I/o complexity: The red-blue pebble game. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, STOC '81, pages 326–333, New York, NY, USA, 1981. ACM.

KDBJ17. Dmitry Khovratovich, Daniel Dinu, Alex Biryukov, and Simon Josefsson. The memory-hard argon2 password hash and proof-of-work function. *memory*, 2017.

Lee11. Charles Lee. Litecoin, 2011.

Len81. Thomas Lengauer. Black-white pebbles and graph separation. *Acta Informatica*, 16(4):465–475, 1981.

LT82. Thomas Lengauer and Robert E. Tarjan. Asymptotically tight bounds on time-space trade-offs in a pebble game. *J. ACM*, 29(4):1087–1130, October 1982.

Per09. Colin Percival. Stronger key derivation via sequential memory-hard functions. 2009. 2009.

Pes14. Alexander Peslyak. yescrypt: password hashing scalable beyond bcrypt and scrypt, 2014.

PHC16. Password hashing competition, 2016. `https://password-hashing.net/`.

Pin14. Krisztián Pintér. Gambit – A sponge based, memory hard key derivation function. Submission to Password Hashing Competition (PHC), 2014.

Pip77. N. Pippenger. Superconcentrators. *SIAM Journal on Computing*, 6(2):298–304, 1977.

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

RD17. Ling Ren and Srinivas Devadas. Bandwidth hard functions for ASIC resistance. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 466–492. Springer, Heidelberg, November 2017.

SJAA$^+$15. Marcos A Simplício Jr, Leonardo C Almeida, Ewerton R Andrade, Paulo CF dos Santos, and Paulo SLM Barreto. Lyra2: Password hashing scheme with improved security against time-memory trade-offs. *IACR Cryptology ePrint Archive*, 2015:136, 2015.

Val77. Leslie G Valiant. Graph-theoretic arguments in low-level complexity. In *International Symposium on Mathematical Foundations of Computer Science*, pages 162–176. Springer, 1977.

Appendix

A Amortized Area-Time Complexity vs. Bandwidth Hardness

The Amortized Area-Time Complexity [AS15, ABH17] metric aims to *capture* the cost of the hardware (e.g., DRAM chips) the attacker must purchase to compute an MHF — amortized by the number of MHF instances computed over the lifetime of that hardware. By contrast, bandwidth hardness [RD17] aims to capture the *energy cost* of the electricity required to compute the MHF once. If the attacker uses an ASIC to computer the function then the *energy* expended during computation will typically be small in comparison with the *energy* expended during a cache-miss. Thus, a bandwidth hard function aims to ensure that *any* evaluation strategy incurs a large number $\Omega(n)$ of cache-misses during computation.

We argue that, in the context of password hashing, the Amortized Area-Time Complexity metric is more appropriate. In particular, our goal is to *maximize* the attackers cost per password guess given a fixed bound N on the maximum acceptable running time e.g., N might be constrained by user patience since larger values of N correspond to larger authentication delays. Suppose that we fix $N=2^{20}$ and assume that we operate over 1KB blocks as in Argon2i so that the total memory consumed is 1GB.

1. It cost approximately $0.3nJ$ per Byte transferred on an ASIC so if we optimistically assume that the attacker has to transfer 1GB of data to/from cache then the attacker will consume $0.3J$ (8.33×10^{-8} kWh) per evaluation. If we optimistically assume that the attacker pays \$0.12 per kWh of electricity[7] then it will cost $\approx \$1\times10^{-8}$ per evaluation (password guess).
2. By contrast if we suppose that the attacker is able to purchase a 1GB DRAM chip for as little as \$5 and that the DRAM chip lasts for up to 2 years under constant utilization then the attacker will be able to compute the MHF approximately 6.3×10^{7} times over the lifetime of the DRAM chip — it takes approximately 1 second to evaluate the Argon2i iMHF with $N=2^{20}$ blocks (1GB of memory) [BDK16]. Thus, if we amortize the cost of the DRAM chip over the total number of guesses then the cost per guess is approximately one order of magnitude higher $\approx 1\times10^{-7}$.
3. If we increase memory consumption to $N=2^{22}$ 1KB blocks (4GB) then the bandwidth costs increases to $\approx \$4\times10^{-8}$ (linear scaling), while the capacity costs increase to $\approx 1.26\times10^{-6}$ (quadradic scaling). To see this why this occurs we note that the attacker would need to pay $4\times\$5$ for four 1GB DRAM chips. Thus, the cost of the memory chips *increased*. However, even though the attacker purchased more DRAM the total number of instances we can evaluate in 2 years *still*

[7] The figure \$0.12 is based on the average price of electricity in the United States, but the attacker might chose to locate in a state/country where electricity is cheaper.

decreases to 1.58×10^7 because it now takes takes 4 seconds per MHF evaluation. In this case we remark that the cost per evaluation due to hardware costs (aAT complexity) exceeds the energy costs by nearly two orders of magnitude.

Furthermore, Blocki et al. [BRZ18] recently demonstrated that *any* MHF with high Amortized Area-Time complexity *must* have relatively high bandwidth cost as well. Thus, we chose to focus on Amortized Area-Time Complexity in this paper. Ideally, an MHF would have high Amortized Area-Time Complexity and high bandwidth cost. We remark that our construction DRSample+BRG has high bandwidth costs [8] in addition to having high aAT complexity.

B The DRSample(+BRG) Algorithm

Algorithm 1: Greedy Pebble.

```
Input  : DAG $G=(V,E)$ of
         size $|V|=N\in\mathbb{N}_{\geq 2}$ with edge set $E=\{(i,i+1) \mid i<N\}\cup\{(r(i),i) \mid i\leq N\}$
Output : A legal pebbling $P$ of $G$
Function GP($G=(V,E)$):
    $P_0:=\{v|\text{parents}(v)=\emptyset\}$
    $i:=1$
    while $\bigcup P_i\neq V$ do
        $\text{SafeToDiscard}:=\{v|\mathsf{gc}(v)<i\}$
        $P_i:=(P_{i-1}\cup\{v|\text{parents}(v)\in P_{i-1}\})\setminus\text{SafeToDiscard}$
        $i\leftarrow i+1$
    end
return $P$
```

Here we provide the original unmodified version of the DRSample algorithm, which is used in the construction of DRS+BRG. It can be seen in Algorithm 3. The algorithm samples a random DAG G which was proven to be $(\Omega(N/\log N),\Omega(N))$-depth-robust with high probability. The algorithm to sample a random DRS+BRG DAG G is presented in Algorithm 4.

C Missing Proofs

Reminder of Theorem 1. *Let* $r:\mathbb{N}_{>0}\rightarrow\mathbb{N}$ *be* any *function with the property that* $r(i)<i-1$ *for all* $i\in\mathbb{N}_{>0}$. *Then the DAG* $G=(V,E)$ *with* N *nodes* $V=\{1,...,N\}$ *and*

[8] Blocki et al. proved that DRSample in particular has asymptotically maximum bandwidth cost. Our iMHF construction inherits this property automatically as it contains the graph DRSample.

Algorithm 2: Greedy Pebble Extension.

```
Input   : DAG $G=(V,E)$ of size
          $|V|=N=2^n$ with edge set $E=\{(i,i+1) \mid i<N\}\cup\{(r(i),i) \mid 1<i\leq N\}$
Output  A legal pebbling $P^{\oplus}$ of $G^{\oplus}$
:
Function GPE($G=(V,E)$):
    $P:=\mathsf{GP}(G)$
    $i:=1$ for $i=1$ to $N/2$-$1$ do
        $P^{\oplus}_i=P_i$
        $P^{\oplus}_{i+N/2-1}=P_i\cup P_{i+N/2-1}$
    end
    $P^{\oplus}_N=\{N\}$
return $P$
```

Algorithm 3: An algorithm for sampling depth-robust graphs.

```
Function DRSample($n\in\mathbb{N}_{\geq 2}$):
    $V:=[v]$
    $E:=\{(1,2)\}$
    for $v\in[3,n]$ and $i\in[2]$ do                          // Populate edges
        $E:=E\cup\{(\mathsf{GetParent}(v,i),v)\}$             // Get $i^{th}$ parent
    end
    return $G:=(V,E)$.

Function GetParent($v$,$i$):
    if $i=1$ then
        $u:=i-1$
    else
        $g'\leftarrow[1,\lfloor\log_2(v)\rfloor+1]$           // Get random range size.
        $g:=\min(v,2^{g'})$                                   // Don't make edges too long.
        $r\leftarrow[\max(g/2,2),g]$                          // Get random edge length.
    end
    return $v-r$
```

edges $E=\{(i-1,i)\ :\ 1<i\leq N\}\cup\{(r(i),i)\ :\ 2<i\leq N\}$ *has* $\Pi_{st}(G)\leq\frac{N^2+2N}{2}$ *and* $\Pi_{cc}(G)\leq\frac{N^2+2N}{4}$ *and* $\mathsf{aAT}_R(G)\leq\frac{N^2+2N}{4}+RN$.

Proof of Theorem 1. We consider the greedy pebbling $P=\mathsf{GP}(G)\in\mathcal{P}(G)$. The sequential pebbling finishes in N rounds and so the computation cost incurred is

$$\sum_{i=1}^{N}|P_i\setminus P_{i-1}|R=RN\ . \qquad (2)$$

Algorithm 4: An algorithm for sampling depth-robust graphs.

```
Input   : Size $n \in \mathbb{N}_{\geq 2}$ of graph.
Output  : A DRS+BRG DAG, the first half sampled
          from DRSample and the second generated by the Bit Reversal Graph
Function DRS+BRG($n \in \mathbb{N}_{\geq 2}$):
    $V = [2^n]$
    $E = \emptyset$
    $DRS \leftarrow DRSample(N/2)$
    $E := E \cup DRS.E$
    for $v \in [N/2+1, N]$ do
        $b \leftarrow v \mod N/2$
        $E := E \cup \{reverse\_bits(b), v\}$
    end
```

Since the pebbling is sequential we observer that $|P_i| \leq i$ for each round i since we can place at most i new pebbles on the graph in i rounds. Thus,

$$\sum_{i=0}^{N/2} |P_i| \leq \sum_{i=0}^{N/2} i = \frac{N^2+2N}{8} \ . \tag{3}$$

Similarly, we also note that for each i we have

$$|P_i| \leq |\mathsf{parents}(\{N, \ldots, i+1\})| + 1 \leq N - i + 1$$

since $P_i \subseteq \{i\} \cup \mathsf{parents}(\{N, \ldots, i+1\})$. Thus, to analyze the last $N/2$ rounds, we look from the opposite direction. $|P_N| = 1, |P_{N-1}| = 2, \ldots, |P_{N-i}| \leq i+1$. Thus we have:

$$\sum_{N/2}^{N} |P_i| = \sum_{i=N/2+1}^{N} N - i + 1 = \frac{N^2+2N}{8} \ . \tag{4}$$

Combining equations 23 and 4 we have

$$\Pi_{cc}(G) \leq 2\frac{N^2+2N}{8} = \frac{N^2+2N}{4} \ ,$$

and

$$\mathsf{aAT}^{\parallel}(G) \leq 2\frac{N^2+2N}{8} + RN = \frac{N^2+2N}{4} + RN \ .$$

Finally, for the maximum space usage is $\max_i |P_i| \leq \max_i \min\{i, N-i+1\}$ which is achieved when $i = \lceil \frac{N}{2} \rceil$. Thus, $\Pi_{st}(G) \leq \frac{N^2}{2} + N$. □

Reminder of Claim 1. *Let G be a randomly sampled DRSample DAG with N nodes and let $Y_{i,j}$ be an indicator random variable for the event that $r(j) < i$ for nodes $i < j \leq N$ then we have $\mathbf{E}[Y_{i,j}] = \Pr[r(j) < i] \leq 1 - \frac{\log(j-i-1)}{\log j}$.*

Proof of Claim 1. Recall that DRSample select the parent node $r(j)$ in several stages. First, we partition all nodes $v < j-1$ into $\log j$ buckets $B_1, \ldots, B_{\log j}$ where $B_1 = [j-2-2^0, j-2), B_2 = [j-2-2^0-2^1, j-2-2^0), \ldots, B_{k+1} = [j-2-2^1-\ldots-2^k, j-2-2^1-\ldots-2^{k-1}), \ldots$ and select a bucket B_k uniformly at random from all $\log j$ buckets. Second, we select a nodes $r(j)$ uniformly at random from the bucket B_k. Note that if $i \leq j-2-2^1-\ldots-2^{k-1}$ (or equivalently, $\log_2(j-i-1) \geq k$) and we select the bucket B_k in step 1 then we cannot possibly select $r(j) < i$ in step 2. Thus, as long as $k \leq \log(j-i-1)$ we will have $r(j) \geq i$ so

$$\Pr[r(j) \geq i] > \frac{\log(j-i-1)}{j}$$

, meaning that $\Pr[r(j) < i] \leq 1 - \frac{\log(j-i-1)}{\log j}$. □

Reminder of Lemma 1. *Given a DAG G on $N = 2^n$ nodes sampled using the randomized DRSample algorithm for any $\delta > 0$ we have*

$$\Pr\left[\max_i |\chi(i)| > (1+\delta)\left(\frac{2N}{n}\right)\right] \leq \exp\left(\frac{-2\delta^2 N}{3n} + n\ln 2\right) .$$

Proof of Lemma 1. We let $Y_{i,j}$ denote and indicator random variable for the event that $r(j) \leq i$ and observe that $\chi(i) \leq \sum_{j=i+1}^{N} Y_{i,j}$, since $\chi(i)$ is upper bounded by the number of edges that "cross" over the node i. Therefore, by linearity of expectation it follows that

$$\mathbb{E}[|\chi(i)|] \leq \sum_{j=i+1}^{N} \mathbb{E}[Y_{i,j}] .$$

Given a specified node i we set $n_i = \lceil \log(N-i) \rceil \leq n$. We note that $i + 2^{n_i - 1} \leq i + N - i \leq N$, but that $i + 2^{n_i} \geq i + N - i \geq N$. By Claim 1 we have

$$\begin{aligned}
\mathbb{E}[|\chi(i)|] &\leq 1 + \sum_{j=i+2}^{N} \left(1 - \frac{\log(j-i-1)}{\log j}\right) \\
&\leq 1 + \sum_{k=1}^{n_i} \sum_{j=i+2+2^{k-1}}^{i+2^k+1} \left(1 - \frac{\log(j-i-1)}{\log j}\right) \\
&\leq 1 + \sum_{k=1}^{n_i} \sum_{j=i+2^{k-1}}^{i+2^k-1} \left(1 - \frac{k-1}{n}\right) \\
&= 1 + \sum_{k=1}^{n_i} 2^{k-1} \left(\frac{n-k+1}{n}\right) \\
&= 1 + \sum_{k=1}^{n_i} 2^{k-1} \left(\frac{n-k}{n}\right) \\
&\leq \frac{2N}{n}
\end{aligned}$$

As the expected value is the sum of independent and identically distributed variables, we can use Chernoff Bounds with $\mu = \frac{2N}{n} \geq \sum_{j=i+1}^{N} \mathbb{E}[Y_{i,j}]$ to show that for any constant δ we have

$$\Pr[|\chi(i)| > (1+\delta)\mu] < \exp\left(\frac{-2\delta^2 N}{3n}\right) .$$

Now we union bound over all $i \leq N$ to recover the original lemma statement. $\square$

Reminder of Theorem 3. *Let* $r : \mathbb{N}_{>0} \to \mathbb{N}$ *be* any *function with the property that* $r(i) < i$ *for all* $i \in \mathbb{N}_{>0}$ *and let* $G = (V,E)$ *be a graph with* N *nodes* $V = \{1,...,N\}$ *and directed edges* $E = \{(i,i+1) \mid i < N\} \cup \{r(i),i \mid 1 < i \leq N\}$*. If* $P = \mathsf{GP}(G) \in \mathcal{P}(G)$ *then the XOR-extension graph* $G^{\oplus}$ *of* G *has amortized Area-Time complexity at most*

$$\mathsf{aAT}^{\|}{}_R\left(G^{\oplus}\right) \leq \sum_{i=1}^{N/2} |P_i| + \sum_{i=1}^{N} |P_i| + \frac{3RN}{2} .$$

Proof of Theorem 3. We consider the pebbling $P^{\oplus} = \mathsf{GPE}(G)$ from Algorithm 2. For $i < N/2$ we have $P_i^{\oplus} = P_i$ where $P = \mathsf{GP}(G)$ is the greedy pebbling.

By definition we have

$$\begin{aligned}
\mathsf{aAT}^{\|}\left(P^{\oplus}\right) &= \sum_{i=1}^{N} \left|P^{\oplus}\right| + R \sum_{i=1}^{N} \left|P_i^{\oplus} \setminus P_i^{\oplus}\right| \\
&= \sum_{i=1}^{N} \left|P^{\oplus}\right| + \frac{3RN}{2} \\
&\leq \sum_{i=1}^{N/2-1} \left|P_i^{\oplus}\right| + \sum_{i=N/2}^{N-1} \left|P_i^{\oplus} \cup P_{i-N/2+1}^{\oplus}\right| + \frac{3RN}{2}
\end{aligned}$$

We first consider the graph $G' = (V,E')$ without the edges of the form $(i - \frac{N}{2}, i)$ i.e., $E' = E \setminus \{(i - \frac{N}{2}, i) | i > \frac{N}{2}\}$ and let $P' = \mathsf{GP}(G') \in \mathcal{P}(G')$ denote the greedy pebbling of this graph. As demonstrated in the proof of Theorem 1 we have

$$\mathsf{aAT}^{\|}{}_R(P') \leq \frac{N^2 + 2N}{4} + RN . \tag{5}$$

However, the pebbling P' may be illegal for the original DAG G since we ignore edges $(i - \frac{N}{2}, i)$. To fix the issue we let $G'' = (V'', E'')$ denote the subgraph of G induced by the first $N/2$ nodes i.e., $V'' = \{1,...,N/2\}$ and $E'' = \{(i,j) \in E \mid i,j \in V''\}$ and we let $P'' = \mathsf{GP}(G'')$ denote the greedy pebbling of this graph. Since the graph G'' only has $N'' = N/2$ nodes by proof of Theorem 1 we have

$$\mathsf{aAT}^{\|}{}_R(P'') \leq \frac{N^2 + 4N}{16} + \frac{RN}{2} . \tag{6}$$

We can now form a legal pebbling $P\in\mathcal{P}^{\parallel}(G)$ by combining P' and P'' as follows: $P_i = P'_i$ for $i < \frac{N}{2}$ and $P_i = P'_i \cup P''_{i+1-N/2}$ for each $i \geq \frac{N}{2}$. Thus, at time i the pebbling configuration P_i includes the node $v=i+1-N/2\in P''_{i+1-N/2}$ and the nodes $i,r(i)\in P'_i$ so it is indeed legal to place the new pebble on node $i+1$ in the next round. Combining equations 5 and 6 we obtain our primary claim

$$\begin{aligned}\mathsf{aAT}^{\parallel}{}_R(G) &\leq \mathsf{aAT}_R(P')+\mathsf{aAT}_R(P'')\\ &= \frac{N^2+2N}{4}+RN+\frac{N^2+4n}{16}+\frac{RN}{2}\\ &= \frac{5N^2+12N}{16}+\frac{3RN}{2}\ .\end{aligned}$$

Consider the graph G' which is a copy of G without the edges of the form $(i-\frac{N}{2},i)$. In addition, consider G to be broken into two layers λ_1,λ_2 which are the first and second half of the vertices being pebbled. To begin, we create an instance of G. We pebble the first half λ_1 using the greedy algorithm described in Theorem 1. When we are at round $\frac{N}{2}$ we begin a second round of pebbling λ_1 by placing a pebble at the first vertex. We then run a second instance of the pebbling of λ_1 along with the currently running pebbling of λ_2. At each step in λ_2 we are guaranteed to have a pebble on $i-\frac{N}{2}$, as it has been placed there by the second instance of pebbling λ_1. This places a pebble on all nodes in G. The complexity of this pebbling is equivalent to the pebbling of G' from the first pebbling run plus the pebbling of λ_1 that begins in round $\frac{N}{2}$. The cost of these two pebblings are bounded by Theorem 1

$$\Pi(G)\leq\Pi(G')+\Pi(\lambda_1)=\frac{N^2+2N}{4}+\frac{N^2}{16}+\frac{N}{4}=\frac{5N^2+12N}{16}. \qquad \square$$

Reminder of Lemma 5. *Suppose that a DAG G is (e,d,b)-Block Depth Robust and that parameters e' and b' satisfies the condition that $e'\left(\frac{b'}{b}\right)+e'\leq\frac{e}{2}$. Then G is (e',d,b')-Block Depth Robust, and for all S with size $|S|\leq e'$ the graph $H=G-\bigcup_{x\in S}[x-b'+1,x]$ is $\left(\frac{e}{2},d,b\right)$-Block Depth Robust.*

Proof of Lemma 5. 1. Suppose that G is not (e',d,b')-Block Depth Robust. Then there exists a set S with size $|S|\leq e'$ such that $\mathsf{depth}\left(G-\bigcup_{x\in S}[x-b'+1,x]\right)<d$. Now, let S' be the set

$$S'=\left\{v-b'i|v\in S \text{ and } 0\leq i<\left\lceil\frac{b'}{b}\right\rceil, i\in\mathbb{Z}\right\}.$$

Then we can claim that $\bigcup_{x\in S}[x-b'+1,x]\subseteq\bigcup_{x\in S'}[x-b+1,x]$ since

$$\begin{aligned} y\in\bigcup_{x\in S}[x-b'+1,x] \quad &\Longrightarrow \quad ^{\exists}u\in S \text{ with } u-b'+1\leq y\leq u\\ &\Longrightarrow \quad u-b\left\lceil\frac{b'}{b}\right\rceil+1\leq u-b'+1\leq y\leq u\\ &\Longrightarrow \quad ^{\exists}i\in\mathbb{Z} \text{ with } 0\leq i<\left\lceil\frac{b'}{b}\right\rceil \text{ with}\end{aligned}$$

$$
\begin{aligned}
&u-b(i+1)+1\le y\le u-bi\\
\implies\quad & y\in \bigcup_{x\in S'}[x-b+1,x].
\end{aligned}
$$

But since G is (e,d,b)-Block Depth Robust, we have

$$
\begin{aligned}
d&\le \mathsf{depth}\left(G-\bigcup_{x\in S'}[x-b+1,x]\right)\\
&\le \mathsf{depth}\left(G-\bigcup_{x\in S}[x-b'+1,x]\right)<d
\end{aligned}
$$

which is a contradiction.

