Bandwidth-Hard Functions: Reductions and Lower Bounds

Jeremiah Blocki[1], Ling Ren[2], and Samson Zhou[1]

[1] Department of Computer Science, Purdue University, West Lafayette, IN.
jblocki@purdue.edu, samsonzhou@gmail.com
[2] CSAIL, MIT, Cambridge, MA.,
renling@mit.edu

Abstract. Memory Hard Functions (MHFs) have been proposed as an answer to the growing inequality between the computational speed of general purpose CPUs and Application Specific Integrated Circuits (ASICs). MHFs have seen widespread applications including password hashing, key stretching and proofs of work. Several metrics have been proposed to quantify the 'memory hardness' of a function. Cumulative memory complexity (CMC) [AS15] (or amortized Area $\times$ Time complexity [ABH17]) attempts to quantify the amortized cost to acquire/build the hardware to evaluate the function — amortized by the number of instances of the function that can be evaluated of this hardware. By contrast, bandwidth hardness [RD17] attempts to quantify the amortized energy costs of evaluating this function on hardware — which in turn is largely dominated by the number of cache misses. Ideally, a good MHF would be both bandwidth hard and have high cumulative memory complexity. While the cumulative memory complexity of leading MHF candidates is well understood, little is known about the *bandwidth hardness* of many of the most prominent MHF candidates.

Our contributions are as follows: First, we provide the first reduction proving that, in the parallel random oracle model, the bandwidth hardness of a Data-Independent Memory Hard Function (iMHF) is described by the red-blue pebbling cost of the directed acyclic graph (DAG) associated with that iMHF. Second, we show that the goals of designing an MHF with high CMC/bandwidth hardness are well aligned. In particular, we prove that *any* function with high CMC also has relatively high bandwidth costs. This result leads to the first *unconditional* lower bound on the bandwidth cost of scrypt in the parallel random oracle model. Third, we analyze the bandwidth hardness of several prominent iMHF candidates such as Argon2i [BDK15], winner of the password hashing competition, aATSample and DRSample [ABH17] — the first practical iMHF with asymptotically optimal CMC. More specifically, we show that Argon2i is maximally bandwidth hard as long as the cache-size m is at most $m \in \mathcal{O}\left(n^{2/3-\epsilon}\right)$ where n is the total number of data-labels produced during computation. We also show that aATSample and DRSample are maximally bandwidth hard as long as the cache-size is $m \in \mathcal{O}\left(n^{1-\epsilon}\right)$. Finally, we show that the problem of finding a red-blue pebbling with minimum bandwidth cost is NP-hard.

1 Introduction

Memory Hard Functions (MHFs) [Per09,ABMW05] are a crucial building block in the design of moderately hard key-derivation function and in the design of egalitarian proof of work schemes [DN92,Bac02]. For example, in password hashing it is critically important to ensure that it is moderately expensive for an offline attacker to check each password — even if the attacker uses customized hardware. The development of Application Specific Integrated Circuits (ASICs) or Field Programmable Gate Arrays (FPGAs) for computing cryptographic hash functions such as SHA256 make this goal increasingly challenging. In fact, Blocki *et al.* [BHZ18] recently argued that key derivation functions based on hash iteration (e.g., PBKDF2-SHA256) *cannot* provide sufficient protection against a rational (economically motivated) offline attacker without introducing unacceptably long authentication delays. The Antminer S9, an ASIC customized for bitcoin mining Bitcoins [Nak08], is able to compute SHA256 hash values at a rate of $13.6TH/s$ (approximately 14 trillion hashes per second) using just 1274 Joules of energy per second (Watts). By contrast, the energy needed to compute SHA256 14 trillion times on a standard CPU would be about six orders of magnitude higher!

MHFs are based on the observation that memory costs (e.g., latency, bandwidth, energy consumption) tend to be equitable across different architectures. Thus, to develop an "egalitarian" function we want to design a function where evaluation costs are dominated by memory costs. Two of the most prominent approaches to measure the "evaluation cost" of MHFs are capacity hardness [Per09,AS15] and bandwidth hardness [RD17]. Capacity hardness [Per09] seeks to quantify construction costs i.e., the building/obtaining the hardware necessary to compute the MHF amortized over the number of instances that can be evaluated over the lifetime of the device. By contrast, bandwidth hardness [RD17] seeks to quantify the energy costs per evaluation i.e., the cost of running the hardware.

Broadly speaking there are two types of MHFs: data-dependent memory hard functions (dMHFs) and data-independent memory hard functions (iMHFs). As the name suggests an iMHF induces a memory access pattern that *must be independent* of the sensitive input (e.g., password) which makes them naturally resistant to certain side channel attacks e.g., cache-timing [Ber05]. Thus, while dMHFs may be easier to construct with high capacity/bandwidth hardness these functions are potentially more vulnerable to side channel attacks.

To a large extent most of the recent cryptanalysis of MHF has focused on metrics for capacity hardness. In particular, *cumulative memory complexity* (CMC) [AS15] and the closely related metric *amortized area-time complexity* (aAT) [AB16,ABH17] aim to lower bound the cost of constructing enough chips to evaluate the function T times per year. For example, if evaluating the function one time requires us to lock up 1GB of DRAM for 1 second then, *at minimum*, an attacker would need to buy roughly 32 (1GB) DRAM chips to evaluate the function a billion times per year. Alwen *et al.* [ACP+17] showed that the dMHF `scrypt` [Per09] has maximal CMC $\Omega(n^2)$. Alwen and Blocki [AB16,AB17]

showed that *any* iMHF has cumulative memory complexity $\mathcal{O}\left(\frac{n^2 \log\log n}{\log n}\right)$ and they exhibited even stronger amortization attacks against against Password Hashing Competition [PHC] (PHC) winner Argon2i [BDK16] along with other candidate MHFs such as balloon hashing [BCS16]. Alwen *et al.* [ABP17] also gave a theoretical construction of an iMHF with CMC $\Omega\left(\frac{n^2}{\log n}\right)$, which is essentially optimal in an asymptotic sense. More recently, Alwen *et al.* [ABH17] designed two *practical* iMHFs called DRSample and aATSample with the same complexity. Blocki and Zhou [BZ17] showed that Argon2i has CMC at most $\mathcal{O}\left(n^{1.767}\right)$ and at least $\tilde{\Omega}\left(n^{1.75}\right)$.

By contrast, the notion of bandwidth-hardness was only introduced recently [RD17] with the intention of lower bounding the energy required to evaluate the function. Ren and Devadas [RD17] observed that metrics such as CMC or aAT do not provide an accurate picture of energy consumption. For example, certain types of memory consume very little energy when idle, but cache misses are costly because we must *retrieve* data from RAM. Memory Bound Functions [ABMW05] are functions whose computation *requires* a large number of cache-misses regardless of computation time. Bandwidth hardness [RD17] relaxes this restriction by requiring that any attacker who evaluates the function must either 1) incur a large number of expensive cache misses, or 2) must perform a larger (e.g., super-linear) amount of computation.

Ideally, one would hope to design a MHF that is *both* bandwidth hard and capacity hard. However, very little is known about the bandwidth hardness of *practical* MHF candidates. Ren and Devadas proposed to cryptanalyze an iMHF using a variant of the red-blue pebbling game in which red-moves (representing computation performed using data in cache) have a smaller cost c_r than blue-moves c_b (representing data-values being *transferred* to/from memory) [RD17] and they gave a theoretical construction of an directed acyclic graph with *provably* high red-blue pebbling costs using stacked expander graphs. However, Ren and Devedas [RD17] did not prove that the red-blue pebbling game captures bandwidth hardness in the parallel random oracle model so it was not known for sure whether or not the corresponding iMHF was maximally bandwidth hard. Similarly, Ren and Devedas [RD17] showed that `scrypt` is bandwidth hard under a restrictive assumption about the cache-architecture adopted by the attacker e.g., they need to assume data from RAM can only be retrieved in larger chunks. Prior to our work nothing was known about the bandwidth hardness of practical MHF candidates such as Argon2i [PHC,BDK16], Balloon Hashing [BCS16], DRSample and aATSample [ABH17].

Our Contributions We formalize the notion of bandwidth hardness in the parallel random oracle model and show that bandwidth hardness is captured by the red-blue pebbling game. This does for bandwidth hardness what Alwen and Serbinenko [AS15] did for CMC when they showed that CMC is captured by the parallel black pebbling game. One immediate consequence of this result is that the stacked expander iMHF is bandwidth hard [RD17].

Second, we demonstrate that CMC *lower bounds* can be used to *directly* lower bound bandwidth costs. In particular, we show that bandwidth costs are at least $\Omega\left(\sqrt{c_b c_r CMC} - c_b \cdot m\right)$. Intuitively, an attacker running in time t will pay computation costs at least $t \cdot c_r$[3] and bandwidth cost *at least* $\left(\frac{CMC}{t \cdot w} - m\right) \cdot c_b$ where m denotes the number of w-bit words that can be stored in cache. Alwen *et al.* [ACP+17] show that `scrypt` has CMC at least $\Omega(n^2 \cdot w)$. Combined with our result this implies that `scrypt` has bandwidth cost at least $\Omega\left(\sqrt{c_b c_r} n\right)$. While we would ideally hope for a lower bound of $\Omega(n \cdot c_b)$, we stress that this is the first lower bound for `scrypt` that *does not* require any restrictive assumptions about cache-architecture [RD17]. The result also demonstrates that the goals of constructing a MHF with high CMC and constructing a MHF with high bandwidth hardness are well aligned.

Third, we introduce a new technique to lower-bound the red-blue pebbling cost of a DAG and we use this new technical hammer to lower-bound the red-blue pebbling cost of several important iMHF candidates including: Argon2iB (the current version of PHC winner Argon2i [BDK16]), Argon2iA (an older version of Argon2, which is similar to balloon hashing [BCS16] [4]), DRSample and aATSample. In particular, we show that if $m = \mathcal{O}\left(n^{2/3-\epsilon}\right)$ then any PROM attacker with cache-size $m \cdot w$ bits *must* incur cost $\min\{\Omega\left(n \cdot c_b\right), \omega(n \cdot c_r)\}$. In particular, Argon2iB is maximally bandwidth hard whenever the attacker's cache size is sufficiently small. Argon2iB uses a round function with word size $w = 2^{10}$ Bytes. Assuming that we set our memory hardness parameter $n = 2^{20}$ (filling $nw = 1$GB of RAM in about 1 second according to Argon2 benchmarks [BDK16]) then $n^{2/3} \cdot w$ corresponds to a cache-size of $10MB$— our lower bounds would not apply if the attacker's cache size is larger. We prove even stronger lower bounds for DRSample and aATSample. In particular, we prove that these functions are *maximally* bandwidth hard as long as $m = \mathcal{O}\left(n^{1-\epsilon}\right)$. Interestingly, DRSample and aATSample have asymptotically higher CMC as well, which is consistent with our observation that the goals of designing a MHF with high CMC is well aligned with the goal of designing an maximally bandwidth hard function.

While we prove that DRSample, aATSample and Argon2iB are all maximally bandwidth hard in an asymptotic sense, it would be nice to gain a more precise understanding of the constant factors in these bounds. To this end it would be useful to develop an efficient algorithm to determine the minimum cost red-blue pebbling of a DAG G. However, our final result is a negative one. We show that it is NP-Hard to compute the minimum cost red-blue pebbling of a DAG G.

[3] In the parallel random oracle measure running time is measured in terms of the number of sequential queries that are made to the random oracle. Thus, there must be at least one query made to the random oracle during each time step — otherwise the round would not have been incremented.

[4] In particular, edges in Argon2iA and Balloon Hashing are both sampled from the uniform distribution.

1.1 Overview of Our Results

An iMHF $f_{G,H}$ is defined by a labeling game over a DAG G and a random oracle H. In particular, the label ℓ_v of an intermediate node v is computed as $\ell_v = H\left(v, \ell_{v_1}, \ldots, \ell_{v_{\mathsf{indeg}}}\right)$ where $v_1, \ldots, v_{\mathsf{indeg}}$ are the parents of node v in G. The output of the random oracle is the label of the final sink node. Before we provide an overview of our technical results it is necessary to first (informally) introduce the black pebbling game and the red-blue pebbling game.

Graph Pebbling

Black Pebbling. Given a directed acyclic graph (DAG) $G = (V, E)$, the goal of the (parallel) black pebbling game is to place pebbles on all sink nodes of G (not necessarily simultaneously). The game is played in rounds and we use $P_i \subseteq V$ to denote the set of currently pebbled nodes on round i. Initially all nodes are unpebbled, $P_0 = \emptyset$, and in each round $i \geq 1$ we may only include $v \in P_i$ if all of v's parents were pebbled in the previous configuration ($\mathsf{parents}(v) \subseteq P_{i-1}$) or if v was already pebbled in the last round ($v \in P_{i-1}$). In the sequential pebbling game we can place at most one new pebble on the graph in any round (i.e., $|P_i \backslash P_{i-1}| \leq 1$), but in the parallel pebbling game no such restriction applies. The space cost of the pebbling is defined to be $\max_i |P_i|$, which intuitively corresponds to minimizing the maximum space required during computation of the associated function, and relates to the space-complexity of the black-pebbling game. Gilbert *et al.* [GLT79] studied the space-complexity of the black-pebbling game and showed that this problem is $\mathsf{PSPACE} - \mathsf{Complete}$ by reducing from the truly quantified boolean formula (TQBF) problem. Given a (parallel) legal black pebbling $P_1, \ldots, P_t$ of a DAG G, we define the cumulative cost to be $|P_1| + \ldots + |P_t|$. Then we define $\Pi_{cc}(G)$ (resp. $\Pi^{\|}_{cc}(G)$) as the minimum cumulative cost of any legal sequential (resp. parallel) black pebbling of G.

Alwen and Serbinenko [AS15] show that under the parallel random oracle model (pROM) of computation, the cryptanalysis of an iMHF, under the amortized time-space metric, can be approximately reduced to the cumulative cost of a pebbling strategy. Intuitively, this quantity corresponds to the total space required during the computation of the associated function and is resilient to adversaries that use multiple iMHF instances (e.g., multiple password guesses) in parallel. Blocki and Zhou show that while the actual computation of the amortized time-space cost of iMHFs is $\mathtt{NP} - \mathtt{Hard}$ [BZ18], upper and lower bounds may be computed for several proposed candidate iMHFs [BZ17], continuing a line of active work [ABP17,AB17,ABH17].

Red-Blue Pebbling. Given a DAG $G = (V, E)$, the goal of the red-blue pebbling game [HK81] is again to place pebbles on all sink nodes of G (not necessarily simultaneously) from a starting configuration that contains no pebbles on any nodes. The game is again played in rounds, with each node possibly containing a blue pebble or a red pebble at each time step. Informally, at each time step, for any node v we can swap between a red pebble at v and a blue pebble at v (and vice versa). Each swap is called a blue move, and while there is

no limit to the number of blue moves at a single time step, they each have an associated cost c_b. Simultaneously, we may place a red pebble at a node v if all of v's parents contained red pebbles in the previous configuration. This manner of placing a new red pebble is a red move and each occurrence incurs cost c_r. We are allowed to have *at most* m (cache-size) red-pebbles on the graph at any point in time. In a sequential red-blue pebbling we are allowed to place at most one new red pebble on the graph during each round, while no such constraint applies to a parallel red-blue pebbling. Finally, there is a parameter m that denotes a threshold on the number of nodes that can contain red pebbles at each time step. The total cost of the red-blue pebbling is the sum of the costs induced by the blue moves and the red moves. We define $\mathsf{rbpeb}^{\|}(G, m)$ (resp. $\mathsf{rbpeb}(G, m)$) to be the minimum cost of *any* legal parallel (resp. sequential) red-blue pebbling of G that places at most m red-pebbles on the graph at any point in time.

Proving that the Red-Blue Pebbling Game Captures Bandwidth Hardness. We consider the variant of the red-blue pebble game proposed by Ren and Devadas [RD17] in which red moves have cost c_r and blue moves have cost c_b — note that if $c_r = 0$ then we recover the traditional goal of minimizing the number of cache misses. Ren and Devadas [RD17] proposed the adoption of red-blue pebbling to model the bandwidth-complexity of iMHFs, with the idea that red moves correspond to hash computations and blue moves correspond to (more expensive) swaps between cache and memory. However, they did not *prove* any connection between red-blue pebbling costs and the *actual* bandwidth-costs of a pROM attacker.

Our contributions are two-fold. First, we formalize the notion of bandwidth cost of a function $f_{G,H}$ in the parallel random oracle model. Second, we prove that $\mathsf{ecost}(f_{G,H})$ the bandwidth cost of $f_{G,H}$ is closely related to red-blue pebbling pebbling costs. This resolves an open question of [RD17].

Theorem 1.1. *(Informal, see Theorem 3.3.) Any* pROM *algorithm that computes* $f_{G,H}$ *has bandwidth cost at least* $\mathsf{ecost}(f_{G,H}) \in \Omega\left(\mathsf{rbpeb}^{\|}(G, 8m)\right)$.

