Efficiently Computing Data-Independent Memory-Hard Functions

Joël Alwen[1] and Jeremiah Blocki[2,3(✉)]

[1] IST Austria, Klosterneuburg, Austria
[2] Microsoft Research, Cambridge, USA
jblocki@microsoft.com
[3] Purdue, West Lafayette, USA

Abstract. A memory-hard function (MHF) f is equipped with a *space cost* σ and *time cost* τ parameter such that repeatedly computing $f_{\sigma,\tau}$ on an application specific integrated circuit (ASIC) is not economically advantageous relative to a general purpose computer. Technically we would like that any (generalized) circuit for evaluating an iMHF $f_{\sigma,\tau}$ has area $\times$ time (AT) complexity at $\Theta(\sigma^2 * \tau)$. A data-independent MHF (iMHF) has the added property that it can be computed with almost optimal memory and time complexity by an algorithm which accesses memory in a pattern independent of the input value. Such functions can be specified by fixing a directed acyclic graph (DAG) G on $n = \Theta(\sigma * \tau)$ nodes representing its computation graph.

In this work we develop new tools for analyzing iMHFs. First we define and motivate a new complexity measure capturing the amount of *energy* (i.e. electricity) required to compute a function. We argue that, in practice, this measure is at least as important as the more traditional AT-complexity. Next we describe an algorithm $\mathcal{A}$ for repeatedly evaluating an iMHF based on an arbitrary DAG G. We upperbound both its energy and AT complexities per instance evaluated in terms of a certain combinatorial property of G.

Next we instantiate our attack for several general classes of DAGs which include those underlying many of the most important iMHF candidates in the literature. In particular, we obtain the following results which hold for all choices of parameters σ and τ (and thread-count) such that $n = \sigma * \tau$.

- The Catena-Dragonfly function of [FLW13] has AT and energy complexities $O(n^{1.67})$.
- The Catena-Butterfly function of [FLW13] has complexities is $O(n^{1.67})$.
- The Double-Buffer and the Linear functions of [CGBS16] both have complexities in $O(n^{1.67})$.
- The Argon2i function of [BDK15] (winner of the Password Hashing Competition [PHC]) has complexities $O(n^{7/4}\log(n))$.
- The Single-Buffer function of [CGBS16] has complexities $O(n^{7/4}\log(n))$.
- *Any* iMHF can be computed by an algorithm with complexities $O(n^2/\log^{1-\epsilon}(n))$ for all $\epsilon > 0$. In particular when $\tau = 1$ this shows

M. Robshaw and J. Katz (Eds.): CRYPTO 2016, Part II, LNCS 9815, pp. 241–271, 2016.
DOI: 10.1007/978-3-662-53008-5_9

that the goal of constructing an iMHF with AT-complexity $\Theta(\sigma^2 * \tau)$ is unachievable.

Along the way we prove a lemma upper-bounding the depth-robustness of any DAG which may prove to be of independent interest.

1 Introduction

Moderately hard to compute functions have proven to be useful security primitives. In this work we focus on "memory-hard functions" (MHF) introduced in [Per09]. These aim to serve as password hashing algorithms for storing passwords in a login system, as Key Derivation Functions (also called "key stretching" functions) for password-based cryptography and for building Proof-of-Effort protocols (in particular for use in cryptocurrencies such as Litecoin [Cha11, Bil13] and others). In each case the main security property we would like to achieve for the MHF is that brute-force attacks (i.e. evaluating the MHF on many inputs) using an application-specific integrated circuit (ASIC) should not be economically viable.

1.1 Memory-Hard Functions and Their Complexity

We interpret this intuitive goal in two ways. Either the cost of *building* the ASIC should be prohibitively expensive (in terms of say USD) or the cost of *running* the ASIC should be prohibitively expensive. In fact, given that the former is a one-time cost which can be amortized over the life-time of the device while the later is a recurring cost, it may often be the case that the later is the most interesting goal to achieve.

The cost of building a circuit is often approximated by its *AT-complexity* [Tho79, BL13, BK15, AS15]; that is the product of the area of the chip and the time it takes the chip to produce the output. In this work we consider MHFs built as modes of operation over an underlying compression function H. Thus, we measure time in units of *tocks*; namely the time it takes to evaluate one instance of H from start to finish[1]. We measure area in units of "memory-area" (MAr); namely the area required to store one output of H (called a *block*). Finally we parametrize our AT-complexity notion AT_R with the *core-memory area ratio* [BK15] $R > 0$, a positive real denoting the number of MAr required to implement one copy of H.[2]

To estimate the cost of running the chip we will use a new notion which we call the "energy-complexity" or *E-complexity* of the circuit. Intuitively, it approximates the energy (say in kilo-Watt-hours) used in an execution of the chip. More precisely the unit of measure for E-complexity is a "memory-Watt-tock" (MWt) – the number of kWh it takes to store one block for one tock. We also parametrize the complexity notion $E_{\bar{R}}$ with the *core-memory energy ratio*,

[1] I.e. without considering pipelining and other such amortized optimizations.

[2] This allows our analysis to be applied regardless of the particular VLSI technology employed and the particular implementation of H used when constructing the ASIC.

a positive real $\bar{R} > 0$ which is the number of MWt required to evaluate one instance of H.

To see why this is an interesting measure for achieving our stated security goal consider the case of password hashing. (The case for KDFs follows essentially the same reasoning.) Suppose an attacker manages to pilfer the credentials file from a login-server and now executes an off-line brute-force attack $\mathcal{A}$ implemented in an ASIC with core-memory energy ratio $\bar{R}$ using $E_{\bar{R}}(\mathcal{A})$ MWt per password guess. We model the monetary income from such an attack as being proportional to the number of password guesses made which we denote by $\#eval$.[3] Conversely, we model the running cost as being proportional to the electricity consumed by the ASIC while executing the attack, namely its $E_{\bar{R}}$-complexity times $\#eval$. The attacker can always increase income (i.e. increase $\#eval$) simply by adding more implementations of $\mathcal{A}$ to the ASIC or running the ASIC for more time. Therefore, the attack is profitable (in this model) if and only if the USD cost c of one MWt and the income i per password guess are such that $i > c * E_{\bar{R}}(\mathcal{A})$. Thus we can use $E_{\bar{R}}(\mathcal{A})$ as a key indicator for which ranges of (c, i) an attack is economically viable.

Quality of an Attack. A candidate MHF F is specified via an algorithm which evaluates it. (E.g. [Per09,FLW13,BDK15].) We refer to this algorithm as the *naïve* algorithm $\mathcal{N}$ for F and it is understood to be the algorithm used by the honest party. $\mathcal{N}$ is intended to be an algorithm that can be evaluated efficiently on *typical* (i.e. general purpose) computer architectures — where we may not be able to evaluate H multiple times in parallel. As usual, we are interested in what advantage an adversarial evaluation algorithm can have over the honest party. Therefore, one measure of the quality of a given algorithm $\mathcal{A}$ for evaluating (multiple instances of) an MHF F is to compare its complexity to that of $\mathcal{N}$. In particular for given core-memory ratios R and $\bar{R}$ the *AT-quality* and *energy-quality* of $\mathcal{A}$ are given by

$$\mathsf{AT\text{-}quality}_R(\mathcal{A}) = \frac{AT_R(\mathcal{N})}{AT_R(\mathcal{A})} \quad \text{and} \quad \mathsf{E\text{-}quality}_{\bar{R}}(\mathcal{A}) = \frac{E_{\bar{R}}(\mathcal{N})}{E_{\bar{R}}(\mathcal{A})}.$$

Here, $AT_R(\mathcal{A})$ (resp. $E_{\bar{R}}(\mathcal{A})$) measures the *amortized* AT complexity (resp. *amortized* energy complexity).[4] That is $AT_R(\mathcal{A})$ is smallest $AT_{\mathbb{R}}$ complexity of a chip implementing $\mathcal{A}$ divided by $\#inst(\mathcal{A})$ — the number of instances of F computed in an execution of $\mathcal{A}$. We consider $\mathcal{A}$ an "attack" if either one of these quality measures is greater than 1. (However we remark that all attacks in this work have both qualities *simultaneously* tending towards infinity as $\#inst$ grows.)

Data-Independent and Ideal MHFs. An *data-independent* memory-hard function (iMHF) is a function f for which the associated naïve algorithm $\mathcal{N}$, on

[3] Intuitively, the more passwords guesses made the higher the expected number of password (equivalents) recovered by the adversary which can then be monetized.

[4] Generally, unless explicitly specified otherwise, we are only interested in the *amortized* AT and energy complexities per instance of the MHF computed.

input x, computes $f(x)$ using a memory access pattern that is independent of x. These take on special importance in applications where the MHF is to be evaluated on secret input in an (at least somewhat) hostile environment. This is because (in contrast to their siblings data-*dependent* MHFs) it is much easier to implement an iMHF in such a way that it avoids information leakage via certain side-channel attacks such as Timing attacks. In these attacks, the variation in the time taken to perform certain operations is used to deduce information about the inputs upon which the MHF is being evaluated. Similar attacks have in the past been mounted by local adversarial processes [BM06], adversarial virtual machines in a cloud environment [RTSS09] or even completely remotely [Ber, ASK07]. Therefore, in the context of both KDFs and password hashing data-independence is a desirable property. All MHFs considered in this work are of this form.

In general, an iMHF f can be described via a fixed DAG G representing its computation graph. Each node represents an intermediary value, which is computed via some deterministic round function, using the values represented by the parent nodes in G (e.g. via a single call to H). The source node of G represents the input x while $f(x)$, the output of the computation, is the value represented by the sink node.

Let f be an iMHF given by some DAG G of size n with constant in-degree. There exists a trivial algorithm triv which can always compute f with AT and energy complexities $\Theta(n^2)$.[5] Given a constant $c > 1$ we consider f to be a *c-ideal* iMHF if, when we take the naïve algorithm to be triv, there exist no attack $\mathcal{A}$ on f with better quality than c (i.e. $\forall \mathcal{A}\ \mathsf{E\text{-}quality}_{\bar{R}}(\mathcal{A}) \leq c$ and $\mathsf{AT\text{-}quality}_{R}(\mathcal{A}) \leq c$). A primary goal of research in this field is to find an ideal iMHF.[6]

1.2 MHF Candidates

Due to the growing interest in MHFs there are a number of candidate functions. For example in the recently completed Password Hashing Competition [PHC] most entrants claimed some form of memory-hardness. The goal of the PHC was to select a winning algorithm to act as a new standard for password hashing.

Catena. To the best of our knowledge the earliest candidate iMHF is the PHC finalist Catena [FLW13]. It received special recognition for its agile framework and its resistance to side-channel attacks. In [FLW13] the authors proposed two different DAGs giving rise two separate functions. The first, called Catena Bit Reversal, is based on an λ-layered graph BRG^n_λ with n nodes. The second is called Catena Double Butterfly and is based on a different $O\big(\lambda \log n\big)$-layered graph DBG^n_λ. The Catena designers recommended choosing $\lambda \in \{1, 2, 3, 4\}$ [FLW13].

[5] Simply compute each intermediary value in topological order, one value at a time, storing all results in memory until the computation is complete.