2. By definition of S', we have

$$|S'|\le\left\lceil\frac{b'}{b}\right\rceil|S|\le\left(\frac{b'}{b}+1\right)e'\le\frac{e}{2},$$

and $\mathsf{depth}\big(G-\bigcup_{x\in S'}[x-b+1,x]\big)\le\mathsf{depth}\big(G-\bigcup_{x\in S}[x-b'+1,x]\big)$. Since we have only deleted at most $\frac{e}{2}$ intervals of length b and the assumption that G is (e,d,b)-Block Depth Robust, we can conclude that the subgraph H is itself still $\left(\frac{e}{2},d,b\right)$-Block Depth Robust. □

Reminder of Lemma 3. *Let $G_1=(V_1=[N],E)$ be a (e,d,b)-block depth-robust graph with $N=2^n$ nodes and let $G=\mathsf{BRG}(G_1)$ denote the bit-reversal extension of G_1 with $2N$ nodes $V(G)=[2N]$. For any interval $I=\left[N+i+1,N+i+1+\frac{4N}{b}\right]\subseteq[2N]$ and any $S\subseteq[1,N+i]$ with $|S|<\frac{e}{2}$, $\mathsf{ancestors}_{G-S}(I)$ is $\left(\frac{e}{2},d,b\right)$-block depth-robust.*

Proof of Lemma 3. Since, G_1 is (e,d,b)-block depth-robust and $|S|\le\frac{e}{2}$ it follows that $H=G_1-\cup_{x\in S}[x-b+1,x]$ is $\left(\frac{e}{2},d,b\right)$ block depth-robust. Thus, it is sufficient to argue that $V(H)\subseteq\mathsf{ancestors}_{G-S}(I)$.

Consider any $v\in[1,N]$, either $v\in\cup_{x\in S}[x-b+1,x]$ (i.e., $v\notin V(H)$) or $\left[v,v+\frac{b}{2}\right]$ contains no vertices of S (i.e., $v\in V(H)$). In the latter case, the graph G must contain an edge of the form (x,y) with $x\in\left[v,v+\frac{b}{2}\right]$ and $y\in I$ because Lemma 2 implies that for *any* interval $I'\subseteq[N]$ of length $\frac{b}{2}$, there is an edge from I' to I. Thus, $v\in\mathsf{ancestors}_{G-S}(I)$ since there is an S-free path from v to I via the edge (x,y). □

Reminder of Theorem 4. *Let G be any (e,d,b)-Block Depth Robust DAG on $N=2^n$ nodes, and $G'=\mathsf{BRG}(G)$ be the bit reversal overlay of G. Then for any pebbling $P\in\Pi^{\parallel}(G)$ and all $e'\le\frac{e}{4}$, we have either $\Pi^{\parallel}_{cc}(P)\ge\frac{e^2db}{64e'}$, or $\Pi_{ss}(P,e')\ge\frac{N}{4}-o(N)$ i.e., at least $\frac{N}{4}-o(N)$ rounds i in which $|P_i|\ge e'$.*

Proof of Theorem 4. Let $P\in\mathcal{P}^{\parallel}(G')$ be a legal pebbling of G' and let t_v be the first time where $v\in P_{t_v}$. Set $b':=\frac{eb}{4e'}$. Partition the nodes (i.e., the top nodes of G') into $b'/4$ intervals $I_1,\ldots,I_{b'/4}$ s.t. for each $j\le b'/4$ the interval I_j contains each of the nodes $\left[N+\frac{4N(j-1)}{b'}+1,N+\frac{4Nj}{b'}\right]$. Let $f_j=N+\frac{4N(j-1)}{b'}+1$ denote the first node in interval I_j and let $t_{j,f}:=t_{f_j}$ denote the first time where the node f_j is pebbles.

Similarly, let $m_j = N + \frac{4N(j-1)+N}{b'}$ denote a node in the middle of the interval and let $t_{j,m} := t_{m_j}$ denote the first time this node is pebbled. Notice that we must have $t_{j,m} - t_{j,f} \geq f_j - m_j \geq \frac{N}{b'}$ since in any legal pebbling it will take at least $f_j - m_j$ steps to walk a pebble from f_j to m_j.

We remark that P either (1) keeps at least $|P_j| \geq e'$ pebbles during each round $j \in [t_{j,f}, t_{j,m}]$ i.e., at least $\frac{N}{b'}$ rounds, or (2) for some $j \in [t_{j,f}, t_{j,m}]$ we drop below $|P_j| < e'$ pebbles. In the second case Lemma 4 implies that

$$\sum_{j=t_{v+N}}^{t_{v+N+\frac{32Ne'}{be}}-1} |P_j| \geq ed/2 \ .$$

We have at least $\frac{b'}{4} - 1$ disjoint intervals of length $\frac{4N}{b'}$ (possibly excluding the last interval $I_{b'/4}$).

1. If case (1) applies to at least $\frac{b'}{8} - 1$ intervals, then we have $\frac{2N}{b'} \times \left(\frac{b'}{8} - 1\right) = \frac{N}{4} - \frac{2N}{b'} = \frac{N}{4} - o(N) = \Omega(N)$ steps with at least e pebbles.
2. Otherwise, case (2) applies to at least $\frac{b'}{8}$ intervals and we must pay the cost

$$\Pi_{cc}^{\parallel}(P) \geq \left(\frac{b'}{8}\right) \times \frac{ed}{2} = \frac{e^2 db}{64e'} = \Omega\left(edb\left(\frac{e}{e'}\right)\right) \ . \qquad \square$$

D New Argon2 Round Function

Figure 4, shown here, is a visual representation of the Argon2 round function with added dependencies shown as arrows.

E Greedy Algorithm for Constructing Smaller Depth-Reducing Sets

A recent line of work establishes a close connection between the aAT completixty of an iMHF and the depth-robustness of and underlying DAG [AB16, ABP17, ABH17]. In particular, if a DAG G is highly depth-robust then the corresponding iMHF *provably* has high aAT complexity if and only if the underlying DAG G is depth-robust. Thus, an improved algorithm for constructing *smaller* depth-reducing sets S would likely yield improved pebbling attacks against the DAG G. We introduce a new greedy algorithm to construct *small* depth-reducing sets and demonstrate the advantage of our new approach empirically.

Intuition Behind Greedy Depth Reduce At a high level the intuition behind Algorithm 5 is simple. Find the node v that is incident to the maximum number of length d paths and delete it. Repeat until no length d paths are left. We use dynamic programming to identify the node v that is incident to the maximum number of length d paths.

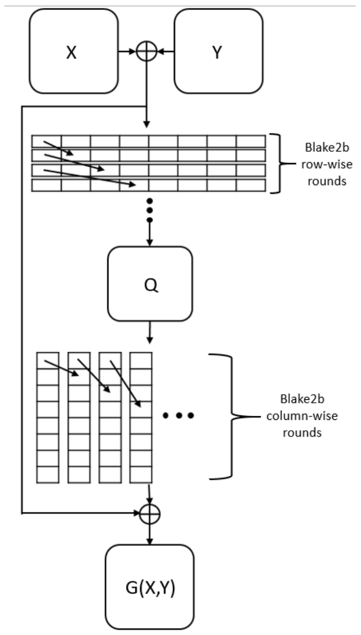


Fig. 4: Dependencies in round function calls with $m = 2^{18}$kB of memory

A dynamic programming method The subroutine CountPaths in Algorithm 5 uses dynamic programming to compute the number of length paths that are incident to each node v. To accomplish this task we fill in *two* dynamic programming tables. Intuitively, $PathsEndingAtNode[v,i]$ (resp. $PathsStartingAtNode[v,i]$) denotes the total number of directed paths of length i (here, length is the total number of directed edges on the path) that *end* (resp. *begin*) at node v. To fill in the dynamic programming table we exploit the fact that

$$PathsEndingAtNode[v,i] = \sum_{(u,v)\in E} PathsEndingAtNode[u,i-1] \tag{7}$$

Intuitively, Equation 7 follows because any path of length i that *ends* at node v has the form $P = P',v$ where P' is a directed path of length $i-1$ ending at some node u and the graph contains the directed edge $(u,v) \in E$ from u to v. Similarly, any path of length i beginning at node v has the form $P = v,P'$ where P' is a directed path of length $i-1$ beginning at node w and the graph contains the directed edge $(v,w) \in E$. Thus, we can use Equation 8 to compute the second array $PathsStartingAtNode[v,i]$

$$IncidentPathsStartingAtNode[v,i] = \sum_{(v,w)\in E} IncidentPathsStartingAtNode[w,i-1] \,. \tag{8}$$

Algorithm 5: An algorithm for a new depth-reducing attack.

```
Input   : DAG $G=([N],E)$ and target depth $d$
Output  : A depth-reducing set $S$
Function BuildDepthReducingSet($n\in\mathbb{N}_{\geq 2},d$):
    $S=\emptyset$
    $NodesInRadius:=\emptyset$
    while $depth(G-S)>d$ do
        $IncidentPathsCount:=$ CountPaths($G,S,d$)
        $B:=$ SelectRemovalNodes($G,d,r,k,NodesInRadius$)
        $S:=S\cup B$
    end
    return $S$

Input   : DAG $G=([N],E)$ and set $S$ of already deleted nodes and a depth $d$
Output  : An array IncidentPathsCount, where $IncidentPathsCount[v]$ denotes
          the number of paths of length $d$ incident to node $v$ in the graph $G-S$.
          Here, the length of a path is given by the number of edges in the path.
Function CountPaths($G$, $S$, $d$):
    for $v\in[N]$ do
        $PathsEndingAtNode[v,0]=1$
        $PathsStartingAtNode[v,0]=1$
    end
    for $i=[1,d]$ do
        for $v\in[N]$ do
            $PathsEndingAtNode[v,i]=\sum_{(u,v)\in E}PathsEndingAtNode[u,i-1]$
            $PathsStartingAtNode[v,i]=\sum_{(v,w)\in E}PathsStartingAtNode[w,i-1]$
        end
    end
    for $v\in[N]$ do
        $IncidentPathsCount[v]=$
          $\sum_{i=0}^{d}PathsEndingAtNode[v,i]\times PathsStartingAtNode[v,d-i]$
    end
    return $IncidentPathsCount$
```

Once we have computed both tables we can compute the total number of length d paths *incident* to each node v as follows

$$IncidentPathsCount[v]=\sum_{i=0}^{d}PathsEndingAtNode[v,i]\times PathsEndingAtNode[v,d-i] \tag{9}$$

Intuitively, Equation 9 follows because each path of length d that is incident to v has the form $P=P_1,P_2$ where P_1 is a length i path ending at node v, P_2 is a length j path beginning at node v and the total length of both paths is $i+j=d$.

We remark that both of the dynamic programming tables has size $\mathcal{O}(Nd)$ and that it takes time $\mathcal{O}(Nd)$ for the subroutine CountPaths to finish. If we want to reduce space usage it is possible to reduce memory consumption to $\mathcal{O}(N \log d)$ if we are willing to increase our running time to $\mathcal{O}(Nd \log d)$. To further increase performance a specific depth can be set i.e. look for the number of nodes with the highest number of incident paths of some length. We use this method in the experiments shown in Figure 2b and Figure 3. [9]

Algorithm 6: A greedy algorithm to select multiple nodes to remove.

```
Input  : DAG $G=([N],E)$, depth $d$, radius $r$, $k$ which denotes the number
         of nodes that will be removed at one time and set $NodesInRadius$
         of nodes that are in the radius of all the already deleted nodes.
Output : A set of selected nodes to remove
Function SelectRemovalNodes($G$, $IncidentPathsCount$, $d$,$r$,$k$):
    if $r=0$ then
        return $\{argmax_v(IncidentPathsCount[v])\}$
    $SelectedCandidates = $ SelectTopKnodes($IncidentPathsCount$)
    for $v \in SelectedCandidates$ do
        if $v \notin NodesInRadius$ then
            $B := B \cup \{v\}$
            UpdateNodesInRadius($G$,$v$,$r$,$NodesInRadius$)
    end
    return $B$

Function UpdateNodesInRadius($G$, $v$, $r$, $NodesInRadius$):
    while $dist(u,v) \leq r$ or $dist(v,u) \leq r$ do
        if $u \notin NodesInRadius$ then
            $NodesInRadius := NodesInRadius \cup \{u\}$
    end
Function SelectTopKnodes($IncidentPathsCount$,$k$):
    return Set of $k$ largest nodes in $IncidentPathsCount$
```

Challenges While the new greedy algorithm for constructing depth-reducing sets yielded superior results (smaller depth-reducing sets for the same target depth d), Algorithm 5 is more expensive computationally than previous approaches. In particular, the algorithm requires $\mathcal{O}(Nde)$ time in the full version to complete where e is the size of the final depth reducing set that is returned. When N,d,e are all large this approach is not always computationally feasible. Another challenge is that Algorithm 5 requires $\mathcal{O}(Nd)$ space for the dynamic programming tables, which can be problematic for larger target depths d e.g., when $N=2^{24}$ with target depth $d=2^{15}$

[9] This can be accomplished by discarding rows of the dynamic programming table from memory and recomputing them later. At each point in time we keep $\log d$ rows of the table in memory to ensure that each row will only need to be repebled.

we would require terabytes of memory to store the dynamic programming table. We remark that it is possible to reduce space usage to $\mathcal{O}(N\log d)$ while maintaining a running time of $\mathcal{O}(Nde)$ by *strategically* recomputing portions of the dynamic programming table to reduce memory usage.

The result of the CountPaths subroutine is $IncidentPathsCount[v]$ an array which, for each vertex v, counts the number of paths of length d incident to node v in graph $G-S$. Here we can employ a heuristic to speed up computation. We note that nodes close to nodes selected for removal in each round have a higher chance of being selected in the next round of CountPaths. However, when node v has a large number of paths of length d incident to it, the nodes that are close to v may also have a large number of paths of length d which share many of the same paths. Thus, deleting nodes that are close together might not contribute much to a decrease the depth of the remaining graph. To counteract this we eliminate the nodes that are within a certain radius of the deleted nodes as shown in UpdateNodesInRadius and SelectRemovalNodes. Thus we add multiple nodes per round, yet do so strategically using this heuristic to help increase depth more per node added to the depth reducing set.

E.1 Empirical Analysis

Figure 3 compares the performance of our new method with prior state-of-the art techniques for constructing depth-reducing sets. As the figure shows our algorithm is able to *consistently* construct *significantly* smaller depth-reducing sets. In particular, the size of the depth-reducing sets we obtain are typically 2–3 times smaller than prior approaches [ABH17, AB17]. Figure 2 compares the attack quality of our new method with previous best depth-reducing attack. Our new method not only reduces the size of the depth-reducing set, but also improves state-of-the art pebbling attacks.

F Candidate CMC-Optimal DAGs

In this section we present two candidate DAGs G which achieve the *best* possible lower bound on $\Pi_{cc}^{\parallel}(G)=\Omega(N^2\log\log N/\log N)$ for constant indegree graphs under *plausible* conjectures of the (block) depth-robustness of these DAGs. Our first results shows that *any* (e,d)-depth-robust DAG G with $e=d=\Omega\left(N^2\log\log N/\log N\right)$ can be used to construct a new graph $G'=\mathsf{superconc}(G)$ s.t. G' has $\Pi_{cc}^{\parallel}(G')=\Omega(N^2\log\log N/\log N)$ by overlaying a superconcentrator on top of G. We first recall the definition of a superconcentrator.

Definition 5. *A graph G with $O(N)$ vertices is a superconcentrator if there exists I,O with $|I|=|O|=N$ such that for all $S_1\subseteq I, S_2\subseteq O$ with $|S_1|=|S_2|=k$, there are k vertex disjoint paths from S_1 to S_2.*

It is well known that there exists superconcentrators with $|I|=|O|=N$, constant indegree and $\mathcal{O}(N)$ nodes total e.g. [LT82, Pip77]. We now define the overlay of a superconcentrator on a graph G_1.

Definition 6. *Let G_1 be a fixed DAG with N nodes and $G_2=(V,E)$ be a (a priori fixed) super-concentrator with N inputs $I=\{i_1,...,i_N\}\subseteq V$ and N outputs $O=\{o_1,...,o_N\}\subseteq V$. We use $G=\mathsf{superconc}(G_1)$ to denote the graph $G=(V,E\cup F_1\cup F_2)$ where $F_1=\{(o_i,o_{i+1})\ :\ 1\leq i<N\}$ and $F_2=\{(i_u,i_v)\ :(u,v)\in E(G_1)\}$.*

Theorem 8 ([ABP17], Theorem 4). *If G is (e,d)-depth robust, then $\Pi^{\parallel}_{cc}(G)>ed$.*

Lemma 7. *Let G be an (e,d)-depth robust graph. Then for all S with $|S|<\frac{e}{2}$, it follows that $\Pi^{\parallel}_{cc}(G-S)\geq\frac{e}{2}d$.*

Proof. Observe that if G is (e,d)-depth robust and $|S|<\frac{e}{2}$, then $G-S$ is $\left(\frac{e}{2},d\right)$-depth robust. Thus, $\Pi^{\parallel}_{cc}(G-S)\geq\frac{e}{2}d$ by Theorem 8. □

Conjecture 1. Let G be a graph with N nodes sampled uniformly at random from the DRSample distribution. Then with high probability, G is (e,d)-depth robust, where $e=\frac{c_1N\log\log N}{\log N}$ and $d=\frac{c_2N\log\log N}{\log N}$, for some constants $c_1,c_2>0$.

Theorem 9. *Let G_1 be an (e,d)-depth robust graph with N nodes. Then for $G=\mathsf{superconc}(G_1)$, $\Pi^{\parallel}_{cc}(G)=\Omega\left(\min\left(\frac{Ne-2e^2}{4},\frac{Nd-2ed}{4}\right)\right)$.*

Proof. Let $P=(P_1,...,P_t)\in\mathcal{P}^{\parallel}(G)$ be a legal pebbling of G and let $\{o_1,...,o_N\}$ denote the output nodes of the super-concentrator. For each node $v\in V(G)$, let t_v be the first time v is pebbled. Notice that $t_{o_i}<t_{o_{i+1}}$ since G includes the edge (o_i,o_{i+1}) for each $i<N$. Partition L into intervals of $2e$ consecutive output nodes $L_1=\{o_1,...,o_{2e}\},L_2=\{o_{2e+1},...,o_{4e}\},...$ and use let $v^i_{start}=o_{2e(i-1)+1}$ denote the first node in interval L_i, $v^i_{mid}=o_{2e(i-1)+e}$ denote the middle node, and $v^i_{last}=o_{2e(i-1)+2e}$ denote the last node in the interval L_i. We also use $L^{first}_i=\{o_{2e(i-1)+1},...,o_{2e(i-1)+e}\}$ to denote the first e nodes on the interval L_i and $L^{last}_i=\{o_{2e(i-1)+e+1},...,o_{2e(i-1)+2e}\}$ to denote the second half of the interval.