While Theorem 3.3 is similar to a result of Alwen and Serbinenko who showed that the cumulative memory complexity of $f_{G,H}$ is captured by the black pebbling game [AS15], we stress that there are several unique challenges in our reduction. In particular, it is easier to extract a black pebbling from the execution trace of a pROM attacker since each new pebble that is placed on the graph during round i corresponds directly to a random oracle query that was made during the previous round. However, in the red-blue pebbling model only red moves correspond to random oracle queries. Intuitively, we expect that blue moves correspond to labels that are transferred to/from memory, but an attacker may encode each of these labels in an unexpected way (e.g., encryption). Thus, even if we can observe the data values being transferred to/from memory we stress that we *cannot directly infer* which labels are being transferred making it difficult to extract a legal red-blue pebbling from the execution trace.

We overcome this difficulty by allowing the red-blue pebbling to use a little bit of extra memory (e.g., if the pROM attacker has $m \cdot w$ bits of cache then the red-

blue pebbling is allowed to use $8m$ red-pebbles) and by introducing the notion of a *red-blue extension pebbling* of a legal black pebbling $P = (P_1, \ldots, P_t)$. Given a legal black pebbling extracted from the execution trace of the pROM attacker running in time t we can partition time into intervals $[t_0 = 1, t_1), [t_1, t_2), \ldots [t_{k-1}, t_k = t]$ such that 1) during each interval $[t_i, t_{i+1})$ the pROM attacker transfers *at least* mw bits from memory (at cost $m \cdot c_b$), and 2) there is a red-blue extension pebbling that makes at most $\mathcal{O}(m)$ blue moves during each interval $[t_i, t_{i+1})$.

To partition time into intervals we introduce a set $\mathbf{QueryFirst}(x, y))$ that intuitively corresponds to the data-labels that *appear first as input* to a random oracle query during the time interval $[x, y)$ *before* the label appears as the output of some random oracle query during the same interval. We then define t_1 to be the minimum pebbling round such that there exists $1 \leq j_1 < t_1$ such that $\mathbf{QueryFirst}(j_1, t_1)$ has size at least $3m$. Similarly, once $t_1 < \ldots < t_{i-1}$ have been defined we can define $t_i > t_{i-1}$ to be the minimum pebbling round such that there exists $t_i > j_i \geq t_{i-1}$ s.t. $\mathbf{QueryFirst}(j_i, t_i)$ has size at least $3m$. At the beginning of each interval $[t_i, t_{i+1})$ our red-blue extension pebbling will place red pebbles on all nodes in the set $\mathbf{QueryFirst}(j_i, t_i)$ (e.g., to "load" these values into cache). We can accomplish this legally since the extension pebbling is allowed to use up to $8m$ red-pebbles. Once we have red pebbles placed on all of these nodes the extension pebbling is able to finish this interval *without* changing any other blue nodes into red-nodes (i.e., zero cache misses).

To prove that the pROM attacker must transfer *at least* mw bits from memory during each interval we rely on extractor argument. In particular, let γ_i encode the messages transferred to/from cache during the interval $[t_i, t_{i+1})$. Our extractor will extract $3m$ labels (without querying the random oracle at these points) by simulating the pROM attacker starting with a hint. The labels we will extract correspond to the nodes in the set $\mathbf{QueryFirst}(t_i, t_{i+1}))$ The hint consists of γ_i along with other information such as the current state of the cache (at most mw bits) indices of the $4m$ labels that we want to extract ($4m \log n$ bits to encode) and the index of the first query in which each label appears as input to a random oracle query ($4m \log q$ bits to encode where q is an upper bound on number of queries made by the attacker). Since, a random oracle is incompressible the extractor's hint must have length at least $4mw$ if we expect the extractor to output $4m$ labels (i.e., $4m$ distinct random oracle outputs of length w assuming there are no hash collisions) without querying the random oracle at these points so it follows that $|\gamma_i| \geq m \cdot w$.

On the Relationship between Bandwidth Complexity and Cumulative Memory Complexity. We show that bandwidth complexity and cumulative memory complexity are intricately related concepts. If $\mathsf{rbpeb}^{\|}(G, m)$ is the minimum energy cost[5] of *any* legal *parallel* reb-blue pebbling of G with cache size m and Π_{cc} is the cumulative complexity of *sequential* black pebbling, then

Theorem 1.2.

$$\mathsf{rbpeb}^{\|}(G, m) \geq 2c_b \left(\frac{\Pi_{cc}(G)}{t} - 2m \right) + c_r t \in \Omega \left(\sqrt{c_b \cdot c_r \cdot \Pi_{cc}(G)} \right),$$

[5] In contrast to [AB17], we use energy cost to refer to bandwidth cost.

where m is the cache size, t is the number of steps in the pebbling, c_b is the cost of a blue move and c_r is the cost of a red move.

Theorem 1.2 demonstrates that the goals of designing an MHF with high cumulative complexity and high bandwidth complexity are well aligned. In fact, we use Theorem 1.2 to show that a family $\{G_n\}_{n=1}^{\infty}$ constant indegree DAGs constructed by Schnitger [Sch83] has high bandwidth costs *because* $\Pi_{cc}(G_n) \in \Omega(n^2)$ [AdRNV16]. As an intermediate step to proving Theorem 1.2 we show that $\mathsf{rbpeb}^{\parallel}(G, m) \geq \mathsf{rbpeb}(G, 2m)$. This result is interesting as it suggests that an attacker will not be able to dramatically decrease bandwidth costs by exploiting parallelism. By contrast, for *any* constant indegree DAG G it is known that the parallel cumulative pebbling cost is at most $\Pi_{cc}^{\parallel}(G) \in \mathcal{O}\left(\frac{n^2 \log\log n}{\log n}\right)$ [AB16] while it is known that $\Pi_{cc}(G_n) \in \Omega(n^2)$ for the constant indegree DAGs constructed by Schnitger [Sch83].

We also prove a similar theorem that directly relates ecost and cmc. In particular, we show that $\mathsf{ecost}(f_{G,H}) \in \Omega\left(\sqrt{c_b c_r \mathsf{cmc}(f_{G,H})} - c_b m\right)$. Crucially, this bound applies to *any* MHF not just for iMHFs. For iMHFs we could use our pebbling reduction to relate $\mathsf{ecost}(f_{G,H})$ to $\mathsf{rbpeb}^{\parallel}(G)$ and we could use [AS15] to relate $\mathsf{cmc}(f_{G,H})$ to $\Pi_{cc}(G)$, but no such pebbling reduction is known for dMHFs. Combining our result with a result of Alwen et al. [ACP+17] we obtain the following lower bound for `scrypt`: $\mathsf{ecost}(\texttt{scrypt}) \in \Omega\left(\sqrt{c_b c_r} n\right)$. While Ren and Devadas [RD17] previously proved that $\mathsf{ecost}(\texttt{scrypt}) \in \Omega(n \cdot c_b)$ their result assumes that the attacker *must* transfer data to/from cache in blocks of size w. While our lower bound is weaker it is the first *unconditional* lower bound on the bandwidth hardness of `scrypt`. We conjecture that our results could be extended to show that $\mathsf{ecost}(\texttt{scrypt}) \in \Omega(n \cdot c_b)$ without condition.

On the Bandwidth Hardness of Important iMHF Candidates. In Section 5, we provide lower bounds on the bandwidth hardness of several important iMHF candidates including Argon2iA, Argon2iB [BDK16], aATSample and DRSample [ABH17]. We use Argon2iA to refer to v1.1 and we use Argon2iB to refer to versions v1.2+ [6]. Thus, Argon2iB (the current version of Argon2i) is particularly important to cryptanalyze as it won the password hashing competition and is being considered for standardization by the Cryptography Form Research Group (CFRG) of the IRTF [BDKJ16].

aATSample and DRSample are important to study as they are the first *practical* iMHF candidate with nearly asymptotically optimal cmc. In particular, $\mathsf{cmc}(\mathsf{DRSample}) \in \Omega\left(\frac{n^2 \cdot w}{\log n}\right)$ and $\mathsf{cmc}(\mathsf{aATSample}) \in \Omega\left(\frac{n^2 \cdot w}{\log n}\right)$ [ABH17] while

[6] The specification of Argon2i has changed several times, but the only changes that affect are analysis are changes that affect the underlying DAG G. A change to the edge distribution was introduced in v1.2 where a non-uniform indexing was introduced. We use Argon2iB to refer to the version that is currently being considered for standardization by the Cryptography Form Research Group (CFRG) of the IRTF[BDKJ16].

any iMHF $f_{G,H}$ has $\mathsf{cmc}(f_{G,H}) \in \mathcal{O}\left(\frac{n^2 \cdot w \cdot \log\log n}{\log n}\right)$. For the families of graphs generated by aATSample and DRSample [ABH17] we show the following:

Theorem 1.3. *Let G be a graph generated by* aATSample*. Then there exists a constant $C > 0$ so that for all $m \leq \frac{Cn}{\log n}$, it follows that*

$$\mathsf{rbpeb}^{\parallel}(G, m) \geq \min(\Omega(n)c_b, \Omega(n \log n)c_r),$$

with high probability.

Theorem 1.4. *Let G be a graph generated by* DRSample *and $0 < \rho < 1$. Then there exists a constant $C > 0$ so that for all $m \leq Cn^{\rho}$, it follows that*

$$\mathsf{rbpeb}^{\parallel}(G, m) \geq \min\left(\Omega(n)c_b, \Omega(n^{3/2-\rho/2})c_r\right)$$

with high probability.

We provide lower bounds on the bandwidth hardness on the family of graphs generated by Argon2iB. The bounds are slightly weaker for Argon2iB in that they only hold if the attacker has cache size $m \leq Cn^{2/3-\epsilon}$.

Theorem 1.5. *Let G be a graph generated by Argon2iB. Then there exists a constant $C > 0$ so that for any $0 < \epsilon < 2/3$ and for all $m \leq Cn^{\frac{2}{3}-\epsilon}$, it follows that*

$$\mathsf{rbpeb}^{\parallel}(G, m) \geq \min(\Omega(n)c_b, \Omega(n^{5/3})c_r),$$

with high probability.

At a technical level our template to establish each of these lower bounds is similar. First, we show that the underlying DAG is "block-depth robust." Second, we show that the graph is "well dispersed." Essentially, if our block size is b, then we show that for every interval $I = [i, j] \subseteq [n/2, n]$ of $\Omega(n/b)$ nodes in the second half and *almost every* block B of b consecutive nodes in $[n/2]$ there is an edge from B to I. We then consider the pebbling interval $[t_i, t_j]$ beginning at the time t_i during which a pebble is first placed on node i and ending at the time t_j during which a pebble is first placed on node j. Because the graph is "well dispersed" we will need to place a red pebble on *at least one node* from almost every block. If the cache size is $m \in o(n/b)$ then most of these $\Omega(n/b)$ blocks will begin with no pebbles in cache. Thus, it is either the case that 1) we make $\Omega(n/b)$ blue moves, or 2) we must re-pebble *almost every* block at some point during the interval $[t_i, t_j]$. Because the DAG is block-depth robust this second step will be prohibitively expensive.

On the Computational Complexity of Minimum Cost Red-Blue Pebbling. While we can establish asymptotic lower bounds on the bandwidth cost of important iMHF candidate, one would ideally want to find the precise bandwidth cost for each function. In particular, given a graph G and a cache parameter m we would like to compute $\mathsf{rbpeb}^{\parallel}(G, m)$ precisely. However, we show in Section 6 that, unfortunately, exactly computing the red-blue pebbling cost of a DAG G is $\mathtt{NP-Hard}$, even under specific assumptions:

Theorem 1.6 (Informal). *Even for $c_b \leq 3nc_r$, the problem of determining the red-blue pebbling cost of a directed acyclic graph G is* $\mathtt{NP} - \mathtt{Hard}$.

We remark that when $c_r = 0$ the problem is PSPACE complete [Liu17]. In particular, it is PSPACE complete to decide whether or not there is a legal black pebbling of a graph G using at most m black pebbles [GLT79]. Demaine and Liu [DL17] show that even determining the minimum number of pebbles to an additive $n^{1/3-\epsilon}$ term is PSPACE-hard. If $c_r = 0$ then a black pebbling that uses at most m black pebbles corresponds to a red-blue pebbling with cost zero (no blue moves) and vice versa. However, to the best of our knowledge, when $c_r > 0$ (i.e., when we charge for computation) nothing was known about the complexity of computing $\mathsf{rbpeb}^{\|}(G, m)$. We remark that whenever $\frac{c_b}{c_r} \in \mathsf{poly}(n)$ this means that the decision problem "is $\mathsf{rbpeb}^{\|}(G, m) \leq k$" is in $\mathtt{NP}$. In particular, for a DAG with $\mathsf{indeg}(G) = \mathcal{O}(1)$ there is always a red-blue pebbling with cost *at most* $\mathcal{O}(n \cdot c_r + n \cdot c_b)$ — pebble nodes in topological order and never remove a pebble from a node. Thus, the optimal red-blue pebbling runs in time at most $t \in \mathcal{O}\left(n + \frac{n \cdot c_b}{c_r}\right)$ since any red-blue pebbling running in time t in the parallel random oracle model has cost *at least* $t \cdot c_r$.

2 Preliminaries

We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$ and $[a, b] = \{a, a+1, \ldots, b\}$ where $a, b \in \mathbb{N}$ with $a \leq b$. Similarly, we use $(a, b]$ to denote the set $[a, b] - \{a\}$.

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

An iMHF can be specified by a DAG G and a random oracle H as in the next definition.

Definition 2.1. *Given a directed acyclic graph $G = (V, E)$ with a set of sink nodes $\mathsf{sinks}(G)$ and a random oracle function $H : \Sigma^* \to \Sigma^\ell$ over an alphabet Σ,*

we define the labeling of graph G as $\mathsf{lab}_{G,H} : \Sigma^* \to \Sigma^*$. 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 (H, x) labeling of G is defined recursively by

$$\mathsf{lab}_{H,x}(v) = \begin{cases} H(v, x), & \mathsf{indeg}(v) = 0 \\ H\left(v, \mathsf{lab}_{H,x}(v_1), \ldots, \mathsf{lab}_{H,x}(v_d)\right), & \mathsf{indeg}(v) > 0, \end{cases}$$

where $v_1, \ldots, v_d$ are the parents of v in G, according to some predetermined lexicographical order. We define $f_{G,H}(x) = \{\mathsf{lab}_{H,x}(s)\}_{s \in \mathsf{sinks}(G)}$. In particular, if there is a single sink node s_G then $f_{G,H}(x) = \mathsf{lab}_{H,x}(s_G)$.

We will often consider graphs obtained from other graphs by removing subsets of nodes. Therefore if $S \subset V$, then we denote by $G - S$ the DAG obtained from G by removing nodes S and incident edges.

Given a directed acyclic graph (DAG) $G = (V, E)$ the goal of the red-blue pebbling game is to place pebbles on all sink nodes of G (not necessarily simultaneously). Let $\mathcal{RB} = ((B_0, R_0), (B_1, R_1), \ldots, (B_t, R_t))$ (resp. $\mathcal{RB}^{\|}$) denote the set of all sequential (resp. parallel) red-blue pebblings of a DAG G. The game is played in rounds and we use $B_i \subseteq V$ (resp. $R_i \subseteq V$) to denote the set of nodes with blue pebbles (resp. red pebbles) in round i. Initially, no nodes contain pebbles, so that $B_0 \cup R_0 = \emptyset$. The goal is to eventually place a red-pebble on every node in V (not-necessarily simultanesouly) so we require that $V \subseteq \bigcup_i R_i$. We also require that in every round $i > 0$ we have (1) $\mathsf{parents}\left(R_i \setminus R_{i-1}\right) \subseteq R_{i-1} \cup B_{i-1}$, (2) $\mathsf{parents}\left(R_i \setminus R_{i-1}\right) \subseteq R_{i-1} \cup B_{i-1}$ and (3) $|R_i| \leq m$ during every time step i.

We let $\mathcal{RB}^{\|}(G, m)$ be the set of all valid parallel red-blue pebblings of G with a cache-size of m pebbles. Intuitively, in each round $i \geq 1$ we may place a red pebble on a node $v \in V$ if either $\mathsf{parents}(v) \subseteq R_{i-1}$ all of v's parents contain red pebbles in the previous configuration (called a *red move*) or v contained a blue pebble in the previous round (called a *blue move*). On the other hand, we may place a blue pebble at $v \in P_i$ (also called a *blue move*) if v contained a red pebble in the previous round. Blue moves represent data transfer to/from memory and are more expensive than red-moves (computation).

We say that a pebbling $((B_0, R_0), (B_1, R_1), \ldots, (B_t, R_t)) \in \mathcal{RB}^{\|}(G, m)$ is sequential if $|R_i \setminus R_{i-1}| \leq 1$ for all $0 < i \leq t$, while for a parallel pebbling we make no such restriction.

Pebbling Costs. We use

$$\#BM_i = |\{v \in R_i \setminus R_{i-1} \ : \ \mathsf{parents}(v) \not\subset R_{i-1}\}| + |B_i \setminus B_{i-1}|$$

to denote the number of blue moves made during round i, and we use

$$\#RM_i = |R_i \setminus R_{i-1}| - |\{v \in R_i \setminus R_{i-1} \ : \ \mathsf{parents}(v) \not\subset R_{i-1}\}|$$

to denote the number of red moves during round i [7].

[7] In some cases we may have $v \in B_{i-1}$ and $\mathsf{parents}(v) \subset R_{i-1}$ so that we could place a pebble on node v using either a red move or a blue move. In such cases we will assume that this is accomplished by a red move, since blue moves will be more expensive.