[6] Hopefully one permuting as simple as possible an explicit description and naïve implementation and as lightweight as possible round-function.

Argon2. One of the most important MHF candidates is Argon2 [BDK15]. Notably, it is the winner of the Password Hashing Competition [PHC]. Argon2 is equipped with a data-dependent mode of operation and an independent mode which is called Argon2i. Argon2i is recommended for password hashing.

Balloon Hashing. Most recently, three new candidate iMHFs are proposed in [CGBS16]. These are called the Single-Buffer (SB), Double-Buffer and Linear constructions respectively and are jointly referred to as the Balloon Hashing constructions. The authors provide strong evidence for the memory-hardness of all three candidates albeit assuming the absence of parallelism.

In general iMHF candidates are equipped with a *space-cost* parameter σ (in which the memory required per evaluation is intended to scale) and a *time-cost* parameter τ (in which the time required for an evaluation is intended to scale). Additionally, Argon2i, the Double-Buffer and the Linear functions are also equipped with a *parallelism* parameter ϕ the property that the naïve algorithm can make efficient use of (up to) ϕ concurrent threads. Viewing these functions as DAGs gives rise to a graph on $n = \sigma * \tau$ nodes with depth n/ϕ. The hope is that for all settings of (σ, τ, ϕ) the AT and energy complexity lie in $\Theta(\sigma^2 * \tau/\phi)$.

1.3 Our Contributions

In this work we introduce and motivate the notion of (amortized) energy complexity. Next we give a generic evaluation algorithm PGenPeb for data-independent iMHFs based on arbitrary DAG G. We analyze PGenPeb's energy and AT complexities in terms of a combinatorial property of G. In particular, we obtain an attack against any iMHF for which G is not *depth-robust*. Informally, a DAG G is not depth-robust if there is a relatively small set S of nodes such that after removing S from G the resulting graph (denoted $G - S$) has low depth (i.e. contains only short paths).

We instantiate the attack for various classes of DAGs. In particular, we exhibit a "depth-reducing" node set S for the Argon2i DAG, both types of Catena DAGs and all three Balloon Hashing DAGs. For example, for any parameters $(\sigma, \tau, \phi = 1)$ with $n = \sigma * \tau$ we obtain an attack on both Catena, the Double-Buffer and the Linear iMHFs with quality $\Omega\big(n^{1/3}\big)$. Similarly we demonstrate an attack on Argon2i and the Single-Buffer iMHF with quality $\Omega\big(\frac{n^{1/4}}{\ln n}\big)$.[7]

In fact we demonstrate that no DAG with constant indegree is sufficiently depth-robust to completely resist the attack. More precisely, we show that any iMHF is at best c-ideal for $c = \Omega\big(\log^{1-\epsilon} n\big)$ and any $\epsilon > 0$. In particular this means that ideal iMHFs, as described above, do not exist.

[7] For the cases when $\phi > 1$ PGenPeb maintains the same complexities but the resulting quality decreases somewhat as the complexity of the naïve algorithm improves for Argon2i, the Double-Buffer and the Linear functions. In other words quality decreases not because memory-hardness increases but because the honest algorithm becomes more efficient.

General Attack on Non-depth Robust DAGs. We first present in Sect. 3, a generic evaluation algorithm GenPeb which takes as inputs a node subset S. Because G is not depth-robust there exists a small set S of nodes such that $d = \mathsf{depth}(G-S)$ is relatively small. The basic idea behind our attack is to divide computation steps into two phases: balloon phases and light phases. Each light phase lasts roughly $g \gg d$ time steps. During light phases we discard most of the values that we have computed from memory keeping only values corresponding to nodes in S, the highest node i whose value has been computed and the parents of the nodes whose values we plan to compute in the next g time steps. As the name suggests, light phases are cheap. Our memory usage is low during these light phases and we will compute one instance of the round function (e.g. call to H) during each time step. During a Balloon Phase we quickly restore all of the discarded values to memory so that we can complete the next light phase. Unlike light phases, the balloon phases are more expensive because we are storing up to $O(n)$ values in memory and because we will often make multiple calls to the round function in parallel. However, the key observation is that we will not incur these higher cost in too many time steps. In particular, because the graph $G - S$ has small depth $d \ll g$ and we never discard values for nodes in S the Balloon Phase can be completed very quickly (i.e., in at most $d \ll g$ times steps) by making parallel calls to the round function.

While for any non-depth-robust graph the GenPeb algorithm has good energy complexity, obtaining an evaluation algorithm with low AT-complexity requires a bit more work. Notice that during a light phase most of the memory capacity and round function implementations needed for a balloon phase are no longer being used. Moreover light phases run for significantly more time than the balloon phases. These observations give rise to the low AT-complexity parallel algorithm PGenPeb which evaluates g/d instances of the iMHF concurrently such that at any given time only a single instance is in a balloon phase while all other instances are in light phases. Intuitively this results in more efficient use of available hardware while technically we get that the energy complexity of the algorithm is approximately equal to the AT complexity (Theorem 3).

Stacked Sandwich Graphs. In Sect. 4 we focus on two classes of DAGs called (strict) stacked sandwich graphs. Informally, a DAG G is a λ-stacked sandwich DAG if the nodes can be partitioned into $\lambda+1$ layers such that, with the possible exception of node i, all of the parents of node $i+1$ are from previous layers. These classes include the DAGs implicit to both Catena iMHFs as well as the Double-Buffer and Linear iMHFs. We prove that no λ-stacked sandwich graph is depth-robust (Lemma 1). For any $t > 1$ there is a set S of n/t nodes such that $\mathsf{depth}(G - S) \leq (\lambda + 1)t$.

(n, δ, w)-Random Graphs. In Sect. 5 we turn to a class of random graphs called (n, δ, w)-random DAGs. We remark that the graphs implicit to Argon2i and the Single-Buffer iMHF (for a randomly chosen salt) fall into this category of random DAGs. We show (in Lemma 4) that, with high probability, by removing just a few nodes these graphs can be transformed into stacked sandwich graphs and are thus not depth-robust.

Attack on any iMHF. In Sect. 6 we prove that no DAG with constant indegree is sufficiently depth-robust to resist at least some form of attack (Theorem 8). In our proof, we rely on a result due to Valiant [Val77] which states that for any DAG G with m edges and depth d there is a set S of $m/\log d$ edges s.t. by deleting them we obtain a graph of depth at most $d/2$ (see Lemma 6). Given $\epsilon > 0$ we can repeatedly apply this result obtain a set S of $o\left(\frac{\delta n}{\log^{1-\epsilon} n}\right)$ nodes s.t $\mathsf{depth}(G - S) \leq \frac{n}{\log^2 n}$. Thus if we let the naïve algorithm be (any algorithm complexity comparable to) triv then we have a generic attack $\mathcal{A}$ with quality $\mathsf{AT\text{-}quality}_R(\mathcal{A}) = \Omega\left(\delta^{-1} \log^{1-\epsilon} n\right)$ and $E_{\bar{R}}(\mathcal{A}) = \Omega\left(\delta^{-1} \log^{1-\epsilon} n\right)$.

Exact Security Analysis. Finally we present exact bounds for the energy and AT complexities of all of our attacks. Our analysis demonstrate that our attacks have high quality for practical values of n and $\bar{R}$ — not just as $n \to \infty$. For example setting $n = 2^{18}$ we already have an attack $\mathcal{A}$ against Argon2i with $\mathsf{AT\text{-}quality}_R(\mathcal{A})$, $\mathsf{E\text{-}quality}_{\bar{R}}(\mathcal{A}) > 1$ — using a realistic value $\bar{R} = 3,000$. In general, $\mathsf{E\text{-}quality}_{\bar{R}}(\mathcal{A})$ will increase as n increases or as $\bar{R}$ decreases.

1.4 Related Work

The intuitive goal of constructing functions for which VLSI implementations are prohibitively expensive was first laid out by Percival in [Per09]. This property was formalized by asking that evaluating such a function on a PRAM requires large ST-complexity. In particular evaluation algorithms with low amortized complexity such as those in this work were not considered. Percival also introduced the first, and currently most widely deployed, candidate MHF called `scrypt`. A full proof of security under a strong security definition remains a central open problem in the area. However recently significant progress has been made in this direction in [ACK+16]. It is interesting to note though that despite `scrypt` being data-*dependent* the (conditional) lower bound in [ACK+16] still does not exceed the upper-bound of Sect. 6 on the best possible quality of an iMHF.

Catena. In [FLW13] the authors of Catena restricted their analysis of its security to a sequential setting. That is they restrict an adversary to only being able to evaluate one instance of the underlying function H at a time. In this setting and for the case when $\lambda = 1$ the results of [LT82] show that, in a simplified computational model, BRG_1^n has ST-complexity $\Omega\left(n^2\right)$. Here ST-complexity denotes the product of the space and time required by any algorithm which evaluates Catena Bit Reversal. The intuition being that large ST-complexity implies large AT-complexity of any implementation in a custom chip.

Argon2. Argon2 [BDK15] was the winner of the Password Hashing Competition [PHC]. Argon2 is equipped with a data-dependent mode of operation and an independent mode which is called Argon2i. The authors recommend using Argon2i for password hashing due to its resistance to side channel attacks [BDK15]. Our attacks only apply to Argon2i, the data independent mode. Recently, Corrigan-Gibbs et al. [CGBS16] gave an attack

on Argon2i which reduces the cost of computing Argon2i by a factor of 4.

Balloon Hashing. In [CGBS16] the authors also proposed three iMHFs which resist their attack on Argon2i. These are called Single-Buffer (SB), Double-Buffer and Linear and collectively referred to as the Balloon Hashing iMHFs.[8] Our attacks reduce the cost of computing both Argon2i and SB by a factor of $\tilde{\Omega}\left(n^{1/4}\right)$.

A Provably Secure MHF. Currently, the only candidate MHF equipped with a full proof of security is the one in [AS15]. There, the authors show an iMHF F for which the energy-complexity of the required storage alone (i.e. disregarding the cost of evaluating the round function) is within a polylogarithmic factor in n of the energy-complexity of the trivial algorithm triv. Moreover triv uses only a single instance of H (i.e. it is sequential) which implies that, roughly speaking, any evaluation algorithm for F can have $\mathsf{E\text{-}quality} = O(\mathsf{polylog}(n))$. The results in Sect. 6 show that this is optimal for any iMHF up to the exponent in the polylogarithmic factor.

Attacking MHFs. The Catena Dragonfly iMHF has been attacked previously [BK15, AS15]. In particular, [AS15] demonstrated an attack on Catena Dragonfly $\mathsf{BRG}^n_{\lambda=1}$ which has energy quality $\mathsf{E\text{-}quality} = O\left(\sqrt{n}\right)$. The attack from [BK15] has slightly worse quality $O\left(n^{1/5}\right)$, but it applies even for Dragonfly variants in which $\lambda > 1$. At a high level the ideas behind both of these attacks is to divide memory into segments, store the leading block in each segment and then recompute the remaining blocks as needed. These attacks only work because the underlying Catena Dragonfly DAG BRG^n_λ allows for quick re-computation of the remaining blocks. In this work we observe that this key idea can be generalized to attack *any* non depth-robust iMHF. In particular, our techniques can be used to attack other iMHFs like Catena Butterfly, Argon2i [BDK15] and SB [CGBS16]. In fact, our attacks can be extended to *any* iMHF because no DAG is sufficiently depth-robust to resist at least some form of attack.