We now lower bound $\sum_{i=t_{v^i_{start}}}^{t_{v^i_{end}}}|P_i|$, the cost incurred during pebbling rounds $[t_{v^i_{start}},t_{v^i_{end}}]$. We consider two cases.

In case 1 we have $|P_j|\geq\frac{e}{2}$ for all $j\in[t_{v_{start}},t_{v_{mid}}]$. In this case we have

$$\sum_{j=t_{v^i_{start}}}^{t_{v^i_{end}}}|P_j|\geq\sum_{j=t_{v^i_{start}}}^{t_{v^i_{mid}}}|P_j|\geq\left(t_{v^i_{mid}}-t_{v^i_{start}}\right)\frac{e}{2}\geq\frac{e^2}{2}\ ,$$

where the last inequality follows from the observation that

$$\left(t_{v^i_{mid}}+1-t_{v^i_{start}}\right)\geq 2e(i-1)+e+1-2e(i-1)+1=e\ .$$

In case 2 there exists some time $j^*\in[t_{v^i_{start}},t_{v^i_{mid}}]$ such that $|P_{j^*}|<\frac{e}{2}$. Now we note that we must completely re-pebble all nodes in the set $\mathsf{ancestors}_{G-P_{j^*}}[L^{last}_i]$ before round $t_{v^i_{last}}$. Let $S=\{i_1,...,i_N\}\setminus\mathsf{ancestors}_{G-P_{j^*}}[L^{last}_i]$ be the inputs of G that are not ancestors of L_i^{last} in the graph $G - P_{j^*}$ i.e., the nodes we *don't necessarily* need to repebble. We first claim that $|S| \le |P_{j^*}| < e/2$. Suppose note, then, since G is a super-concentrator, there are $\min(|S|, e) > |P_{j^*}|$ vertex disjoint paths from S to L_i^{last} in G. It follows that there is a path from some node $v \in S$ to L_i^{last} which avoids P_{j^*}, but this implies that $v \in \mathsf{ancestors}_{G-P_{j^*}}[L_i^{last}]$. Contradiction, by construction S is disjoint from $\mathsf{ancestors}_{G-P_{j^*}}[L_i^{last}]$! It follows that $|S| \le |P_{j^*}|$.

We now let $S_1 = \{v \in V(G_1) : i_v \in S\}$ be the nodes in G_1 corresponding to the inputs $S \subseteq V(G)$. Note that $|S_1| = |S|$. We have

$$\sum_{j=t_{v^i_{start}}}^{t_{v^i_{end}}} |P_j| \ge \sum_{j=j^*}^{t_{v^i_{end}}} |P_j| \ge \Pi_{cc}^{\parallel}(G_1 - S_1) \ge \frac{ed}{2}$$

where the second to last inequality follows because we need to re-pebble every input node that is not in S and we G contains a copy of G_1 overlayed on top of the inputs. The last inequality follows from Lemma 7.

We have show that for each interval L_i we have

$$\sum_{j=t_{v^i_{start}}}^{t_{v^i_{end}}} |P_j| \ge \min\left\{\frac{e^2}{2}, \frac{ed}{2}\right\} .$$

Since there are at least $\left\lfloor \frac{n}{2e} \right\rfloor$ such intervals, the total cost of pebbling G is at least

$$\sum_{j=1}^{t} |P_j| \ge \sum_{i=1}^{\left\lfloor \frac{N}{2e} \right\rfloor} \sum_{j=t_{v^i_{start}}}^{t_{v^i_{end}}} |P_j| \ge \left(\frac{N}{2e} - 1\right) \min\left\{\frac{e^2}{2}, \frac{ed}{2}\right\} . \qquad \square$$

If Conjecture 1 holds, then we can take $e = \Omega\left(\frac{N \log\log N}{\log N}\right)$ and $d = \Omega\left(\frac{N \log\log N}{\log N}\right)$ so that $\Pi_{cc}^{\parallel}(G) = \Omega\left(\frac{N^2 \log\log N}{\log N}\right)$.

Corollary 4. *Let G_1 be a graph with N nodes sampled uniformly at random from the DRSample distribution. If Conjecture 1 holds, then $\Pi_{cc}^{\parallel}(\mathsf{superconc}(G_1)) = \Omega\left(\frac{N^2 \log\log N}{\log N}\right)$ with high probability.*

However, a superconcentrator overlay is not the most practical construction. Before we describe a more practical construction, we first set up some notation.

We now make a conjecture slightly stronger than Conjecture 1 and show it also leads to an asymptotically cmc-optimal DAG may be easier for practical implement.

Conjecture 2. Let G be a graph with N nodes sampled uniformly at random from the DRSample distribution. Then with high probability, G is (e,d,b)-block depth robust, where $e = \frac{c_1 N \log\log N}{\log N}$, $d = \frac{c_2 N \log\log N}{\log N}$, and $b = \frac{c_3 \log N}{\log\log N}$ for some constants $c_1, c_2, c_3 > 0$.

Reminder of Theorem 5. *Let G_1 be an (e,d,b)-block depth-robust graph with $N = 2^n$ nodes. Then $\Pi_{cc}^{\parallel}(\mathsf{BRG}(G_1)) \ge \min\left(\frac{eN}{2}, \frac{edb}{32}\right)$.*

Proof of Theorem 5. Suppose $G_1=(V_1=[N],E_1)$ is (e,d,b)-block depth-robust DAG with and let $G=\mathsf{BRG}(G_1)$ be a graph with $V=[2N]$. We partition the nodes $[N+1,2N]$ from the second half of G into $\left\lfloor\frac{b}{16}\right\rfloor$ disjoint intervals of length $\frac{16N}{b}$:

$$J_1=\left[N+1,N+\frac{16N}{b}\right],J_2=\left[N+\frac{16N}{b}+1,N+\frac{32N}{b}\right],...$$

and split each interval J_i into $F_i=\left[N+\frac{16iN}{b}+1,N+\frac{16iN}{b}+\frac{8N}{b}\right]$, the first half of the interval, and $L_i=\left[N+\frac{16iN}{b}+\frac{8N}{b}+1,N+\frac{16(i+1)N}{b}\right]$, the last half of the interval. Similarly, partition the first half of G into disjoint intervals of length $\frac{b}{4}$: $I_1=\left[1,\frac{b}{4}\right],I_2=\left[\frac{b}{4}+1,\frac{b}{2}\right],...$. By Lemma 2, each L_i is connected to all of the intervals $\{I_k\}$. For any fixed i let t_{start} be first time the first node in F_i is pebbled and let t_{last} be the first time the last node in F_i is pebbled. Note that either for all $j\in[t_{start},t_{last}]$, $|P_j|\geq\frac{e}{2}$, or there exists a $j\in[t_{start},t_{last}]$ with $|P_j|<\frac{e}{2}$.

In the first case, $|P_j|\geq\frac{e}{2}$ for at least $\frac{b}{8}$ steps, so that the cost of pebbling the interval is at least $\frac{8en}{b}$. In the second case, there exists a $j\in[t_{start},t_{last}]$ with $|P_j|<\frac{e}{2}$. Observe that L_i has length $\frac{8n}{b}$. Thus by Lemma 3, $\mathsf{ancestors}_{G-P_j}(L_i)$ is $\left(\frac{e}{2},d,b\right)$ block depth-robust, so by Theorem 8, the cost to repebble $\mathsf{ancestors}_{G-P_j}(L_i)$ is at least $\frac{ed}{2}$.

Hence, the cost to pebble each interval of length $\frac{16N}{b}$ is at least $\min\left(\frac{8eN}{b},\frac{ed}{2}\right)$. Accounting for each of the $\frac{b}{16}$ intervals, the total pebbling cost is at least $\min\left(\frac{eN}{2},\frac{edb}{32}\right)$. □

Corollary 5. *Let G_1 be a graph with N nodes sampled uniformly at random from the DRSample distribution. If Conjecture 2 holds, then $\Pi^{\parallel}_{cc}(\mathsf{BRG}(G_1))=\Omega\left(\frac{N^2\log\log N}{\log N}\right)$ with high probability.*

G Sequential Complexity of the Bit Reversal Graph

In this section, we show that the bit reversal graph has high sequential cumulative memory cost.

Definition 7. *We say an interval $I=[a,b]$* contains *a pebble in round j if $I\cap P_j\neq\emptyset$. We also interchangeably say that we place a pebble on I in round j if $I\cap P_j\neq\emptyset$, but $I\cap P_{j-1}=\emptyset$.*

Reminder of Lemma 6. *Let $G=\mathsf{BRG}_n$ for some integer $n>0$ and $N=2^n$. Let $P=(P_1,...,P_t)\in\mathcal{P}(G)$ be some legal sequential pebbling of G. For a given b, partition $[N]$ into $\frac{N}{2^b}=2^{n-b}$ intervals $I_x=\left[(x-1)2^b+1,x\times 2^b\right]$, each having length 2^b, for $1\leq x\leq 2^{n-b}$. Suppose that at time i, at most $\frac{N}{2^{b'+3}}$ of the intervals contain a pebble with $b'\geq b$ and at time j, at least $\frac{N}{2^{b'+1}}$ of the intervals contain a pebble. Then*

$$|P_i|+...+|P_j|\geq\frac{N^2}{2^{b'+5}}\quad and\quad (j-i)\geq\frac{2^{b-b'}N}{4}\ .$$

Proof of Lemma 6. We say an interval I_x is *far from being pebbled* at time k if both $I_{x-1} \cap P_k = \emptyset$ and $I_x \cap P_k = \emptyset$. Let z be the last pebbling round $i < z < j$ in which $|P_z| < \frac{N}{2^{b'+3}}$. Since at most $\frac{N}{2^{b'+3}}$ intervals contain pebbles at time z, there are at least $\frac{N}{2^{b'+1}} - \frac{2N}{2^{b'+3}} = \frac{N}{2^{b'+2}}$ intervals I_x that are far from being pebbled at time z, but also contain a pebble at time j. For each such interval I_x, at least 2^b sequential pebbling steps are necessary to place a pebble on I_x, since $I_{x-1} \cap P_z = \emptyset$ and each interval has length 2^b. Thus, at least $j - z \geq \frac{N}{2^{b'+2}} \cdot 2^b = \frac{2^{b-b'}N}{4}$ steps are necessary to reach round j. Hence,

$$|P_i| + \ldots + |P_j| \geq |P_{z+1}| + \ldots + |P_j| \geq \frac{2^{b-b'}N}{4} \cdot \frac{N}{2^{b+3}} = \frac{N^2}{2^{b'+5}}. \qquad \square$$

Reminder of Theorem 6. *Let $G = \mathsf{BRG}_n$ and $N = 2^n$. Then $\Pi_{\mathsf{cc}}(G) = \Omega\left(N^2\right)$.*

Proof of Theorem 6. We first let $P \in \mathcal{P}(G)$ be a legal sequential pebbling of the bit reversal graph G. Let t_i denote the first time a pebble is placed on node i. We first show that we can assume $|P_k| < \frac{N}{32}$ for all pebbling rounds k.

If $|P_k| \geq \frac{N}{32}$, then let $z < k$ be the last time at which $|P_z| < \frac{N}{64}$ and note that $k - z \geq |P_k| - |P_z| \geq \frac{N}{64}$, since at most one pebble can be added in each round in a sequential pebbling. Thus, it follows that $\mathsf{aAT}(P) \geq (k-z)\frac{N}{64} \geq \left(\frac{N}{64}\right)^2$.

In the remainder of the proof we assume that $|P_k| < \frac{N}{32}$ for all k and in particular, $|P_{t_N}| < \frac{N}{32}$. We define a sequence $b_0, b_1, \ldots, b_{i^*}$ recursively as follows. Let $b_0 > 0$ be the largest integer such that at most $\frac{N}{2^{b_0+3}}$ of the $\frac{N}{2^{b_0}}$ intervals $I_1^0, \ldots, I_{2^{n-b_0}}^0$ contain pebbles at time $y_0 = t_N$, where $I_x^0 := [(x-1) \cdot 2^{b_0} + 1, x \cdot 2^{b_0}]$ denotes the intervals that partition the nodes $[N]$ in the first layer of the bit reversal graph. Let $b_1 > 0$ be the largest integer so that at most $\frac{N}{2^{b_1+3}}$ of the $\frac{N}{2^{b_1}}$ intervals $I_1^1, \ldots, I_{2^{n-b_1}}^1$ with $I_x^1 := [(x-1) \cdot 2^{b_1} + 1, x \cdot 2^{b_1}]$ contain pebbles at time $y_1 = t_{N+4N/2^{b_0}}$, and in general, once $b_0, \ldots, b_i$ have been defined, let b_{i+1} denote the largest integer so that at most $\frac{N}{2^{b_{i+1}+3}}$ of the $\frac{N}{2^{b_{i+1}}}$ intervals $I_1^{i+1}, \ldots, I_{2^{n-b_{i+1}}}^{i+1}$ contain pebbles at time $y_{i+1} = t_{N+4N/2^{b_0}+\ldots+4N/2^{b_i}}$, where $I_x^{i+1} := [(x-1) \cdot 2^{b_{i+1}} + 1, x \cdot 2^{b_{i+1}}]$.

We halt the sequence whenever $\sum_{i=0}^{i^*} 2^{-b_i+2}N \geq 3N/4$. We now prove two useful claims. Claim 3, which follows from Lemma 6, lower bounds the initial pebbling cost. In particular, if $b_0 = O(1)$ is *any* constant then Claim 3 implies that $\mathsf{aAT}(P) = \Omega(N^2)$. Lemma 6 also implies that if *for any* b_i in our sequence we have $b_i = O(1)$ then $\mathsf{aAT}(P) = \Omega(N^2)$. Thus, in the rest of the proof we can safely assume that for each $i \leq i^*$ we have $b_i \geq 4$ so that $N2^{-b_i} < N/4$. In particular, this implies that $y_{i^*} \leq t_{2N}$ and

$$\sum_{i=0}^{i^*-1} 2^{-b_i+2}N \geq N/2 \quad \text{, and thus} \quad \sum_{i=0}^{i^*-1} 2^{-b_i+2}2^{-b_i} \geq 2^{-3} .$$

Claim 3

$$\sum_{i=1}^{y_0-1} |P_i| \geq \frac{N^2}{2^{b_0+8}} .$$

Proof of Claim 3. The claim follows by setting $b=b_0+1$ and $b'=b+2$ in Lemma 6. In particular, at time $j=y_0$ we must have pebbles on *at least* $\frac{N}{2^{b_0+4}}=\frac{N}{2^{b+3}}=\frac{N}{2^{b'+1}}$ of the intervals $I_x^0=[(x-1)2^b+1,x\times 2^b]$ — otherwise we could have selected a larger value of b_0. Similarly, at time $i=0$ we have no pebbles on the graph. Thus, by Lemma 6

$$\sum_{i=1}^{y_0-1}|P_i|\geq \frac{N^2}{2^{b'+5}}=\frac{N^2}{2^{b_0+8}} \ . \qquad \square$$

Claim 4 lower bounds the *elapsed time* between two phases. In particular, $y_{i+1}-y_i\geq\frac{3N}{4}$.

Claim 4 *For all* $0\leq i<i^*$ *we have* $y_{i+1}-y_i\geq\frac{3N}{4}$.

Proof of Claim 4. Each of the nodes in

$$I:=\left[N+4N\left(2^{-b_0}+...+2^{-b_{i-1}}\right),N+4\left(2^{-b_0}+...+2^{-b_i}\right)\right]$$

are all pebbled for the first time during the time interval $[y_i,y_{i+1}]$ — if $i=0$ then set $I:=\left[N+1,N+4N2^{-b_0}\right]$. By Lemma 2, each interval I_k^i of the form $I_k^i=\left[(k-1)2^{b_i},k\times 2^{b_i}\right]$ has an edge to I. Hence, for each interval of the form $I_k^i=\left[(k-1)2^{b_i},k\times 2^{b_i}\right]$ there must be some pebbling round $j\in[y_i,y_{i+1}]$ s.t. I_k^i contains a pebble at time j i.e., $\left|P_j\cap I_k^i\right|>0$.

We have $\frac{N}{2^{b_i}}$ intervals of the form I_k^i and, by definition of b_i, at most $\frac{N}{2^{b_i-3}}$ of these intervals I_k^i contain pebbles at time y_i. Let F denote the set of all such intervals I_k^i s.t. *both* of the intervals I_k^i and I_{k+1}^i contain no pebbles at time y_i. We note that

$$|F|\geq\frac{N}{2^{b_i}}-2\times\frac{N}{2^{b_i-3}}\geq\frac{3}{4}\times\frac{N}{2^{b_i}} \ .$$

Finally, we note that each interval I_k^i in F will require 2^{b_i} steps as it will need to be *completely repebbled* before we can place a pebble on the next interval I_{k+1}^i. Thus, we have

$$y_{i+1}-y_i\geq 2^{b_i}|F|\geq\frac{3N}{4} \ . \qquad \square$$

We now define a potential function Φ to help analyze the amortized cost of pebbling during each interval $[y_i,y_{i+1}]$. In particular, we initially set $\Phi(y_0)=\frac{N^2}{2^{8+b_0}}$ and then prove that for each such interval we have

$$\Phi(y_i)-\Phi(y_{i+1})+\sum_{t=y_i}^{y_{i+1}-1}|P_t|\geq\frac{N^2}{2^{b_i+10}} \ .$$

It follows that

$$-\Delta\Phi+\sum_{i=0}^{i^*-1}\sum_{t=y_i}^{y_{i+1}-1}|P_t| = \sum_{i=0}^{i^*-1}\left(\Phi(y_{i+1})-\Phi(y_i)+\sum_{t=y_i}^{y_{i+1}-1}|P_t|\right)$$

$$\geq \sum_{i=0}^{i^*-1} \frac{N^2}{2^{10+b_i}}$$
$$= \frac{N^2}{2^9} \sum_{i=0}^{i^*-1} 2^{-b_i}$$
$$\geq \frac{N^2}{2^{13}} \ .$$

We then separately prove that $\Delta\Phi \geq -\Phi(y_o)$ which means that

$$\sum_{t=0}^{y_0-1} |P_i| + \sum_i \sum_{t=y_i}^{y_{i+1}-1} |P_t| \geq \sum_{t=0}^{y_0-1} |P_i| + \frac{N^2}{2^{13}} - \Phi(y_o) \geq \frac{N^2}{2^{12}} \ ,$$

where the last inequality follows because Claim 3 implies $\sum_{t=0}^{y_0-1} |P_i| \geq \Phi(y_0)$.

We now consider several cases based on the difference $b_{i+1} - b_i$:

Case 1 (small increase): $b_i \leq b_{i+1} \leq b_i + 2$. We consider two sub-cases: either in every round $z \in [y_i, y_{i+1})$ we have $|P_z| > \frac{N}{2^{b_i+8}}$, or at some point $z \in [y_i, y_{i+1})$ we have $|P_z| \leq \frac{N}{2^{b_i+8}}$. In the first sub-case, the cost of pebbling during rounds $[y_i, y_{i+1})$ is at least

$$\sum_{t=y_i}^{y_{i+1}-1} |P_t| \geq (y_{i+1} - y_i) \frac{N}{2^{b_i+8}} \geq \frac{3N}{4} \cdot \frac{N}{2^{b_i+8}} \geq \frac{N^2}{2^{b_i+9}} \ .$$

To obtain a lower bound in the second sub-case we rely on the observation that at time y_{i+1} there must be pebbles on at least $\frac{1}{8}$-fraction of the intervals of length $2^{b_{i+1}+1}$. Now we set $b = b_{i+1} + 1$ and $b' = b + 2 \leq b_i + 5$ in Lemma 6 to obtain the lower bound

$$\sum_{t=y_i}^{y_{i+1}-1} |P_t| \geq \frac{N^2}{2^{b'+5}} \geq \frac{N^2}{2^{b_i+10}} \ .$$

Note that we can apply Lemma 6 since we end with pebbles on at least $N/2^{b_{i+1}+4} = N/2^{b'+1}$ of the intervals of length 2^b at time y_{i+1} (otherwise, we would have selected $b_{i+1} = b$) and start with pebbles on at most $|P_z| \leq \frac{N}{2^{b_i+8}} \leq \frac{N}{2^{b'+3}}$ such intervals.