Given cost parameters c_r and c_b, we define the energy cost of a red-blue pebbling $(R, B) = ((R_1, B_1), \ldots, (R_t, B_t))$ to be

$$\mathsf{rbpeb}^{\|}((R, B)) = \sum_{i=1}^{t} c_b \# BM_i + c_r \# RM_i \ .$$

Generally, it is assumed that c_r is much larger than c_b. Finally, we define

$$\mathsf{rbpeb}^{\|}(G, m) = \min_{(R,B) \in \mathcal{RB}^{\|}(G,m)} \mathsf{rbpeb}^{\|}((R, B))$$

to be the cost of the optimal red-blue pebbling of G with maximum cache-size of m red pebbles. We consider the computational complexity of computing $\mathsf{rbpeb}^{\|}(G, m)$, defining a decision version below and showing it is $\mathtt{NP-Complete}$.

The decision problem $\mathsf{rbpeb}^{\|}$ is defined as follows:
Input: a DAG G on n nodes, parameter c_b, c_r, and integers $m, d > 0$.
Output: *Yes*, if $\mathsf{rbpeb}^{\|}(G, m) \leq d$; otherwise *No*.

2.1 Depth-Robustness

The cumulative memory complexity of an iMHF is very closely related to the notion of depth-robustness [AB16,ABP17,ABH17,BZ17]. In particular, we know that $\Pi^{\|}_{cc}(G) \geq ed$ for a depth-robust DAG and that $\Pi^{\|}_{cc}(G) \in \mathcal{O}\left(en + n \cdot \sqrt{dn}\right)$. We will show that $\Pi^{\|}_{cc}(G)$ can be used to lower bound $\mathsf{rbpeb}^{\|}(G, m)$, thus depth-robustness can also be a useful tool to lower bound bandwidth costs.

Definition 2.2 (Block Depth-Robustness). *Given a node v, let $N(v, b) = \{v - b + 1, \ldots, v\}$ denote a segment of b consecutive nodes ending at v. Similarly, given a set $S \subseteq V$, let $N(S, b) = \cup_{v \in S} N(v, b)$. We say that a DAG G is (e, d, b)-block-depth-robust if for every set $S \subseteq V$ of size $|S| \leq e$, we have $\mathsf{depth}(G - N(s, b)) \geq d$. If $b = 1$, we simply say G is (e, d)-depth-robust and if G is not (e, d)-depth-robust, we say that G is (e, d)-depth-reducible.*

Note that when $b > 1$ (e, d, b)-block-depth robustness is a strictly stronger notion that (e, d)-depth-robustness since the set $N(S, b)$ of nodes that we remove may have size as large as $|N(S, b)| = eb$. Hence, $(e, d, b \geq 1)$-block depth robustness implies (e, d)-depth robustness. On the other hand, (e, d)-depth robustness only implies $(e/b, d, b)$-block depth robustness.

2.2 Metagraphs

We will also frequently use the notion of a metagraph in our analysis. For a fixed integer $m \in [n]$, let $n' = \lfloor n/m \rfloor$. For all $i \in [n']$, let $M_i = [(i-1)m + 1, im] \subseteq V$. Moreover, we denote the first and last thirds respectively of M_i with

$$M_i^F = \left[(i-1)m + 1, (i-1)m + \left\lfloor \frac{m}{3} \right\rfloor\right] \subseteq M_i \ ,$$

and

$$M_i^L = \left[(i-1)m + \left\lceil \frac{2m}{3} \right\rceil + 1, im\right] \subseteq M_i \ .$$

Given a DAG G, we call a DAG $G_m = (V_m, E_m)$ with the following properties the *metagraph* of G.

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

Edges: If the end of a meta-node M_i^L is connected to the beginning M_j^F of another meta-node we connect their simple nodes.

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

3 Modeling Energy Complexity as Red-Blue Pebbling

In this section we show that the bandwidth cost of the function $f_{G,H}$ is characterized by the reb-blue pebbling cost $\mathsf{rbpeb}^{\|}(G, m)$ in the parallel random oracle model just as Alwen and Serbinenko [AS15] showed that cumulative memory complexity can be characterized by the black pebbling game. Similar to [AS15] our reduction uses Lemma 3.1 as a core building block. In particular, if the bandwidth cost is significantly smaller than $\mathsf{rbpeb}^{\|}(G, 8m)$ for a pROM attacker with $m \cdot w$ bits of cache then we will be able to build an extractor that receives a small hint and predicts the random oracle output on a larger set of indices contradicting Lemma 3.1. One of the unique challenges we face when designing our extractor is that it is not obvious how to relate messages between cache and main memory to specific blue pebbling moves. By contrast, a black pebbling move always corresponds to a specific random oracle query.

Lemma 3.1. *[DKW11b] 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 $\frac{|H|}{2^k}$.*

3.1 Memory and Cache in the Parallel Random Oracle Model

Before we present our reduction it is first necessary to give a formal definition of bandwidth costs in the pROM model.

We define a state of an algorithm $\mathcal{A}^{H(\cdot)}$ to be the tuple (σ, ξ), where σ contains the contents of the cache and has size at most mw bits, to store at most m labels, and ξ contains the contents of the memory. We consider a pROM attacker $\mathcal{A}^{H(\cdot)}$ with cache size $m \cdot w$ who is given oracle access to a random oracle $H : \{0,1\}^* \to \{0,1\}^w$. In particular, the cache is large enough to store m labels. An execution of $\mathcal{A}^{H(\cdot)}$ on input x proceeds in rounds as follows. Initially, the state at time 0 is (σ_0, ξ_0) where ξ_0 is empty and σ_0 encodes the initial input x. At the

beginning of round i the attacker is given the initial state $(\sigma_{i-1}, \xi_{i-1})$ 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/or transfer data between memory and cache. The round ends when the attacker outputs a new state (σ_i, ξ_i) along with a batch of queries $Q_i = \{q_1^i, q_2^i, \ldots, q_i^{k_i}\}$. Since the attacker only has cache-size $m \cdot w$ we only allow the attacker to make *at most* $|Q_i| \leq m$ queries during a single step (otherwise the attacker won't even have room to store all of the responses).

Message Passing. We allow the attacker to specify *arbitrary* functions F_1, F_2, F_3 and F_4 for communication between cache and memory during each round so long as the specification of each function is independent of the random oracle H (e.g., we cannot query to the random oracle in between rounds). In particular, the function $F_1(\sigma_{i-1}, A_{i-1}) = r_i^1$ is used to specify the *first* message we will send to memory during round i — in the event that we don't send any message to memory we define $F_1(\sigma_{i-1}) = \perp$. Similarly, the function $F_2(\xi_{i-1}, r_i^1) = s_i^1$ specifies the response from memory (or $\perp$ if there is no response). One $r_i^1, s_i^1, \ldots, r_i^{j-1}, s_i^{j-1}$ have been defined we set $r_i^j = F_1\left(\sigma_{i-1}, A_{i-1}, r_i^1, s_i^1, \ldots, r_i^{j-1}, s_i^{j-1}\right)$ and $r_i^j = F_2\left(\xi_{i-1}, r_i^1, s_i^1, \ldots, r_i^{j-1}, s_i^{j-1}, r_i^j\right)$. We terminate when $r_i^j = \perp$ or when $s_i^j = \perp$. We let $R_i = \{r_i^1, r_i^2, \ldots, r_i^{\ell_i}\}$ denote the sequence of messages sent from cache to memory during round i and we let $S_i = \{s_i^1, s_i^2, \ldots, s_i^{\ell_i}\}$ denote the responses send from memory back to the cache. Finally, the round ends when the attacker uses the function $F_3(\xi_{i-1}, R_i, S_i) = \xi_i$ to output a new state ξ_i for memory and $F_4(\sigma_{i-1}, R_i, S_i)$ to ouput a new state σ_i for cache and a new batch Q_i of at most m random oracle queries. At this point $\mathcal{A}^{H(\cdot)}$ outputs the next state (σ_i, ξ_i) along with the next batch of queries Q_i

Crucially, the functions F_2 and F_3, which are used to generate response from main memory and update the state of main memory at the end of the round, do not have access to σ_{i-1} (the state of cache) or A_{i-1} (the answers to random oracle queries). In particular, any information about σ_{i-1} (cache-state) and A_{i-1} (most recent answers to random oracle queries) that main memory receives must be communicated through one of the messages in the set R_i. Similarly, the functions F_1 and F_4 are used to generate the requests sent from cache to main memory, to update the state of cache σ_i at the end of the round and to output the next batch Q_i of random oracle queries. Crucially these functions do not have access to ξ_{i-1} (the state of memory). Thus, any information about ξ_{i-1} must be communicated through one of the responses in the set S_i.

Dziembowski *et al.* [DKW11a] also addresses communication between two parties, A_{small} (e.g., a space-bounded virus) and A_{big}, over a bounded channel. However, both parties in this model can query the random oracle. This is a crucial difference, since one of the parties in our model, the main memory, is strictly forbidden from querying the random oracle to avoid trivialization of the problem (e.g., the attacker can perform all computation in RAM with no blue moves).

Execution Trace. We define the execution trace of the algorithm $\mathcal{A}^{H(\cdot)}$ by the sequence of cache states, memory states, messages passed between cache and memory, and queries made to the random oracle H. Formally, the execution trace is $\mathsf{Trace}_{\mathcal{A},R,H}(x) = \{(\sigma_i, \xi_i, R_i, S_i, Q_i)\}_{i=1}^{t}$, 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 we say the cost of the execution trace is

$$\mathsf{cost}(\mathsf{Trace}_{\mathcal{A},R,H}(x)) = \sum_{i=1}^{t} \left(c_r k_i + \sum_{j=1}^{\ell_i} \frac{c_b}{w} \left(|r_i^j| + |s_i^j| \right) \right) .$$

Intuitively, the c_r term is the cost of all of the queries we make to the random oracle H and the c_b terms result from the messages passed between cache and memory — here c_b denotes the cost of transferring w bits between cache and memory.

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

Definition 3.2. *Given constants c_b and c_r, the energy cost* ecost *of a function $f_{G,H}$ is defined by*

$$\mathsf{ecost}_{q,\epsilon}(f_{G,H}, m \cdot w) = \min_{\mathcal{A},x} \mathbb{E}\left[\mathsf{cost}(\mathsf{Trace}_{\mathcal{A},R,H}(x))\right],$$

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

3.2 Red-Blue Extension Pebbling

We are now ready to prove our main result in this section. Theorem 3.3 lower bounds the energy cost $\mathsf{ecost}_{q,\epsilon}(f_{G,H}, m \cdot w)$ of the function $f_{G,H}$ with cache size $m \cdot w$ using $\mathsf{rbpeb}^{\|}(G, 8m)$ the red-blue pebbling cost of the DAG G with $8m$ red pebbles.

Theorem 3.3. *For any G and any $\mathcal{A}_{mw}^{H(\cdot)}$ making at most $q < 2^{w/20}$ queries, then for $4 \log n < w$,*

$$\mathsf{ecost}_{q,\epsilon}\left(f_{G,H}, m \cdot w\right) \geq \frac{\epsilon}{16} \mathsf{rbpeb}^{\|}\left(G, 8m\right).$$

Given a DAG G and a legal black pebbling $P = (P_1, \ldots, P_t) \in \mathcal{P}^{\|}(G)$ with $|P_{i+1} \setminus P_i| \leq m$ we say that a (legal) red-blue pebbling $((B_1, R_1), \ldots, (B_t, R_t)) \in \mathcal{RB}^{\|}(G, m)$ is a k-extension of a black pebbling P if for all $i \in [t]$ we can find a small set $E_i \subseteq V(G)$ such that $|E_i| \leq k$, $P_i \subseteq B_i \cup R_i$ and $R_i \cup B_i = P_i \cup E_i$. We let $\mathbf{RBExt}(P, m, k) \subseteq \mathcal{RB}^{\|}(G, m)$ denote the set of all possible k-extensions of P. To prove Theorem 3.3 we extract a legal black pebbling $P = (P_1, \ldots, P_t) \in \mathcal{P}^{\|}(G)$ from the execution trace of $\mathcal{A}^{H(\cdot)}$, and then use P to build a legal extension

pebbling $((B_1, R_1), \ldots, (B_t, R_t)) \in \mathbf{RBExt}(P, m, 7m)$. Our extension pebbling may use up to $8m = (m + 7m)$ red-pebbles, but the pROM attacker $\mathcal{A}$ has a cache size mw bits — enough to store exactly m labels. We then show how to *upper bound* the cost of the extension pebbling and *lower bound* the energy cost of of the attacker $\mathcal{A}$ in the random oracle.

Step 1: Since the ϵ term is needed to account for algorithms that fail cheaply, we focus on the instances in which $\mathcal{A}_{mw}^{H(.)}$ correctly computes $f_{G,H}$. We start by using $\mathcal{A}_{mw}^{H(.)}$ to extract a legal $P = (P_1, \ldots, P_t) \in \mathcal{P}^{\parallel}(G)$ following Alwen and Serbinenko [AS15], who show that the computation of a function $f_{G,H}$ with hash function H and underlying directed acyclic graph G yields a legal black pebbling with high probability. Given an execution trace $\mathsf{Trace}_{\mathcal{A},R,H}(x)$, the corresponding pebbling $\mathtt{BlackPebble}^{H}(\mathsf{Trace}_{\mathcal{A},R,H}(x)) = P_0, \ldots, P_t$ is defined by setting $P_0 = \emptyset$ and define the pebbles at each subsequent time step i based on the corresponding batch of queries q_i made during iteration i. We then apply the following rules:

- For each query q in batch q_i, if the query has the form $v, \mathsf{lab}_{H,x}(v_1), \ldots, \mathsf{lab}_{H,x}(v_d)$ for some vertex v and its parents $v_1, \ldots, v_d$, then we add a pebble to node v in P_i.
- If there exists another query for v before v is used as input for a query, then v is deleted from P_i.

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. In this manner, we define $\mathtt{BlackPebble}^{H}(\mathsf{Trace}_{\mathcal{A},R,H}(x)) = P_1, \ldots, P_t \subseteq V$, which Alwen and Serbineneko show is legal with high probability,

$$\mathtt{BlackPebble}^{H}(\mathsf{Trace}_{\mathcal{A},R,H}(x)) \in \mathcal{P}^{\parallel}(G).$$

Theorem 3.4. *[AS15] The pebbling extracted from an execution trace,*

$$\mathtt{BlackPebble}^{H}(\mathsf{Trace}_{\mathcal{A},R,H}(x)) \in \mathcal{P}^{\parallel}(G),$$

is a legal black pebbling with probability at least $1 - \frac{q}{2^w}$*, where* w *is the label size and* q *is the number of queries made by* $\mathsf{Trace}_{\mathcal{A},R,H}$*.*

Once our black pebbling has been defined we can define the set $\mathbf{QueryFirst}(t_1, t_2)$ to be the data-labels that will appear as input to a random oracle query during rounds $[t_1, t_2]$ *before* the data-label appears as the *output* of some random oracle query made during the same interval. Formally, given P and an interval $[t_1, t_2]$ we let

$$\mathbf{QueryFirst}(t_1, t_2) = \bigcup_{i=t_1}^{t_2} \left(\mathsf{parents}\left(P_i \setminus P_{i-1}\right) \setminus \left(\bigcup_{j=t_1}^{i-1} \left(P_j \setminus P_{j-1}\right) \right) \right) .$$

Step 2: We partition the pebbling rounds $[t]$ into sub time-intervals $(t_0 = 0, t_1], (t_1, t_2], \ldots$ recursively follows. Let t_1 be the minimum pebbling round such

that there exists $j < t_1$ such that $|\mathbf{QueryFirst}(j, t_1)| \geq 3m$ — if no such t_1 exists then set $t_1 = t$ and output $(t_0, t_1]$. We emphasize the importance of using $\mathbf{QueryFirst}(j, t_1)$, rather using than the naive $\mathbf{QueryFirst}(0, t_1)$: the latter never admits an index t_1. As a special case if $|\mathbf{QueryFirst}(i, j)| \leq 3m$ for all $i < j \leq t$ (i.e., no such intervals exist) we remark that there is a red-blue extension pebbling in $\mathbf{RBExt}(P, 8m, k)$ that requires 0 blue moves at at most $\sum_i |P_i \setminus P_{i-1}|$ red-moves.

Once $t_1 < \ldots < t_{i-1}$ have been defined we let $t_i > t_{i-1}$ denote the minimum pebbling round such that there exists $t_{i-1} \leq j < t_i$ such that $|\mathbf{QueryFirst}(j, t_i)| \geq 3m$ — if no such t_i exists then we set $t_i = t$. Thus, we can assume that we have partitioned time into several intervals $(t_0 = 0, t_1], (t_1, t_2], \ldots$. We now argue that the attacker must pay $\Omega(m)$ blue moves during each sub-interval except the last one and that the extension pebbling has *at most* $4m$ blue moves during each interval — except for the first one where the the cost is 0.