Memory-Bound Functions. An important precursor to memory-hard functions are memory-*bound* functions. First introduced in [ABMW05] here the complexity measure of interest is the number of cache misses required to evaluate the function. On the highest level the motivation is the same as that of memory-hard functions; namely to build moderately hard functions which are more equally hard across different computational devices (compared to the rather unbalanced notion of plain computational complexity). In particular it was observed that while computational speeds may vary greatly between different devices the same is not as true for memory latency speeds [DGN03]. In contrast memory-hard

[8] Corrigan-Gibbs et al. [CGBS16] use "Balloon Hashing" as a title for their iMHF however this similarity with the balloon phase in our evaluation algorithm is a slightly unfortunate coincidence.

functions aim to achieve egalitarian hardness by making the cost of custom hardware prohibitively large [Per09]. The first provably secure memory-bound function was (implicitly) given in [DGN03] where it was used to construct a protocol for fighting SPAM email. The construction was later improved in [DNW05] which was also the first result in cryptography to make use of a version of the pebbling model of computation; a technique later adapted in [AS15].

Password Storage. Recent high-profile security breaches (e.g., RockYou, Sony, LinkedIN, Ashley Madison[9]) highlight the importance of proper password storage practices like salting [Ale04] and key stretching [MT79][10]. However, hash iteration, the technique used by password hash functions like PBKDF2 [Kal00] and `bcrypt` [PM], is typically an insufficient defense against an adversary who could build customized hardware to evaluate the underlying hash function. In particular, the cost of computing a hash function H like SHA256 or MD5 on an ASIC is orders of magnitude smaller than the cost of computing H on traditional hardware [DGN03, NB+15]. By contrast, memory costs tend to be relatively stable across different architectures [DGN03], which motivates the use of memory-hard functions for password hashing [Per09].

Several orthogonal lines of research have explored defenses such as: distributing the storage and/or computation of a password hash across multiple servers (e.g., [BJKS03, CLN12]), storing fake password hashes on the server (e.g., [JR13, BBBB10]), the inclusion of secret salt values (e.g., "pepper") in password hashes [Man96, BD16] and the inclusion of the solution(s) to hard AI challenges in password hashes [CHS06, DC08, BBD13].

2 Preliminaries

We begin with some notation. Given a directed acyclic graph (DAG) $G = (V, E)$ of *size* $|V| = n$ and a subset $S \subseteq V$ we use $G - S$ to denote the resulting DAG after removing all nodes in S. We denote by $\mathsf{depth}(G)$ the length of the longest (directed) path in G and we denote by $\mathsf{indeg}(G)$ the maximum number of directed edges entering a single node. For integers $a \leq b$ we write $[a, b]$ as shorthand for the set $\{a, a+1, \ldots, b\}$ and we write $[a]$ for the set $[1, a]$.

We use $H_\lambda = \sum_{i=1}^{\lambda} \frac{1}{i}$ to denote the λ'th harmonic number. In particular H_λ can be approximated by the natural logarithm $H_\lambda \approx \ln \lambda$.

[9] See http://www.privacyrights.org/data-breach/ (Retrieved 9/1/2015).

[10] Users routinely select lower entropy password [Bon12], which are especially vulnerable to an offline attacker when the underling password hash function is inexpensive to compute. Furthermore, stricter password restrictions (e.g., requiring a mix of numbers and upper/lower case letters) [SS09] have not been found to greatly improve the entropy of the resulting passwords [KSK+11, BKPS13]. In fact, sometime these policies reduced the entropy of user selected passwords [KSK+11]. These policies are often associated with high usability costs [FH10].

2.1 Complexity and Quality of Attacks

We consider algorithms in the parallel random oracle model (pROM) [AS15] of computation.[11] That is an algorithm is repeatedly invoked. At invocation $i \in \{1, 2, \ldots\}$ the algorithm is given the state (bit-string) σ_{i-1} it produced at the end of the previous invocation. Next $\mathcal{A}$ can make a batch of calls $\mathbf{q}_i = (q_{1,i}, q_{2,i}, \ldots)$ to the underlying round function H (modeled as a random oracle (RO)). Then it receives the response from H and can perform arbitrary computation before finally outputting an updated state σ_i. The initial state σ_0 contains the input to the computation which terminates once a special final state is produced by $\mathcal{A}$. Apart from the explicit states σ the algorithm may keep no other state between invocations. For a input x and coins r we denote by $\mathcal{A}(x; r; H)$ the corresponding (deterministic) execution of $\mathcal{A}$.

We define the runtime $\mathsf{time}(\mathcal{A})$ to be the maximum running time of $\mathcal{A}$ in any execution (over all choices of x, r and H). Then the *cumulative memory complexity* (CMC) and *cumulative RO complexity* are defined as

$$\mathsf{cmc}(\mathcal{A}) = \max_{x,r,H} \sum_{i \in [T-1]} |\sigma_i| \qquad \mathsf{crc}(\mathcal{A}) = \max_{x,r,H} \sum_{i \in [T]} |\mathbf{q}_i|$$

where $|\sigma|$ is the bit-length of state σ, $|\mathbf{q}|$ is the dimension of the vector $\mathbf{q}$ and $\max_{x,r,H}$ denotes the maximum over all possible executions of $\mathcal{A}$. Similarly the *absolute memory complexity* (AMC) and *absolute RO complexity* are defined to be (ARC)

$$\mathsf{amc}(\mathcal{A}) = \max_{x,r,H} \max_{i \in [T-1]} |\sigma_i| \qquad \mathsf{arc}(\mathcal{A}) = \max_{x,r,H} \max_{i \in [T]} |\mathbf{q}_i|.$$

We remark that these complexity measures are stricter then is common, especially with respect to maximizing over all random oracles H. However we use them to upper-bound the complexity of our attacks so this strictness can only serve to strengthen the results.

Using these tools we can now define the complexity of an algorithm as follows.

Definition 1 (AT and Energy Complexities). *Let $\mathcal{A}$ be a pROM algorithm which computes $\#inst(\mathcal{A})$ instances of an iMHF in parallel. Then for any* core-memory area ratio $R > 0$ *and any* core-memory energy ratio $\bar{R} > 0$ *the (amortized)* AT-complexity *and the (amortized)* energy-complexity *of $\mathcal{A}$ are defined to be*

$$AT_R(\mathcal{A}) = [\mathsf{amc}(\mathcal{A}) + R \cdot \mathsf{arc}(\mathcal{A})] \times \frac{\mathsf{time}(\mathcal{A})}{\#inst(\mathcal{A})} \qquad E_{\bar{R}}(\mathcal{A}) = \frac{\mathsf{cmc}(\mathcal{A}) + \bar{R} \cdot \mathsf{crc}(\mathcal{A})}{\#inst(\mathcal{A})}.$$

Finally we can define the quality of an attack in terms of how much (if at all) it improves on the naïve algorithm

[11] Alternatively the results in this work also apply to the random access machine model of computation.

Definition 2 (Attack Quality). *Let f be an MHF with naïve algorithm $\mathbb{N}$ and let $\mathcal{A}$ be a pROM algorithm for evaluating $\#inst(\mathcal{A})$ instance(s) of f. Then for any* core-memory area ratio $R > 0$ *and any* core-memory energy ratio $\bar{R} > 0$ *the* AT-quality *and* energy-quality *of $\mathcal{A}$ is defined to be*

$$\mathsf{AT\text{-}quality}_R(\mathcal{A}) = \frac{AT_R(\mathcal{N})}{AT_R(\mathcal{A})} \qquad \mathsf{E\text{-}quality}_{\bar{R}}(\mathcal{A}) = \frac{E_{\bar{R}}(\mathcal{N})}{E_{\bar{R}}(\mathcal{A})}.$$

In particular if either quantity is less than 1 *then we call $\mathcal{A}$ an* attack *on f.*

Let f be an iMHF based on some DAG G of size n with constant in-degree. Observe f can always be evaluated by computing one intermediate value at a time in topological order while never deleting a computed value. Clearly this always results in correctly computing f and it corresponds to a well defined pROM algorithm triv for evaluating f. Moreover $AT_R(\mathsf{triv}) = \Theta(n(n+R))$ and $E_{\bar{R}}(\mathsf{triv}) = \Theta(n(n+\bar{R}))$. Given a constant $c > 0$ we say that f is a *c-ideal* iMHF if, when $\mathsf{triv} = \mathcal{N}$ is the naïve, for any attack $\mathcal{A}$ we have $\mathsf{AT\text{-}quality}_R(\mathcal{A}) \geq c$ and $\mathsf{E\text{-}quality}_{\bar{R}}(\mathcal{A}) \geq c$. This is motivated by the observation that for any iMHF algorithm triv is always a possible way to evaluate it. An ideal iMHF captures the property that triv is (approximately) the best evaluation strategy possible.

Unfortunately we will later show that c-ideal iMHFs do not exist for any constant $c > 0$. As $n \to \infty$ we will have $\mathsf{AT\text{-}quality}_R(\mathcal{A}) = \omega(1)$ and $\mathsf{E\text{-}quality}_{\bar{R}}(\mathcal{A}) = \omega(1)$.

2.2 Pebbling and Graph Theory

We provide some shorthand for describing algorithms and give some useful graph theoretic definitions and lemmas.

Graph Pebbling. To simplify exposition, our attacks are often described in the language of parallel graph pebbling [AS15]. However, unlike in [AS15], we merely think of this as shorthand for describing an evaluation strategy of an iMHF rather then describing an algorithm in a distinct model of computation.

In particular any iMHF f which we consider is based on some fixed underlying DAG G with (a single source and sink node) which describes which values are used as inputs to which calls to the round function. To compute f on some input x each node of G is assigned a value (bit-string). The source receives the value x. The value of any other node v is defined to be the output of the round function applied to the values of the parent nodes of v. Finally $f(x)$ is defined to be the value of the sink node.[12]

With this in mind, each round of pebbling corresponds to one invocation in an execution. Placing a pebble on a node v in some round is shorthand for computing the value of v by computing the round function on the values of v's

[12] For concreteness, though not relevant to this work, in most cases the round function is simply the compression function H (with the exception of the Linear iMHF of [CGBS16]).

parents. Clearly this can only be done if $(x_1, \ldots, x_z)$ are stored in memory and so, if an algorithm places a pebble on a node whose parents do not all contain a pebble then we call such a move *illegal*. Thus we will always show that our pebbling strategies only produce legal pebblings in order to ensure that they correspond to a feasible pROM algorithm for evaluating iMHF. Finally having a pebble on a node at the end of a round corresponds to storing the value of that node in the state σ for that invocation.

Graph Theory. The key insight behind our attacks is that if a graph is not depth-robust enough then it can be efficiently pebbled.

Definition 3 (Depth Robust and Depth Reducible DAGs). *For $e, d \in \mathbb{N}$ a DAG $G = (V, E)$ is called* (e, d)-depth-robust *if*

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

If G is not (e, d)-depth-robust *then we say that G is* (e, d)-reducible.