Thus, in this sub-case we have cost at least $\sum_{t=y_i}^{y_{i+1}-1} |P_t| \geq \frac{N^2}{2^{b_i+10}}$. In both sub-cases we set $\Phi(y_{i+1}) := \Phi(y_i)$ so that the potential function does not change i.e., $\Phi(y_i) - \Phi(y_{i+1}) = 0$.

Case 2 (decrease): $b_{i+1} = b_i - k$ *with* $k \geq 1$. In this case the pebbling costs will be quite large which will allow us to "recharge" the potential function using excess costs. We again consider two subcases. In subcase one we assume that $|P_z| \geq \frac{N}{2^{b_{i+1}+6}}$ for all $z \in [y_i, y_{i+1})$ which immediately implies that

$$\sum_{t=y_i}^{y_{i+1}-1} |P_t| \geq \frac{3N}{4} \cdot \frac{N}{2^{b_{i+1}+6}} \geq \frac{3N^2}{2^{b_i-k+8}} \geq \frac{N^2}{2^{b_i-k+7}} \ .$$

In the second case we let $z\in[y_i,y_{i+1})$ be the latest time for which $|P_z|\leq\frac{N}{2^{b_{i+1}+6}}$. By setting $b=b_{i+1}+1$ and $b'=b+2=b_{i+1}+3$ in Lemma 6 we obtain the lower bound

$$\sum_{t=y_i}^{y_{i+1}-1}|P_t|\geq\frac{N^2}{2^{b'+5}}\geq\frac{N^2}{2^{b_{i+1}+8}}=\frac{N^2}{2^{b_i-k+8}}\ .$$

Note that we can apply Lemma 6 since at most $|P_z|\leq\frac{N}{2^{b_{i+1}+6}}=\frac{N}{2^{b'+3}}$ of the intervals of size 2^b are pebbled at time z and at least $\frac{N}{2^{b+3}}=\frac{N}{2^{b'+1}}$ of these intervals must be pebbled by time y_{i+1} — otherwise we would have selected a larger b_{i+1}. In both sub-cases we have

$$\sum_{t=y_i}^{y_{i+1}-1}|P_t|\geq\frac{2^{k+2}N^2}{2^{b_i+10}}\ .$$

We will define $\Phi(y_{i+1}):=\Phi(y_i)+\left(2^{k+2}-1\right)\frac{N^2}{2^{b_i+10}}$. Notice that while the potential does increase significantly in this case we still have

$$\Phi(y_i)-\Phi(y_{i+1})+\sum_{t=y_i}^{y_{i+1}-1}|P_t|\geq\frac{N^2}{2^{b_i+10}}\ .$$

Case 3 (large increase): $b_{i+1}>b_i+2$. In this case we will simply define $\Phi(y_{i+1})=\Phi(y_i)-\frac{N^2}{2^{b_i+10}}$ so that trivially we have

$$\Phi(y_i)-\Phi(y_{i+1})+\sum_{t=y_i}^{y_{i+1}-1}|P_t|\geq\frac{N^2}{2^{b_i+10}}\ .$$

In particular, we don't attempt to lower bound the pebbling costs in this case and instead reduce the potential function.

In the final case the potential decreases, but, as we later prove in Lemma 8, we maintain the invariant that $\Phi(y_i)\geq 0$ which means that for any $i>0$ we have

$$\Phi(y_0)-\Phi(y_i)\leq\Phi(y_0)\leq\sum_{t=1}^{y_0-1}|P_t|\ .$$

It remains to prove that the potential function never becomes negative. Lemma 8 shows that a stronger invariant holds.

Lemma 8. *For each i we have $\Phi(y_i)\geq\frac{N^2}{2^{b_i+8}}$.*

Proof of Lemma 8. Clearly, when $i=0$ we have $\Phi(y_0)\geq\frac{N^2}{2^{b_i+8}}$ by definition. Now suppose that the invariant holds at time i and consider $\Phi(y_{i+1})$. There are three cases. In the first case (small increase) we have $b_i\leq b_{i+1}\leq b_i+2$. In this case we defined $\Phi(y_{i+1})=\Phi(y_i)$. It follows that

$$\Phi(y_{i+1})=\Phi(y_i)\geq\frac{N^2}{2^{b_i+8}}\geq\frac{N^2}{2^{b_{i+1}+8}}\ .$$

In the second case (decrease) we have $b_{i+1} = b_i - k$ with $k > 0$ where we had set

$$\begin{aligned}\Phi(y_{i+1}) &= \Phi(y_i) + \left(2^{k+2} - 1\right)\frac{N^2}{2^{b_i+10}} \\ &= \Phi(y_i) + \left(2^{k+2} - 1\right)\frac{N^2}{2^{b_{i+1}+k+10}} \\ &\geq \frac{N^2}{2^{b_{i+1}+8}} + \left(\Phi(y_i) - \frac{N^2}{2^{b_i+10}}\right) \\ &\geq \frac{N^2}{2^{b_{i+1}+8}} \ .\end{aligned}$$

In the third case (large increase) we have $b_{i+1} > b_i + 2$ and we defined $\Phi(y_{i+1}) = \Phi(y_i) - \frac{N^2}{2^{b_i+10}}$. Thus,

$$\Phi(y_{i+1}) \geq \frac{N^2}{2^{b_i+8}} - \frac{N^2}{2^{b_i+10}} \geq \frac{N^2}{2^{b_i+9}} \geq \frac{N^2}{2^{b_{i+1}+6}} \ . \qquad \square$$

This completes the proof of Theorem 6. □

Definition 8. *A pebbling $P = P_1, \ldots$ is* c-parallel *if $|P_{i+1} \setminus P_i| \leq c$ for all i. We define $\Pi^c_{\mathsf{cc}}(G)$ to be the cumulative pebbling cost by any c-parallel pebbling.*

Note that any c-parallel pebbling P places at most c new pebbles in each step, so that $|P_{i+1} \setminus P_i| \leq c$ for all i. Thus, each step P_i in a c-parallel pebbling can be emulated by a sequence of c steps $Q_{ci+1}, \ldots, Q_{c(i+1)}$ in a sequential pebbling where $Q_{ci} = P_i$ and $Q_{c(i+1)} = P_{i+1}$ so that $|Q_{j+1} \setminus Q_j| \leq 1$ for all j and $|Q_{ci+j}| \leq |P_{i+1}|$ for all $1 \leq j \leq c$. Thus, for any *c-parallel* pebbling P there exists a sequential pebbling Q with $\Pi_{\mathsf{cc}}(Q) \leq c \times \Pi_{\mathsf{cc}}(P)$.

Remark 3. For any graph G and any integer $c \geq 1$,

$$\Pi_{\mathsf{cc}}(G) \leq c \times \Pi^c_{\mathsf{cc}}(G) \ .$$

Corollary 6. *Let $G = \mathsf{BRG}_n$ and $N = 2^n$ for some integer $n > 0$. In particular, for any constant $c \geq 1$ we have $\Pi^c_{\mathsf{cc}}(G) = \Omega(\Pi_{\mathsf{cc}}(G)) = \Omega\left(N^2\right)$.*

H Pebbling Reduction for XOR Labeling Rule

Alwen and Serbinenko [AS15] previously showed that, in the parallel random oracle model, cumulative memory complexity of an iMHFs $f_{G,H}$ can be characterized by the black pebbling cost $\Pi^{\parallel}_{cc}(G)$ of the underlying DAG. However, their reduction assumed that the output of $f_{G,H}$ is $f_{G,H}(x) := \mathsf{lab}_{G,H,x}(N)$ is the label of the last node N of G where labels are defined recursively using the rule $\mathsf{lab}_{G,H,x}(v) = H(v, \mathsf{lab}_{G,H,x}(v_1), \ldots, \mathsf{lab}_{G,H,x}(v_\delta))$ where $v_1, \ldots, v_\delta = \mathsf{parents}_G(v)$. To improve performance real world implementations of iMHFs such as Argon2i, DRSample and our own construction BRG(DRSample) are defined using the XOR labeling rule

$$\mathsf{lab}_{G,H,x}(v) = H\left(\bigoplus_{i=1}^{\delta} \mathsf{lab}_{G,H,x}(v_i)\right)$$

$$= H(\mathsf{lab}_{G,H,x}(v_1) \oplus \mathsf{lab}_{G,H,x}(v_2) \oplus \ldots \mathsf{lab}_{G,H,x}(v_\delta)),$$

where $v_1,\ldots,v_\delta$ are the parents of node v.

In this section we prove that, in the parallel random oracle model, the cumulative memory complexity of $f_{G,H}$ is still captured by $\Pi^{\parallel}_{cc}(G)$ when using the XOR labeling rule. There are several additional challenges we must handle when using the XOR labeling rule. First, in [AS15] we effectively use an *independent* random oracle $H_v(x) = H(v,x)$ to compute the label of each node v — a property that does not hold for XOR labels. Second, even if H is a random oracle the hash function $F(x,y) = H(x \oplus y)$ is used to generate the labels.

We remark that G is not even collision resistant e.g., $F(x,y) = F(y,x)$. Because of this we will not be able to prove a pebbling reduction for *arbitrary* DAGs G. In fact, one can easily find examples of DAGs G where $\mathsf{cmc}(f_{G,H}) \ll \Pi^{\parallel}_{cc}(G)$ i.e., the cumulative memory complexity is much less than the cumulative pebbling cost by exploiting the fact that $\mathsf{lab}_{G,H,x}(u) = \mathsf{lab}_{G,H,x}(v)$ whenever $\mathsf{parents}(u) = \mathsf{parents}(v)$. In such a case if $\mathsf{parents}(N) = \{u,v\}$ we would have

$$f_{G,H}(x) = \mathsf{lab}_{G,H,x}(N) = H(\mathsf{lab}_{G,H,x}(u) \oplus \mathsf{lab}_{G,H,x}(v)) = H(0^w) \ ,$$

so that $f_{G,H}(x)$ becomes a constant function!

For this reason we only prove that $\mathsf{cmc}(f_{G,H}) = \Omega\left(\Pi^{\parallel}_{cc}(G) \times w\right)$ when $G = (V = [N],E)$ contains all of the edges of the form $(i,i+1)$ with $i < N$. This *ensures* that for any $u < v$ we have $\mathsf{parents}(v) \neq \mathsf{parents}(u)$ since $v-1 \notin \mathsf{parents}(u)$. Fortunately, this happens to be true of all of the iMHFs we consider. We can use this to argue that it is not possible for an attacker to find a pair $(x,v) \neq (x',v')$ such that $\mathsf{lab}_{G,H,x'}(v') = \mathsf{lab}_{G,H,x}(v)$.

Definition 9 (XOR Labeling). *Suppose $G = (V,E)$ is a directed acyclic graph with indegree δ and a single sink node N. Given a family of random oracle functions $H = \{H_1,H_2\}$ with $H_1,H_2 : \Sigma^* \to \Sigma^\ell$ over an alphabet Σ, we define the* prelabel *of a node $v \in [N]$ as $\mathsf{prelab}_{G,H,x}(i) : [N] \to \Sigma^U$. We omit the subscripts G,H when the dependency on the graph G and hash function H is clear. In particular, given an input x the prelabel of node v is defined by*

$$\mathsf{prelab}_{G,H,x}(v) = \begin{cases} H_1(x), & \mathsf{indeg}(v) = 0 \\ \mathsf{lab}_{G,H,x}(v-1), & \mathsf{indeg}(v) = 1 \\ \bigoplus_{i=1}^{\delta} \mathsf{lab}_{G,H,x}(v_i), & \mathsf{indeg}(v) > 1. \end{cases}$$

where $v_1,\ldots,v_\delta$ are the parents of node v. The (H,x) XOR labeling *of G is then defined recursively by*

$$\mathsf{lab}_{G,H,x}(v) = \begin{cases} H_2(H_1(x)), & \mathsf{indeg}(v) = 0 \\ H_2(\mathsf{lab}_{G,H,x}(v-1)), & \mathsf{indeg}(v) = 1 \\ H_2\left(\bigoplus_{i=1}^{\delta} \mathsf{lab}_{G,H,x}(v_i)\right), & \mathsf{indeg}(v) > 1. \end{cases} .$$

We then define $f_{G,H}(x) = \mathsf{lab}_{G,H,x}(N)$.

Lemma 9 states that, except with negligible probability, all nodes will have distinct labels and pre-labels as long as the original DAG satisfies the property that $\mathsf{parents}(u) \neq \mathsf{parents}(v)$ for all pairs $u \neq v \in V$. The assumption that $\mathsf{parents}(u) \neq \mathsf{parents}(v)$ for all $u \neq v \in V$ is necessary so that each node in G has a unique prelabel with high probability. Otherwise, we cannot accurately view the label of each node as an independent strings. See Figure 5 for an example of a DAG whose prelabels are not necessarily different.

Lemma 9. *Suppose $G=(V,E)$ is a DAG with N nodes, such that $\mathsf{parents}(u) \neq \mathsf{parents}(v)$ for all pairs $u \neq v \in V$. Let $\mathcal{H}=(H_1,H_2)$ be a family of random oracle functions with outputs of label length w. Then*

$$\Pr_{H \in \mathcal{H}}\left[\texttt{COLLISION}] \leq \frac{N^2}{2^{w-1}-2N}\right] .$$

where $\texttt{COLLISION}$ *denotes the event* $\exists a \neq b \in V, \mathsf{lab}_{G,H,x}(a) = \mathsf{lab}_{G,H,x}(b) \vee \mathsf{prelab}_{G,H,x}(a) = \mathsf{prelab}_{G,H,x}(b)$.

Proof of Lemma 9. Suppose without loss of generality that the nodes $1,\dots,N$ are in topological order and let H be chosen uniformly at random from $\mathcal{H}$. Let $\mathcal{L}_m$ be the event that $\mathsf{lab}_{G,H,x}(a) = \mathsf{lab}_{G,H,x}(b)$ for some $a \neq b$ with $a,b \leq m$. Let $\mathcal{P}_m$ be the event that $\mathsf{prelab}_{G,H,x}(a) = \mathsf{prelab}_{G,H,x}(b)$ for some $a \neq b$ with $a,b \leq m$.

Consider an induction on m after observing that $\mathbf{Pr}[\mathcal{L}_1] = \mathbf{Pr}[\mathcal{P}_1] = 0$. For any $v \in V$, let $r_1(v),\dots,r_{\delta_v}(v)$ denote the parents of v with $r_1(v) < r_2(v) < \dots < r_{\delta_v}$. For fixed $i \in V$ with $i < m+1$, $\mathsf{prelab}_{G,H,x}(i) = \mathsf{prelab}_{G,H,x}(m+1)$ if and only if

$$\bigoplus_{j=1}^{\delta_i} \mathsf{lab}_{G,H,x}(r_j(i)) = \bigoplus_{j=1}^{\delta_i} \mathsf{lab}_{G,H,x}(r_j(m+1)) .$$

Conditioned on $\mathcal{L}_m$, the probability that $\mathsf{prelab}_{G,H,x}(i) = \mathsf{prelab}_{G,H,x}(m+1)$ for a fixed $i < m+1$ is at most $\frac{1}{2^w - m}$ since $\mathsf{lab}_{G,H,x}(r_1(i)) = H\left(\mathsf{prelab}_{G,H,x}(r_1(i))\right)$ is essentially a uniformly random w bit string — the condition $\mathcal{L}_m$ that the first m pre-labels are pairwise distinct rules out *at most* m possible values of $\mathsf{lab}_{G,H,x}(r_1(i))$. Taking a union bound over all choices of $i \leq m$ we have

$$\phi \mathbf{Pr}[\neg \mathcal{P}_{m+1} | \neg \mathcal{L}_m \wedge \neg \mathcal{P}_m] \leq \frac{m}{2^w - m} \leq \frac{N}{2^w - N} .$$

Conditioned on $\neg \mathcal{P}_{m+1}$, it follows that for a fixed $i < m+1$, $\mathsf{prelab}_{G,H,x}(i) \neq \mathsf{prelab}_{G,H,x}(m+1)$. Hence, the probability that $\mathsf{lab}_{G,H,x}(i) = \mathsf{lab}_{G,H,x}(m+1)$ for a fixed $i < m+1$ is at most $\frac{1}{2^w}$ since we can view $\mathsf{lab}_{G,H,x}(m+1)$ as a uniformly random w bit string. Taking a union bound over all choices of $i \leq m$,

$$\mathbf{Pr}[\neg \mathcal{L}_{m+1} | \neg \mathcal{L}_m \wedge \neg \mathcal{P}_{m+1}] \leq \frac{m}{2^w} \leq \frac{N}{2^w - N} .$$

Thus it follows that $\mathbf{Pr}[\neg \mathcal{P}_N \wedge \neg \mathcal{L}_N] \leq \frac{2N^2}{2^w - N} = \frac{N^2}{2^{w-1} - 2N}$. □

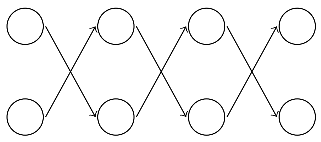


Fig. 5: An example of a DAG without independent prelabels.

Technical Note: Argon2i actually defines the first *two blocks* according to a special rule. Essentially, the labels of the first two nodes will be $H_1(x,0)$ and the label of the second node will be $H_1(x,1)$, while the rest of the nodes will be defined as above. We remark that Lemma 9 will still hold with this modification since the prelabels for the first two nodes are *guaranteed* to be distinct.

H.1 Memory in the Parallel Random Oracle Model

Before describing our reduction, we formally recall the definition of cumulative memory complexity in the pROM model. Let the state of an algorithm $\mathcal{A}^{H(\cdot)}$ at time i to be σ_i, which contains the contents of the memory. Let $\mathcal{A}^{H(\cdot)}$ be a pROM attacker $\mathcal{A}^{H(\cdot)}$ who is given oracle access to a random oracle $H:\{0,1\}^* \rightarrow \{0,1\}^w$. An execution of $\mathcal{A}^{H(\cdot)}$ on input x proceeds in rounds as follows. Initially, the state at time 0 is σ_0, which encodes the initial input x. At the beginning of round i the attacker is given the initial state σ_{i-1} as well as the answers A_{i-1} to any random oracle queries that were asked at the end of the last round. The algorithm $\mathcal{A}^{H(\cdot)}$ may then perform arbitrary computation and choose to update the memory, outputting a new state σ_i, along with a batch of queries $Q_i = \{q_1^i, q_2^i, ..., q_i^{k_i}\}$.

Execution Trace. The execution trace of the algorithm $\mathcal{A}^{H(\cdot)}$ is defined by the sequence of memory states and queries made to the random oracle H. Formally, the execution trace is $\mathsf{Trace}_{\mathcal{A},R,H}(x) = \{(\sigma_i, Q_i)\}_{i=1}^t$, where the trace $\mathsf{Trace}_{\mathcal{A},R,H}(x)$ is dependent on the algorithm $\mathcal{A}^{H(\cdot)}$, random oracle H, internal randomness R, and input value x. Then the cumulative memory cost of the execution trace is

$$\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}(x)) = \sum_i |\sigma_i| \ .$$

Note that the attacker is not charged for space used for computation between queries. This is justified since we will *lower bound* the cumulative memory cost. We say that an execution trace $\mathsf{Trace}_{\mathcal{A},R,H}(x)$ is *successful* if the final output was correct i.e., $f_{G,H}(x) = \mathsf{lab}_{G,H,x}(N)$.

H.2 Ex-Post-Facto Pebbling

Let $f_{G,H}$ be a function with random oracle H and underlying directed acyclic graph G. We show that computation of $f_{G,H}$ yields a legal black pebbling with high probability. We first define an *ex-post-facto* pebbling for any computation of $f_{G,H}$ using the following terminology.

Definition 10. *We say that random oracle query q* targets *node v if $q = \mathsf{prelab}(v)$. We say that the node v is an* input *for query q if $q = \mathsf{prelab}(w)$ for some node w such that $v \in \mathsf{parents}(w)$. We use the predicate $\mathsf{targets}(q,v)$ (resp. $\mathsf{input}(q,v)$) to indicate that query q targets node v (resp. node v is an input for query q).*

We remark that multiple nodes v could be the *target* of a query q if two pre-labels collide i.e., $\mathsf{prelab}_{G,H,x}(v) = \mathsf{prelab}_{G,H,x}(u)$, though Lemma 9 implies that that this only happens with negligible probability when all nodes in G have a distinct set of parents.