First, note that for $j < t_i$, any node in $\mathbf{QueryFirst}(j + 1, t_{i+1})$ must either be in $\mathbf{QueryFirst}(t_i, t_{i+1})$ or have been pebbled between rounds t_i and j. Hence, we have the following invariant:

Invariant 1 *For any $j \in (t_i, t_{i+1})$,*

$$\mathbf{QueryFirst}(j + 1, t_{i+1}) \setminus \bigcup_{i=t_i}^{j} (P_i / P_{i-1}) \subseteq \mathbf{QueryFirst}(t_i, t_{i+1}).$$

Step 3: We can argue that there is an extension pebbling that has at most $4m$ blue moves during each interval. In particular, we set $k = 7m$ and we will define an extension pebbling

$$(B^*, R^*) \in \mathbf{RBExt}(P, 8m, k)$$

by dividing the cache into two sets of size $4m$ at each time i.

We now define sets R_i^{inter} and R_i^{legal} with the purpose of showing that $R_i = R_i^{legal} \cup R_i^{inter}$ is a legal red-blue pebbling with $B_i \supset P_i$. We set $R_{t_i}^{inter} = \{\}$ at the start of each time interval $(t_i, t_{i+1}]$ and for each $j \in (t_i, t_{i+1}]$ we have

$$R_j^{inter} = \left(R_{j-1}^{inter} \cup (P_j \setminus P_{j-1})\right) \cap \mathbf{QueryFirst}(j, t_{i+1}).$$

Intuitively, R_j^{inter} stores all of the red-pebbles we have computed during the interval $(t_i, j]$ that are later needed in the interval $[j + 1, t_{i+1})$. Thus, any node that is pebbled during rounds $(t_i, j]$ and subsequently needed in round $(j+1, t_{i+1})$ must be in R_j^{inter}, yielding the following invariant.

Invariant 2 *For any $j \in (t_i, t_{i+1})$,*

$$\mathbf{QueryFirst}(j + 1, t_{i+1}) \cap \bigcup_{i=t_i}^{j} (P_i / P_{i-1}) \subseteq R_j^{inter}$$

To maintain legality across all time steps, we add a few rules about blue moves:

(1) We convert a pebbled node v from blue to red if node v is in $\mathbf{QueryFirst}(t_i, t_{i+1})$. That is for any $j \in [t_i, t_{i+1})$, we define $R_j^{legal} = \mathbf{QueryFirst}(t_i, t_{i+1})$.
(2) We convert a pebbled node $v \in R_j$ from red to blue at time j if node v is in $\mathbf{QueryFirst}(t_i, t_{i+1})$ for some later interval (t_i, t_{i+1}) with $j < t_i$ and if $v \notin B_j$ is not already stored in cache. In this case it will be helpful to "charge" the cost c_b of this blue move to the future interval (t_i, t_{i+1}).

We now show the following bound on the size of $\mathbf{QueryFirst}(j, t_{i+1})$.

Lemma 3.5. *For any $j \in (t_i, t_{i+1})$,*

$$|\mathbf{QueryFirst}(j, t_{i+1})| \leq 4m.$$

Proof. Observe that $\mathbf{QueryFirst}(i+1, j) \setminus \mathbf{QueryFirst}(i, j) \subseteq (P_{i+1} \setminus P_i)$ and that $\mathbf{QueryFirst}(i, j+1) \subseteq \mathbf{QueryFirst}(i, j) \cup \mathsf{parents}\,(P_{j+1} \setminus P_j)$. Otherwise, we would have $|\mathbf{QueryFirst}(j, t_{i+1} - 1)| \geq 4m - |\mathsf{parents}\,(P_{j+1} \setminus P_j)| \geq 3m$, which contradicts our choice of t_{i+1}.

Lemma 3.6. *For each (t_i, t_{i+1}), $\{R_i\} = \{R_i^{legal} \cup R_i^{inter}\}$ is a legal pebbling.*

Proof. First, observe that $\mathsf{parents}(P_{j+1}/P_j) \subseteq \mathbf{QueryFirst}(j, t_{i+1})$ since any parents of P_{j+1} are either in P_j or not pebbled in round j, which inherently places them in $\mathbf{QueryFirst}(t_i, t_{i+1})$. Moreover, observe that $\mathbf{QueryFirst}(j, t_{i+1}) \subseteq R_j$, since any node in $\mathbf{QueryFirst}(j, t_{i+1})$ must either be in $\mathbf{QueryFirst}(t_i, t_{i+1})$ or have been pebbled at some point during time steps (t_i, j). In the former case, the node would be in R_j^{legal}, and in the latter case, the node would be in R_j^{inter}. Thus, $\mathsf{parents}(P_{j+1}/P_j) \subseteq \mathbf{QueryFirst}(j, t_{i+1}) \subset R_j$, so the pebbling is legal.

Lemma 3.7.

$$\left|R_j^{inter}\right| \leq 4m.$$

Proof. First observe that $R_j^{inter} \subseteq \mathbf{QueryFirst}(j+1, t_{i+1})$ since elements are only kept in R_j^{inter} if they are needed for some later pebbling round. On the other hand, $|\mathbf{QueryFirst}(j+1, t_{i+1})| \leq 4m$ by Lemma 3.5 and the attacker being limited to cache size m. Thus,

$$\left|R_j^{inter}\right| \leq |\mathbf{QueryFirst}(j+1, t_{i+1})| \leq 4m.$$

Step 4: Argue that during each interval the optimal pebbling is "charged" at most

$$8mc_b + \sum_{j \in (t_i, t_{i+1}]} c_r \left|P_j \setminus P_{j-1}\right| \ ,$$

and thus $\mathsf{rbpeb}^{\|}(B^*, R^*) \leq \sum_i \left(8mc_b + \sum_{j \in (t_i, t_{i+1}]} c_r \left|P_j \setminus P_{j-1}\right|\right)$. The point is that even if we start with a cold cache $R_{t_i}^{inter} = \{\}$ we can still have *at most* $4m$ cache-misses during this interval since we never discard red pebbles from $R_{t_i}^{inter}$. That is, by Lemma 3.7, the number of cache misses during the interval (t_i, t_{i+1}) is at most $|\mathbf{QueryFirst}(t_i, t_{i+1})| \leq 4m$. We will "charge" double for

every cache-miss $v \in \mathbf{QueryFirst}(t_i, t_{i+1})$ to account for the previous blue move where a blue pebble was initially placed on node v. In this way we can also charge the cost of all new blue pebbles that are placed on the graph during rounds $[t_i, t_{i+1})$ to future rounds.

Step 5: Argue that during each interval the algorithm $\mathcal{A}$ must pay red-blue cost *at least* $mc_b + \sum_{j \in (t_i, t_{i+1}]} c_r \, |P_j \setminus P_{j-1}|$. The point is that we can set up an extractor that extracts $3m$ random oracle labels (i.e., $3mw$ truly random bits) by simulating $\mathcal{A}$ during this time interval. The extractor needs a hint of size $mw + w(\#words_i)$ where $\#words_i$ is the total amount of data $\mathcal{A}$ transfers to/from cache. If $\#words_i \leq m$ then we will arrive at a contradiction as we compressed a random string of length $3mw$ — contradicting Lemma 3.1. Thus, $\mathcal{A}$ must pay blue cost *at least* mc_b during each interval, and by construction of $P = \mathtt{BlackPebble}^H(\mathsf{Trace}_{\mathcal{A},R,H}(x))$ the red-cost is at least $\sum_{j \in (t_i, t_{i+1}]} c_r \, |P_j \setminus P_{j-1}|$. We detail this step formally in Section 3.3.

3.3 Extractor

We now show that if an attacking strategy does not yield a corresponding legal red-blue pebbling, then the attacking strategy can be modified to form an extractor for the labels of a subset of nodes. That is, an extractor with access to the attacking strategy, the state of the cache, and a few select hints can successfully predict a large number of random bits, which cannot happen with high probability. The hints we give the extractor will dictate the location of the random bits, and ensure these bits remain "random" (that is, we do not explicitly query these locations later). Figure 1 illustrates this setup. In particular, the extractor will use a hint to simulate $\mathcal{A}^{H(\cdot)}$ but this hint *does not* include the current state of memory ξ_i. Instead, the hint will encode the messages that the attacker expects to receive from main memory which allows us to *simulate* the attacker without storing the entire (large) state ξ_i.

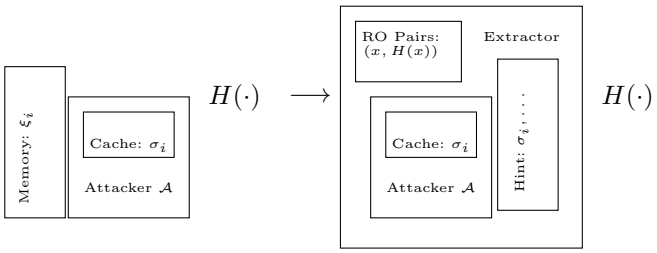

Fig. 1. Using the attacker to create an extractor that tries to predict $3m$ distinct outputs of random oracle $H(\cdot)$.

Lemma 3.8. *Given a randomly sampled execution trace*

$$\mathsf{Trace}_{\mathcal{A},R,H}(x) = \{(\sigma_i, \xi_i, R_i, S_i, Q_i)\}_{i=1}^{t},$$

let $\texttt{BlackPebble}^H(\mathsf{Trace}_{\mathcal{A},R,H}(x))$ *be the extracted black pebbling and* (t_i, t_{i+1}) *be defined from this black pebbling as above. If* $q < 2^{w/20}$ *and* $4 \log n < w$*, then with probability at least* $1 - \frac{2}{2^{w/2}}$*, the following holds for* all i*:*

$$\sum_{i=t_i}^{t_{i+1}} \left(c_r k_i + \sum_{j=1}^{\ell_i} \frac{c_b}{w} \left(|r_i^j| + |s_i^j| \right) \right) \geq m,$$

where q *is the total number of random oracle queries made in the execution trace and the probability is taken over the random coins of* $\mathcal{A}$ *and the selection of the random oracle* H*.*

Proof. Suppose, by way of contradiction, that for interval (t_i, t_{i+1}), an attacker transfers less than mw bits between cache and memory. We define an extractor that can predict $3m$ labels given access to the attacker's algorithm, the random oracle, and a small set of hints to help the extractor. Recall that $\mathsf{lab}_{H,x}(v) = H\left(v, \mathsf{lab}_{H,x}(v_1), \ldots, \mathsf{lab}_{H,x}(v_d)\right)$ for all nodes after the first node. Thus for nodes $y \neq z$, the values of $\mathsf{lab}_{H,x}(y)$ and $\mathsf{lab}_{H,x}(z)$ correspond to different inputs to H. That is, there are no *input* collisions and so the adversary must separately determine the hash outputs for each of the $3m$ inputs, which correspond to $3mw$ truly random bits in total.

The hint given to help the extractor consists of four components, which we now detail.

(1) The set $\mathbf{QueryFirst}(t_i, t_{i+1})$ is given as a hint to denote the indices that form the string that the extractor will ultimately predict. Since $|\mathbf{QueryFirst}(t_i, t_{i+1})| \leq 4m$, then the size of this component of the hint is at most $4m \log n$ bits.

(2) For each $v \in \mathbf{QueryFirst}(t_i, t_{i+1})$, the index of the first query that appears in which $\mathsf{lab}(v)$ is needed as input. This component of the hint tells the extractor the queries that require the prediction of random strings, and has size at most $4m \log q$ bits, where q is the total number of queries made by the attacker.

(3) For each $v \in \mathbf{QueryFirst}(t_i, t_{i+1})$, 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. In the later case, the extractor is not predicting a random string at all! To avoid this, we give the extractor a hint of the 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 $4m \log q$ bits.

(4) The cache state at t_i is given as a hint to the extractor to simulate the attacker beginning at time step t_i. Since the cache has size m, each containing w-bit words, the size of this component of the hint is at most mw bits.

(5) Messages between the cache and memory during time steps (t_i, t_{i+1}) are also given as a hint to the extractor to simulate the attacker beginning at time step t_i. By assumption, the attacker transfers less than mw bits between cache and memory, so the size of this component of the hint is at most mw bits in total.

Since $q < 2^{w/20}$ and $4 \log n < w$, then the total size, in bits, of the hint is at most

$$4m \log n + 4m \log q + 4m \log q + mw + mw \leq \frac{13}{5} mw.$$

However, $|\mathbf{QueryFirst}(t_i, t_{i+1})| \geq 3m$, so the extractor must successfully predict the output of $3m$ hash outputs, each of size w, given a hint of size at most $\frac{13}{5}mw$ bits. Thus, the extractor must predict $3mw - \frac{13}{5}mw = \frac{2}{5}mw$ random bits. The probability that the extractor predicts $\frac{2}{5}mw$ random bits is $2^{-2mw/5}$, so the probability that the extractor does not predict $\frac{2}{5}mw$ random bits over any set of time intervals (t_i, t_{i+1}) is at least $1 - \frac{n^2}{2^{2mw/5}} > 1 - \frac{1}{2^{mw/5}}$, for sufficiently large m, by a union bound over the intervals. Since the probability that the execution does not yield a legal black pebbling is $\frac{q}{2^w}$ by Theorem 3.4, then the probability that an attacker must transfer mw bits between cache and memory for each time interval (t_i, t_{i+1}) is at least $1 - \frac{1}{2^{mw/5}} - \frac{q}{2^w} > 1 - \frac{2}{2^{w/2}}$.

We now justify the correctness of Theorem 3.3.

Proof of Theorem 3.3: Recall that $\mathsf{ecost}_{q,\epsilon}(f_{G,H}, m \cdot w)$ is taken over algorithms that compute $f_{G,H}(x)$ correctly with probability at least ϵ. Thus, $\mathsf{ecost}_{q,\epsilon}(f_{G,H}, m \cdot w)$ is at least ϵ times the expected cost of an execution trace that correctly computes $f_{G,H}(x)$. Lemma 3.8 implies that an execution trace that correctly computes $f_{G,H}(x)$ must transfer at least m words between memory and cache for each interval (t_i, t_{i+1}) with probability at least $\left(1 - \frac{2}{2^{w/2}}\right) > \frac{1}{2}$. Recall that by construction, the red-cost for each interval is at least $\sum_{j \in (t_i, t_{i+1}]} c_r \left| P_j \setminus P_{j-1} \right|$. Therefore with probability at least $\frac{1}{2}$, the bandwidth cost of an execution trace that correctly computes $f_{G,H}(x)$ is *at least* $mc_b + \sum_{j \in (t_i, t_{i+1}]} c_r \left| P_j \setminus P_{j-1} \right|$ for each interval $(t_i, t_{i+1}]$. On the other hand, recall that $\mathsf{rbpeb}^{\|}(B^*, R^*) \leq \sum_i \left(8mc_b + \sum_{j \in (t_i, t_{i+1}]} c_r \left| P_j \setminus P_{j-1} \right| \right)$. Hence,

$$\mathsf{ecost}_{q,\epsilon}(f_{G,H}, m \cdot w) > \frac{\epsilon}{16} \mathsf{rbpeb}^{\|}(B^*, R^*).$$

□

4 Relating Memory Hardness and Bandwidth Hardness

In this section, we show that any function with high cumulative memory complexity also has high bandwidth costs. Namely,
Reminder of Theorem 1.2.

$$\mathsf{rbpeb}^{\parallel}(G,m) \geq 2c_b\left(\frac{\Pi_{cc}(G)}{t} - 2m\right) + c_r t \in \Omega\left(\sqrt{c_b \cdot c_r \cdot \Pi_{cc}(G)}\right),$$

where m is the cache size, t is the number of steps in the pebbling, c_b is the cost of a blue move and c_r is the cost of a red move.

We also show that this connection can be exploited to design a maximally bandwidth hard iMHF. Thus, the goals of designing an MHF with high cumulative memory complexity/bandwidth hardness are well aligned.

Lemma 4.1.

$$\mathsf{rbpeb}^{\parallel}(G,m) \geq \min_t\left(2\left(\frac{\Pi^{\parallel}_{cc}(G)}{t} - m\right)c_b, tc_r\right),$$

where c_b is the cost of a blue move and c_r is the cost of a red move.

Proof. For any red-blue pebbling P of DAG G, let R_i be the set of red pebbles at timestep i and let B_i be the set of blue pebbles at timestep i. Setting $D_i = B_i \cup R_i$ we remark that $(D_1, \ldots, D_t)$ is a valid black pebbling of G. Thus, by the optimality of $\Pi^{\parallel}_{cc}(G)$,

$$\Pi^{\parallel}_{cc}(G) \leq \sum_{i=1}^{t}|R_i \cup B_i| \leq \sum_{i=1}^{t}|R_i| + \sum_{i=1}^{t}|B_i| \leq t\max|B_i| + \sum_{i=1}^{t}|R_i| \leq t\left(\max_i|B_i| + m\right).$$

Rearranging terms we have

$$\max_i|B_i| \geq \frac{\Pi^{\parallel}_{cc}(G)}{t} - m\ .$$

On the other hand, in the optimal red-blue pebbling, each blue pebble must eventually be converted back to a red pebble, or else it should be discarded instead. Additionally, without loss of generality, we can assume that during each step we make at least one red move. Otherwise, we could combine consecutive steps into one single step. Thus,

$$\begin{aligned}
\mathsf{rbpeb}^{\parallel}(G,m) &\geq 2\left|\cup_{i=1}^{t}B_i\right|c_b + tc_r \\
&\geq 2\max_i|B_i|c_b + tc_r \\
&\geq 2\left(\frac{\Pi^{\parallel}_{cc}(G)}{t} - m\right)c_b + tc_r \\
&\geq \min_t\left(2\left(\frac{\Pi^{\parallel}_{cc}(G)}{t} - m\right)c_b, tc_r\right)
\end{aligned}$$

Corollary 4.2. *For an (e,d)-depth reducible graph,*

$$\mathsf{rbpeb}^{\parallel}(G,m) \geq \min_t\left(2\left(\frac{ed}{t} - m\right)c_b, tc_r\right).$$

Proof. [ABP17] show that any (e, d)-depth reducible DAG G has $ed \leq \Pi_{cc}^{\|}(G)$.