In order to prove the generic attack on any iMHF we rely on a lemma, originally due to Valiant [Val77], to show that no graph is depth-robust enough not to permit at least some sort of attack.

3 Generic Attack

In this section we describe a general pebbling attack GenPeb against any (e, d)-reducible graph. $\mathsf{GenPeb}(G, S, g, d)$ takes as input a DAG $G = (V, D)$ and a set $S \subseteq V$ of size e such that $\mathsf{depth}(G - S) \leq d$ and a parameter $g \geq d$ which we will define below. In every round GenPeb makes progress (i.e., places a pebble on node i in the i'th round). Thus, $\mathsf{time}\big(\mathsf{GenPeb}\big) = n$ as the algorithm will place a pebble on the final node n in the n'th rounds. Intuitively, GenPeb is divided into two types of phases: Balloon Phases and a Light Phases. During light phases we throw out most of the pebbles on the graph keeping only pebbles on nodes in S, the highest pebbled node i and the parents of the nodes $[i, i + g]$ that we plan to pebble in the next g rounds. Every g rounds we execute a balloon phase to ensure that we will always have pebbles placed on the parents of the nodes that we plan to pebble in the next g rounds. Because we never remove pebbles on nodes in S and the DAG $G - S$ has depth $\leq d$ we will be able to accomplish this goal in at most d rounds. During light phases we keep at most $\delta g + e$ pebbles on the graph and we place at most one new pebble on G in every round. Thus the total cost during all light phases is at most $n\big(\delta g + e + \bar{R}\big)$. While we may incur higher costs during a balloon phase we are only in the balloon phase for at most $\frac{dn}{g}$ rounds.

We analyze the energy complexity of GenPeb in terms of the depth reduction parameters e and d. These results are summarized in Theorem 2. While GenPeb will lead to attacks with good energy-quality $\mathsf{E\text{-}quality}_{\bar{R}}$ the attack may not necessarily have good AT-quality $\mathsf{AT\text{-}quality}_R$. This is because GenPeb

may still have high absolute memory and RO complexity due to the balloon phase. However, we can easily circumvent this problem by pebbling multiple copies of the DAG G in parallel, which corresponds to evaluating multiple independent instances of the iMHF. In particular, PGenPeb pebbles $\lfloor g/d \rfloor$ instances of G in parallel. We stagger evaluation of the different iMHF instances so that at most one of the $\lfloor g/d \rfloor$ pebbling instances is in a balloon phase at any point in time. To accomplish this PGenPeb simply waits $(i-1)d$ steps to begin pebbling the i'th instance of G. Thus, PGenPeb takes at most $n + \lfloor g/d \rfloor d \leq 2n$ steps to complete. The absolute memory and RO complexity of PGenPeb is essentially just the cost of the balloon phase for a single iMHF instance. Thus, PGenPeb leads to attacks with good AT-quality $\mathsf{AT\text{-}quality}_R$ because the cost of the balloon phase can be amortizes among the $\lfloor g/d \rfloor$ iMHF instances we compute. Theorem 3 states both the energy and AT complexity of PGenPeb. The energy complexity of PGenPeb is roughly equivalent to the energy complexity of GenPeb, and the AT-complexity of PGenPeb is roughly twice the energy complexity of PGenPeb.

In the rest of the paper we will consider several specific families of DAGs like the underlying DAGs in the Catena and Argon2i iMHFs. For Catena, we can find a set $S \subseteq V$ of size $e = n/t$ such that $\mathsf{depth}(G - S) = O(t)$ for every $t > 1$. For Argon2i we can find a set S with expected size $O\left(n/t + (n \ln \lambda)/\lambda\right)$ such that $\mathsf{depth}(G - S) \leq t \cdot \lambda$. Combined with Theorem 3 we will obtain an attack on Catena with quality $\Omega\left(n^{1/3}\right)$ and an attack on Argon2i with quality $\Omega\left(n^{1/4}/\ln n\right)$.

GenPeb makes use of two subroutines need and keep. In our complexity analysis we omit the cost of computing these functions. However we stress that in all our attacks they are either trivial (constant) or very easy to compute. By "easy to compute" we mean that the sets returned by these subroutines will have a short description size (e.g., "all nodes" or $[i, j]$) and that it will be trivial to decide whether a given node v is in these sets.

We begin with some useful notation. Fix a DAG G of size n and number its nodes in (arbitrary) topological order from 1 to n. For $i \in [n]$ and $j \geq i$ we write $\mathsf{parents}(i, j)$ for the set of nodes v with an edge (v, u) for some $u \in [i, \min\{j, n\}]$. Next we fix the class of functions from which need and keep must be chosen in order to prove that GenPeb produces a legal pebbling (and thus defines a pROM evaluation algorithm).[13]

Definition 4 (Needed Pebbles). *Fix a subset of target nodes $T \subseteq V$ and a pebbling configuration $C \subseteq V$ of G.*[14] *Then a node $v \in V$ is* needed for T within d' steps *if there exists a completely unpebbled path P*[15] *of length $\geq d'$ from v to some node in T. We use $N_{C,T,d'}$ to denote the set of all such nodes. We use $K_{C,T}$ to denote the set of all nodes $v \in C$ such that $v \in T$ or v has a child v' such that $v' \in \bigcup_{i=0}^{n} N_{C,T,i}$.*

[13] Later on we instantiate need and keep in several ways but will always prove that they are valid for the inputs we use them for.

[14] That is fix a set C of nodes of V which currently have a pebble on them.

[15] That is $P \cap C = \emptyset$.

Algorithm 1. GenPeb $(G,\ S,\ g,\ d)$

Arguments : $G = (V, E)$, $S \subseteq V$, $g \in [\mathsf{depth}(G-S), |V|]$, $d \geq \mathsf{depth}(G-S)$
Local Variables: $n = |V|$

for $i = 1$ *to* n **do**
  Pebble node i.
  $l \leftarrow \lfloor i/g \rfloor * g + d + 1$
  if $i \mod g \in [d]$ **then** `// Balloon Phase`
    $d' \leftarrow d - (i \mod g) + 1$
    $N \leftarrow \mathsf{need}(l, l+g, d')$
    Pebble every $v \in N$ which has all parents pebbled.
    Remove pebble from any $v \notin K$ where $K \leftarrow S \cup \mathsf{keep}(i, i+g) \cup \{n\}$.
  else `// Light Phase`
    $K \leftarrow S \cup \mathsf{parents}(i, i+g) \cup \{n\}$
    Remove pebbles from all $v \notin K$.
  end
end

Definition 5 (Valid need and keep). *We say that the pair of functions* need *and* keep *is* valid *for* $\mathsf{GenPeb}(G, S, d, g)$ *if we always have* $\mathsf{need}(i, j, d') \supseteq N_{C,[i,j],d'}$ *and* $\mathsf{keep}(i, j) \supseteq K_{C,[i,j]}$ *whenever* $\mathsf{GenPeb}(G, S, \mathsf{depth}(G-S), g)$ *queries* need *or* keep.

In our generic iMHF attack we use the trivial functions need and keep which always output V (e.g., during the balloon phase we pebble every node we can during each round and we never discard any pebbles). The following fact is easy to see:

Fact 1 (Generic Valid Subroutine). *Fix a DAG* $G = (V, E)$ *and let* need *and* keep *be the constant function returning* V. *Then the pair* need *and* keep *is valid for* $\mathsf{GenPeb}(G, S, d, g)$ *for any set* $S \subseteq V$ *and any parameters* $g \geq d \geq \mathsf{depth}(G-S)$.

While we would already obtain high quality attacks on Catena and Argon2i by using the generic need and keep subroutines, we show how our attacks can be optimized further by defining the subroutines need and keep more carefully.

We remark that by leaving need and keep undefined for now we leave some flexibility in the implementation of the balloon phase in $\mathsf{GenPeb}(G, S, g, d)$. During each round of a balloon phase we may pebble any $v \in V$ which has all parents pebbled, but we are only required to add pebbles to these nodes once it becomes absolutely necessary to finish the balloon phase in time (e.g., there are only d' rounds left in the balloon phase and the vertex v is part of an completely unpebbled path to T of length $\geq d'$. Similarly, we are allowed to remove pebbles provided that they are no longer needed for the balloon phase (e.g., every path to T from that node has an intermediate pebble).

The easiest way to satisfy these conditions is to simply pebble every $v \in V$ which has all parents pebbled, and to never remove pebbles during the balloon

phase (Fact 1). Indeed this is exactly what we do in our general attack on iMHFs. However, we demonstrate that further optimizations are possible against the Catena and Argon2i iMHFs (e.g., each of the new pebbles we add during a Catena balloon phase does not need to remain on the DAG very long). In each case the subroutines need and keep will have very simple instantiations — we will not need to perform complicated computations like breadth first search to find these sets.

Fix any G, S, g and d and let $\mathsf{M}(G, S, g, d)$ be the largest number of pebbles simultaneously on $G - S - \mathsf{parents}(i, i + g)$ during any round i which is in a Balloon phase of $\mathsf{GenPeb}(G, S, g, d)$[16]. Similarly let $\mathsf{C}(G, S, g, d)$ be the largest number of pebbles placed on G during any single round in a Balloon Phase. In the following we prove that GenPeb always produces a legal pebbling. Thus it describes a well formed pROM algorithm $\mathcal{A}$ for evaluating an iMHF based on G. We also show how to use $\mathsf{M}(G, S, g, d)$ and $\mathsf{C}(G, S, g, d)$ to upper-bound the energy-complexity of $\mathcal{A}$ with hardcoded inputs (G, S, g, d).

Theorem 2 (Energy Complexity of GenPeb). *Let $G = (V, E)$ be a DAG, with $\mathsf{indeg}(G) = \delta$. Further let $S \subseteq V$ with $|S| = e$ and $d \geq \mathsf{depth}(G - S)$ and let integer $g \in [d, n]$. Fix any valid pair of subroutines need and keep and let $\mathcal{A}$ be the pROM algorithm described by GenPeb with hardcoded inputs (G, S, g, d). Then $\mathcal{A}$ produces a valid pebbling and for any core-memory energy ratio $\bar{R}$ and $\mathsf{M} = \mathsf{M}(G, S, g, d)$ and $\mathsf{C} = \mathsf{C}(G, S, g, d)$ it holds that:*

$$\mathsf{cmc}(\mathcal{A}) \leq n\left(\frac{d \cdot \mathsf{M}}{g} + \delta g + e\right) \qquad \mathsf{crc}(\mathcal{A}) \leq n\left(\frac{\min\{d \cdot \mathsf{C}, n\}}{g} + 1\right)$$

$$E_{\bar{R}}(\mathcal{A}) \leq n\left(\frac{d \cdot \mathsf{M} + \min\{d\mathsf{C}, n\} \cdot \bar{R}}{g} + \delta g + e + \bar{R}\right).$$

Proof. We first prove that $\mathsf{GenPeb}(G, S, g, d)$ produces a legal pebbling to ensure that $\mathcal{A}$ is a well defined algorithm. Then we upper-bound its energy complexity.