We use $\mathcal{A}^{H(\cdot)}$ to extract a legal $P = (P_1,...,P_t) \in \mathcal{P}^{\|}(G)$ from a *successful* execution trace. Given a *successful* execution trace $\mathsf{Trace}_{\mathcal{A},R,H}(x) = \{(\sigma_i,Q_i)\}_{i=1}^t$ we let $T_i = \{v : \exists q \in Q_i \text{ s.t. } \mathsf{targets}(q,v)\}$ be the set of targeted nodes in round i and we let $I_i = \{v : \exists q \in Q_i \text{ s.t. } \mathsf{input}(q,v)\}$ be the set of required input nodes in round i. Given a node v and a round i we define $\mathtt{NextTargetRound}(v,i)$ (resp. $\mathtt{NextRequiredRound}(v,i)$) to be the earliest round $j \geq i$ s.t. $v \in T_j$ (resp. $v \in I_j$). We define $R_i = \{v : \mathtt{NextRequiredRound}(v,i) \leq \mathtt{NextTargetRound}(v,i)\}$ to denote the set of all nodes which are required as inputs for random oracle queries before they are observed as outputs. Now to obtain the corresponding pebbling $\mathtt{BlackPebble}^H(\mathsf{Trace}_{\mathcal{A},R,H}(x)) = (P_0,...,P_t)$ where $P_0 = \emptyset$ and $P_i = R_i \cap (T_i \cup P_{i-1})$ for each round $0 < i \leq t$.

Intuitively, at each time j, P_j contains all nodes v whose label will appear as input to a future random oracle query *before* the label appears as the output of a random oracle query.

Definition 11. *Given an execution trace $\mathsf{Trace}_{\mathcal{A},R,H}(x) = \{(\sigma_i,Q_i)\}_{i=1}^t$ we say that a round i query $q \in Q_i$ is* lucky *if for some nodes x and $v \in \mathsf{parents}(x)$ we have :*

- $\mathtt{targets}(q,x)$ *(query q targets node x), and*
- *For all prior queries $q' \in \bigcup_{j=1}^{i-1} Q_j$ we have $\mathtt{input}(q',v) = 0$ i.e., v has not been the target of any prior query.*

We say that the output *is* lucky *if the execution trace is successful, but the final node N was never the target of a query i.e., $\mathtt{targets}(q,N) = 0$ for all $q \in \bigcup_{j=1}^t Q_j$.*

Observe that if there are no lucky guesses, then $\mathtt{BlackPebble}^H(\mathsf{Trace}_{\mathcal{A},R,H}(x)) = P_0,...,P_t$ corresponds to a legal black pebbling.

Theorem 10. *Suppose $G = (V,E)$ is a DAG with N nodes, such that $\mathsf{parents}(u) \neq \mathsf{parents}(v)$ for all pairs $u \neq v \in V$. and that $\mathcal{A}$ computes $f_{G,H}$ correctly with probability at least ϵ while making* at most *q queries to the random oracle. Then with probability at least $\epsilon - \frac{qN+1}{2^w} - \frac{N^2}{2^{w-1}-2N}$ the ex-post-facto pebbling extracted from $\mathcal{A}$ is a legal pebbling* and *the event $\mathtt{COLLISION}$ from Lemma 9 does not occur.*

Proof. Fix an execution trace $\mathsf{Trace}_{\mathcal{A},R,H}(T) = \{(\sigma_i,Q_i)\}_{i=1}^t$. By Lemma 9 we have that the probability of the event $\mathtt{COLLISION}$ that two nodes have the same label or pre-label is at most $\frac{N^2}{2^{w-1}-2N}$. We now upper bound the probability that there is a lucky query conditioning on the event that $\mathtt{COLLISION}$ does not occur. Let $NQ_i = \{v : \mathsf{input}(q',v) = 0 \ \ \forall q' \in \bigcup_{j=1}^{i-1} Q_j\}$ denote the set of nodes that were an input for a query q before round i.

Fixing a query $q' \in Q_i$ along with a node u we have

$$\Pr[\mathtt{targets}(q',u) \, : \, \mathsf{parents}(u) \cap NQ_i \neq \emptyset] \leq 2^{-w} \ .$$

This follows because, if some $v \in \mathsf{parents}(u)$ has never been the target of a prior random oracle query, then we can view the label $\mathsf{lab}_{G,H,x}(v)$ as a random w-bit string that is yet to be fixed. In particular, we can view the prelabel $\mathsf{prelab}(u) = \mathsf{lab}_{G,H,x}(v) \oplus \left(\bigoplus_{i \in \mathsf{parents}(u) \setminus \{v\}} \mathsf{lab}_{G,H,x}(i) \right)$ as a random w-bit string that is yet to be fixed. Thus, $\Pr[\mathsf{prelab}(u) = q' \, : \, \mathsf{parents}(u) \cap NQ_i \neq \emptyset] = 2^{-w}$. Union bounding over all nodes u we have

$$\Pr[\exists u \, : \, \mathtt{targets}(q',u) \wedge \mathsf{parents}(u) \cap NQ_i \neq \emptyset] \leq N \times 2^{-w} \ .$$

Finally, we can union bound over all q queries to see that the probability there exists a lucky query is at most

$$\Pr[\exists u,q' \, : \, \mathtt{targets}(q',u) \wedge \mathsf{parents}(u) \cap NQ_i \neq \emptyset] \leq qN \times 2^{-w} \ .$$

A similar argument shows that the probability of a *lucky output* is at most 2^{-w}. Thus, the probability of a lucky query or lucky output is at most $\frac{qN+1}{2^w}$.

If there are no lucky queries, then for each node $u \in V$ and round i such that there exists $q \in Q_i$ with $\mathsf{targets}(q,u)$, then for all nodes $v \in \mathsf{parents}(u)$, there exists a query q' in *some* previous round $j < i$ with $q' \in Q_j$ and $\mathsf{targets}(q',u)$. Notably, let $z < i$ be the last round in which there exists a query $q_0 \in Q_z$ with $\mathsf{targets}(q_0,v)$ — which means that we have $v \in R_y$ for any round $z < y \leq i$. Thus, by definition of $\mathtt{BlackPebble}^H(\mathsf{Trace}_{\mathcal{A},R,H}(T))$ we have $v \in P_y$ for any $z \leq y < i$ because we never discard pebbles $v \in R_y$. It follows that $\mathsf{parents}(P_{i+1} \setminus P_i) \subseteq P_i$.

Furthermore, if the output is not lucky, then there exists a round j and a query $q \in Q_j$ such that $\mathsf{targets}(q,N)$. By definition of $\mathtt{BlackPebble}^H(\mathsf{Trace}_{\mathcal{A},R,H}(T))$ this means that $N \in P_j$. Thus with probability at least $\epsilon - \frac{qN+1}{2^w} - \frac{N^2}{2^{w-1}-2N}$, $\mathtt{BlackPebble}^H(\mathsf{Trace}_{\mathcal{A},R,H}(T))$ is a legal black pebbling. $\square$

H.3 Extractor Argument

We now argue that with high probability, the execution cost of an adversary that computes an instance of $f_{G,H}$ corresponds to the cumulative memory cost of its ex-post-facto pebbling.

We now formally define the cost of computing a function based on its execution trace.

Definition 12. *The memory cost* cost *of a function* $f_{G,H}$ *is defined by*

$$\mathsf{cmc}_{q,\epsilon}(f_{G,H}) = \min_{\mathcal{A},x} \mathbb{E}[\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}(x))],$$

where the minimum is taken over all valid inputs x and all algorithms $\mathcal{A}$ making at most q queries that compute $f_{G,H}(x)$ correctly with probability at least ϵ.

We now show that any algorithm $\mathcal{A}^{H(\cdot)}$ that computes a function that follows the XOR labeling rule correctly with probability at least ϵ has cost corresponding to the cumulative cost of the resulting legal black pebbling, $\Pi^{\parallel}_{cc}(G)$. The proof uses that fact that if an attacking strategy does not yield a corresponding legal black pebbling, then the attacking strategy can be modified to form an extractor for the labels of a subset of nodes. Specifically, an extractor with access to the attacking strategy, the state of the memory, and a few select hints can successfully predict a large number of random bits, which cannot happen with high probability. The hints given to the extractor describes the positions of the random bits, and ensure these bits remain "random" (that is, we do not explicitly query these locations later).

In particular, the extractor uses the hints to simulate $\mathcal{A}^{H(\cdot)}$ but the hints do not include the current state of memory σ_i.

Lemma 10. *[DKW11] Let B be a series of random bits and let $\mathcal{A}$ be an algorithm that receives a hint $h \in H$ and can query B at specific indices. If $\mathcal{A}$ outputs a subset of k indices of B that were previously not queried, as well as guesses for each of the bits, the probability there exists some $h \in H$ so that all the k guesses are correct is at most $\frac{|H|}{2^k}$.*

Theorem 11. *Let G be a DAG with N nodes, maximum indegree $\delta \geq 2$, and $\mathsf{parents}(u) \neq \mathsf{parents}(v)$ for all pairs $u \neq v \in V$, and let $f_{G,H}$ be a function that follows the XOR labeling rule, with label size w. Let $q < 2^{w/32}$ be a number of queries to a random oracle, $32\log N < w$, and $\frac{\epsilon}{4} > 2^{-w/2+2} > \frac{qN+1}{2^w} + \frac{N^2}{2^{w-1}-2N}$. Then*

$$\mathsf{cmc}_{q,\epsilon}(f_{G,H}) \geq \frac{\epsilon w}{8\delta} \cdot \Pi^{\parallel}_{cc}(G),$$

where w is the size of each label.

Proof. Suppose by way of contradiction, that $\mathbb{E}[\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}x)] \leq \frac{w}{8\delta}\Pi^{\parallel}_{cc}(G)$ where $\mathsf{Trace}_{\mathcal{A},R,H}(x) = \{(\sigma_i, Q_i)\}_{i=1}^t$ is a random execution trace of $\mathcal{A}^{H(\cdot)}$. By Markov's inequality we have $\Pr\left[\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}x) \leq \frac{|P_i|w}{4\delta}\right] \leq \frac{\epsilon}{2}$. By assumption we have

$$\mathcal{A}^{H(\cdot)}(x) = f_{G,H}(x) \geq \epsilon \ .$$

Similarly, we have $\sum_i |Q_i| \leq q$ since $\mathcal{A}^{H(\cdot)}$ makes at most q random oracle queries.

Consider a random execution trace $\mathsf{Trace}_{\mathcal{A},R,H}(x) = \{(\sigma_i, Q_i)\}_{i=1}^t$ of $\mathcal{A}^{H(\cdot)}(x)$. By Markov's inequality we have $\Pr\left[\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}x) \leq \frac{w\Pi^{\parallel}_{cc}(G)}{4\delta}\right] \leq \frac{\epsilon}{2}$. By Theorem 10, with probability at least $\epsilon/2 - \frac{qN+1}{2^w} - \frac{N^2}{2^{w-1}-2N} \geq \epsilon/4$ we get a *successful* execution trace with $\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}x) \geq \frac{w\Pi^{\parallel}_{cc}(G)}{4\delta}$ (i.e., $\mathcal{A}^{H(\cdot)}$ that succeeds in calculating $f_{G,H}(x)$) *and* we can extract a legal black pebbling $P = (P_1, \ldots, P_t) = \mathtt{BlackPebble}^H(\mathsf{Trace}_{\mathcal{A},R,H}(T))$ from this execution trace *and* the event $\mathtt{COLLISION}$ did not occur (i.e., all nodes have distinct labels and pre-labels). Let $\Pi^{\parallel}_{cc}(P) \geq \Pi^{\parallel}_{cc}(G)$ be the cumulative complexity of this pebbling.

If $\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}x) \leq \frac{w\Pi^{\parallel}_{cc}(P)}{4\delta}$ then for some step i we must have $|\sigma_i| < \frac{|P_i|w}{4\delta}$. By construction of the pebbling, for each node $v \in P_i$ we have $i_v := \mathtt{NextRequiredRound}(v,i) \leq$

$\texttt{NextTargetRound}(v,i)$. Let $q_v \in Q_{i_v}$ denote the first query in which $\texttt{input}(q_v,v)=1$. Assuming that the event $\texttt{COLLISION}$ did not occur (i.e., all nodes have distinct labels and pre-labels) there is a unique node u_v such that $\mathsf{prelab}(u_v)=q_v$. We will let u_v be the node targeted by the query q_v.

We would like to extract $\mathsf{lab}(v)$ from the query q_v. However, under the XOR labeling rule, the situation is complicated since $\mathsf{lab}(v)$ is not explicitly revealed in the random oracle query q_v, instead $q_v = \mathsf{lab}(v)\left(\bigoplus_{j\in\mathsf{parents}(u_v)\setminus\{v\}}\mathsf{lab}(j)\right)$. Let $A_v=\mathsf{parents}(u_v)\setminus\{v\}$ denote the set of additional nodes whose labels are XORed with $\mathsf{lab}(v)$. To solve the problem we fix some arbitrary ordering $v_1,\ldots,v_{|P_i|}$ over the set P_i and define the set C_i by following the following procedure: 1) Set $P_i^0=P_i$ and $j=0$, 2) Select the first element $v\in P_i^j$ 3) Set $C_i=C_i\cup A_v$. 4) Set $P_i^{j+1}=P_i^j\setminus(\{v\}\cup A_v)$ and increment j. 5) repeat steps 2–4 until P_i^j is empty.

We set $S=C_i\cup P_i$. Note that $|C_i|\le(\delta-1)|P_i|$ where δ is the indegree of the graph. This follows because in each step we add at most $|A_v|\le\delta-1$ nodes to C_i and discard *at least one* node from P_i^j. Similarly, we note that $|P_i\setminus C_i|\ge\frac{|P_i|}{\delta}$. Thus, $|S|\ge|C_i|+\frac{|P_i|}{\delta}$. In case $|P_i|<\delta$ we still must have $|P_i\setminus C_i|\ge1$.

We argue that an extractor using $\mathcal{A}^{H(\cdot)}$ can predict $(|S|)w$ random bits using $\frac{|P_i|}{4\delta}w$ bits of information from the state of $\mathcal{A}^{H(\cdot)}$, along with the following hint, which consists of three parts:

1. The sets $S=C_i\cup P_i$ and $|C_i|$ itself is given as a hint to tell the extractor which labels to extract. The size of this component of the hint is at most $|S|\log N+|C_i|\log N$ bits.
2. For each node $v\in P_i\setminus C_i$ the hint includes the index of the first query $q_v\in Q_{i_v}$ denote the first query in which $\texttt{input}(q_v,v)=1$ as well as the target u_v of this query q_v. Since, there are *at most* q queries total it will take at most $\log_2 q$ bits to encode the index of each query and $\log_2 N$ bits to encode the target of each query. Thus, this component of the hint is at most $|P_i\setminus C_i|\log q+|P_i\setminus C_i|\log N$ bits.
3. For each node $v\in C_i$ the hint includes $\mathsf{lab}(v)$. This component of the hint is at most $|S|w$ bits.
4. For each $v\in S$, the index of the first query when $\mathsf{lab}(v)$ might be compromised. Observe that if the extractor successfully predicts a random string at a location v, but then $\mathsf{lab}(v)$ is later queried by the attacker, we cannot distinguish this case at the end from the case that the extractor simply read $\mathsf{lab}(v)$ after making the query. Effectively, the extractor is no longer predicting a random string. To avoid this, the hint given to the extractor details queries that would compromise the randomness of the desired locations. Formally, the hint is the minimal index i such that $q_i^j=v$, which yields returns the query $H(q_i^j)=\mathsf{lab}(v)$. This component of the hint tells the extractor the locations of the random strings to be predicted, and has size at most $|S|\log q$ bits.

We remark that the total length of the hint h is $|S|(2\log q+2\log N)+|C_i|w+|\sigma_i|\le\delta(2\log q+2\log N)+|C_i|w+\frac{|P_i|}{2\delta}$, while the extractor will output $|S|w\ge\left(|C_i|+\frac{|P_i|}{\delta}\right)w$

random bits. Since we assume that $\log N < \frac{w}{32}$ and $q < 2^{w/32}$ we have

$$|S|w - |h| \geq |P_i \setminus C_i| w \left(1 - \frac{1}{2\delta}\right) - w/8 \geq w/2 \ .$$

The extractor will simulate $\mathcal{A}^{H(.)}$ starting from initial state σ_i. The extractor maintains a list $L = \{(v,\mathsf{lab}(v)\}$ of known labels — initially $L = \{(v,\mathsf{lab}(v)) \ : v \in C_i\}$ — as well as a list $L_{pre} = \{(v,\mathsf{prelab}(v)\}$ of known pre-labels — initially empty. In each round $j \geq i$ we observe a new batch of random oracle queries Q_j. For each query $y \in Q_j$ we check if the query is of interest.

1. If our hint indicates that y has some node $v \in P_i \setminus C_i$ as input then we will compute $\mathsf{lab}(v) = y \oplus \left(\bigoplus_{j \in \mathsf{parents}(u_v) \setminus \{v\}} \mathsf{lab}(j)\right)$ add $(v,\mathsf{lab}(v))$ to our list — here we rely on the fact that the hint contains the target u_v of the query y to identify $\mathsf{parents}(u_v)$ and, to compute $\bigoplus_{j \in \mathsf{parents}(u_v) \setminus \{v\}} \mathsf{lab}(j)$, we rely on the fact that L must contain each of the pairs $(j,\mathsf{lab}(j))$ since $\mathsf{parents}(u_v) \subseteq C_i \cup \{v\}$. In this case we have $\mathsf{prelab}(u_v) = y$ so we will also add the pair (u_v,y) to L_{pre} in this case. If we happen to have $(u_v,\mathsf{lab}(u_v)) \in L_{pre}$ for some node v then we write $\mathsf{lab}(u_v)$ on $\mathcal{A}$'s response tape; otherwise we query $H(y)$, add $(u_v,H(y))$ to L and record $H(y)$ on $\mathcal{A}$'s response tape.
2. If our hint indicates that y targets some node $v \in P_i$ then we will look for the pair $(v, \mathsf{lab}(v))$ in our list L and write this response on $\mathcal{A}$'s response tape. Note that $(v, \mathsf{lab}(v))$ *must* be in L because for *any* node $v \in P_i$ we have $\texttt{NextRequiredRound}(v,i) \leq \texttt{NextTargetRound}(v,i)$. Thus, we will extract $(v,\mathsf{lab}(v))$ before the current round $j \geq \texttt{NextTargetRound}(v,i)$. In this case we have $\mathsf{prelab}(v) = y$ so we will add (v,y) to the list L_{pre}.
3. If we have $(v,y) \in L_{pre}$ for some node $v \in C_i$ then the extractor looks for a pair $(v,\mathsf{lab}(v)) \in L$ and writes the response $\mathsf{lab}(v)$ on $\mathcal{A}$'s response tape. Note that we *must* have $(v,\mathsf{lab}(v)) \in L$ in this case. Clearly, if $v \in C_i$ then $(v,\mathsf{lab}(v)) \in L$ since we start with all of these labels in L. If $v \in P_i \setminus C_i$ then y cannot the first query to target v then we would have already added $(v,\mathsf{lab}(v))$ in the previous case (If y were first query to target v then we would be in case 2 since the hint encodes the index of the first query to target v).
4. Otherwise, we simply query $H(y)$ and write $H(y)$ on $\mathcal{A}$'s response tape.

After $\mathcal{A}^{H(.)}$ finishes the extractor the list L will contain $(v,\mathsf{lab}(v))$ for each node $v \in C_i \cup P_i$, but we may not have $(v,\mathsf{prelab}(v))$ for each node. Thus, the extractor will begin computing $f_{G,H}$ using the honest evaluation algorithm. As before the extractor will check each random oracle query y to see if $(v,y) \in L_{pre}$ for some node $v \in C_i$. If so then extractor finds $(v,\mathsf{lab}(v)) \in L$ and uses $\mathsf{lab}(v)$ as the output without querying $H(.)$. As we progress the extractor maintains a list $(v,\mathsf{lab}(v))$ for each of the labels computed so far, and the extractor immediately adds $(v,\mathsf{prelab}(v))$ to L_{pre} once all of the labels of the node in $\mathsf{parents}(v)$ are known. Finally, the extractor will output $(\mathsf{prelab}(v),\mathsf{lab}(v))$ for each node $v \in P_i \cup C_i$.