We show that there exists a similarly relationship between sequential black pebbling cost and sequential red-blue pebbling cost.

Theorem 4.3.

$$\mathsf{rbpeb}(G) \geq 2c_b \left(\frac{\Pi_{cc}(G)}{t} - m \right) + c_r t,$$

where m is the cache size, t is the number of steps in the pebbling, c_b is the cost of a blue move and c_r is the cost of a red move.

Proof. Given a sequential black pebbling $P_1, \ldots, P_t$ of G, let B_i be the set of blue pebbles at time step i. Then

$$\max_i |B_i| \geq \max_i (|P_i| - m) \geq \left(\frac{\Pi_{cc}(G)}{t} - m \right),$$

where the last step results from a simple averaging argument over all t steps. Finally, each item in B_i requires cost c_b to load into cache and another cost c_b to be retrieved from memory (if the item is not ever retrieved from memory, it would not be in B_i for an optimal pebbling).

Observe that Theorem 1.2 can also be related to parallel pebblings [8] through the following relationship:

Lemma 4.4.

$$\mathsf{rbpeb}(G, 2m) \leq \mathsf{rbpeb}^{\|}(G, m) \leq \mathsf{rbpeb}(G, m)$$

Proof. We show that $\mathsf{rbpeb}(G, 2m) \leq \mathsf{rbpeb}^{\|}(G, m)$ since $\mathsf{rbpeb}^{\|}(G, m) \leq \mathsf{rbpeb}(G, m)$ follows immediately from definition. We argue that any move performed by a parallel pebbling with capacity m can also be performed by a sequential pebbling with capacity $2m$. Note that at any step, a parallel pebbling with capacity m can have at most m labels stored and similarly, at most m new pebbles can be placed in each step. Thus, a sequential pebbling with capacity $2m$ can emulate this by retaining the stored labels of the parallel pebbling, while also adding the new pebbles at each step.

[8] To see that $\mathsf{rbpeb}^{\|}(G, m)$ and $\mathsf{rbpeb}(G, m)$ are not identically equivalent quantities, consider the complete directed bipartite graph $K_{m,m}$ with m sources A and m sink nodes B(m is also the cache size). In the parallel model we can finish in two steps with zero blue moves: $R_0 = \emptyset$, $R_1 = A$, $R_2 = B$. In the sequential pebble game we would have to keep pebbles on A while we begin placing pebbles on B one by one. Each time we place a red-pebble on a node $y \in B$ we need to evict some node $x \in A$ by converting x into a blue node (and then bring it back into the cache-later).

Combining Theorem 4.3 and Lemma 4.4 yields Theorem 1.2.

Alwen and Blocki [AB16] show that for any graph G with constant indegree, $\Pi^{\|}_{cc}(G) = \mathcal{O}\left(\frac{n^2 \log\log n}{\log n}\right)$. Moreover, there exist a family of DAGs $\{G_n\}_{n=1}^{\infty}$ with constant indegree with $\Pi_{cc}(G_n) \in \Omega(n^2)$ [Sch83,AdRNV16].

We now show a relationship similar to Theorem 1.2 between the bandwidth cost and cumulative memory cost of an execution trace. Recall that Alwen and Serbinenko [AS15] formally define cumulative memory cost of an execution trace as $\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}(x)) = \sum |\alpha_i|$, where α_i encodes the state of the attacker at round i. Similarly, $\mathsf{cmc}_{q,\epsilon}(f_{G,H}) = \min_{\mathcal{A},R,x} \mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}(x))$ where the minimum is taken over all $\mathcal{A}$ making at most q random oracle queries that compute $f_{G,H}$ correctly with probability at least ϵ. While there is no notion of a cache in their pROM model, we could trivially set $\alpha_i = (\sigma_i, \xi_i)$. We note that for $\mathsf{ecost}_{q,\epsilon}(f_{G,H})$ minimum is taken over all $\mathcal{A}$ making at most q random oracle queries that compute $f_{G,H}$ correctly with probability at least ϵ *and* having cache size at most m, which means that the set of attackers we consider is even more restrictive.

Theorem 4.5. *For any execution trace* $\mathsf{Trace}_{\mathcal{A},R,H}(x)$ *of an algorithm* $\mathcal{A}$ *with cache size* $m \cdot w$ *bits we have*

$$\mathsf{ecost}(\mathsf{Trace}_{\mathcal{A},R,H}(x)) \geq \left(\frac{\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}(x))}{tw} - m\right) c_b + tc_r,$$

where m *is the cache size,* t *is the number of steps,* c_b *is the cost of a blue move and* c_r *is the cost of a red move.*

Proof. Recall that the bandwidth cost of an execution trace $\mathsf{Trace}_{\mathcal{A},R,H}(x) = \{(\sigma_i, \xi_i, R_i, S_i, Q_i)\}_{i=1}^{t}$ is defined as

$$\begin{aligned}
\mathsf{ecost}(\mathsf{Trace}_{\mathcal{A},R,H}(x)) &= \sum_{i=1}^{t} \left(c_r|Q_i| + \frac{c_b}{w}\left(|R_i| + |S_i|\right)\right) \\
&\geq \max_i \frac{|\xi_i|}{w} c_b + tc_r \\
&\geq \left(\frac{\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}(x))}{tw} - m\right) c_b + tc_r
\end{aligned}$$

Where the second step follows from the observation that for all j we have $|\xi_j| \leq \sum_{i=1}^{j} (|R_i| + |S_i|)$, and the third observation follows from the observation that $\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}(x)) - mtw = \sum_{i=1}^{t} (|\xi_i| + |\sigma_i|) - mtx \leq tw \max_i |\sigma_i|$.

In particular, by minimizing over all t it follows that for any trace $\mathsf{Trace}_{\mathcal{A},R,H}(x)$ we have

$$\mathsf{ecost}(\mathsf{Trace}_{\mathcal{A},R,H}(x)) \in \Omega\left(\sqrt{\frac{\mathsf{cmc}(\mathsf{Trace}_{\mathcal{A},R,H}(x)) \cdot c_b \cdot c_r}{w}} - mc_b\right) \tag{1}$$

Since Alwen *et al.* [ACP+17] note that that $\mathsf{cmc}_{q,\epsilon}(\mathtt{scrypt}) \in \Omega(\epsilon n^2 \cdot w)$ for any $q > 0$ and $\epsilon > 2^{-w/2} + 2^{-n/20+1}$ it follows that

Corollary 4.6. *For any $q > 0$ and $\epsilon > 2^{-w/2} + 2^{-n/20+1}$,*

$$\mathsf{ecost}_{q,\epsilon}(\mathtt{scrypt}) \in \Omega\left(n\sqrt{c_b \cdot c_r}\right).$$

5 Bandwidth Hardness of Candidate iMHFs

In this section, we provide lower bounds on the bandwidth hardness on the family of graphs generated by Argon2i [BDK15], aATSample, and DRSample [ABH17]. Given a DAG $G = ([n], E)$, a target set $T \subset [n]$ and red/blue subsets $B, R \subseteq [n]$ with $|R| \leq m$ we let $\mathsf{rbpeb}^{\parallel}(G, m, T, B, R)$ denote the red-blue cost to place red pebbles on a target set T starting from an initial red-blue pebbling configuration B, R.

5.1 Analysis Framework

We follow a similar strategy for each proof by defining a target set $T = ((i-1)c\ell, ic\ell]$, and analyzing the structure of the DAG to prove that the following bound is large:

$$\min_{R \subseteq [(i-1)c\ell]:|R| \leq \ell} \min_{B' \subseteq [(i-1)c\ell]} \left(|B'|\, c_b + |\mathsf{ancestors}_{G-R-B'}(((i-1)c\ell, ic\ell])|\, c_r\right)$$

We show in Theorem 5.2 that this quantity suffices to lower bound the bandwidth hardness.

Lemma 5.1. *$\forall T, B, R \subseteq [n]$ such that $|R| \leq m$ we have*

$$\mathsf{rbpeb}^{\parallel}(G, m, T, B, R) \geq \min_{B' \subseteq B} \left(|B'|\, c_b + |\mathsf{ancestors}_{G-R-B'}(T)|\, c_r\right),$$

where c_b is the cost of a blue move and c_r is the cost of a red move.

Proof. Let $P = (B_0, R_0), (B_1, R_1) \ldots, (B_t, R_t)$ denote a legal red-blue pebbling sequence given starting configuration $B_0 = \{v \in B \ : \ \exists j \leq t. v \in R_j\}$ (e.g., B_0 is the subset of all blue pebbles in B that we will use at some point during the pebbling) and $R_0 = R$. By construction of B_0 the pebbling contains at least B_0 blue moves at cost $|B_0|\, c_b$. Similarly, we remark that we must place a red-pebble on all of the nodes in $\mathsf{ancestors}_{G-R-B'}(T)$ at some point. Thus, we have at least $|\mathsf{ancestors}_{G-R-B'}(T)|\, c_r$ red-moves. It follows that

$$\mathsf{rbpeb}^{\parallel}(G, m, T, B, R) \geq \min_{B' \subseteq B} \left(|B'|\, c_b + |\mathsf{ancestors}_{G-R-B'}(T)|\, c_r\right) \ .$$

Theorem 5.2. *Let $G = ([n], E)$ be any DAG such that $(i, i+1) \in E$ for each $i < n$ and let $c \geq 1$ be a positive integer then*

$$\mathsf{rbpeb}^{\parallel}(G, m) \geq \sum_{i=1}^{\lfloor \frac{n}{c\ell} \rfloor} \min_{R \subseteq [(i-1)c\ell]:|R| \leq \ell} \min_{B' \subseteq [(i-1)c\ell]} \left(|B'|\, c_b + |\mathsf{ancestors}_{G-R-B'}(((i-1)c\ell, ic\ell])|\, c_r\right) \ ,$$

where c_b is the cost of a blue move and c_r is the cost of a red move.

Proof. (Sketch) Repeatedly invoke the previous theorem. Consider an optimal red-blue pebbling and let t_i denote the first time we place a pebble on node $ic\ell$. For each i the red-blue cost incurred between steps $t_{i-1}+1$ and t_i starting from some red-blue configuration $B_{t_{i-1}}, R_{t_{i-1}}$ is at least

$$\begin{aligned}\mathsf{rbpeb}^{\|}(G,m,T=&[(i-1)c\ell+1,ic\ell],B_{t_{i-1}},R_{t_{i-1}})\\ &\geq \min_{B'\subseteq[(i-1)c\ell]}\left(|B'|\,c_b+|\mathsf{ancestors}_{G-R-B'}(T)|\,c_r\right)\\ &\geq \min_{R\subseteq[(i-1)c\ell]:|R|\leq m}\ \min_{B'\subseteq[(i-1)c\ell]}\left(|B'|\,c_b+|\mathsf{ancestors}_{G-R-B'}(T)|\,c_r\right)\ .\end{aligned}$$

To complete the proof we observe that

$$\mathsf{rbpeb}^{\|}(G,m)\geq\sum_{i=1}^{\left\lfloor\frac{n}{c\ell}\right\rfloor}\mathsf{rbpeb}^{\|}(G,m,T=[(i-1)c\ell+1,ic\ell],B_{t_{i-1}},R_{t_{i-1}})\ .$$

Now for our lower bounds on specific graphs we simply need to lower bound

$$\min_{R\subseteq[(i-1)cm]:|R|\leq m}\ \min_{B'\subseteq[(i-1)cm]}\left(|B'|\,c_b+|\mathsf{ancestors}_{G-R-B'}(T)|\,c_r\right)$$

for each $i\geq\left\lfloor\frac{n}{2cm}\right\rfloor$.

Observe that if $m=n$ then we have red-blue cost at most $\mathsf{rbpeb}^{\|}(G,m)\leq nc_r$ for *any* graph G. Thus, we must assume some upper bound on m to establish lower-bounds for red-blue pebbling cost.

5.2 Underlying DAGs

We now describe each of the underlying DAGs whose energy complexity we analyze. The underlying graph for Argon2iB [BDK16] has a directed path of length n nodes. Each node i has parents $i-1$ and $r(i)=i\left(1-\frac{x^2}{N^2}\right)$, where $N=2^{32}$ and x is chosen uniformly at random from $[N]$. See Algorithm 3 for details.

While Argon2iA (v1.1) is an outdated version of the password hash function it is still worthwhile to study for several reasons. First, the uniform edge distribution is a natural one which has been adopted by other iMHF constructions [BCS16]. Second, it is possible that this older version of Argon2i may have seen some adoption.

Each node i in Argon2iA has two parents: $i-1$ and $r(i)=i\left(1-\frac{x}{N}\right)$, where $N=2^{32}$ and x is chosen uniformly at random from $[N]$. Thus, the parents in Argon2iA are slightly less biased towards closer nodes than in Argon2iB. See Algorithm 4 for details.

DRSample, introduced by Alwen, Blocki, and Harsha [ABH17], is a family of graphs $\mathbb{G}_n$ with $\Pi^{\|}_{cc}(G)\in\Omega\left(\frac{n^2}{\log n}\right)$ with high probability for any $G\in\mathbb{G}_n$. Like Argon2i and Argon2iB, the underlying graph for DRSample has a directed path

of length n nodes. Each node i has parents $i-1$ and $r(i)$, but the distribution for $r(i)$ differs greatly from Argon2i and Argon2iB. Roughly speaking, DRSample samples an index j uniformly at random from $[1, \log i]$, an index k uniformly at random from $[1, 2^j]$, and sets $r(i) = i - k$. See Algorithm 1 for details.

A close relative to DRSample, aATSample [ABH17] is also a family of graphs $\mathbb{G}_n$ with $\Pi^{\|}_{cc}(G) \in \Omega\left(\frac{n^2}{\log n}\right)$ with high probability for any $G \in \mathbb{G}_n$. aATSample modifies DRSample by appending another directed path with n nodes that strategically connects to the first half of the graph so that the resulting computational complexity is high. See Algorithm 2 for details.

5.3 aATSample

We now consider the family of graphs generated by aATSample (Algorithm 2) [ABH17]. Alwen *et al.* showed that the first half of the nodes of aATSample is a (e, d, b)-block depth robust graph with $e = \Omega(n/\log n)$, $d = \Omega(n)$ and $b = \Omega(\log n)$ [ABH17]. This graph can be viewed as a metagraph G_k with $\Omega(n/b)$ meta-nodes M_i. Recall that we connect two meta-nodes $i < j$ in G_k if there exists a node in the last $k/3$ nodes of i to a node in the first $k/3$ nodes of j. The second half of the nodes of aATSample is a chain of $n/2$ nodes so that for each meta-node M_i and each interval $[v, v+\ell-1]$ of length $\ell = \Omega(n/b)$ in the second half of the graph, it holds that $\mathsf{parents}([v, v+\ell-1])$ contains some node in the last $k/3$ nodes of M_i. In order to move a pebble from node v to node $v+\ell-1$ in this construction, either the starting configuration must have at least $e/2$ pebbles on the graph, or a significant fraction of the block depth robust graph in the first half of the graph must be re-pebbled.

Lemma 5.3. *Let $i > \frac{n}{2}$ and $T = [i, i+\ell-1]$ be an interval of length $\ell = \frac{n}{\log n}$. Then there exists a constant $C > 0$ so that for $m \leq \frac{Cn}{\log n}$, a graph generated by* aATSample *satisfies the following with high probability:*

$$\min_{R \subseteq [i-1]:|R| \leq m} \min_{B' \subseteq [i-1]} \left(|B'|\, c_b + |\mathsf{ancestors}_{G-R-B'}(T)|\, c_r\right) \geq \min\left(\Omega\left(\frac{n}{\log n}\right) c_b, \Omega(n) c_r\right).$$

Using Lemma 5.3, whose proof appears in Appendix A.1, we have:

Reminder of Theorem 1.3. *Let G be a graph generated by* aATSample*. Then there exists a constant $C > 0$ so that for all $m \leq \frac{Cn}{\log n}$, it follows that*

$$\mathsf{rbpeb}^{\|}(G, m) \geq \min(\Omega(n) c_b, \Omega(n \log n) c_r),$$

with high probability.