Recall that pebbles can be removed at will and by definition, in Step 7, GenPeb only places a pebble if it is legal to do so. Thus the only illegal move could come due to Step 2. Assume no illegal pebble has been placed up to node i. To show that i is then also pebbled legally it suffices to show that each of its parents $P \subseteq V$ are have a pebble at the beginning of round i. The most recent Balloon Phase to have completed before round i (if any) consisted of rounds $B = [i', i'+d]$ where $i-(i'+d) \leq g$. Consider the partition $P_1 = P \cap [i'+d+1, i-1]$, $P_2 = P \cap [i', i' + d]$ and $P_3 \cap [1, i' - 1]$ of the set of parents P. By assumption all $v \in P_1$ were pebbled (legally) in the previous g rounds using Step 2 and so were not removed (by definition of K in Step 10). Moreover by assumption all nodes in P_2 where pebbled by Step 2 during B and so were not removed (by definition of K in Step 8 and the validity of the subroutine keep the pebble is not removed during the balloon phase $B = [i', i' + d]$ and by definition of K in

[16] Recall that there are n/g Balloon Phases and the j^{th} Balloon Phase consists of rounds $\{jg + 1, \ldots, jg + d\}$.

Step 10 the pebble was not removed during rounds $[i'+d+1, i-1]$). Thus it suffices to prove that all $v \in P_3$ contained a pebble at some point during B since then by definition of K in Steps 2 and 10 they too will not be removed.

Let P_4 be the subset of P_3 which don't contain a pebble at the start of B. (If it is empty we are done.) Otherwise, for a given round j let $\mathbf{p}_j$ be all paths which end with a node in P_4 and are unpebbled at the beginning of the round. Let l_j be the length of the longest path in $\mathbf{p}_j$. We argue that $\forall j \in [i', i'+d-1]$ then $l_j \leq d-(j-i')$. If this is the case then we are done. Entering the final round $i'+d-1$ the length of the longest unpebbled path is $l_{i'+d-1} \leq 1$ so by the end of the final round of B all nodes P_4 – the end points of paths $\mathbf{p}_j$ – are pebbled.

We argue that $l_j \leq d-(j-i')$ by induction. Clearly, this is true when $j = i'$ as $l_j \leq \mathsf{depth}(G-S) \leq d$. Now assume that $l_j \leq d-(j-i')$ for some $j \in [i', i'+d-1]$, let $p \in \mathbf{p_j}$ denote a longest path and let v denote the starting node of p. We first observe that either the starting node v of p has no parents or they are all pebbled.[17] Second, we observe that, because need is valid, in round j we must either have $v \in N$ or we must have $l_j < d-(j-i')$. In the latter case we have $l_{j+1} \leq l_j \leq d-(j+1-i')$ — because keep is valid we are not allowed to remove pebbles from any of the parents of v. In the former case we have $l_{j+1} \leq l_j - 1 \leq d-(j+1-i')$ because $v \in \mathsf{need}\big(i', i'+g, d' = d-(j+1-i')\big)$ by the validity of need. Thus in Step 7 of round j node v is pebbled. Finally it remains there till the end of the round since $v \in \mathsf{keep}(j, j+g)$ because there is a completely unpebbled path from v's children in p to $[i', i'+g+d]$. This completes the proof that GenPeb produces a legal pebbling.

Recall that the energy-complexity of $\mathcal{A}$ can be computed as $E_{\bar{R}}(\mathcal{A}) = \mathsf{cmc}(\mathcal{A}) + \bar{R} \cdot \mathsf{crc}(\mathcal{A})$. To upper-bound $\mathsf{cmc}(\mathcal{A})$ we can sum upper-bounds on the cmc of the Balloon phases and the cmc of the Light phases. To compute the Balloon phase term notice that GenPeb is in a Balloon Phase for nd/g steps and during each round i of a balloon phase there are, by definition, at most $\mathsf{M}(G,S,g,d)$ extra pebbles on $G-S-\mathsf{parents}(i, i+g)$. On the other hand, there are clearly at most n Light phase steps and at the start of each round i of a light phase there are no pebbles on $G-S-\mathsf{parents}(i, i+g)$. Finally, during each round i we pay cumulative memory cost at most e to keep pebbles on nodes in S and at most δg to keep pebbles on nodes in the set $\mathsf{parents}(i, i+g)$, which can be of size at most δg. Adding these three terms and factoring out an n term we get that $\mathsf{cmc}(\mathcal{A}) \leq n\left(\frac{d \cdot \mathsf{M}(G,S,g,d)}{g} + \delta g + e\right)$.

Placing a pebbled on G corresponds to making a call to H. To upper-bound $\mathsf{crc}(\mathcal{A})$ we observe that in any round of a Light phase only one pebble is ever placed on G (namely in Step 2). During each balloon phase we place at most $\mathsf{C}(G,S,g,d)$ pebbles on the graph in each rounds, and at most n pebbles on the graph in total. Thus we can write $\mathsf{crc}(\mathcal{A}) \leq n\left(\frac{\min\{n, d \cdot \mathsf{C}(G,S,g,d)\}}{g} + 1\right)$. Combing this with the bound on cmc and rearranging terms we obtain the theorem. □

[17] As otherwise it wouldn't be a longest path in $\mathbf{p_j}$.

Algorithm 2. PGenPeb $(G,\ S,\ g,\ d,\ k)$

Arguments : $G,\ S \subseteq V,\ g \in [\mathsf{depth}(G-S), |V|]\ d \geq \mathsf{depth}(G-S),\ k \leq \lfloor \frac{g}{d} \rfloor$
Local Variables: $n = |V|$, copies $G_1, \ldots, G_k = G,\ S_1, \ldots, S_k = S$

for $t = 1$ *to* $n + kd$ **do**
  Parallel for $j = \max\{1, \frac{t-n}{d}\}$ *to* $\min\left\{k, \frac{t-1}{d}\right\}$ **do**
    $i \leftarrow t - jd$
    Pebble node i in G_j.
    if $i = n$ **then**
      Remove pebbles from all $v \notin \{n\}$ in G_j
      Break
    end
    $l \leftarrow \lfloor i/g \rfloor * g + d + 1$
    if $i \mod g \in [d]$ **then** `// Balloon Phase`
      $d' \leftarrow d - (i \mod g) + 1$
      $N_j \leftarrow \mathsf{need}_j(l, l+g, d')$
      Pebble any $v \in N_j$ which has all parents pebbled.
      Remove pebble from any $v \notin K_j$ where
      $K_j \leftarrow S_j \cup \mathsf{keep}_j(i, i+g) \cup \{n\}$.
    else `// Light Phase`
      $K_j \leftarrow S_j \cup \mathsf{parents}_j(i, i+g) \cup \{n\}$
      Remove pebbles from all $v \notin K_j$.
    end
  end
end

The following Theorem 3 upper-bounds the complexity of PGenPeb. The proof in the full version [AB16] closely follows the analysis of GenPeb in Theorem 2. The key difference is that we evaluate multiple instances, and at that at most one of these instances is in a balloon phase at any point in time. Thus, we get a much tighter bound on AT-complexity because the worst-case memory usage M is approximately the same as the average memory usage of PGenPeb.

Theorem 3 (Complexity of PGenPeb). *Let $G = (V, E)$ be a DAG, with $\mathsf{indeg}(G) = \delta$. Further let $S \subseteq V$ with $|S| = e$ and $d \geq \mathsf{depth}(G-S)$ and let integer $g \in [d, n]$. Fix any valid pair of subroutines* need *and* keep *and let $\mathcal{A}$ be the pROM algorithm described by* PGenPeb *with hardcoded inputs $\left(G, S, g, d, \lfloor \frac{g}{d} \rfloor\right)$. Then for any core-memory area and energyratios $R > 0$ and $\bar{R} > 0$ and $\mathsf{M} = \mathsf{M}(G, S, g, d)$ and $\mathsf{C} = \mathsf{C}(G, S, g, d)$ it holds that:*

$$AT_R(\mathcal{A}) \leq 2n \left\lceil \frac{d(\mathsf{M} + R\mathsf{C})}{g} + \delta g + e + R \right\rceil \text{ and}$$

$$E_{\bar{R}}(\mathcal{A}) \leq n \left\lceil \frac{d\mathsf{M} + \min\{d\bar{R}\mathsf{C}, n\bar{R}\})}{g} + \delta g + e + \bar{R} + 1 \right\rceil.$$

4 Sandwich Graph Attacks

In this section we focus on the two Catena hash functions [FLW13] as well as the second two Balloon Hashing constructions of [CGBS16]. The first Catena iMHF is given by the *Catena Bit Reversal Graph* (which we denote BRG^n_λ); an n node DAG which consists of a stack of $\lambda \in \mathbb{N}_{\geq 1}$ bit-reversal graphs [LT82]. Each node in a layer is associated with a $\log_2\left(\frac{n}{\lambda+1}\right)$ bit string and edges between layers correspond to the bit reversal operation[18]. The Catena designers recommended choosing $\lambda \in \{1, 2, 3, 4\}$ [FLW13]. The second Catena hash function is an iMHF based on the *Catena Double Butterfly Graph*, denoted DBG^n_λ. It is an n node DAG with $O(\lambda \log n)$ layers of nodes.

The "Double-Buffer" and "Linear" iMHFs of [CGBS16] consist of τ layers of σ nodes for a total of $n = \tau * \sigma$ nodes. Each layer is a path with its origin connected to the final node in the path of the previous layer. Moreover all nodes at layers $\tau \geq i \geq 1$ have 20 incoming edges from nodes selected uniformly and independently in the previous layer. In the "Double-Buffer" construction the hash of a node is given by hashing the concatenation of all parent node labels while in the "Linear" construction the parent node labels are first XORed together before being hashed for greater throughput. However this difference will not affect the results in this work.[19]

In this section we demonstrate that all of these DAGs can be computed with lower then hoped for energy and AT complexities (simultaneously) regardless of the random choices made when constructing the graphs. In particular, the iMHF corresponding to both BRG^n_λ and DBG^n_λ can be evaluated with amortized AT complexity $AT_R(\mathcal{A}) = O\left(n^{5/3} + Rn^{4/3}\right)$ and energy complexity $E_{\bar{R}}(\mathcal{A}) = O\left(n^{5/3} + Rn^{4/3}\right)$ for any value of λ. In fact, our attacks hold for a more general class of graphs characterized by Definition 6 below. Thus, to understand our attacks it is not critical to know the exact specification of these DAGs just that both DAGs are strict sandwich graphs. We refer an interested reader to the full version [AB16] of this paper for the actual definitions of the Catena DAGs BRG^n_λ and DBG^n_λ.

Definition 6 ((Strict) λ-Stacked Sandwich Graphs). *Let $n, \lambda \in \mathbb{N}_{\geq 1}$ be a integers such that $\lambda + 1$ divides n and let $k = n/(1+\lambda)$ and let G be a DAG with n nodes. We say that G is a* λ-stacked sandwich DAG *if G contains a directed path of n nodes $(v_1, \ldots, v_n)$ with arbitrary additional edges connecting*

[18] The parameter λ in Catena is related to the parameter τ in Argon2i. The intended space complexity of Catena is $\sigma = 2n/(\lambda+1)$ and the intended computation time is n. Thus, the intended energy complexity is $2n^2/(\lambda+1)$.