Assuming that we were able to extract a legal pebbling from the execution trace $\mathsf{Trace}_{\mathcal{A},R,H}(x)$ the extractor will always succeed in extracting $|S|$ input/output pairs

$(\mathsf{prelab}(v),\mathsf{lab}(v))$ without querying the random oracle at $\mathsf{prelab}(v)$ for each node $v \in P_i \cup C_i$, and if all of the pre-labels are distinct then we have $|S|$ input/output pairs.

In general, the probability that our extractor can extract $|S|$ input/output pairs from our short hint will be *at least*

$$\epsilon - \frac{qN+1}{2^w} - \frac{N^2}{2^{w-1}-2N} \geq \frac{\epsilon}{2} \geq 2^{-w/2+2} ,$$

where ϵ is the probability $\mathcal{A}^{H(\cdot)}$ correctly computes $f_{G,H}(x)$ and $\frac{N^2}{2^{w-1}-2N} + \frac{N^2}{2^{w-1}-2N}$ upper bounds the probability that we fail to extract a black pebbling or two labels/prelabels collide.

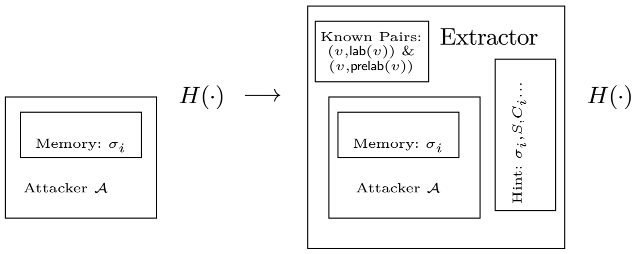
However, by Lemma 10 the probability the extractor can successfully output $|S|$ input output pairs from a hint of size $|S|$ is at most

$$2^{-|S|w+|h|} \leq 2^{-w/2} .$$

However, this is a contradiction as it implies that

$$2^{-w/2+2} \leq \frac{\epsilon}{2} \leq 2^{-w/2}. \qquad \square$$

See Figure 6 for intuition.



Memory: σ_i
Attacker $\mathcal{A}$
$H(\cdot)$ $\longrightarrow$
Known Pairs: $(v,\mathsf{lab}(v))$ & $(v,\mathsf{prelab}(v))$
Extractor
Memory: σ_i
Attacker $\mathcal{A}$
Hint: $\sigma_i, S, C_i \ldots$
$H(\cdot)$

Fig. 6: An extractor that uses the attacker to predict distinct outputs of random oracle $H(\cdot)$.

Reminder of Theorem 7.

Let G be a DAG with N nodes, indegree $\delta \geq 2$, and $\mathsf{parents}(u) \neq \mathsf{parents}(v)$ for all pairs $u \neq v \in V$, and let $f_{G,H}$ be a function that follows the XOR labeling rule, with label size w. Let $\mathcal{H}$ be a family of random oracle functions with outputs of label length w and $H = (H_1, H_2)$, where $H_1, H_2 \in \mathcal{H}$. Let m be a number of parallel instances such that $mN < 2^{w/32}$, $q < 2^{w/32}$ be the maximum number of queries to a random oracle, and let $\frac{\epsilon}{4} > 2^{-w/2+2} > \frac{qmN+1}{2^w} + \frac{m^2N^2}{2^{w-1}-2mn}$. Then

$$\mathsf{cmc}_{q,\epsilon}(f_{G,H}^{\times m}) \geq \frac{\epsilon m w}{8} \cdot \Pi_{cc}^{\parallel}(G).$$

Proof of Theorem 7. [Sketch] The proof is quite similar to Theorem 11. Except that the number of nodes in $G^{\times m}$ is mN. We need to show that the event COLLISION is negligible as in Lemma 9. The proof is *almost* identical except that we need to add a special case for all of the m *sink* nodes in $G^{\times m}$. We note that all of these source nodes are *guaranteed* to have distinct pre-labels since the *inputs* $X=(x_1,x_2,...,x_m)$ to each instance of $f_{G,H}$ are distinct. With this observation we can easily adapt the proof of Lemma 9 to conclude that

$$\Pr[\texttt{COLLISION}] \leq \frac{m^2N^2}{2^{w-1}-2mn}$$

and we can adapt the proof of Theorem 10 to show that we extract a legal black pebbling with probability at least

$$\epsilon - \frac{qN+1}{2^w} - \frac{m^2N^2}{2^{w-1}-2mn} \ .$$

At this point the extractor argument follows the proof of Theorem 11 exactly. □

I Cryptanalysis of DRSample as Evidence for Conjectures

In this section of the appendix we provide strong evidence in support of our conjectures about the depth-robustness of DRSample. The first conjecture states that DRSample is $(e=c_1N\log\log N/\log N, d=c_2N\log\log N/\log N)$-depth robust for suitable constants $c_1,c_2,c_3>0$, and the second (stronger) conjecture states that DRSample is $(e=c_1N\log\log N/\log N, d=c_2N\log\log N/\log N, b=c_3\log N/\log\log N)$-block-depth robust. We consider three different state-of-the-art depth-reducing attacks: the layered attack [AB16], Valiant's Lemma [Val77, AB16] and the pebbling reduction of Alwen et al. [ABP17] which constructs a depth-reducing set S of size at most $e=|S|\leq \Pi_{cc}^{\parallel}(G)/d$ — we use the greedy pebble attack [BCS16] as our starting point since this is the best known pebbling attack against DRSample. The layered attack [AB16] was originally used to attack Argon2i and Balloon Hash, while Valiant's Lemma [Val77, AB16] was originally used to derive the general upper bound that any DAG G with constant indegree is $(e=c_1N\log\log N/\log N, d=c_2N/\log^2 N)$ reducible for some constants c_1 and c_2 — hence $\Pi_{cc}^{\parallel}(G)=\mathcal{O}\left(N^2\log\log N/\log N\right)$ by a result from [AB16].

We show that when we want to reduce the depth to just $d=N/\sqrt{\log N}$ that each of the above attacks *require* the removal of at least $\Omega(N\log\log N/\log N)$ nodes. Along the way we introduce a *general* framework for analyzing Valiant's Lemma [Val77, AB16] for a specific DAG and use our techniques to analyze the performance of the attack against Argon2iA and Argon2iB. Interestingly, the performance *exactly* matches the performance of the layered attack against these graphs. This provides theoretical justification to a surprising finding of Alwen et al. [AB17, ABH17] that the layered attack seems to perform slightly worse than Valiant's Lemma despite the fact that (at the time) the best asymptotic upper bounds for the layered attack were much better. Another interesting finding is that the performance of *all three distinct* attacks are asymptotically equivalent despite the fact that the depth-reducing sets are chosen in very different ways.

I.1 Valiant's Lemma: Basic Setting and Observation

We introduce two variant's of Valiant's Lemma attack in Algorithm 7 and Algorithm 8. Both algorithms start by partitioning the edges E into sets $E_1,...,E_n$ where $E_i=\{(u,v)\ |\mathrm{MSDB}(\ell_u,\ell_v)=i\}$ (resp. $S_i=\{u:(u,v)\in E_i\}$) is the set of all edges (u,v) with the property that the most-significant different bit between the binary strings ℓ_u and ℓ_v is i. Here, ℓ_u is the n bit binary string corresponding to the integer $u-1$ i.e. $\ell_1=0^n$. If the nodes are given in topological order then it is easy to show that for any $X\subseteq[n]$ we have $\mathsf{depth}(G-\bigcup_{i\in X}S_i)\leq N/2^{|X|}$. Algorithm 8 takes as input a target depth d and selects the $\log_2(N/d)$ smallest sets S_i to add to X to ensure the depth of the graph $\mathsf{depth}(G-\bigcup_{i\in X}S_i)$ is at most d. By contrast, Algorithm 7 takes as input a maximum size e and repeatedly finds $i\in[n]\setminus X$ with minimum size $|S_i|$ to add to X until we have we cannot find any $i\in[n]\setminus X$ such that $\left|\bigcup_{j\in X\cup\{i\}}S_i\right|\leq e$. We now introduce a general technique to analyze DAGs $G=(V=[N],E)$ with edge set $E=\{(i,i+1):1\leq i<N\}\cup\{(r(i),i):1<i\leq N\}$ for some predecessor function $r(i)<i$.

Definition 13. *Let $G=(V=[N],E)$ be a DAG with $N=2^n$ nodes. Define E_i as follows:*

$$E_i:=\{(u,v)\in E: MSDB(\ell_u,\ell_v)=i\}.$$

Where ℓ_u is the n bit binary string corresponding to the integer $u-1$ i.e. $\ell_1=0^n$.

All of the DAGs $G=(V=[N],E)$ we consider in this section have edge set $E=\{(i,i+1):1\leq i<N\}\cup\{(r(i),i):1<i\leq N\}$ where $r(i)$ is a random predecessor of the node i and the selection of the predecessor $r(i)$ can be viewed as an independent choice for each node i. Note that the function $r(i)<i$ varies for different constructions e.g., Argon2iA,Argon2iB and DRSample. Thus, we can split the set E_i into two sets:

$$\begin{aligned}E_i&=\{(u,v)\in E:\mathrm{MSDB}(\ell_u,\ell_v)=i\}\\ &=\underbrace{\{(v,v+1):\mathrm{MSDB}(\ell_v,\ell_{v+1})=i\}}_{=:A_i}\cup\underbrace{\{(r(v),v):\mathrm{MSDB}(\ell_{r(v)},\ell_v)=i\}}_{=:B_i}\end{aligned}$$

We introduce an indicator random variable which checks whether $\mathrm{MSDB}(\ell_{r(v)},\ell_v)=i$ or not.

Definition 14. *Let $G=(V,E)$ be a DAG and $v\in V$ be any node in the set V. Then an indicator random variable $X_{v,i}$ to check MSDB is defined by*

$$X_{v,i}=\begin{cases}1 & \textit{when } MSDB(\ell_{r(v)},\ell_v)=i\\ 0 & \textit{otherwise}\end{cases}$$

The following fact states that $|A_i|=2^{-i}N$ for each $i\leq n$. The fact applies to *all* DAGs we will analyze. Thus, to analyze the size of E_i for a particular DAG it suffices to focus on the set B_i.

Fact 12 *Let $i\leq n$ be given and set $A_i=:\{(v,v+1):MSDB(\ell_v,\ell_{v+1})=i\}$ then $|A_i|=2^{-i}N$.*

Proof. It is clear that $\mathrm{MSDB}(\ell_v,\ell_{v+1})=i$ if and only if $v=2^{i-1}+2^i\cdot k$ for any integer $0\le k<2^{-i}N$. Therefore, we have that $|A_i|=|\{k:0\le k<2^{-i}N,k\in\mathbb{Z}\}|=2^{-i}N$. $\square$

Lemma 11 is a general tool which we will use to bound the size of B_i.

Lemma 11. *MSDB$(\ell_{r(v)},\ell_v)=i$ if and only if $2^{i-1}+m\ge\ell_v-\ell_{r(v)}>m$ where $\ell_v=j\cdot 2^{i-1}+m$ where $0\le m<2^{i-1}$ and j is a nonnegative odd integer.*

Proof. If $\ell_{r(v)}<\ell_v-m$, then clearly $\mathrm{MSDB}(\ell_{r(v)},\ell_v)$ will never be i. And if $\ell_v-\ell_{r(v)}>2^{i-1}+m$, then we have $\mathrm{MSDB}(\ell_{r(v)},\ell_v)\ge i+1$. Finally, if j is even, then the i-th bit should be 0, which means that $\mathrm{MSDB}(\ell_{r(v)},\ell_v)\ne i$ because flipping the i-th bit would increase the label which contradicts to the definition of the predecessor $r(v)$. $\square$

Example 1. Suppose that $i=3$ and choose $m=3<2^{i-1}=4$ and $j=3$. Then $\ell_v=3\cdot 2^2+3$ and the bit representation for ℓ_v is 1111. Then we observe that if $\ell_{r(v)}$ is greater than or equal to $1100=1111-11=\ell_v-m$, then $\mathrm{MSDB}(\ell_v,\ell_{r(v)})$ should be 1 or 2, which is not 3. Moreover, if $\ell_{r(v)}$ is less than $1000=1111-100-11=\ell_v-2^{i-1}-m$, then $\mathrm{MSDB}(\ell_v,\ell_{r(v)})$ should be 4, which is not 3. Therefore, we have that to yield $\mathrm{MSDB}(\ell_v,\ell_{r(v)})=3$, then we need to pick $r(v)$ which satisfies $2^{i-1}+m\ge\ell_v-\ell_{r(v)}>m$.

When the function $r(v)$ is random we expected size of the set B_i is given in Corollary 7. While $|B_i|$ is a random variable it is easy to show that $|B_i|$ will be close to its expectation because the values $r(v)$ are independently chosen for each vertex v.

Corollary 7. *The expected value of the size of the set B_i is given by*

$$\mathbb{E}[|B_i|]=\sum_{j=0}^{\frac{N}{2^i}-1}\sum_{m=0}^{2^{i-1}-1}\Pr\left[2^{i-1}+m\ge\ell_v-\ell_{r(v)}>m\right]$$

where $\ell_v=(2j+1)\cdot 2^{i-1}+m$. Furthermore, $|B_i|$ is tightly concentrated around its mean

$$\Pr[|B_i|>2\mathbb{E}[|B_i|]]<\left(\frac{e}{4}\right)^{\mathbb{E}[|B_i|]}\quad\text{, and}\quad \Pr\left[|B_i|<\frac{1}{2}\mathbb{E}[|B_i|]\right]<\left(\frac{\sqrt{2}}{\sqrt{e}}\right)^{\mathbb{E}[|B_i|]}.$$

In particular, if $\mathbb{E}[|B_i|]>\log^2 N$ then, except with negligible probability, we have $\frac{1}{2}\mathbb{E}[|B_i|]\le|B_i|\le 2\mathbb{E}[|B_i|]$. If $\mathbb{E}[|B_i|]<\mu$ for any upper bound $\mu\ge\log^2 N$ then, except with negligible probability $|B_i|\le 2\mu$.

Proof. Recall an indicator random variable from Definition 14. Clearly, we have $|B_i|=\sum_{v=1}^{N}X_{v,i}$. Taken together with Lemma 11, we conclude that

$$\begin{aligned}\mathbb{E}[|B_i|]&=\mathbb{E}\left[\sum_{v=1}^{N}X_{v,i}\right]=\sum_{v=1}^{N}\mathbb{E}[X_{v,i}]=\sum_{v=1}^{N}\Pr\left[\mathrm{MSDB}(\ell_{r(v)},\ell_v)=i\right]\\&=\sum_{j=0}^{\frac{N}{2^i}-1}\sum_{m=0}^{2^{i-1}-1}\Pr\left[\mathrm{MSDB}(\ell_{r(v)},\ell_v)=i\right]\end{aligned}$$

$$=\sum_{j=0}^{\frac{N}{2^i}-1}\sum_{m=0}^{2^{i-1}-1}\Pr\left[2^{i-1}+m\geq \ell_v-\ell_{r(v)}>m\right]$$

provided by $\ell_v=(2j+1)\cdot 2^{i-1}+m$. Moreover, one can observe that $X_{1,i},\cdots,X_{N,i}$'s are all independent. Then applying a multiplicative Chernoff bound, we have

$$\Pr[|B_i|>2\mathbb{E}[|B_i|]]<\left(\frac{e}{4}\right)^{\mathbb{E}[|B_i|]}$$

and

$$\Pr\left[|B_i|<\frac{1}{2}\mathbb{E}[|B_i|]\right]<\left(\frac{e^{-1/2}}{(1/2)^{1/2}}\right)^{\mathbb{E}[|B_i|]}=\left(\frac{\sqrt{2}}{\sqrt{e}}\right)^{\mathbb{E}[|B_i|]}. \qquad \square$$

Now we have the following Lemma from [AB16] which is essentially equivalent to Valiant's Lemma [Val77]:

Lemma 12 ([AB16], Lemma 6.1). *Given a DAG G with m edges and depth $\mathsf{depth}(G)\leq d=2^i$ there is a set of m/i edges such that by deleting them we obtain a graph of depth at most $d/2$.*

Lemma 13 ([AB16], Lemma 6.2). *Let $G=(V,E)$ be an arbitrary DAG of size $|V|=N=2^n$ with $\mathsf{indeg}(G)=\delta$. Then for every integer $t\geq 1$ there is a set $S\subseteq V$ of size $|S|\leq \frac{t\delta N}{\log(N)-t}$ such that $\mathsf{depth}(G-S)\leq 2^{n-t}$. Furthermore, there is an efficient algorithm to find such S.*

Applying this lemma to well-known graphs such as Argon2i-A, B, and DRSample, one should be able to get the reducibility of such graphs. We are going to argue that the results by applying Valiant's lemma matches the known results for Argon2i and DRSample, with the interesting observation that layered attack against DRSample is not effective based on the upcoming analysis. To see this, we need the following algorithm (Algorithm 7) from invoking Lemma 12 and Lemma 13:

I.2 Analysis on Argon2i (Improved Results)

Theorem 13. *Let G be Argon2i-A with N nodes and let $S=\mathsf{Valiant}(G,e)$ with $e^2>N\log N$. Then with high probability, $\mathsf{depth}(G-S)=\mathcal{O}((N/e)^2\log N)$.*

Proof. In Argon2i-A, the edge distribution is uniformly random. In particular, for $v>1$ the predecessor $r(v)$ is chosen uniformly at random from the set $\{1,...,v-2\}$. By Corollary 7, we have

$$\mathbb{E}[|B_i|]=\sum_{j=0}^{\frac{N}{2^i}-1}\sum_{m=0}^{2^{i-1}-1}\Pr\left[2^{i-1}+m\geq \ell_v-\ell_{r(v)}>m\right] \qquad \text{where } \ell_v=(2j+1)\cdot 2^{i-1}+m$$

$$=\sum_{j=0}^{\frac{N}{2^i}-1}\sum_{m=0}^{2^i-1}\frac{2^{i-1}}{v}=\sum_{m=0}^{2^{i-1}-1}\sum_{j=0}^{\frac{N}{2^i}-1}\frac{2^{i-1}}{(2j+1)\cdot 2^{i-1}+m}$$