Proof of Theorem 1.3: Applying Lemma 5.3 to each of the disjoint $\log n$ intervals in the second half of graph G, it follows from Theorem 5.2 that

$$\mathsf{rbpeb}^{\|}(G, m) \geq \min(\Omega(n) c_b, \Omega(n \log n) c_r).$$

□

5.4 Argon2iB

We now consider the family of graphs generated by Argon2iB (Algorithm 3) [BDK15]. Let G be a graph generated by Argon2iB and G_k be the metagraph with $\frac{n}{k}$ nodes, so that each meta-node in G_k represents k nodes in G. Again, we connect two meta-nodes $i < j$ in G_k if there exists a node in the last $k/3$ nodes of i to a node in the first $k/3$ nodes of j. Then G_k has the following property:

Lemma 5.4. *[BZ17] For any two meta-nodes $x < y$ of G_k, the last third of x is connected to the first third of y with probability at least $\frac{k\sqrt{k}}{k\sqrt{k}+36\sqrt{n(y-x+1)}}$.*

Lemma 5.5. *Suppose there exists a $C > 0$ and $0 < \epsilon < 2/3$ so that $m = Cn^{\frac{2}{3}-\epsilon}$. Let $i > \frac{n}{2}$ and let $T = [i, i+\ell-1]$ be an interval of length $\ell = \mathcal{O}\left(n^{\frac{2}{3}-\epsilon}\right)$. Then a graph generated by Argon2iB satisfies the following with high probability:*

$$\min_{R\subseteq[i-1]:|R|\leq m} \min_{B'\subseteq[i-1]} \left(|B'|\, c_b + |\mathsf{ancestors}_{G-R-B'}(T)|\, c_r\right) \geq \min\left(\Omega(n^{\frac{1}{3}-\epsilon})c_b, \Omega(n)c_r\right).$$

Proof. Consider the metagraph G_k with $\frac{n}{k}$ nodes for some constant k that we shall define. Let B be the set of nodes in G that have a blue pebble at some point, and let B_m be the set of meta-nodes that contain some node in B. Partition the second half of graph G into intervals of size $\ell = \frac{n}{k}$: $[i, i+\ell-1]$. By Lemma 5.4, there exists a constant α such that for $k = \alpha n^{\frac{2}{3}+\epsilon}$, the meta-nodes in $[i, i+\ell-1]$ are connected to $\Omega\left(\frac{n}{k}\right)$ meta-nodes in G_k with high probability. Thus, pebbling the interval $[i, i+\ell-1]$ requires pebbling at least $\beta n/k - m - |B_k|$ meta-nodes in G_k for some constant β. Noting that $m = \mathcal{O}\left(n^{\frac{2}{3}-\epsilon}\right)$ and that the middle $k/3$ nodes of a meta-node must be pebbled for two meta-nodes that are connected in G_k, it follows that at least $\left(\frac{c_1 n}{k} - |B_k|\right)\frac{k}{3} = \Omega(n)$ nodes must be pebbled. Thus, the cost of pebbling $[i, i+\ell-1]$ is at least $\min(\Omega(n/k)c_b, \Omega(n)c_r)$ nodes.

Reminder of Theorem 1.5. *Let G be a graph generated by Argon2iB. Then there exists a constant $C > 0$ so that for any $0 < \epsilon < 2/3$ and for all $m \leq Cn^{\frac{2}{3}-\epsilon}$, it follows that*

$$\mathsf{rbpeb}^{\|}(G, m) \geq \min(\Omega(n)c_b, \Omega(n^{5/3})c_r),$$

with high probability.

Proof of Theorem 1.5: Applying Lemma 5.5 to each of the disjoint $\frac{n}{\ell} = k = \mathcal{O}\left(n^{\frac{2}{3}+\epsilon}\right)$ intervals in the second half of graph G, it follows from Theorem 5.2 that

$$\mathsf{rbpeb}^{\|}(G, m) \geq \min(\Omega(n)c_b, \Omega(n^{5/3})c_r).$$

□

5.5 DRSample

We now consider the family of graphs generated by DRSample (Algorithm 1) [ABH17]. This graph can be viewed as a metagraph G_b with $\Omega(n/b)$ meta-nodes M_i. Recall that we connect two meta-nodes $i < j$ in G_b if there exists a node in the last $b/3$ nodes of i to a node in the first $b/3$ nodes of j.

Lemma 5.6. *Suppose $m = \mathcal{O}(n^\rho)$ for some constant $0 < \rho < 1$ and $i > \frac{n}{2}$. Let $T = [i, i + \ell - 1]$ be an interval of length $\ell = \mathcal{O}(n^\rho)$. Then a graph generated by* DRSample *satisfies the following with high probability:*

$$\min_{R \subseteq [i-1]:|R| \leq m} \min_{B' \subseteq [i-1]} \left(|B'|\, c_b + |\mathsf{ancestors}_{G-R-B'}(T)|\, c_r\right) \geq \min\left(\Omega(n^\rho)c_b, \Omega(n^{1/2+\rho/2})c_r\right).$$

Using Lemma 5.6, whose proof appears in Appendix A.2, we have:

Reminder of Theorem 1.4. *Let G be a graph generated by* DRSample *and $0 < \rho < 1$. Then there exists a constant $C > 0$ so that for all $m \leq Cn^\rho$, it follows that*

$$\mathsf{rbpeb}^{\|}(G, m) \geq \min\left(\Omega(n)c_b, \Omega(n^{3/2-\rho/2})c_r\right)$$

with high probability.

Proof of Theorem 1.4: Applying Lemma 5.6 to each of the disjoint $\frac{n}{\ell}$ intervals in the second half of graph G and observing that $\ell = \mathcal{O}(n^\rho)$, it follows from Theorem 5.2 that

$$\mathsf{rbpeb}^{\|}(G, m) \geq \min\left(\Omega(n)c_b, \Omega(n^{3/2-\rho/2})c_r\right).$$

□

We also give a stronger bound for DRSample when the cache has size $\mathcal{O}(n^\rho/\log n)$ for any $0 < \rho < 1$ in Appendix A.2.

6 NP-Hardness of the Red-Blue Pebbling Cost

In this section we prove that it is $\mathtt{NP} - \mathtt{Hard}$ to compute $\mathsf{rbpeb}^{\|}(G, m)$. Quanquan Liu [Liu17] observed that when $c_r = 0$ the problem is $\mathsf{PSPACE} - \mathsf{Hard}$ via a straightforward reduction from minimum space black pebbling. As we observed previously, when $c_b/c_r \in \mathcal{O}(\mathsf{poly}(n))$ the decision problem is in NP and has a fundamentally different structure. We show that even when the cost of red moves is significant, the problem remains $\mathtt{NP} - \mathtt{Hard}$.

Theorem 6.1. *For $c_b \leq 3nc_r$, the problem* $\mathsf{rbpeb}^{\|}$ *is* $\mathtt{NP} - \mathtt{Hard}$.

Gilbert *et al.* showed that the minimum space black pebbling problem was $\mathsf{PSPACE} - \mathsf{Hard}$ by reduction from the Truly Quantified Boolean Formula (TQBF) problem. We note that an instance ϕ of $3 - \mathsf{SAT}$ with n variables is still a TQBF instance (albeit with no $\forall$ quantifiers). Thus, given an instance ϕ of $3 - \mathsf{SAT}$ with n variables, we can create the corresponding DAG G_ϕ, as described in the

reduction of Gilbert *et al.* [GLT79]. The graph G_ϕ has the property that it can be pebbled with at most $3n + 3$ black pebbles if and only if ϕ is satisfiable.

We detail in Appendix A.4 a gadget to append to G_ϕ to create a graph H_ϕ so that $\mathsf{rbpeb}^{\|}(H) = x_1$ if ϕ is a satisfiable assignment, but $\mathsf{rbpeb}^{\|}(H_\phi) > x_1$ if ϕ is not a satisfiable assignment. The key goal of the additional gadget is to ensure that we cannot *significantly* reduce the number of red moves (computation costs) by including a few blue move. Thus, if ϕ is satisfiable, the optimal pebbling will correspond to the minimum space black pebbling and will require 0 blue moves.

Lemma 6.2. *If ϕ is satisfiable, then there exists a pebbling strategy of H_ϕ with capacity $3n + 4$ and cost exactly*

$$\left(\frac{3n^3 + 15n^2 + 40n + 14 + 12c}{2}\right) c_r.$$

Lemma 6.3. *If ϕ is unsatisfiable, then the pebbling cost of H_ϕ with capacity $3n + 4$ is greater than*

$$\left(\frac{3n^3 + 15n^2 + 40n + 14 + 12c}{2}\right) c_r.$$

Together, Lemma 6.2 and Lemma 6.3, whose proofs appear in Appendix A.4, imply Theorem 6.1.

Acknowledgements

The authors would like to thank Daniel Wichs for helpful discussion and anonymous reviewers for important comments that improved the presentation of the paper. The work was supported by the National Science Foundation under NSF Awards #1649515 and #1704587. The opinions expressed in this paper are those of the authors and do not necessarily reflect those of the National Science Foundation.

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. 1, 1.1, 2.1, 4

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. 1, 1.1, 5

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. 1, 1, 1.1, 1.1, 2.1, 5, 5.2, 5.3, 5.5, 1, 2

ABMW05. Martín Abadi, Michael Burrows, Mark S. Manasse, and Ted Wobber. Moderately hard, memory-bound functions. *ACM Trans. Internet Techn.*, 5(2):299–327, 2005. 1

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 II*, volume 10211 of *LNCS*, pages 3–32. Springer, Heidelberg, May 2017. 1, 1.1, 2.1, 4

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 II*, volume 10211 of *LNCS*, pages 33–62. Springer, Heidelberg, May 2017. 1, 1.1, 4

AdRNV16. Joël Alwen, Susanna F. de Rezende, Jakob Nordstrm, and Marc Vinyals. Cumulative space in black-white pebbling and resolution. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, 9-11 January 2017, Berkeley, California USA*, 2016. 1.1, 4

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. 1, 1, 1.1, 1.1, 1.1, 3, 3.2, 3.4, 4

Bac02. Adam Back. Hashcash-a denial of service counter-measure, 2002. 1

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. 1, 5.2, A.3

BDK15. Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Fast and tradeoff-resilient memory-hard functions for cryptocurrencies and password hashing. *IACR Cryptology ePrint Archive*, 2015:430, 2015. 1, 5, 5.4, A.3, 3, 4

BDK16. Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Argon2: New generation of memory-hard functions for password hashing and other applications. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 292–302, 2016. 1, 1.1, 5.2

BDKJ16. Alex Biryukov, Daniel Dinu, Dmitry Khovratovich, and Simon Josefsson. The memory-hard argon2 password hash and proof-of-work function, March 2016. 1.1, 6

Ber05. Daniel J. Bernstein. Cache-timing attacks on aes, 2005. 1

BHZ18. Jeremiah Blocki, Ben Harsha, and Samson Zhou. On the economics of offline password cracking. In *IEEE Symposium on Security and Privacy, SP*, pages 35–53, 2018. 1

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. 1, 1.1, 2.1, 5.4

BZ18. Jeremiah Blocki and Samson Zhou. On the computational complexity of minimal cumulative cost graph pebbling. *Financial Cryptography and Data Security, 22nd International Conference, FC*, 2018. (to appear). 1.1, B.2, B

DKW11a. Stefan Dziembowski, Tomasz Kazana, and Daniel Wichs. Key-evolution schemes resilient to space-bounded leakage. In *Advances in Cryptology - CRYPTO - 31st Annual Cryptology Conference, Proceedings*, pages 335–353, 2011. 3.1

DKW11b. 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. 3.1

DL17. Erik D. Demaine and Quanquan C. Liu. Inapproximability of the standard pebble game and hard to pebble graphs. In *Algorithms and Data Structures - 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 - August 2, 2017, Proceedings*, pages 313–324, 2017. 1.1

DN92. Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology - CRYPTO, 12th Annual International Cryptology Conference, Proceedings*, pages 139–147, 1992. 1

GLT79. John R. Gilbert, Thomas Lengauer, and Robert Endre Tarjan. The pebbling problem is complete in polynomial space. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing (STOC)*, pages 237–248, 1979. 1.1, 1.1, 6, A.4, B, B, B.1, B.2, B

HK81. Jia-Wei Hong and H. T. Kung. I/O complexity: The red-blue pebble game. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing, May 11-13, 1981, Milwaukee, Wisconsin, USA*, pages 326–333, 1981. 1.1

Liu17. Quanquan Liu. Red-blue and standard pebble games: Complexity and applications in the sequential and parallel models. Master's thesis, Massachusetts Institute of Technology, Feburary 2017. 1.1, 6

Nak08. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. 1

Per09. Colin Percival. Stronger key derivation via sequential memory-hard functions. *BSDCan*, 2009. 1

PHC. Password hashing competition. 1

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. 1, 1, 1.1, 1.1

Sch83. Georg Schnitger. On depth-reduction and grates. In *24th Annual Symposium on Foundations of Computer Science*, pages 323–328, 1983. 1.1, 4

A Complete Proofs

A.1 aATSample

Reminder of Lemma 5.3. *Let $i > \frac{n}{2}$ and $T = [i, i+\ell-1]$ be an interval of length $\ell = \frac{n}{\log n}$. Then there exists a constant $C > 0$ so that for $m \leq \frac{Cn}{\log n}$, a graph generated by* aATSample *satisfies the following with high probability:*

$$\min_{R\subseteq[i-1]:|R|\leq m} \min_{B'\subseteq[i-1]} \left(|B'|\, c_b + |\mathsf{ancestors}_{G-R-B'}(T)|\, c_r\right) \geq \min\left(\Omega\left(\frac{n}{\log n}\right) c_b, \Omega(n) c_r\right).$$

Proof. Lemma 5.3 Consider the metagraph G_k with $\frac{n}{\log n}$ nodes and recall that G_k is $(\Omega(n/\log n), \Omega(n/\log n))$ depth robust. Let B be the set of nodes in G that have a blue pebble at some point, and let B_k be the set of meta-nodes that contain some node in B. Then by depth-robustness, there exists a path of length at least $c_1 n/\log n - |B_k|$ in G_k.

Now, partition the second half of graph G into intervals of size $\ell = n/\log n$: $[i, i+\ell-1]$. Since $\mathsf{parents}([i, i+\ell-1]) \cap M_i \neq \emptyset$, then pebbling the interval $[i, i+\ell-1]$ requires pebbling at least $c_1 n/\log n - m - |B_k|$ meta-nodes in G_k. Noting that $m = \mathcal{O}\left(\frac{n}{\log n}\right)$ and the middle $\log n/3$ nodes of a meta-node must be pebbled for two meta-nodes that are connected in G_k, it follows that at least $\left(\frac{c_1 n}{\log n} - |B_k|\right)\frac{\log n}{3} = \Omega(n)$ nodes must be pebbled. Thus, the cost of pebbling $[i, i+\ell-1]$ is at least $\min\left(\Omega\left(\frac{n}{\log n}\right) c_b, \Omega(n) c_r\right)$ nodes.

A.2 DRSample

Reminder of Lemma 5.6. *Suppose $m = \mathcal{O}(n^\rho)$ for some constant $0 < \rho < 1$ and $i > \frac{n}{2}$. Let $T = [i, i+\ell-1]$ be an interval of length $\ell = \mathcal{O}(n^\rho)$. Then a graph generated by* DRSample *satisfies the following with high probability:*

$$\min_{R \subseteq [i-1]: |R| \leq m} \min_{B' \subseteq [i-1]} \left(|B'|\, c_b + |\mathsf{ancestors}_{G-R-B'}(T)|\, c_r\right) \geq \min\left(\Omega(n^\rho) c_b, \Omega(n^{1/2+\rho/2}) c_r\right).$$

Proof of Lemma 5.6: Let $T = [i, i+\ell]$ where $\ell = \Omega(n^\rho)$ for some constant $\frac{1}{2} < \rho < 1$ and let $r(j)$ denote the predecessor of a node j in the graph (besides $j-1$). Let X_j be an indicator random variable for the event $\mathsf{far}(j)$, which we define to be the event that $|r(j) - r(k)| > b = \sqrt{\frac{n}{16\ell}}$ for all $k \in [1, j-1]$. Observe that

$$\mathbf{Pr}\left[\mathsf{far}(j)\right] \geq \mathbf{Pr}\left[\mathsf{far}(j) | r(i) = i-b, r(i+1) = i-2b, \ldots, r(j-1) = i-(j-i+1)b\right]$$

since the conditioned event has the maximum number of invalid nodes for $r(j)$, and with the highest probability of hitting each of these invalid nodes, since they are the closest to i. Thus,

$$\begin{aligned}
\mathbf{Pr}\left[\mathsf{far}(j)\right] &\geq \mathbf{Pr}\left[r(j) < i - (j-i+1)b\right] \\
&\geq \mathbf{Pr}\left[j - r(j) > \ell + (j-i+1)b\right] \\
&\geq \mathbf{Pr}\left[j - r(j) > \ell + (\ell+1)b\right] \\
&\geq \mathbf{Pr}\left[j - r(j) > \sqrt{\frac{n\ell}{2}}\right]
\end{aligned}$$

since $j \leq i + \ell$ and $b = \sqrt{\frac{n}{16\ell}}$. Hence,

$$\begin{aligned}
\mathbf{Pr}\left[\mathsf{far}(j)\right] &\geq \frac{\log(j) - \log\sqrt{n\ell}}{\log(j)} \\
&\geq 1 - \left(\frac{1}{2} - \frac{\rho}{2}\right)\left(\frac{\log(n)}{\log(n)-1}\right) \\
&\geq \frac{1}{2} - \frac{\rho}{2} - o(1) = \Omega(1).
\end{aligned}$$

Let $X = \sum_{k=i}^{i+\ell} X_k$. With high probability, $X > c\ell$ for some constant c that depends on ρ. Picking ℓ to satisfy $m < \frac{c\ell}{4}$, then with high probability, the number of ancestors of T in $G - R - B'$ is at most

$$\begin{aligned}(X - |R| - |B'|)b &\geq (X - m - |B'|)b \\ &\geq \left(\frac{c\ell}{4}\right)\sqrt{\frac{n}{16\ell}}.\end{aligned}$$

Thus to pebble T, either $\frac{c\ell}{2}$ blue moves are required, or at least $\left(\frac{c\ell}{4}\right)\sqrt{\frac{n}{16\ell}}$ red moves are required, to pebble the ancestors of T in $G - R - B'$. Hence, the cost is at least $\min\left(\frac{c\ell}{2}c_b, \left(\frac{c\ell}{4}\right)\sqrt{\frac{n}{16\ell}}c_r\right)$. Since $\ell = \Omega(n^\rho)$, then the cost is at least

$$\min\left(\Omega(n^\rho)c_b, \Omega(n^{1/2+\rho/2})c_r\right).$$

□

We now give a stronger bound for DRSample when the cache has size $\mathcal{O}\left(n^\rho / \log n\right)$ for any $0 < \rho < 1$.