[19] We remark that we have assumed that the thread count parameter $p = 1$. However we observe that for $p > 1$ the resulting DAG has an almost identical distribution except that every $s/p^t h$ edge along the path forming a layer is removed. This can only make the job easier of an evaluation algorithm. In particular the complexity of our attacks for the case $p = 1$ are an upper-bound on the complexities of these constructions for $p > 1$.

nodes from lower layers $L_j \doteq \{v_{jk+1}, \ldots, v_{jk+k}\}$ *with* $j \leq i$ *to the* $i+1^{st}$ *layer* L_{i+1}. *If the DAG has no edges of the form* (u, v) *with* $u \in L_j$ *and* $v \in L_{j+2+i}$ *for* $i \geq 0$ *then we say it is a* strict λ-stacked sandwich DAG.

In particular, the Catena bit reversal graph BRG^n_λ is a strict λ-stacked sandwich DAG with n nodes and maximum indegree $\mathsf{indeg} = 2$. The Catena double butterfly graph DBG^n_λ is a strict $\big(\lambda(2x-1)+1\big)$-stacked sandwich DAG with n nodes, where $x \leq \log n$ is the integer such that $n = 2^x \cdot \big(\lambda(2x-1)+1\big)$ — see the full version [AB16] of this paper for additional details about the construction of BRG^n_λ and DBG^n_λ. Finally, for any parameters t and s, a randomly chosen DAG for the Double-Buffer and Linear iMHFs is a strict t-stacked sandwich graph on $n = ts$ nodes with probability 1.

Summary of the Results in this Section. Lemma 1 upper-bounds the depth-robustness of any λ-stacked sandwich DAG G — any λ-stacked sandwich DAG is $(n/t, \lambda t + t)$-reducible. Thus we can apply the generic attack (Theorem 3) to get an upper-bound on the energy and AT complexities of such graphs (Theorem 4) and so, in particular, also for the 4 constructions mentioned above. Theorem 4 states that there is an attack $\mathcal{A}$ with $AT_R(\mathcal{A})$ and energy complexity $E_{\bar{R}}(\mathcal{A}) = O\left((\lambda+\delta)n^{5/3} + \bar{R}n^{4/3}\right)$, where δ denotes the maximum indegree of the DAG.

While these results will also be useful in the next section focused on Argon2i for the 4 constructions above the results can be improved somewhat by observing that the constructions are actually based on *strict* stacked sandwich DAGs. In particular, we can further decrease the resulting complexity if we first define more targeted need and keep functions and prove that they are valid for $\mathsf{PGenPeb}$ when G is a *strict* λ-stacked sandwich DAG (Lemma 2). Theorem 5 says that there is an attack $\mathcal{A}$ with $AT_R(\mathcal{A})$ and energy complexity $E_{\bar{R}}(\mathcal{A}) = O\left(\delta n^{5/3} + \bar{R}n^{4/3}\right)$.

The following Lemma upper-bounds the depth-robustness of any λ-stacked sandwich DAG G. By combining this observation with the generic attack from the previous section we can obtain strong attacks on any λ-stacked sandwich DAG G.

Lemma 1 (Sandwich Graphs are Reducible). *Let* G *be a* λ*-stacked sandwich DAG then for any integer* $t \geq 1$ G *is* $(n/t, \lambda t + t - \lambda - 1)$*-reducible.*

Proof. Let $S = \left\{v_{it} \,\middle|\, 1 \leq i \leq n/t\right\}$. We claim that $\mathsf{depth}(G-S) \leq \lambda t + t - \lambda - 1$. Consider any path P in $G - S$. For each layer L_j the path P can contain at most $t-1$ nodes from layer L_j because any sequence of t consecutive nodes $v_i, v_{i+1} \ldots, v_{i+1}$ must contain at least one node in S. Thus,

$$|P| \leq \sum_{i=0}^{\lambda} \left|P \bigcap L_j\right| \leq \big(\lambda+1\big)\big(t-1\big).$$

□

The next lemma states that keep and need from Algorithm 3 and Algorithm 4 are valid for strict sandwich DAGs.

Algorithm 3. Function: $\mathsf{need}(x, y, d')$

Arguments: x, $y \geq x$, $d' \geq 0$
Constants : Pebbling round i, g, t.
$j \leftarrow (i \mod g)$ // Current Layer is $L_{\lfloor j/t \rfloor}$
Return $L_{\lfloor j/t \rfloor} \cap \left\{ it + j \,\middle|\, i \leq \frac{n}{t} \right\}$

Algorithm 4. Function: $\mathsf{keep}(x, y)$

Arguments: x, $y \geq x$
Constants : Pebbling round i, g, t.
$j \leftarrow (i \mod g)$
$\ell \leftarrow \lfloor j/t \rfloor$ // Current Layer
Return $L_{\geq \ell - 1}$

Lemma 2 (Valid need and keep for Strict Sandwich Graphs). *Let G be a strict λ-stacked sandwich DAG on n nodes, let $S = \left\{ it \,\middle|\, i \leq \frac{n}{\lambda+1} \right\}$, $d = (\lambda+1)t$ and $g \geq d$ then the functions* need *and* keep *from Algorithms 3 and 4 are valid for* $\mathsf{GenPeb}(G, S, g, d)$.

If we modify keep to simply return the entire vertex set then we obtain a valid pair need and keep for general sandwich DAGs. The proofs of Lemmas 2 and 3 are in the full version [AB16].

Lemma 3 (Valid need and keep for Sandwich Graphs). *Let G be a λ-stacked sandwich DAG on n nodes, let $S = \left\{ it \,\middle|\, i \leq \frac{n}{\lambda+1} \right\}$, $d = (\lambda+1)t$ and $g \geq d$. Further, let* keep *be the constant function that returns V and let* need *be the function from Algorithm 3. Then the pair* need *and* keep *are valid for* $\mathsf{GenPeb}(G, S, g, d)$.

Theorem 4 follows easily from Theorem 3, Lemma 3 and Lemma 1 by setting $g = n^{2/3}$ and $t = n^{1/3}$.

Theorem 4 (Complexity of Sandwich Graph). *Let F be an iMHF based on DAG G; a λ-stacked sandwich DAG on n nodes with $\lambda < n^{1/3}$ and maximum indegree* $\mathsf{indeg}(G) = \delta$. *Then for any core-memory area and energy ratios R and $\bar{R}$ there exists an evaluation algorithm $\mathcal{A}$ with*

$$AT_R(\mathcal{A}) \leq 2n^{5/3}\left[(1+\delta) + (\lambda+1) + \frac{R}{n^{1/3}} + \frac{2R + (\lambda+1)R}{n^{2/3}}\right] \text{ and}$$

$$E_{\bar{R}}(\mathcal{A}) \leq n^{5/3}\left[(\lambda+1) + (\delta+1) + \frac{3\bar{R}+1}{n^{2/3}} + \frac{\bar{R}}{n^{1/3}}\right]$$

For *strict* λ-stacked sandwich DAGs Theorem 5 improves on the attack Theorem 4 by instantiating the keep function with Algorithm 4 instead of the constant function $\mathsf{keep}(\cdot) = V$ that returns all vertices (permissible by Lemma 2).

A formal proof of Theorems 4 and 5 can be found in the full version of this paper [AB16].

Theorem 5 (Complexity of Strict Sandwich Graph). *Let G be a strict λ-stacked sandwich DAG on n nodes with $\lambda < n^{1/3}$ and maximum indegree $\mathsf{indeg}(G) = \delta$ then for any core-memory area and energy ratios R and $\bar{R}$ there exists an evaluation algorithm $\mathcal{A}$ for the corresponding iMHF with*

$$AT_R(\mathcal{A}) \leq 2n^{5/3} \times \left[3 + \delta + \frac{R}{n^{1/3}} + \frac{3R}{n^{2/3}}\right] \text{ and}$$
$$E_{\bar{R}}(\mathcal{A}) \leq n^{5/3} \times \left[3 + \delta + \frac{\bar{R}}{n^{1/3}} + \frac{\bar{R}+1}{n^{2/3}}\right].$$

Theorem 5 follows easily from Theorem 3, Lemma 2 and Lemma 1 by setting $t = n^{1/3}$ and $g = n^{2/3}$. While Theorems 4 and 5 only hold for $\lambda < n^{1/3}$ we note there is an trivial pebbling algorithm for strict sandwich graphs with complexity $O(n^2/\lambda)$, or $O(n^{5/3})$ whenever $\lambda > n^{1/3}$ — see [AB16].

We remark that for special cases (e.g., $\lambda = 1$) an alternative instantiation of the functions need and keep in GenPeb allows us to immediately generalize a result of [AS15]. For any $\lambda = 1$-sandwich DAG G with maximum $\mathsf{indeg} = 2$ there is an algorithm $\mathcal{A}$ with $\mathsf{amc}(\mathcal{A}) = O(\mathsf{indeg}\sqrt{n})$ and $\mathsf{cmc}(\mathcal{A}) = O(\mathsf{indeg} \times n^{1.5})$ — our result is slightly more general in that we do not require that G has maximum $\mathsf{indeg} = 2$. Briefly, we can use need from Algorithm 3 and we can redefine keep to simply return the exact same set as need in each round of GenPeb (so that we don't immediately throw out pebbles in step 14 of the same round). It is easy to show that the pair keep and need is valid whenever G is a $\lambda = 1$-stacked sandwich DAGs. We refer an interested reader to the full version [AB16] of this paper for details.

5 (n, δ, w)-Random Graph Attacks

In this section we demonstrate how to extend our attacks to two recent iMHF proposals. The first is the Argon2i iMHF — the variant of Argon2 in which data access patterns are data independent [BDK15]. The authors recommended using Argon2i for password hashing applications to avoid potential side-channel leakage through data dependent memory access patterns [BDK15]. The basic Argon2i DAG is a (pseudo) randomly generated DAG with maximum indegree $\mathsf{indeg} = 2$. Thus, we view the Argon2i DAG as a distribution over n node DAGs — see Definition 7. The second iMHF considered is the Single-Buffer (SB) construction of [CGBS16] (we considered the Double-Buffer and Linear iMHFs from [CGBS16] in Sect. 4).

We begin with the following definition which fixes a class of random graphs key to our analysis.

Definition 7 ((n, δ, σ)-random DAG). *Let $n \in \mathbb{N}$, $1 < \delta < n$, and $1 \leq \sigma \leq n$ such that σ divides n. An (n, δ, w)-*random DAG *is a randomly generated directed acyclic (multi)graph with n nodes $v_1, \ldots, v_n$ and with maximum in-degree δ for each vertex. The graph has directed edges (v_i, v_{i+1}) for $1 \leq i < n$ and random forward edges $(v_{r(i,1)}, v_i), \ldots, (v_{r(i,\delta-1)}, v_i)$ for each vertex v_i. Here, $r(i, j)$ is independently chosen uniformly at random from the set $[\max\{0, i - \sigma\}, i - 1]$.*

We observe that a τ-pass instance of Argon2i iMHF[20] is an $(n, 2, n/\tau)$-random DAG[21]. Similarly the τ-pass "Single-Buffer" construction of [CGBS16] is based on an $(n, 21, n/\tau)$-random DAG. Here, $\sigma = n/\tau$ denotes the size of the memory window used by the naive pebbling algorithm $\mathcal{N}$ (e.g., $\mathsf{amc}(\mathcal{N}) = \sigma$). The hope is that such graphs will have complexity $\theta(\sigma^2 \times \tau)$.