Algorithm 7: An algorithm to sample a depth-reducing set.

```
Input   : DAG $G=(V(G),E(G))$ with $|V(G)|=N=2^n$, and a target size $e$.
Output  : A depth-reducing set $S$ with $|S|\leq e$ to remove
Function Valiant($G$, $e$):
    for $i\leq n$ do
        $E_i:=\{(u,v)|MSDB(u,v)=i\}$
        $S_i:=\{u|(u,v)\in E_i\}$
    end
    $S:=\varnothing$
    $X:=\varnothing$
    while $|S|\leq e$ do
        $i=\text{argmin}_{i\notin X}|E_i|$ // Find smallest $|E_i|$ that hasn't been picked yet
        $S=S\cup S_i$
        $X=S$
    end
    return $S$
```

$$\leq \sum_{m=0}^{2^{i-1}-1}\left[\underbrace{\int_0^{\frac{N}{2^i}-1}\frac{2^{i-1}dj}{2^i\cdot j+2^{i-1}+m}}_{(1)}+\underbrace{\frac{2^{i-1}}{2^{i-1}+m}}_{\text{when } j=0}\right]$$

because for a positive decreasing function f, we have $\sum_{k=1}^{n}f(k)\leq\int_0^n f(t)dt$. Moreover, we have

$$\begin{aligned}(1)&=\left[\frac{\ln(2^i\cdot j+2^{i-1}+m)}{2}\right]_0^{\frac{N}{2^i}-1}\\&=\frac{1}{2}\ln\left[\frac{N-2^{i-1}+m}{2^{i-1}+m}\right]=\frac{1}{2}\ln\left[1+\frac{N-2^i}{2^{i-1}+m}\right]\end{aligned}$$

which leads to

$$\begin{aligned}\mathbb{E}[|B_i|]&\leq\sum_{m=0}^{2^{i-1}-1}\left[\frac{1}{2}\ln\left[1+\frac{N-2^i}{2^{i-1}+m}\right]+\frac{2^{i-1}}{2^{i-1}+m}\right]\\&\leq\int_0^{2^{i-1}-1}\frac{1}{2}\underbrace{\ln\left[1+\frac{N-2^i}{2^{i-1}+m}\right]}_{\leq\ln\left[1+\frac{N-2^i}{2^{i-1}}\right]\leq\ln\left[\frac{N}{2^{i-1}}\right]}+\frac{2^{i-1}}{2^{i-1}+m}dm+\underbrace{\frac{1}{2}\ln\left[1+\frac{N-2^{i-1}}{2^{i-1}}\right]+1}_{\text{when } m=0}\\&\leq2^{i-1}\cdot\frac{1}{2}\ln\left[\frac{N}{2^{i-1}}\right]+\underbrace{\int_0^{2^{i-1}-1}\frac{2^{i-1}}{2^{i-1}+m}dm}_{\leq[2^{i-1}\ln(2^{i-1}+m)]_0^{2^{i-1}}=2^{i-1}\ln2}+\frac{1}{2}\ln\left[\frac{N}{2^{i-1}}\right]+1\end{aligned}$$

$$\leq 2^{i-1}\cdot\frac{1}{2}\log_2\left[\frac{N}{2^{i-1}}\right]+\underbrace{2^{i-1}+\frac{1}{2}\log_2\left[\frac{N}{2^{i-1}}\right]+1}_{\leq 2^{i-1}\cdot\frac{1}{2}\log_2 N}$$

$$\leq 2^{i-1}\log_2 N.$$

Therefore, we can argue that with high probability, we have $|B_i|\leq 2\mathbb{E}[|B_i|]\leq 2^i\log N$ for each $i\geq 1+\log\log N$ by Corollary 7 and with high probability $|B_i|\leq 2\log^2 N$ for each $i\leq \log\log N$. Taken together, for $i\leq \log_2 N$, with high probability we have

$$|E_i|\leq 2^i\log_2 N+\frac{N}{2^i}\quad \text{for } i\geq 1+\log\log N$$

We also observe that, except with negligible probability, we have

$$\sum_{i=1}^{\log\log N}|B_i|\leq \log^2 N(\log\log N)=n^2\log n\ .$$

The algorithm will find the j smallest sets $E_{i_1},\ldots,E_{i_j}$ to delete such that $\sum_{k=1}^{j}|E_{i_k}|\leq e$ to reduce the depth to $d=N/2^j$. We can achieve this when $i\simeq(n-\log_2 n)/2$. Then we will delete all sets in the interval $[\frac{n-\log_2 n}{2}-\frac{j}{2},\frac{n-\log_2 n}{2}+\frac{j}{2}]$. Hence, with high probability, the total number of deleting edges is

$$e=\sum_{k=\frac{n-\log_2 n-j}{2}}^{\frac{n-\log_2 n+j}{2}}|E_k|\leq \sum_{k=\frac{n-\log_2 n-j}{2}}^{\frac{n-\log_2 n+j}{2}}2^k\log_2 N+\frac{N}{2^k}+\sum_{i=1}^{\log\log N}|B_i|$$

$$=\sum_{k=0}^{j}2^{\frac{n-\log_2 n-j}{2}+k}\log_2 N+N\cdot 2^{\frac{-n+\log_2 n+j}{2}-k}+n^2\log n$$

$$\leq 2^{\frac{n-\log_2 n-j}{2}+j+1}\log_2 N+2^n\cdot 2^{\frac{-n+\log_2 n+j}{2}+1}+n^2\log n$$

$$=2^{\frac{n}{2}-\frac{\log_2 n}{2}+\frac{j}{2}+1}\left(\underbrace{\log_2 N}_{=n}+\underbrace{2^{\log_2 n}}_{=n}\right)+n^2\log n$$

$$=\sqrt{N}\cdot\frac{1}{\sqrt{n}}\cdot\sqrt{\frac{N}{d}}\cdot 2\cdot 2n+n^2\log n\leq\frac{5\sqrt{n}N}{\sqrt{d}}$$

which implies that $d=\mathcal{O}\left(\left(\frac{N}{e}\right)^2\right)\ln N$. □

Theorem 14. *Let G be Argon2i-B with N nodes and let $S=\mathsf{Valiant}(G,e)$ with $e^3>N^2$. Then with high probability,* $\mathsf{depth}(G-S)=\mathcal{O}((N/e)^3)$.

Proof. In Argon2i-B, we have

$$\Pr[r(i)=j]=\Pr_{x\in[N]}\left[i\left(1-\frac{x^2}{N^2}\right)\in(j-1,j]\right]=\sqrt{1-\frac{j-1}{i}}-\sqrt{1-\frac{j}{i}}$$

since $i\left(1-\frac{x^2}{N^2}\right)\in(j-1,j]$ is equivalent to $N\sqrt{1-\frac{j}{i}}\leq x<N\sqrt{1-\frac{j-1}{i}}$. Similarly, we have

$$\Pr[a\leq r(i)<b]=\Pr_{x\in[N]}\left[i\left(1-\frac{x^2}{N^2}\right)\in(a-1,b-1]\right]=\sqrt{1-\frac{a-1}{i}}-\sqrt{1-\frac{b-1}{i}}.$$

Therefore, in Argon2i-B, we have

$$\begin{aligned}
\mathbb{E}[|B_i|]&=\sum_{j=0}^{\frac{N}{2^i}-1}\sum_{m=0}^{2^{i-1}-1}\Pr\left[\mathrm{MSDB}(\ell_{r(v)},\ell_v)=i\right]\qquad\text{where }\ell_v=(2j+1)\cdot 2^{i-1}+m\\
&=\sum_{j=0}^{\frac{N}{2^i}-1}\sum_{m=0}^{2^{i-1}-1}\Pr\left[2^{i-1}+m\geq \ell_v-\ell_{r(v)}>m\right]\\
&=\sum_{j=0}^{\frac{N}{2^i}-1}\sum_{m=0}^{2^{i-1}-1}\Pr\left[2^{i-1}+m\geq v-r(v)>m\right]\\
&=\sum_{j=0}^{\frac{N}{2^i}-1}\sum_{m=0}^{2^{i-1}-1}\Pr\left[v-2^{i-1}-m\leq r(v)<v-m\right]\\
&=\sum_{j=0}^{\frac{N}{2^i}-1}\sum_{m=0}^{2^{i-1}-1}\sqrt{\frac{2^{i-1}+m+1}{v}}-\sqrt{\frac{m+1}{v}}\\
&=\sum_{m=0}^{2^{i-1}-1}\left[\left(\sqrt{2^{i-1}+m+1}-\sqrt{m+1}\right)\underbrace{\sum_{j=0}^{\frac{N}{2^i}-1}\frac{1}{\sqrt{2^i\cdot j+2^{i-1}+m}}}_{(2)}\right]
\end{aligned}$$

where

$$\begin{aligned}
(2)&\leq\int_0^{\frac{N}{2^i}}\frac{dj}{\sqrt{2^i\cdot j+2^{i-1}+m}}+\frac{1}{\sqrt{2^{i-1}+m}}\\
&=\left[\frac{\sqrt{2^i\cdot j+2^{i-1}+m}}{2^{i-1}}\right]_0^{\frac{N}{2^i}}+\frac{\sqrt{2^{i-1}+m}}{2^{i-1}+m}\\
&\leq\frac{\sqrt{N+2^{i-1}+m}-\sqrt{2^{i-1}+m}}{2^{i-1}}+\frac{\sqrt{2^{i-1}+m}}{2^{i-1}}\\
&=\frac{\sqrt{N+2^{i-1}+m}}{2^{i-1}}
\end{aligned}$$

which leads to

$$\mathbb{E}[|B_i|]\leq\sum_{m=0}^{2^{i-1}-1}\left(\sqrt{2^{i-1}+m+1}-\sqrt{m+1}\right)\cdot\frac{\sqrt{N+2^{i-1}+m}}{2^{i-1}}$$

$$
\begin{aligned}
&= \sum_{m=0}^{2^{i-1}-1} \frac{2^{i-1}}{\sqrt{2^{i-1}+m+1}+\sqrt{m+1}} \cdot \frac{\sqrt{N+2^{i-1}+m}}{2^{i-1}} \\
&\leq \sum_{m=0}^{2^{i-1}-1} \frac{2^{i-1}}{\sqrt{2^{i-1}}} \cdot \frac{\sqrt{3N}}{2^{i-1}} \\
&\leq \frac{\sqrt{3N}}{\sqrt{2^{i-1}}} \cdot 2^{i-1} = \sqrt{3N \cdot 2^{i-1}}.
\end{aligned}
$$

Therefore, since $\sqrt{3N\cdot 2^{i-1}} \gg \log^2 N$ for any $i \geq 0$, by Corollary 7 we can argue that with high probability, we have $|B_i| \leq 2\mathbb{E}[|B_i|] \leq 2\sqrt{3N\cdot 2^{i-1}} = \sqrt{6N\cdot 2^i}$ for $i \geq 2+2\log\log N$. Taken together, for $i \leq \log_2 N$, with high probability we have

$$|E_i| \leq \sqrt{6}\left(\sqrt{N\cdot 2^i} + \frac{N}{2^i}\right).$$

Our goal is to find j smallest sets to delete to make smallest E_i when $d = N/2^j$. We can achieve this when $i = n/3$. Then we will delete all sets in the interval $[\frac{n}{3}-\frac{j}{3}, \frac{n}{3}+\frac{2j}{3}]$. Hence, the total number of deleting edges is

$$
\begin{aligned}
e \leq \sum_{k=\frac{n}{3}-\frac{j}{3}}^{\frac{n}{3}+\frac{2j}{3}} |E_k| &\leq \sqrt{6} \sum_{k=\frac{n}{3}-\frac{j}{3}}^{\frac{n}{3}+\frac{2j}{3}} \left(\sqrt{N\cdot 2^i} + \frac{N}{2^i}\right) \\
&= \sqrt{6} \underbrace{\sum_{k=0}^{j} 2^{\frac{n}{2}} \cdot 2^{\frac{1}{2}\left(\frac{n}{3}-\frac{j}{3}+k\right)}}_{\leq 2^{\frac{2n}{3}-\frac{j}{6}+\frac{j}{2}} \cdot \frac{1}{\sqrt{2}-1}} + \sqrt{6} \underbrace{\sum_{k=0}^{j} 2^{n-\left(\frac{n}{3}-\frac{j}{3}+k\right)}}_{\leq 2^{\frac{2n}{3}+\frac{j}{3}+1}} \\
&\leq \sqrt{6}\cdot(\sqrt{2}+3)\cdot 2^{\frac{2n}{3}+\frac{j}{3}} \leq 11 \cdot 2^{\frac{2n}{3}+\frac{j}{3}}
\end{aligned}
$$

which implies that

$$e^3 \leq 11^3 \cdot 2^{2n+j} = 11^3 \cdot N^2 \cdot \frac{N}{d}.$$

Therefore, we can conclude that $d = \mathcal{O}\left(\left(\frac{N}{e}\right)^3\right)$. □

I.3 Analysis on DRSample

In DRSample which has been introduced in [ABH17] and is specified in Algorithm 3 in the appendix. To see how the distribution $r(v)$ is defined consider partitioning the set of all potential predecessors u into buckets $D_1^v, D_2^v, \cdots$ where $D_i^v := \{u : 2^{i-1} < \mathbf{dist}(u,v) \leq 2^i\}$ where $\mathbf{dist}(u,v) = v-u$. Intuitively, to sample $r(v)$ we first select a bucket D_i^v with $i \leq \log_2 v$ uniformly at random and then select a parent $u = r(v)$ uniformly at random from this bucket D_i^v i.e., $(r(v),v) \in D_i^v$, or equivalently, subject to the constraint that $2^{i-1} < \mathbf{dist}(r(v),v) \leq 2^i$. We remark that the pebbling attacks of Alwen and Blocki [AB16] have cost $\Theta(Ne + N\sqrt{Nd})$. If we wanted to obtain an attack with cumulative cost $o\left(N^2 \log\log N/\log N\right)$ then we would need a depth-reducing set S of size $e \le \frac{N \log\log N}{\log N}$ such that $\mathsf{depth}(G-S) \le \frac{N(\log\log N)^2}{\log^2 N}$. We show that no *known* techniques for producing depth-reducing sets will produce a set S which satisfy both criteria. We first consider an attack based on Valiant's Lemma.

Here, we consider the variant of the attack (Algorithm 8) which is guaranteed to find S s.t. $\mathsf{depth}(G-S) \le \frac{N}{\log N}$. As Theorem 15 shows that with high probability $|S| = \Omega\left(\frac{N \log\log N}{\log N}\right)$ — of course we would need to ensure that $\mathsf{depth}(G-S) \le \frac{N(\log\log N)^2}{\log^2 N} \ll \frac{N}{\log N}$ is even smaller to obtain an attack with cost $o\left(N^2 \log\log N/\log N\right)$.

Algorithm 8: An algorithm to sample a depth-reducing set.

```
Input   : DAG $G=(V(G),E(G))$ with $|V(G)|=N=2^n$, and a target size $e$.
Output  : A depth-reducing set $S$ with $\mathsf{depth}(G-S)>d$ to remove
Function Valiant2($G$, $d$):
    for $i \le n$ do
        $E_i := \{(u,v) | MSDB(u,v)=i\}$
        $S_i := \{u | (u,v) \in E_i\}$
    end
    $S := \varnothing$
    $X := \varnothing$
    while $\mathsf{depth}(G-S) > d$ do
        $i = \mathrm{argmin}_{i \notin X} |E_i|$ // Find smallest $|E_i|$ that hasn't been picked yet
        $S = S \cup S_i$
        $X = S$
    end
    return $S$
```

Theorem 15. *Let G be a randomly sampled DRSample DAG with N nodes and let $S = \mathsf{Valiant2}\left(G, \frac{N}{\sqrt{\log N}}\right)$ with $e=|S|$. Then there exist constants c_1 and c_2 such that with high probability, $\frac{c_1 N \log\log N}{\log N} \le e \le \frac{c_2 N \log\log N}{\log N}$.*

Proof. Similar to Argon2i-A and B, we have that $|A_i| = \frac{N}{2^i}$. As stated before, from Lemma 11 we have that

$$\Pr[\mathrm{MSDB}(\ell_{r(v)}, \ell_v) = i] = \Pr[2^{i-1} + m \ge v - r(v) > m].$$

Now observe that if m is large, i.e., $m = 2^{i-1} - 1$, then there is up to only one bucket to select which satisfies the inequality $2^{i-1} + m \ge v - r(v) > m$. Similarly, if $m = 2^{i-j} + 1$, then there are totally up to j buckets to select which satisfies the inequality. Taken together with the assumption that $v \ge \sqrt{N}$, we can compute the expectation

$$\mathbb{E}[|B_i|] \le \sum_{j=0}^{i-2} 2^{j-i+1} \cdot N \cdot \frac{i-j}{\log v} \le \sum_{j=0}^{i-2} 2^{j-i+1} \cdot N \cdot \frac{i-j}{\frac{1}{2}\log N}$$

$$=\frac{2N}{n}\sum_{j=0}^{i-2}(i-j)2^{j-i+1}=\frac{2N}{n}\cdot\underbrace{\frac{3\cdot 2^{i-1}-2i-2}{2^{i-1}}}_{\leq 3}\leq\frac{6N}{n}$$

Therefore, we can argue that with high probability, we have $|B_i|\leq\frac{12N}{n}$. Taken together, for $i\leq\log_2 N$, with high probability we have

$$|E_i|\leq\frac{12N}{n}+\frac{N}{2^i}.$$

Our goal is to find j smallest sets to delete to make smallest E_i when $d=N/2^j$. We can achieve this when $i=n$. Then we will delete all sets in the interval $[n-j+1,n]$. Hence, the total number of deleting edges is

$$\begin{aligned}e\leq\sum_{k=n-j+1}^{n}|E_k|&\leq\sum_{k=n-j+1}^{n}\left(\frac{12N}{n}+\frac{N}{2^k}\right)\\&\leq j\cdot\frac{12N}{n}+N\cdot 2^{-n+j}\\&=\log_2\left(\frac{N}{d}\right)\cdot\frac{12N}{n}+2^j\\&=\log_2\left(\frac{N}{d}\right)\cdot\frac{12N}{n}+\frac{N}{d}.\end{aligned}$$

Putting $d=\frac{N}{\sqrt{n}}=\frac{N}{\sqrt{\log N}}$, we have that

$$\begin{aligned}e&\leq\log_2\sqrt{n}\cdot\frac{12N}{n}+\sqrt{n}\\&\leq c\cdot\frac{N\log\log N}{\log N}\end{aligned}$$

for some constant $c>0$.

Now, in terms of the lower bound, one can observe that the number of bucket in each case is lower bounded by $j-1$ buckets if $m=2^{i-j}+1$. Hence, we have that

$$\begin{aligned}\mathbb{E}[|B_i|]&\geq\sum_{j=0}^{i-2}2^{j-i+1}\cdot N\cdot\frac{i-j-1}{\log v}\geq\sum_{j=0}^{i-2}2^{j-i+1}\cdot N\cdot\frac{i-j-1}{\log N}\\&=\frac{N}{n}\sum_{j=0}^{i-2}(i-j-1)2^{j-i+1}=\frac{N}{n}\cdot\underbrace{\frac{\cdot 2^i-i-1}{2^{i-1}}}_{\geq 1/2}\geq\frac{N}{2n}\end{aligned}$$

which, by Corollary 7, leads to

$$|E_i|\geq\frac{N}{4n}+\frac{N}{2^i}$$

with high probability since $\mathbb{E}[|B_i|]=\frac{N}{2n}\gg\log^2 N$. Again, our goal is to find j smallest sets to delete to make smallest E_i when $d=N/2^j$. We can achieve this when $i=n$. Then we will delete all sets in the interval $[n-j+1,n]$. Hence, the total number of deleting edges is

$$\begin{aligned}
e &\geq \sum_{k=n-j+1}^{n} |E_k| \geq \sum_{k=n-j+1}^{n} \left(\frac{N}{4n}+\frac{N}{2^k}\right)\\
&\geq j\cdot\frac{N}{4n}+N\cdot\sum_{k=0}^{j-1} 2^{-n+j-1-k}\\
&=\log_2\left(\frac{N}{d}\right)\cdot\frac{N}{4n}+(2^j-1)\\
&=\log_2\left(\frac{N}{d}\right)\cdot\frac{N}{4n}+\frac{N}{d}-1.
\end{aligned}$$

Putting $d=\frac{N}{\sqrt{n}}=\frac{N}{\sqrt{\log N}}$, we have that

$$\begin{aligned}
e &\geq \log_2\sqrt{n}\cdot\frac{N}{4n}+\sqrt{n}-1\\
&\geq \log_2\sqrt{n}\cdot\frac{N}{4n}=\frac{1}{8}\cdot\frac{N\log\log N}{\log N}\ .
\end{aligned}$$

Hence, one can conclude that Valient's lemma fails to do better than $e=\Omega\left(\frac{N\log\log N}{\log N}\right)$ even when the target depth is just $d=\frac{N}{\sqrt{n}}$. □

Therefore, we can safely conclude that DRSample is optimally resistant to Valient's Lemma.

I.4 Layered Attack against DRSample

Next we consider the layered attack of Alwen and Blocki [AB16] for constructing depth-reducing sets and show that it fails to produce a set S of size $e\leq\frac{N\log\log N}{\log N}$ s.t. $\mathsf{depth}(G-S)\leq\frac{N(\log\log N)^2}{\log^2 N}$ as required to obtain a pebbling attack with cumulative cost at most $eN+N\sqrt{Nd}=o\left(\frac{N^2\log\log N}{\log N}\right)$. In fact, Lemma 14 and Corollary 8 show that the layered attack fails to produce such an effective depth-reducing set S.

Before introducing Lemma 14, define an algorithm which samples the depth-reducing set from layered attack (Algorithm 9).