Lemma A.1. *For each $x, y \in G_b$ with $y > x$ and node i in meta-node y, there exists an edge from the last third of meta-node x to node i with probability at least $\frac{1}{6|y-x|\log y}$.*

Proof. Recall that for node i, DRSample creates an edge from i to parent node $r(i)$ by first sampling j from $[1, \lfloor \log i \rfloor]$. Then DRSample sets $r(i) = i - k$ by randomly sampling k from $(2^{j-1}, 2^j]$. Thus, for nodes $i, j \in G$ with $i > j$, there exists an edge from node j to i with probability at least $\frac{1}{2b|y-x|\log i}$. Taking the union bound over all $\frac{b}{3}$ nodes in the last third of meta-node x and observing that $i < y$ yields the desired result.

Lemma A.2. *For any two meta-nodes $x, y \in G_b$ with $x < y$, the last third of x is connected to the first third of y with probability at least $\frac{b}{b+18|y-x|\log n}$.*

Proof. Let p be the probability that the final third of x is connected to the first third of y. Let E_i be the event that the i^{th} node of meta-node y is the first node in y to which there exists an edge from the last third of meta-node x, so that by Lemma A.1, $\mathbf{Pr}\left[E_1\right] \geq \frac{1}{6|y-x|\log y}$. Note that furthermore, $\mathbf{Pr}\left[E_i\right]$ is the probability that there exists an edge from the last third of meta-node x to the i^{th} node of meta-node y and no previous meta-node of y. Hence, $\mathbf{Pr}\left[E_i\right] \geq \frac{1}{6|y-x|\log y}(1-p)$. Thus,

$$\begin{aligned}p &= \mathbf{Pr}\left[E_1\right] + \mathbf{Pr}\left[E_2\right] + \ldots + \mathbf{Pr}\left[E_{b/3}\right] \\ &\geq \left(\frac{b}{3}\right)\frac{1}{6|y-x|\log y}(1-p).\end{aligned}$$

Setting $\alpha = \left(\frac{b}{3}\right)\frac{1}{6|y-x|\log y}$, then it follows that $p + \alpha p \geq \alpha$, so that $p \geq \frac{\alpha}{1+\alpha}$. Since $y \leq \frac{n}{b}$,

$$p \geq \frac{b}{b + 18|y-x|\log y} \geq \frac{b}{b + 18|y-x|\log n}.$$

Lemma A.3. *Suppose $m = \mathcal{O}(n^\rho/\log n)$ and $i > \frac{n}{2}$. Let $T = [i, i+\ell-1]$ be an interval of length $\ell = \mathcal{O}(n^\rho)$ for some $0 < \rho < 1$. Then a graph generated by* DRSample *satisfies the following with high probability:*

$$\min_{R\subseteq[i-1]:|R|\leq m} \min_{B'\subseteq[i-1]} (|B'|\, c_b + |\mathsf{ancestors}_{G-R-B'}(T)|\, c_r) \geq \min(\tilde{\Omega}(n^\rho)c_b, \tilde{\Omega}(n)c_r).$$

Proof. Let T be an interval of length ℓ. Consider a metagraph $G_{n/\ell}$ with ℓ nodes. Then by Lemma A.2, a meta-node y in G_ℓ is connected to a previous meta-node x with probability $\frac{\ell}{\ell+18|y-x|\log n} = \Omega(1)$ for $|y-x| = \mathcal{O}\left(\frac{n^\rho}{\log n}\right)$, since $\ell = \Omega(n^\rho)$. Thus in expectation, y is connected to $\frac{Cn^\rho}{\log n}$ previous meta-nodes for some constant y.

Now, partition the second half of graph G into intervals of size n/ℓ. For each interval of length n/ℓ, to pebble a meta-node, either there already exists a blue pebble in the previous meta-node or we must spend $\frac{2n}{3\ell}$ red moves to repebble the previous meta-node. Applying the same argument for $\frac{Cn^\rho}{\log n} - m$ of the previous meta-nodes, and noting that $\ell = \mathcal{O}(n^\rho)$ and $m = \mathcal{O}(n^\rho/\log n)$, then the cost of pebbling $[i, i+\ell-1]$ is at least $\tilde{\Omega}(n^\rho)\min\left(c_b, \frac{2n}{3\ell}c_r\right) = \min(\tilde{\Omega}(n^\rho)c_b, \tilde{\Omega}(n)c_r)$.

Theorem A.4. *Let G be a graph generated by* DRSample *and $0 < \rho < 1$. Then there exists a constant $C > 0$ so that for all $m \leq Cn^\rho/\log n$, it follows that*

$$\mathsf{rbpeb}^{\parallel}(G, m) \geq \min(\tilde{\Omega(n)}c_b, \tilde{\Omega}(n^{2-\rho})c_r)$$

with high probability.

Proof. Applying Lemma A.3 to each of the disjoint $\frac{n}{\ell}$ intervals in the second half of graph G and observing that $\ell = \mathcal{O}(n^\rho)$, it follows from Theorem 5.2 that

$$\mathsf{rbpeb}^{\parallel}(G, m) \geq \min(\tilde{\Omega}(n)c_b, \tilde{\Omega}(n^{2-\rho})c_r).$$

A.3 Argon2iA

We now consider the family of graphs generated by Argon2iA (Algorithm 4) [BDK15]. Notably, the same underlying graph is also used in Balloon Hashing (Boneh et al. [BCS16]) Let G be a graph generated by Argon2i and G_b be the metagraph with $\frac{n}{b}$ nodes, so that each meta-node in G_b represents b nodes in G. Again, we connect two meta-nodes $i < j$ in G_b if there exists a node in the last $b/3$ nodes of i to a node in the first $b/3$ nodes of j.

Lemma A.5. *For each $x, y \in G_b$ with $y > x$ and node i in meta-node y, there exists an edge from the last third of meta-node x to node i with probability at least $\frac{1}{6y}$.*

Proof. Recall that for node i, Argon2i creates an edge from i to parent node $i\left(1 - \frac{k}{N}\right)$, where $k \in [N]$ is picked uniformly at random. Thus, for nodes $i, j \in G$

with $i > j$, there exists an edge from node j to i with probability at least

$$\begin{aligned}
&\mathbf{Pr}\left[(x-1)m+1 \leq i\left(1-\frac{k}{N}\right) \leq \left(x-1+\frac{1}{3}\right)m\right] \\
&= \mathbf{Pr}\left[\left(x-1+\frac{1}{6}\right)m \leq ym\left(1-\frac{k}{N}\right) \leq \left(x-1+\frac{1}{3}\right)m\right] \\
&\geq \mathbf{Pr}\left[\frac{y-x+\frac{5}{6}}{y} \geq \frac{k}{N} \geq \frac{y-x+\frac{2}{3}}{y}\right] \\
&\geq \left(\frac{y-x+\frac{5}{6}}{y}\right) - \left(\frac{y-x+\frac{2}{3}}{y}\right) \\
&\geq \frac{1}{6y}
\end{aligned}$$

.

Lemma A.6. *For any two meta-nodes $x, y \in G_b$ with $x < y$, the last third of x is connected to the first third of y with probability at least $\frac{b^2}{9y\log y + b^2}$.*

Proof. Let p be the probability that the final third of x is connected to the first third of y. Let E_i be the event that the i^{th} node of meta-node y is the first node in y to which there exists an edge from the last third of meta-node x, so that by Lemma A.5, $\mathbf{Pr}[E_1] \geq \frac{1}{6y}$. Note that furthermore, $\mathbf{Pr}[E_i]$ is the probability that there exists an edge from the last third of meta-node x to the i^{th} node of meta-node y and no previous meta-node of y. Hence, $\mathbf{Pr}[E_i] \geq \frac{1}{6y}(1-p)$. Thus,

$$\begin{aligned}
p &= \mathbf{Pr}[E_1] + \mathbf{Pr}[E_2] + \ldots + \mathbf{Pr}\left[E_{b/3}\right] \\
&\geq \left(\frac{b}{3}\right)\frac{1}{6y}(1-p).
\end{aligned}$$

Setting $\alpha = \left(\frac{b}{3}\right)\frac{1}{6y}$, then it follows that $p + \alpha p \geq \alpha$, so that $p \geq \frac{\alpha}{1+\alpha}$. Since $y \leq \frac{n}{b}$, then $p \geq \frac{b}{18y+b} \geq \frac{b^2}{18n+b^2}$.

Lemma A.7. *Let $i > \frac{n}{2}$ and $T = [i, i+\ell-1]$ be an interval of length $\ell = \mathcal{O}\left(n^{1/2}\right)$. There exists a constant $C > 0$ so that for $0 < \epsilon < 1/2$ and $m \leq Cn^{1/2}$, a graph generated by Argon2i satisfies the following with high probability:*

$$\min_{R\subseteq[i-1]:|R|\leq m} \min_{B'\subseteq[i-1]} \left(|B'|\, c_b + |\mathsf{ancestors}_{G-R-B'}(T)|\, c_r\right) \geq \min\left(\Omega(n^{1/2})c_b, \Omega(n)c_r\right).$$

Proof. Consider the metagraph G_k with $\frac{n}{k}$ nodes for some constant k that we shall define. Let B be the set of nodes in G that have a blue pebble at some point, and let B_m be the set of meta-nodes that contain some node in B. Partition the second half of graph G into intervals of size $\ell = \frac{n}{k}$: $[i, i+\ell-1]$. By Lemma A.6, there exists a constant α such that for $k = \alpha n^{1/2}$, the meta-nodes in $[i, i+\ell-1]$ are connected to $\Omega\left(\frac{n}{k}\right)$ meta-nodes in G_k with high probability. Thus, pebbling the interval $[i, i+\ell-1]$ requires pebbling at least $\beta n/k - m - |B_k|$ meta-nodes in G_k for some constant β. Noting that $m = \mathcal{O}(n/k)$ and that the middle $k/3$ nodes of a meta-node must be pebbled for two meta-nodes that are connected in G_k, it follows that at least $\left(\frac{c_1 n}{k} - |B_k|\right)\frac{k}{3} = \Omega(n)$ nodes must be pebbled. Thus, the cost of pebbling $[i, i+\ell-1]$ is at least $\min(\Omega(n/k)c_b, \Omega(n)c_r)$ nodes.

Theorem A.8. *Let G be a graph generated by Argon2i. Then there exists a constant $C > 0$ so that for all $m \leq Cn^{1/2}$, it follows that*

$$\mathsf{rbpeb}^{\|}(G, m) \geq \min(\Omega(n)c_b, \Omega(n^{3/2})c_r),$$

with high probability.

Proof of Theorem A.8: Applying Lemma A.7 to each of the disjoint $\frac{n}{\ell} = k = \mathcal{O}\left(n^{1/2}\right)$ intervals in the second half of graph G, it follows from Theorem 5.2 that

$$\mathsf{rbpeb}^{\|}(G, m) \geq \min(\Omega(n)c_b, \Omega(n^{1/2})c_r).$$

□

A.4 NP-Hardness of the Red-Blue Pebbling Cost

For more details about the Gilbert *et al.* [GLT79] reduction, we refer an interested reader to Appendix B. Since an instance ϕ of $3-\mathsf{SAT}$ with n variables is still a TQBF instance (albeit with no $\forall$ quantifiers), we can create the corresponding DAG G_ϕ, as described in the reduction of Gilbert *et al.* [GLT79]. For DAG G_ϕ with t vertices, there exist unique pyramid gadgets with $3n+3, 3n+2, 3n+1, \ldots, 1$ vertices in the bottom layer. Let Δ_i be the pyramid gadget with ϕ vertices in the bottom layer. Additionally, let α_i be the vertex *above* the apex of pyramid Δ_i. Create a directed path P_1 with $3n+3$ vertices so that for each $1 \leq i \leq 3n+3$, connect an edge to vertex $3i-2$ of P_1 from the top vertex of Δ_{3n+4-i}.

We then connect the final vertex of P_1 to a directed path P_2 with

$$\left(\frac{(3n+1)(3n)}{2}+1\right)+\left(\frac{(3n-2)(3n-3)}{2}+1\right)+\ldots+(21+1)+(6+1) = \frac{3}{2}n(n+1)^2+n$$

vertices. Moreover, the first $\frac{(3n+1)(3n)}{2}$ vertices of P_2 each have an edge from separate vertices of Δ_{3n+1}, starting with the vertices in the bottom layer and moving upwards. We also create an edge to the following vertex from the vertex α_{3n+1}. The next $\frac{(3n-2)(3n-3)}{2}$ vertices of P_2 each have an edge from separate vertices of Δ_{3n-2}, starting with the vertices in the bottom layer and moving upwards. We also create an edge to the following vertex from the vertex α_{3n-2}. We continue this process until all vertices from all pyramids of the form Δ_{3i+1} are connected to P_2, as well as the vertices α_{3i+1}. Finally, we connect P_2 to a sink node. Then by setting P to be the path P_1 concatenated with P_2, we have the following result:

Lemma A.9. *P contains exactly $3n+3+\sum_{i=1}^{n}\left(\frac{(3i+1)(3i)}{2}+1\right) = 4n+3+\frac{3}{2}n(n+1)^2 = \frac{3n^3+6n^2+11n+6}{2}$ vertices.*

See Figure 5 for an example. Let $H_\phi = G_\phi \cup P$. We claim that H_ϕ with capacity $3n+4$ will have a certain pebbling cost if and only if ϕ is satisfiable.

Reminder of Lemma 6.2. *If ϕ is satisfiable, then there exists a pebbling strategy of H_ϕ with capacity $3n+4$ and cost exactly*

$$\left(\frac{3n^3+15n^2+40n+14+12c}{2}\right)c_r.$$

Proof of Lemma 6.2: The total number of nodes in G_ϕ corresponding to variable assignments from the GLT construction is

$$6n+\sum_{i=4}^{3n+3} i = \frac{9n^2+33n+12}{2}.$$

This can be visualized in Figure 4 by the nodes on the left hand side, excluding the nodes q_i. Additionally, there are n nodes q_i, six nodes for each of the c clauses p_i for $1 \le i \le c$, and an additional node for p_0. Thus,

$$6c+1+7n+\sum_{i=4}^{3n+3} i = \frac{9n^2+35n+14}{2}+6c$$

nodes must be pebbled in G_ϕ.

By Lemma A.9, the number of nodes in the additional path P is $\frac{3n^3+6n^2+21n+18}{2}$. However, pebbling P requires that each of the pyramids Δ_{3i+1} are pebbled a second time, as well as each α_{3i+1}, requiring an additional

$$\sum_{i=1}^{n}\left(\frac{(3i+1)(3i)}{2}+1\right) = n+\frac{3}{2}n(n+1)^2 = \frac{3n^3+6n^2+5n}{2}$$

steps.

Thus, the total number of steps required to pebble H_ϕ is

$$\frac{9n^2+35n+14}{2}+6c+\frac{3n^3+6n^2+5n}{2} = \frac{3n^3+15n^2+40n+14+12c}{2}.$$

The GLT construction has pebbling number $3n+3$. Since the nodes in P are ordered corresponding to the natural pebbling order in G_ϕ, a single additional pebble suffices for P. Thus, if the capacity of G_ϕ is $3n+4$, then all pebbling moves can be achieved with red moves, so there exists a pebbling strategy with total cost is $\left(\frac{3n^3+15n^2+40n+14+12c}{2}\right)c_r$.

By construction, the pebbling strategy of Lemma 6.2 is the optimal pebbling with only red moves. Thus, it remains to show that no strategy containing any blue pebbles has better cost.

Since blue pebbles are more expensive than red pebbles, the only place the above strategy can be possibly improved would be using blue pebbles on nodes in H_ϕ that are pebbled multiple times in the strategy of Lemma 6.2. As it turns out, the only nodes that are pebbled multiple times are the pyramids Δ_{3i+1},

which are each pebbled twice, as well as the vertices α_{3i+1}. However, all of the vertices in Δ_{3i+1} are parents of vertices in P. Thus, a blue pebble on any of these nodes must be returned to red, and then subsequently discarded. That is, a blue pebble will replace at most one red move in the pebbling strategy of Lemma 6.2. The number of the other red moves is unaffected so that the overall cost cannot be cheaper. Therefore, no pebbling strategy has a better cost than $\left(\frac{3n^3+15n^2+40n+14+12c}{2}\right) c_r$. □

Reminder of Lemma 6.3. *If ϕ is unsatisfiable, then the pebbling cost of H_ϕ with capacity $3n+4$ is greater than*

$$\left(\frac{3n^3+15n^2+40n+14+12c}{2}\right) c_r.$$

Proof of Lemma 6.3: By the construction of the DAG H_ϕ, if ϕ is unsatisfiable, then H_ϕ has pebbling number at least $3n+5$. Thus, if H_ϕ has capacity $3n+4$, any pebbling strategy must have a blue pebble at some point.