We cannot directly apply our results from the previous section because a (n, δ, σ)-random DAG will not be λ-stacked sandwich DAG with high probability. However, we can show that these graphs are 'close' to λ-stacked sandwich DAGs in the sense that, with high probability, there is a 'small' set S such that we can turn G into a λ-stacked sandwich DAGs just by removing edges incident to vertices in S. Definition 8 formalizes this intuition.

Definition 8 ((m, λ)-Layered Graph). *Let G be a DAG with $n = k\lambda$ nodes $v_1, \ldots, v_n$ with directed edges (v_i, v_{i+1}) for $1 \leq i < n$. Given a set S of nodes we use G_S to denote the resulting DAG if we removed all directed edges (v, v') that are incident to nodes in S (e.g., $v \in S$ or $v' \in S$) except for edges of the form (v_i, v_{i+1}). We say that G is a (m, λ)-*layered DAG *if we can find a set S with at most m nodes such that G_S is a λ-stacked sandwich DAG.*

Demonstrating that a graph is (m, λ)-layered is useful because Theorem 6 upper-bounds the AT and energy complexities of an iMHF based on such a graph. In particular, Theorem 6 relies on Lemma 4 which states that any layered graph is also depth reducible. In Lemma 5, for any given probability $\gamma > 0$ we upper-bound the size m when viewing an Argon2i graph as an (m, λ)-layered graph. Thus we get Theorem 7 which describes an evaluation algorithm for Argon2i and the Balloon Hashing algorithms and bound their AT and energy complexities. In particular it states both their expected values and upper-bounds holding with a given probability γ.

Lemma 4 (Layered Graphs are Reducible). *Let G be a (m, λ)-layered DAG then for any integer $t \geq 1$ G is $(n/t + m, \lambda t + t - \lambda - 1))$-reducible.*

[20] In the notation of [BDK15] the case when $\tau = 1$ corresponds to a single pass.

[21] In response to our attack and the attack of Gibbs et al. [CGBS16] the Argon2i team recently 'tweaked' their construction by adding additional edges of the form $(i, i + \sigma)$, where $\sigma = n/\tau$ is the size of the memory window. While this tweak does seem to eliminate the attack of Gibbs et al. [CGBS16], it is ineffective against our attack. In fact, as long as $\tau < n^{1/4}$ we can completely ignore these extra edges when constructing our depth-reducing set S in Lemma 5.

Proof By definition there is a set S_1 of m nodes such that G_{S_1} is a λ-stacked sandwich DAG. Now by Lemma 1 we can find a set $S_2 \subseteq V(G)$ of size n/t such that $G_{S_1} - S_2$ has depth at most $\mathsf{depth}\big(G_{S_1} - S_2\big) \leq \lambda t + t - \lambda - 1$. Now we set $S = S_1 \bigcup S_2$ we have $|S| \leq m + n/t$ and $\mathsf{depth}(G - S) \leq \mathsf{depth}(G_{S_1} - S_2) \leq \lambda t - \lambda - 1$.

Theorem 6, which upper bounds the complexity of a layered graph, now follows directly from Lemmas 3, 4 and Theorem 2. The proof is in the full version [AB16] of the paper.

Theorem 6 (Complexity of Layered Graph). *Let G be a (m, λ)-layered DAG on n nodes with $\lambda < n^{1/3}$ and maximum indegree $\mathsf{indeg}(G) = \delta$ and fix any $t > 0$, $g \geq t(\lambda + 1)$ then there exists an attack $\mathcal{A}$ on the corresponding iMHF such that for any core-memory ratio $R > 0$ and $\bar{R} > 0$ the energy and AT complexities of $\mathcal{A}$ are at most*

$$AT_R(\mathcal{A}) \leq 2n \left[\frac{n}{t} + m + \delta g + R + \frac{(\lambda+1)t \cdot (n + 2R) + R \cdot n}{g} \right] \text{ and}$$

$$E_{\bar{R}}(\mathcal{A}) \leq n \left[\frac{(\lambda+1)t\big(n + 2\bar{R}\big) + \bar{R} \cdot n}{g} + \delta g + \frac{n}{t} + m + \bar{R} \right].$$

Lemma 5 is the key technical result in this section. It states that, in particular, for any $m \leq n$ and any constants τ and δ a $(n, \delta, \sigma = n/\tau)$-random DAG will be a $\big(O\big(m \log n\big), O\big(\frac{n}{m}\big)\big)$-layered DAG with high probability.

In a bit more detail we show how to construct a set S of (expected) size $O\big(m \log n\big)$. We note that our construction is computationally efficient. Intuitively, we partition the nodes of G into equal sized layers of consecutive nodes $L_0, \ldots, L_\lambda$. We add a node $v \in L_i$ to our set S if *any* of v's parents are also in layer L_i. In the single pass case ($w = n$) we will add a vertex $v \in L_i$ to S with probability at most $(\delta - 1)/i$. Thus, in expectation we will add at most $(\delta - 1)\,|L_i|\,/i$ nodes from layer L_i to S. In total we add at most $\frac{n}{\lambda+1} \sum_{i=1}^{\lambda} \frac{1}{i} = \frac{nH_\lambda}{\lambda+1}$ nodes from layers $L \geq 1$ in expectation. Recall that by H_λ we denote the λ^{th} harmonic number.

Lemma 5 ((n, δ, σ)-random DAGs are Layered). *Fix any $\lambda \geq 1$ and let $q = \lceil \frac{n}{\lambda+1} \rceil$. Then a $(n, \delta, \sigma = n/\tau)$-random DAG is a (m, λ)-layered DAG, where the random variable m has expected value $\mathbb{E}[m] = \tau(\delta - 1)q + q + \frac{(\delta-1)q \cdot H_\lambda}{2}$. Furthermore, for any $\gamma > e^{-3\left(\mathbb{E}[m]-q\right)/4}$ we have $m \leq \mathbb{E}[m] + \sqrt{3\left(\mathbb{E}[m] - q\right)\ln \gamma^{-1}}$ except with probability γ.*

Proof of Lemma 5. Let G be a $(n, \delta, \sigma = n/\tau)$-random DAG with nodes $v_1, \ldots, v_n$. For simplicity assume that $\lambda = n/q - 1$ is an integer so that we can divide G into λ layers $L_0, \ldots, L_\lambda$ of equal size q (If λ is not an integer then

the first $\lceil\lambda\rceil - 1$ layers will contain $q+1$ nodes each and the last layer will contain $\leq q$ nodes.). Layer L_i contains nodes $L_i = \{v_{iq+1}, \ldots, v_{(i+1)q}\}$.We construct $S \subseteq V(G)$ as follows:

$$S = \bigcup_i \left\{ v_j \in L_i \mid \exists s \leq d-1 . v_{r(j,s)} \in L_i \right\}.$$

That is if any of v's parents are in the same layer as v we add v to S.

Given $v_j \in L_i$ we let x_j denote the indicator random variable that is 1 if and only if $v_{r(j,t)} \in L_i$ for some index $t \leq \delta - 1$. We note that by linearity of expectation we have

$$\mathbb{E}\left[|S|\right] \leq \sum_{t=0}^{\lambda} \sum_{j=1}^{q} \mathbb{E}\left[x_{iq+j}\right].$$

Let $\lambda' = \lfloor \frac{\sigma}{q} \rfloor - 1$. Suppose first that $i \leq \lambda'$ and $0 < j \leq q$ so that $iq+j \leq \sigma$ and $v_{iq+j} \in L_i$ $(i > 0)$ then the probability that we add v_{iq+j} to S because one of its parents is also in layer L_i is at most $(\delta-1)(j-1)/(iq)$. Thus, $\mathbb{E}\left[x_{iq+j}\right] \leq (\delta-1)/i$ for each $v_j \in L_i$. Now suppose that $iq + j > \sigma$ then the probability that we add $v_{r(iq+j)}$ to S because one of its parents is in the same layer is at most $(\delta-1)(j-1)/\sigma = (\delta-1)(j-1)\tau/n$. Thus, $\mathbb{E}\left[x_{iq+j}\right] \leq (\delta-1)(q-1)\tau/n$. Thus, in expectation we have

$$\begin{aligned}
\mathbb{E}\left[|S|\right] &\leq q + (\delta-1)\sum_{i=1}^{\lambda'} \frac{(q-1)}{2i} + q(\delta-1)\sum_{i=\lambda'+1}^{\lambda} \frac{(q-1)\tau}{n} \\
&\leq q + (\delta-1)\sum_{i=1}^{\lambda'} \frac{(q-1)}{2i} + q(\delta-1)\left(\lambda - \lambda'\right)\frac{(q-1)\tau}{n} \\
&\leq q + (\delta-1)\frac{(q-1)H_{\lambda'}}{2} + q(\delta-1)\tau,
\end{aligned}$$

where $H_\lambda \approx \ln \lambda$ denotes the λ'th harmonic number and the last inequality follows because $(q-1)\lambda \leq n$. Observe that the DAG G_S with all directed edges originating in S deleted (i.e., delete edges of the form (v, v') with $v \in S$) will be a λ-stacked sandwich graph because there are no forward edges within each layer apart from the chain (v_i, v_{i+1}). Let $X = \sum_{j=q+1}^{n} x_j$. Because the random variables $x_j \in \{0,1\}$ are independent standard concentration bounds imply that $\Pr\left[X \geq \mathbb{E}\left[X\right] + \tau\right] \leq \exp\left(\frac{-\tau^2}{2\mathbf{Var}(X)+2\tau/3}\right)$. In our case we have

$$\begin{aligned}
\mathbf{Var}(X) &= \sum_{i=q+1}^{n}\sum_{j=q+1}^{n} \mathbb{E}\left[x_i x_j\right] - \mathbb{E}\left[x_i\right]\mathbb{E}\left[x_j\right] = \sum_{i=q+1}^{n} \mathbb{E}\left[x_i\right] - \mathbb{E}\left[x_i\right]^2 \\
&\leq \mathbb{E}\left[X\right] = (\delta-1)\frac{(q-1)H_{\lambda'}}{2} + q(\delta-1)\tau.
\end{aligned}$$

Thus, for any $\gamma > \exp\left(-3\mathbb{E}\left[X\right]/4\right)$ we can set $\tau = \sqrt{3\mathbb{E}\left[X\right]\ln\gamma^{-1}}$ to obtain

$$\Pr\left[X \geq \mathbb{E}\left[X\right] + \tau\right] \leq \exp\left(\frac{-\tau^2}{\mathbb{E}\left[X\right] + 2\tau/3}\right) \leq \exp\left(\frac{3\mathbb{E}\left[X\right]\ln\gamma}{3\mathbb{E}\left[X\right]}\right) \leq \gamma.$$

Where the second inequality follows from the observation that $2\tau/3 < \mathbb{E}\left[X\right]$ whenever $\gamma > \exp\left(-3\mathbb{E}\left[X\right]/4\right)$. As $|S| = X + q$ we have

$$|S| \leq q + (\delta-1)\frac{(q-1)H_{\lambda'}}{2} + q(\delta-1)\tau + \sqrt{3\left((\delta-1)\frac{(q-1)H_{\lambda'}}{2} + q(\delta-1)\tau\right)\ln\gamma^{-1}}$$

except with probability γ[22]. □

As an immediate consequence of Lemma 5 we get an attack on the static mode of operation of the Password Hashing Competition winner Argon2. Specifically we can attack the Argon2i variant in which memory accesses patterns are not input dependent — a desirable property to prevent side-channel attacks based on cache timing. As another immediate consequence we also get an attack on the Single-Buffer construction of [CGBS16].