Lemma 14. *Let G be a randomly sampled DRSample DAG with N nodes, $\lambda,g>0$ be given such that $\lambda\log\lambda>N$, and $S=\mathsf{Layered}(G,\lambda,g)$. Then with high probability,*

$$\frac{N}{g}+\frac{N\log(N/2\lambda)}{4\log\lambda}\leq|S|\leq\frac{N}{g}+\frac{8N\log(N/\lambda)}{\log\lambda}.$$

Algorithm 9: An algorithm to sample the depth-reducing set from layered attack.

```
Input   : DAG $G=(V(G)=[N],E(G))$ with $|V(G)|=N=2^n$ and
          $E(G)=\cup_{i=3}^{N}\{(i-1,i),(r(i),i)\}\cup\{(1,2)\}$, the number of layer $\lambda$, and a gap $g$.
Output  : A depth-reducing set $S$ to remove
Function Layered($G$, $\lambda$, $g$):
    $S_1:=\{g,2g,3g,\cdots\}$ $S:=V(G)$
    for $i=1$ to $\lambda$ do
        $L_i:=\{k\in\mathbb{Z} | (i-1)\lceil\frac{N}{\lambda}\rceil<k\leq i\lceil\frac{N}{\lambda}\rceil\}$
        $E_i:=\{v\in L_i$ s.t. $r(v)\in L_i\}$
    end
    $S_2:=\cup_{i=1}^{\lambda}E_i$
    return $S=S_1\cup S_2$
```

Proof. The probability that the predecessor of the node v is in the same layer has the following upper and lower bound. Note that we could get the lower bound by only considering the case that v lies in the second half of the layer.

$$\frac{\log(N/2\lambda)}{\log(iN/\lambda)}\leq \Pr[r(v) \text{ in the same layer}]\leq \frac{\log(N/\lambda)}{\log(iN/\lambda)}\leq\frac{\log(N/\lambda)}{\log i}.$$

Then we have

$$\begin{aligned}\mathsf{depth}(G-S)&=\lambda\cdot g \quad \text{and}\\ |S|&\leq \frac{N}{g}+\frac{N}{\lambda}\sum_{i=2}^{\lambda}\frac{\log(N/\lambda)}{\log i}\\ &\leq \frac{N}{g}+\frac{N}{\lambda}\cdot 8\cdot\frac{\lambda}{\log\lambda}\cdot\log\left(\frac{N}{\lambda}\right) && \triangleleft\text{Theorem 17}\\ &=\frac{N}{g}+\frac{8N\log(N/\lambda)}{\log\lambda}\end{aligned}$$

Now, when it comes to lower bounds, from the condition $\lambda\log\lambda>N$, we have that $\lambda^2>N$ which is equivalent to $\lambda>N/\lambda$. Hence, this leads to

$$\begin{aligned}|S|&\geq\frac{N}{g}+\frac{N}{\lambda}\sum_{i=2}^{\lambda}\frac{\log(N/2\lambda)}{\log(iN/\lambda)}\\ &\geq\frac{N}{g}+\frac{N}{\lambda}\sum_{i=2}^{\lambda}\frac{\log(N/2\lambda)}{\log(i\lambda)}\\ &=\frac{N}{g}+\frac{\lambda-1}{\lambda}\cdot\frac{N\log(N/2\lambda)}{2\log\lambda}\\ &\geq\frac{N}{g}+\frac{N\log(N/2\lambda)}{4\log\lambda}\end{aligned}$$

which proves the lemma. □

Corollary 8 demonstrates demonstrates that the layered attack fails to produce an effective depth-reducing set to obtain a pebbling with cost $o\left(\frac{N^2\log\log N}{\log N}\right)$ for *any* settings of the parameters λ and g.

Corollary 8. *Let G be a randomly sampled DRSample DAG with N nodes. Then with high probability for all $\lambda,g>0$ s.t. $\lambda g \le \frac{N}{\sqrt{\log N}}$ we have $|S| \ge \frac{N\log\log N}{8\log N}$ where S is the depth reducing set generated by the layered attack with parameters λ and g.*

Proof. By Lemma 14, with high probability for any λ,g we have

$$|S| \ge \frac{N}{g} + \frac{N\log(N/2\lambda)}{4\log\lambda} \ .$$

If we want $|S| < \frac{N\log\log N}{\log N}$ then we must have $g \ge \frac{\log N}{\log\log N}$ due to the $\frac{N}{g}$ term. To ensure that $\mathsf{depth}(G-S) \le \frac{N}{\sqrt{\log N}}$ we also require that $\lambda g \le \frac{N}{\sqrt{\log N}}$. Thus, we have $\lambda \le \frac{N\log\log N}{\log^{1.5} N}$. But this implies that

$$|S| \ge \frac{N\log(N/2\lambda)}{4\log\lambda} \ge \frac{N\log\frac{\log^{1.5}N}{2\log\log N}}{4\log\left(\frac{N\log\log N}{\log^{1.5}N}\right)} \ge \frac{N\log\log N}{8\log N} \ .$$ □

Corollary 9 shows that the layered attack and Valiant's Lemma generate comparable sizes of the depth reducing set when we have target depth $d = \frac{N}{\sqrt{\log N}}$ despite generating the depth-reducing sets in very different ways. In fact, for any constant $c>0$ we could also achieve target depth $N/\log^c N$ with $|S| = \mathcal{O}(N\log\log N/\log N)$.

Corollary 9. *Let G be a randomly sampled DRSample DAG with N nodes. Then there exist $\lambda,g>0$ such that the layered attack on yields a depth-reducing set S of size $|S| = \mathcal{O}\left(\frac{N\log\log N}{\log N}\right)$ s.t. $\mathsf{depth}(G-S) \le \frac{N}{\sqrt{\log N}}$.*

Proof. Let $g = \log N$ and $\lambda = N/\log^2 N$. Then by Lemma 14 we have $|S| = \mathcal{O}\left(\frac{N\log\log N}{\log N}\right)$ and $\mathsf{depth}(G-S) = \frac{N}{\log N}$. □

I.5 Analysis of GreedyPebble Attack along with [ABP17]

Alwen et al. [ABP17] proved that any DAG G that is (e,d)-depth robust has cumulative pebbling cost at least $\Pi_{cc}^{\parallel}(G) > ed$. Their argument was by contradiction. In particular, for any target depth $d>0$ they show how to transform *any* legal pebbling $P \in \mathcal{P}^{\parallel}(G)$ into a depth-reducing set S of size *at most* $|S| \le \Pi_{cc}^{\parallel}(P)/d$ s.t. $\mathsf{depth}(G-S) \le d$. Thus, one natural approach to construct a depth-reducing set would be to find an efficient pebbling $P \in \mathcal{P}^{\parallel}(G)$ and this transformation to yield S. We focus on the Greedy Pebbling of DRSample since this is the most-effective pebbling of the DAG that is known. Once again, if we set our target depth $d = \frac{N}{\log N}$ we can show that with high probability the size of our depth-reducing set S is $e=\Theta\left(\frac{N\log\log N}{\log N}\right)$. Thus, the transformation matches the performance of the layered attack and Valiant's lemma, but does *not* yield a sufficiently small set to obtain a depth-reducing attack [AB16] with cost $o\left(\frac{N\log\log N}{\log N}\right)$.

Recall that the GreedyPebble configuration is $GP(G)=(P_1,\cdots,P_n)\in\mathcal{P}^{\|}(G)$ where $P_i=\{i\}\cup\{j \text{ s.t. } \mathsf{gc}(j)>i\}$. Here $\mathsf{gc}(j)=\max\{v : j\in\mathsf{parents}(v)\}$. Let $S_i=P_i\cup P_{i+d}\cup P_{i+2d}\cup\cdots\cup P_{i+kd}\cup\cdots$ for $i<d$ and consider the interval $I_k=[i+(k-1)d, i+kd]$. We can observe that if $\mathsf{gc}(v)\in I_k$, then v will be discarded before reaching P_{i+kd}. Therefore, we have that $S_i=\cup_k\{i+kd\}\cup\{v|\mathsf{gc}(v)-v>d-m_v\}$ where m_v denotes the distance between $i+(k-1)d$ and v. We provide such algorithm to sample the minimum depth-reducing set in Algorithm 10.

Algorithm 10: An algorithm to sample the minimum depth-reducing set from greedypebble.

```
Input  : DAG $G=(V(G),E(G))$ with $|V(G)|=N=2^n$, and a target depth $d$.
Output : A depth-reducing set $S$ to remove
Function GPDR($G$, $d$):
    $P:=\mathsf{GP}(G)$                 // A legal pebbling $P$ from Algorithm 1.
    $S:=V(G)$
    for $i=0$ to $d-1$ do
        $S_i:=P_i\cup P_{i+d}\cup P_{i+2d}\cup\cdots$
        if $|S_i|<|S|$ then
            $S=S_i$
    end
    return $S$
```

Theorem 16. *Let G be a randomly sampled DRSample DAG with N nodes and let $S=\mathsf{GPDR}(G,\frac{N}{\sqrt{\log N}})$. Then there exists a constant $c_1,c_2>0$ such that with high probability, $\frac{c_1 N\log\log N}{\log N}\le|S|\le\frac{c_2 N\log\log N}{\log N}$.*

Proof. Let Y_v be the random variable representing the event that $v-r(v)>m_v$ where $r(v)$ is the predecessor of v. That is, we have that

$$Y_v=\begin{cases}1 & \text{if } v-r(v)>m_v\\ 0 & \text{otherwise}\end{cases}$$

Similarly, define the random variable Z_v as follows:

$$Z_v=\begin{cases}1 & \text{if } Y_v=1 \text{ and } \mathsf{gc}(r(v))=v\\ 0 & \text{otherwise}\end{cases}$$

Then we have that the expectation of the size of the set $\{v|\mathsf{gc}(v)-v>d-m_v\}$ equals to the sum of the value $\mathbb{E}[Z_v]$ over the nodes v, which leads to

$$\mathbb{E}[|S_i|]=\frac{N}{d}+\sum_v\mathbb{E}[Z_v].$$

In DRSample, when v lies between the interval $[u-2^{k+1}+1,u-2^k+1]$, the size of the bucket is 2^k and the probability that $r(u)=v$ is $\frac{1}{\log u}\cdot\frac{1}{2^k}$. Moreover, the fact that $u-2^{k+1}+1\le v\le u-2^k+1$ implies $\frac{1}{2^k}\le\frac{2}{u-v}$. Taken together, we have

$$\Pr[r(u)=v]=\frac{1}{\log u}\cdot\frac{1}{2^k}\le\frac{1}{\log u}\cdot\frac{2}{u-v}.$$

With the choice of $d=\frac{N}{\sqrt{\log N}}$, we would get

$$\begin{aligned}
\Pr\left[\exists x>u+\frac{d}{2}\text{ s.t. } r(x)=u\right] &\le \sum_{x>u+\frac{d}{2}}^{N}\frac{2}{x-u}\cdot\frac{1}{\log x}\\
&\le \frac{2}{\log\left(u+\frac{d}{2}\right)}\sum_{x>u+\frac{d}{2}}^{N}\frac{1}{x-u}\\
&\le \frac{2}{\log\left(u+\frac{d}{2}\right)}\cdot\ln\left[\frac{2(N-u)}{d}\right]\\
&\le \frac{2}{\log\left(u+\frac{N}{2\sqrt{\log N}}\right)}\cdot\ln\left[2\sqrt{\log N}\right]\\
&\le \frac{2}{\log\left(\frac{N}{2\sqrt{\log N}}\right)}\cdot\log\log N\\
&\le \frac{4}{\log N}\cdot\log\log N=\mathcal{O}\left(\frac{\log\log N}{\log N}\right)
\end{aligned}$$

because $\log N\ge 8\log\log N$ for large N. Therefore, we have

$$\Pr[Z_v|Y_v]\ge 1-\frac{4}{\log N}\cdot\log\log N\ge\frac{1}{2}$$

and

$$\begin{aligned}
\mathbb{E}[|S_i|] &= \frac{N}{d}+\sum_v\mathbb{E}[Z_v]\\
&\ge \frac{N}{d}+\left(\frac{N}{4}\right)\cdot\frac{1}{2}\cdot\Pr\left[v-r(v)>d\,\middle|\,v>\frac{N}{2}\right]\\
&\ge \frac{N}{d}+\left(\frac{N}{4}\right)\cdot\frac{1}{2}\cdot\frac{\log v-\log d}{\log v}\\
&\ge \frac{N}{d}+\left(\frac{N}{4}\right)\cdot\frac{1}{4}\cdot\frac{\log\log N}{\log(N/2)}\ge\Omega\left(\frac{N\log\log N}{\log N}\right).
\end{aligned}$$

Now, we can simply get the upper bound by replacing Z_v by Y_v if we assume that $\mathsf{gc}(r(v))=v$ always happens in the best scenario. Assuming that $Y_v=1$ for every

$v \leq \frac{N}{\log N}$, we have

$$\begin{aligned}\mathbb{E}[|S_i|] &\leq \frac{N}{d} + \sum_v \mathbb{E}[Y_v] \\ &\leq \frac{N}{d} + \frac{N}{\log N} + \underbrace{\sum_{v > N/\log N} \mathbb{E}[Y_v]}_{(1)}.\end{aligned}$$

Considering (1), split the interval with length d and consider the case $v \in [x, x+d]$. If $x + \frac{d}{2^i} < v \leq x + \frac{d}{2^{i-1}}$, then at least $\log(d/2^i)$ buckets overlap $[x, x+d]$. If b_v denotes the total number of buckets before v, then we have

$$\mathbb{E}[Y_v] \leq \frac{b_v - \log(d/2^i)}{b_v} = 1 - \frac{\log(d/2^i)}{b_v} \leq 1 - \frac{\log(d/2^i)}{\log N}$$

when $v \in \left(x + \frac{d}{2^i}, x + \frac{d}{2^{i-1}}\right]$. Since we have $\frac{d}{2^i}$ such v's and we have at most $\frac{N}{d}$ such $[x, x+d]$'s, with the choice of $d = \frac{N}{\sqrt{\log N}}$, we would get

$$\begin{aligned}(1) = \sum_{v > N/\log N} \mathbb{E}[Y_v] &\leq \frac{N}{d} \sum_{i=1}^{\infty} \frac{d}{2^i} \cdot \left(1 - \frac{\log(d/2^i)}{\log N}\right) \\ &= N \left[\sum_{i=1}^{\infty} \frac{1}{2^i} - \sum_{i=1}^{\infty} \frac{\log N - \frac{1}{2}\log\log N - i}{2^i \log N}\right] \\ &= N \left[\frac{1}{2} \frac{\log\log N}{\log N} \sum_{i=1}^{\infty} \frac{1}{2^i} - \frac{1}{\log N} \left(\sum_{i=1}^{\infty} \frac{i}{2^i}\right)\right] \\ &= \frac{N \log\log N}{2 \log N} - \frac{2N}{\log N} = \mathcal{O}\left(\frac{N \log\log N}{\log N}\right).\end{aligned}$$

Taken together, we finally have

$$\begin{aligned}\mathbb{E}[|S_i|] &\leq \frac{N}{d} + \frac{N}{\log N} + \sum_{v > N/\log N} \mathbb{E}[Y_v] \\ &= \log N + \frac{N}{\log N} + \mathcal{O}\left(\frac{N \log\log N}{\log N}\right) \\ &= \mathcal{O}\left(\frac{N \log\log N}{\log N}\right).\end{aligned}$$

Therefore, we can conclude that $\mathbb{E}[|S_i|] = \Theta\left(\frac{N \log\log N}{\log N}\right)$ and the rest follows from the Chernoff bound argument. $\square$

Remark 4. If $d = \frac{N}{\sqrt{\log N}} \gg \frac{N \log\log N}{\log N}$, we still have that $\mathbb{E}[|S_i|] \geq \Omega\left(\frac{N \log\log N}{\log N}\right)$, which implies that we cannot remove below $\Omega\left(\frac{N \log\log N}{\log N}\right)$ nodes to achieve that target depth.

J The Summation of 1/log

In this section we prove some useful facts for our cryptanalysis of depth-reducing attacks.

Lemma 15. *For* $s \geq 1$*, we have* $\sum_{i=1}^{s} \frac{2^i}{i} \leq 4 \cdot \frac{2^s}{s}$.

Proof. We can prove this by induction:

- **(Base Case)** $\frac{2}{1} \leq 4 \cdot \frac{2}{1}$ when $s=1$ and $\frac{2}{1}+\frac{2^2}{2}=4\leq 8=4\cdot\frac{2^2}{2}$ when $s=2$.
- **(Induction Hypothesis)** Suppose that $\sum_{i=1}^{t} \frac{2^i}{i} \leq 4 \cdot \frac{2^t}{t}$ for $t \geq 2$.
- **(Induction Step)** Then we have

$$\begin{aligned}\sum_{i=1}^{t+1}\frac{2^i}{i} &= \sum_{i=1}^{t}\frac{2^i}{i}+\frac{2^{t+1}}{t+1}\\ &\leq \underbrace{4\cdot\frac{2^t}{t}+\frac{2^{t+1}}{t+1}\leq 4\cdot\frac{2^{t+1}}{t+1}}_{(1)}\end{aligned}$$

because we have

$$\begin{aligned}(1) &\Longleftrightarrow 4\cdot\frac{2^t}{t}\leq 3\cdot\frac{2^{t+1}}{t+1}\\ &\Longleftrightarrow 4\cdot 2^t(t+1)\leq 3t\cdot 2^{t+1}\\ &\Longleftrightarrow 4t\cdot 2^t+4\cdot 2^t\leq 6t\cdot 2^t\\ &\Longleftrightarrow 4\cdot 2^t\leq 2t\cdot 2^t\\ &\Longleftrightarrow 2\leq t.\end{aligned}$$

□

Lemma 16. *For* $s \geq 1$*, we have* $\sum_{i=2}^{2^s} \frac{1}{\log i} \leq 4 \cdot \frac{2^s}{\log 2^s}$.

Proof. By Lemma 15 we have

$$\begin{aligned}\sum_{i=2}^{2^s}\frac{1}{\log i} &= \left[\frac{1}{\log 2}+\frac{1}{\log 3}\right]+\left[\frac{1}{\log 4}+\cdots+\frac{1}{\log 7}\right]+\\ &\qquad \cdots+\left[\frac{1}{\log 2^{s-1}}+\cdots+\frac{1}{\log(2^s-1)}\right]+\frac{1}{\log 2^s}\\ &\leq \frac{2}{\log 2}+\frac{2^2}{\log 2^2}+\cdots+\frac{2^{s-1}}{\log 2^{s-1}}+\frac{1}{\log 2^s}\\ &= \frac{1}{\log 2}\left[\sum_{i=1}^{s-1}\frac{2^i}{i}+\frac{1}{s}\right]\leq\frac{1}{\log 2}\sum_{i=1}^{s}\frac{2^i}{i}\leq\frac{1}{\log 2}\cdot 4\cdot\frac{2^s}{s}=4\cdot\frac{2^s}{\log 2^s}.\end{aligned}$$

□

Lemma 17. *For* $t \geq 2$*, we have* $\sum_{i=2}^{t} \frac{1}{\log i} \leq 8 \cdot \frac{t}{\log t}$.

Proof. We can write $t=2^s+k$ where $0\leq k<2^s$ and $s\geq 1$. With the similar technique from Lemma 16, we have

$$\begin{aligned}\sum_{i=2}^{t}\frac{1}{\log i}&=\left[\frac{1}{\log 2}+\frac{1}{\log 3}\right]+\\&\qquad\cdots+\left[\frac{1}{\log 2^{s-1}}+\cdots+\frac{1}{\log(2^s-1)}\right]+\left[\frac{1}{\log 2^s}+\cdots+\frac{1}{\log(2^s+k)}\right]\\&\leq\frac{2}{\log 2}+\frac{2^2}{\log 2^2}+\cdots+\frac{2^{s-1}}{\log 2^{s-1}}+\frac{k+1}{\log 2^s}\\&\leq\frac{1}{\log 2}\left[\sum_{i=1}^{s-1}\frac{2^i}{i}+\frac{k+1}{s}\right]\leq\frac{1}{\log 2}\sum_{i=1}^{s}\frac{2^i}{i}\\&\leq\frac{1}{\log 2}\cdot 4\cdot\frac{2^s}{s}=\frac{2^s}{\log 2}\cdot\frac{4}{s}\\&\leq\frac{2^s}{\log 2}\cdot\frac{8}{s+1}=8\cdot\frac{2^s}{\log 2^{s+1}}=8\cdot\frac{2^s}{\log(2^s+2^s)}\leq 8\cdot\frac{2^s+k}{\log(2^s+k)}\\&=8\cdot\frac{t}{\log t}.\end{aligned}$$

□

Theorem 17. *For $t\geq 2$, we have $\frac{1}{2}\cdot\frac{t}{\log t}\leq\sum_{i=2}^{t}\frac{1}{\log i}\leq 8\cdot\frac{t}{\log t}$.*

Proof. The second inequality comes directly from Lemma 17. Now we have

$$\sum_{i=2}^{t}\frac{1}{\log i}\geq\frac{t-1}{\log t}\geq\frac{t/2}{\log t}=\frac{1}{2}\cdot\frac{t}{\log t}$$

for $t\geq 2$. □