As in the proof of Lemma 6.2, any optimal strategy only places blue pebbles on nodes that the strategy of Lemma 6.2 pebbles twice. Again, the only nodes that are pebbled multiple times are the pyramids Δ_{3i+1}, which are each pebbled twice, as well as the vertices α_{3i+1}. Since each vertex in Δ_{3i+1} is a parent of a vertex in P, a blue pebble on any of these nodes must be returned to red, and then subsequently discarded. Thus, a blue pebble will replace at most one red move in the pebbling strategy of Lemma 6.2. The remaining red moves are unaffected, so the overall cost is more expensive, under the assumption that blue pebbles are more expensive than red moves. Therefore, any pebbling strategy has a cost greater than $\left(\frac{3n^3+15n^2+40n+14+12c}{2}\right) c_r$. □

Reminder of Theorem 6.1. *For $c_b \leq 3nc_r$, the problem* $\mathsf{rbpeb}^{\|}$ *is* $\mathtt{NP}-\mathtt{Hard}$.

Proof of Theorem 6.1: First, we remark that given a DAG H_ϕ with some capacity m, as well as a complete pebbling strategy as the certificate, the certificate can be verified in polynomial time by checking the validity of each step in the pebbling strategy. Thus, the computation of $\mathsf{rbpeb}^{\|}(H_\phi)$ is in $\mathtt{NP}$.

We now reduce $3-\mathsf{SAT}$ to the computation of $\mathsf{rbpeb}^{\|}(H_\phi)$. Now, given an instance ϕ of $3-\mathsf{SAT}$ with n variables, we construct the above DAG H_ϕ. This procedure clearly takes polynomial time. Moreover, by Lemma 6.2, if ϕ is satisfiable, then the optimal pebbling cost of H_ϕ with capacity $3n+4$ is exactly

$$\left(\frac{3n^3+15n^2+40n+14+12c}{2}\right) c_r.$$

On the other hand, by Lemma 6.3, if ϕ is unsatisfiable, then the pebbling cost of H_ϕ with capacity $3n+4$ is greater than

$$\left(\frac{3n^3+15n^2+40n+14+12c}{2}\right) c_r.$$

Thus, the computation of $\mathsf{rbpeb}^{\|}(H_\phi)$ distinguishes whether ϕ is satisfiable or not. Since $3-\mathsf{SAT}$ is $\mathtt{NP}-\mathtt{Hard}$, it follows that the computation of $\mathsf{rbpeb}^{\|}(H_\phi)$ is $\mathtt{NP}-\mathtt{Hard}$. □

B Background on the Gilbert *et al.* Black Pebbling Reduction

Recall that Gilbert *et al.* [GLT79] showed that the minimum space black pebbling problem was $\mathsf{PSPACE}-\mathsf{Hard}$ by reduction from the Truly Quantified Boolean Formula (TQBF) problem. Gilbert *et al.* [GLT79] provide a construction from any instance of TQBF to a DAG G_{TQBF} with pebbling number $3n+3$ if and only if the instance is satisfiable. Here, the pebbling number of a DAG G is $\min_{P=(P_1,\ldots,P_t)\in\mathcal{P}^{\|}} \max_{i\leq t} |P_i|$ is the number of pebbles necessary to pebble G. An important gadget in their construction is the so-called pyramid DAG. We use both Δ_k and a triangle with the number k inside to denote a k-pyramid (see Figure 2 for an example of a 3-pyramid). The key property of these DAGs is that *any* legal pebbling $P=(P_0,\ldots,P_t)\in\mathcal{P}^{\|}(\Delta_k)$ of a k-pyramid requires at least $\min_i |P_i| \geq k$ pebbles on the DAG at some point in time.

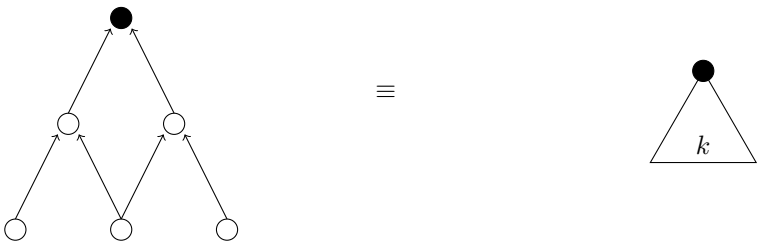

Fig. 2. A 3-Pyramid.

Another gadget, which appears in Figure 3, is the existential quantifier gadget, which requires that s_i, s_i-1, and s_i-2 pebbles must be placed in each of the pyramids to ultimately pebble q_i.

Any instance of TQBF in which each quantifier is an existential quantifier requires at most a quadratic number of pebbling moves. Specifically, we look at instances of `3-SAT`, such as in Figure 4. In such a graph representing an instance of `3-SAT`, the sink node to be pebbled is q_n. By design of the construction, any true statement requires exactly three pebbles for each pyramid representing a clause. On the other hand, a false clause requires four pebbles, so that false statements require more pebbles. Thus, by providing extraneous additions to the construction which force the number of pebbling moves to be a known constant, we can extract the pebbling number, given the space-time complexity. For more details, see the full description in [GLT79].

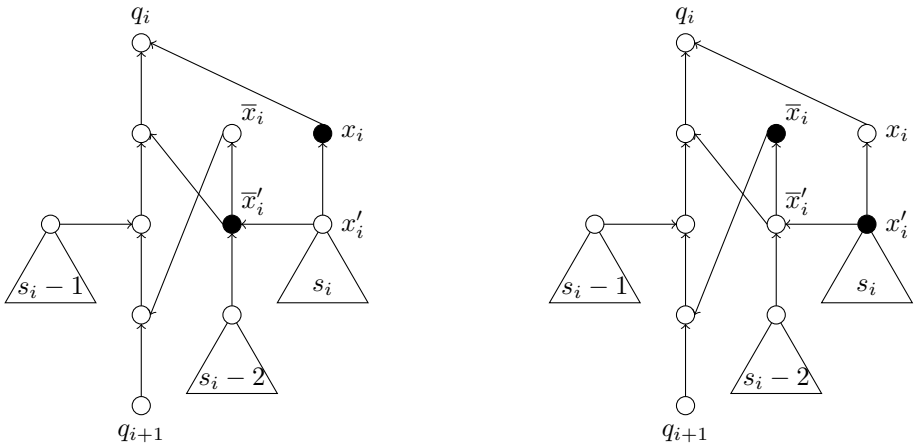

Fig. 3. An existential quantifier, with x_i set to true in the left figure and x_i set to false in the right figure.

Lemma B.1. *[GLT79] The quantified Boolean formula $Q_1x_1Q_2x_2\cdots Q_nx_nF_n$ is true if and only if the corresponding DAG G_{TQBF} has pebbling number $3n+3$.*

Lemma B.2. *[BZ18] Suppose that we have a satisfiable TQBF formula $Q_1x_1Q_2x_2\cdots Q_nx_nF_n$ with $Q_i = \exists$ for all $i \leq n$. Then there is a legal sequential pebbling $P = (P_0, \ldots, P_t) \in \mathcal{P}(G_{TQBF})$ of the corresponding DAG G_{TQBF} from [GLT79] with $t \leq 6n^2 + 33n$ pebbling moves and $\max_{i\leq t} |P_i| \leq 3n+3$.*

Proof. We repeat the proof of [BZ18] for completion. We describe the pebbling strategy of Gilbert *et al.* [GLT79], and analyze the pebbling time of their strategy. Let $T(i)$ be the time it takes to pebble q_i in the proposed construction for any instance with i variables, i clauses, and only existential quantifiers.

Suppose that x_i is allowed to be true for the existential quantifier $Q_i = \exists$. Then vertex x'_i is pebbled using s_i moves, where $s_i = 3n-3i+6$. Similarly, vertices d_i and f_i are pebbled using $s_i - 1$ and $s_i - 2$ moves respectively. Additionally, f_i is moved to $\overline{x}'_i$ and then x_i is moved to x'_i in the following step, for a total of two more moves. We then pebble q_{i+1} using $T(i+1)$ moves and finish by placing a pebble on $\overline{x}_i$ and moving it to c_i, b_i, a_i, and q_i, for five more moves. Finally, we use six more moves to pebble an additional clause. Thus, in this case,

$$T_{true}(i) = s_i + (s_i - 1) + (s_i - 2) + 13 + T(i+1).$$

On the other hand, if x_i is allowed to be false for the existential quantifier $Q_i = \exists$, then first we pebble x'_i, d_i, and f_i sequentially, using s_i, $s_i - 1$, and $s_i - 2$ moves respectively. We then move the pebble from f_i to $\overline{x}'_i$ and then to $\overline{x}_i$, for a total of two more moves. We then pebble q_{i+1} using $T(i+1)$ moves. The pebble on q_{i+1} is subsequently moved to c_i and then b_i, using two more moves. Picking up all pebbles except those on b_i and x'_i, and using them to pebble f_i takes $s_i - 2$

more moves. Additionally, the pebble on f_i is moved to $\overline{x}'_i$ and then a_i, while the pebble on x'_i is moved to x_i and then q_i, for four more moves. Finally, we use six more moves to pebble an additional clause. In total,

$$T_{false}(i) = s_i + (s_i - 1) + (s_i - 2) + (s_i - 2) + 14 + T(i + 1).$$

Therefore,

$$T(i) \leq 4s_i + 10 + T(i + 1),$$

where $s_i = 3n - 3i + 6$. Thus,

$$T(i) \leq 12(n - i) + 34 + T(i + 1).$$

Writing $R(i) = T(n - i)$ then gives

$$R(i) \leq 12i + 34 + R(i - 1),$$

so that $R(n) \leq \sum_{i=1}^{n}(12i + 34) = 6n^2 + 40n$. Hence, it takes at most $6n^2 + 40n$ moves to pebble the given construction for any instance of TQBF which only includes existential quantifiers.

C Figures and Candidate iMHFs

In this section we give provide detailed descriptions of the iMHFs analyzed in the main body of the paper. DRSample is described in Algorithm 1, aATSample is described in Algorithm 2, Argon2iB is described in Algorithm 3 and Argon2iA is described in Algorithm 4.

Algorithm 1: An algorithm for sampling depth-robust graphs. [ABH17]

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
  $E := E \cup \{(v, \mathsf{GetParent}(v, i))\}$ // 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 $v - r$

Algorithm 2: An algorithm for sampling a high aAT graph. [ABH17]

Function aATSample($H = (\bar{V} = [n], \bar{E})$, $c \in (0, 1)$):

$V := [2n]$
$E := \bar{E} \cup \{(i, i + 1) \ : i \in [2n - 1]\}$
for $v \in [n + 1, 2n]$ *and* $i \in [2]$ **do** // Populate new edges of graph.
  $E := E \cup \{(v, \mathsf{GetParent}^c(v, i))\}$ // Get i^{th} parent of node v
end
return $G := (V, E)$.

Function GetParent^c(v,i):

if $i = 1$ **then**
  $u := i - 1$
end
else if $v \leq n$ **then**
  $u := \mathsf{GetParent}_H(v, i)$
  // DRSample
end
else
  $m := \lfloor c \log(n) \rfloor$
  $b := (v - n) \mod \lfloor n/m \rfloor$
  $u := bm$
end
return u

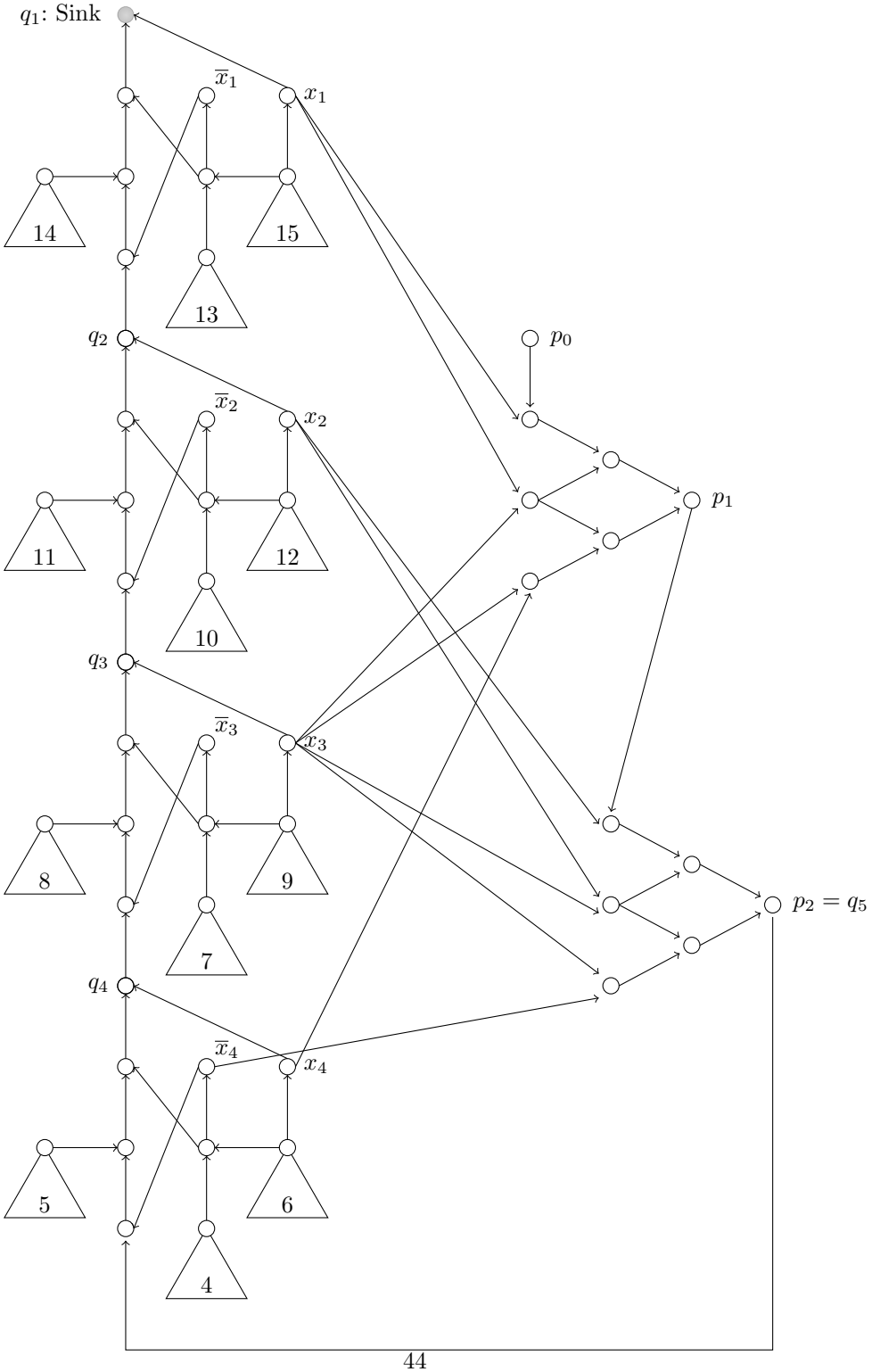

Fig. 4. Graph G_{TQBF} for $\exists x_1, x_2, x_3, x_4$ s.t. $(x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee \overline{x}_4)$.

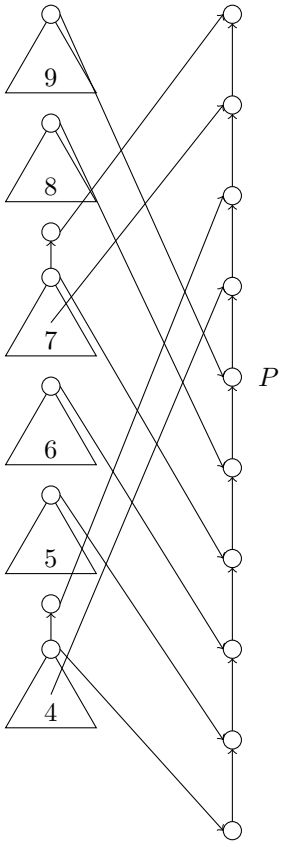

Fig. 5. Path P for H_ϕ.

Algorithm 3: An algorithm for sampling depth-robust graphs. [BDK15]

Function Argon2iB($n \in \mathbb{N}_{\geq 2}$):

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

Function GetParent(v,i):

if $i = 1$ **then**
  $u := i - 1$
else
  $N := 2^{32}$ // Set sample range.
  $g \leftarrow [1, N]$ // Get random range length.
  $r := \left\lceil \frac{g^2}{N^2} v \right\rceil$ // Set quadratic dependency.
end
return $v - r$

Algorithm 4: An algorithm for sampling depth-robust graphs. [BDK15]

Function Argon2iA($n \in \mathbb{N}_{\geq 2}$):

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

Function GetParent(v,i):

if $i = 1$ **then**
  $u := i - 1$
else
  $N := 2^{32}$ // Set sample range.
  $g \leftarrow [1, N]$ // Get random range length.
  $r := \left\lceil \frac{g}{N} v \right\rceil$ // Set linear dependency.
end
return $v - r$