Theorem 7. *Let G be a $(n, \delta, n/\tau)$-random DAG on n nodes. There exists an evaluation algorithm $\mathcal{A}$ for the corresponding iMHF such that for any core-memory ratios $R > 0$ and $\bar{R} > 0$ the expected AT and energy complexities of $\mathcal{A}$ are at most $\mathbb{E}\left[AT_R(\mathcal{A})\right] \leq U_{AT}$ and $\mathbb{E}\left[E_{\bar{R}}(\mathcal{A})\right] \leq U_E$, where*

$$U_{AT} = 2n^{7/4}\left[1 + 2\delta + \frac{(\delta-1)H_{n^{1/4}/\tau}}{2} + \frac{1}{\tau} + \frac{n^{1/4}R + \sqrt{n} + R}{n^{3/4}}\right] \text{ and}$$

$$U_E = n^{7/4}\left[1 + 2\delta + \frac{(\delta-1)H_{n^{1/4}/\tau}}{2} + \frac{1}{\tau} + \frac{n^{1/4}\bar{R} + \sqrt{n} + \bar{R}}{n^{3/4}}\right].$$

Furthermore, except with probability $\gamma > e^{-n^{3/4}\left(\delta-1\right)\left(1+H_{n^{1/4}/\tau}/2\right)}$, we have

$$AT_R(\mathcal{A}) \leq U_{AT} + 2n^{7/4} \times \sqrt{\frac{3(\delta-1)\left(H_{n^{1/4}/\tau} + 2\right)\ln\gamma^{-1}}{2n^{3/4}}} \text{ and}$$

$$E_{\bar{R}}(\mathcal{A}) \leq U_E + n^{7/4} \times \sqrt{\frac{3(\delta-1)\left(H_{n^{1/4}/\tau} + 2\right)\ln\gamma^{-1}}{2n^{3/4}}}.$$

In the special case $\tau = 1$ (single-pass variants) Theorem 7 follows by setting $q = n^{3/4}$ in Lemma 5 and $\lambda = n^{1/4} - 1$, $g = n^{3/4}$ and $t = n^{1/4}$ in Theorem 6. For

[22] If $\gamma < \exp\left(-3\mathbb{E}\left[X\right]/4\right)$ then we can set $\tau = \sqrt{3\mathbb{E}\left[X\right]\ln\gamma^{-1}}$ to obtain a slightly weaker concentration bound. However, the term $\exp\left(-3\mathbb{E}\left[X\right]/4\right)$ is already negligibly small in all of our applications.

the general case (multi-pass variants) we can customize the function keep. The basic idea is simple: during a balloon phase we only need to keep about σ extra pebbles on the $(n, \delta, \sigma = n/\tau)$-random DAG because we can discard pebbles outside of the current memory window. The proof is included in the full version of this paper [AB16].

6 Ideal iMHFs Don't Exist

In this section we show that ideal iMHFs do not exist. More specifically we show that for every graph G there exists node set S and positive integer $g \geq \mathsf{depth}(G-S)$ such that the iMHF evaluation algorithm $\mathcal{A} = \mathsf{PGenPeb}(G, S, g, d, \lfloor g/d \rfloor)$ has AT and energy-complexity $o(n^2/\log^{1-\epsilon} n)$ for any constant $\epsilon > 0$. In particular, if we take the naïve algorithm to be $\mathcal{N} = \mathsf{triv}$ then $\mathcal{A}$ is an attack with energy-quality $\omega(\log^{1-\epsilon} n)$. We first prove (Lemma 7) that all DAGs are reducible provided that the maximum indegree δ is sufficiently small (e.g., $\delta \leq \log^{0.999} n$). The proof of Lemma 7 follows from a result of Valiant [Val77]. Once we have established that all DAGs are reducible we can use PGenPeb to obtain a high quality attack on any iMHF.

Lemma 6 ([Val77] Extension). *Given a DAG G with m edges and depth $\mathsf{depth}(G) \leq d = 2^i$ there is a set of m/i edges s.t. by deleting them we obtain a graph of depth at most $d/2$.*

Lemma 7 (All DAGs are Reducible). *Let $G = (V, E)$ be an arbitrary DAG of size $|V| = n = 2^k$ with $\mathsf{indeg}(G) = \delta$. Then for every integer $t \geq 1$ there is a set $S \subseteq V$ of size $|S| \leq \frac{t\delta n}{\log(n)-t}$ such that $\mathsf{depth}(G - S) \leq 2^{k-t}$. Furthermore, there is an efficient algorithm to find S.*

In a nutshell, Lemma 7 follows by invoking Lemma 6 t times. The detailed proof is in the full version [AB16]. Theorem 8 now follows from Lemma 7 and Theorem 3.

Theorem 8 (Complexities of any iMHF). *Let F be an iMHF based on arbitrary DAG $G = (V, E)$ of size $|V| = n$ with in-degree $\mathsf{indeg}(G) = \delta$. Then for every constant $\epsilon > 0$ and fixed ratios $R > 0$ and $\bar{R} > 0$ there exists an evaluation algorithm $\mathcal{A}$ such that*

$$E_{\bar{R}}(\mathcal{A}) = o\left(n\left(\frac{\delta n}{\log^{1-\epsilon}} + \hat{R}\right)\right) = AT_R(\mathcal{A})$$

where $\hat{R} = \max\{R, \bar{R}\}$.

In particular if we let the naïve algorithm for F be $\mathcal{N} = \mathsf{triv}$ then algorithm $\mathcal{A}$ is an attack with qualities

$$\mathsf{E\text{-}quality}(\mathcal{A}) = \Omega\left(\frac{\hat{R} + n}{\hat{R} + \delta n/\log^{1-\epsilon} n}\right) = \mathsf{AT\text{-}quality}(\mathcal{A})$$

or, for constant R and $\bar{R}$, simply $\Omega\big(\delta^{-1}\log^{1-\epsilon} n\big)$.

In a nutshell, in the proof we set $t = O(\log \log n)$ in Lemma 7 to obtain a set S s.t. $|S| \leq O(n \log \log(n)/\log(n))$ and $d = \mathsf{depth}(G - S) \leq n/\log^2(n)$. We then set $g = n/\log^{1+\epsilon}(n)$ in Theorem 3. See the full version of this paper for a detailed proof [AB16].

Remarks. We remark that for any constants δ and $\epsilon > 1$ Theorem 8 yields an attack with quality $\mathsf{E\text{-}quality}(\mathcal{A}) = \Omega(\log^{1-\epsilon} n) = \mathsf{AT\text{-}quality}(\mathcal{A})$. Furthermore, provided that $\delta \leq \log^{1-\epsilon'} n$ where $\epsilon' > 1$ Theorem 8 yields an attack with quality $\mathsf{E\text{-}quality}(\mathcal{A}) = \omega(1) = \mathsf{AT\text{-}quality}(\mathcal{A})$. If $\bar{R} \neq R$ then we can obtain separate attacks $\mathcal{A}_1$ and $\mathcal{A}_2$ optimizing E-quality and AT-quality respectively.

Erdos et al. [EGS75] constructed a graph G of any size n with $\mathsf{indeg}(G) = O(\log(n))$ which is $(\alpha n, \beta n)$-depth robust for some constants $0 < \beta < \alpha < 1$.[23] This would imply that our bounds in Theorem 8 and Lemma 7 are essentially tight. Alwen and Serbinenko [AS15] used the depth robust DAGs from [MMV13] as a building block to construct a family of DAGs with provably high pebbling complexity $\tilde{\Omega}(n^2)$. Thus, our general attack in Theorem 8 is optimal up to polylogarithmic factors.

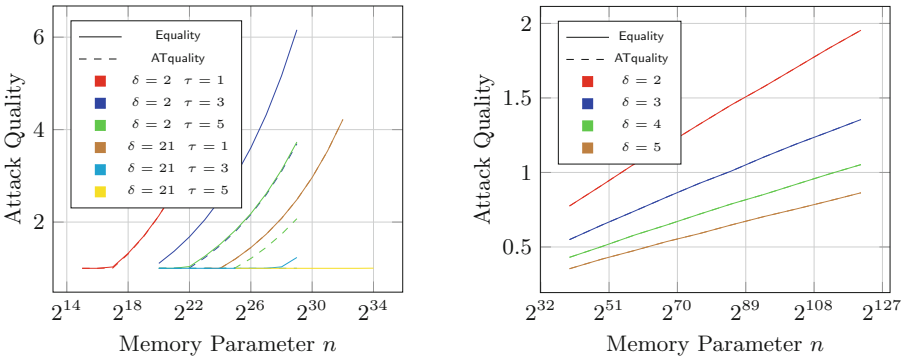
7 Practical Considerations

In this section we demonstrate that our attacks have high quality for practical values of n. For example, we obtain positive attack quality against Argon2i, the winner of the Password Hashing Competition, when n is only 2^{18}. Figure 1a plots attack quality vs n for Argon2i and the Single-Buffer (SB) construction for various values of varies $\tau \in \{1, 3, 5\}$, the number of passes through memory. The full version [AB16] includes additional plots showing that we achieve an even greater attack quality against Catena Dragonfly and Butterfly variants. Figure 1b shows the results for our generic attack on any iMHF.

Parameter Optimization. We remark that we optimized the parameters of our attack for each specific value of n in our plots. For example, we showed that any λ-stacked sandwich DAG is $(n/t, t(\lambda + 1))$-reducible for any $t \geq 1$. For each different value of n we ran a script to find the optimal values of t and $g \geq t(\lambda + 1)$ which minimize the energy complexity (resp. AT-complexity) of $\mathsf{PGenPeb}(G, S, g, d = t(\lambda + 1), k = g/d)$. In our general iMHF attack we used a script to find the optimal value of t in Lemma 7 and the optimal value of g.

Naïve Algorithms. The naïve algorithm $\mathcal{N}$ for the Catena Butterfly iMHF has absolute memory complexity $\mathsf{amc}(\mathcal{N}) = n/(\lambda \log n)$ and energy complexity $E_{\bar{R}}(\mathcal{N}) = n(\mathsf{amc}(\mathcal{N}) + \bar{R})$. Similarly, the naïve algorithm $\mathcal{N}$ for Catena Dragonfly has absolute memory complexity $\mathsf{amc}(\mathcal{N}) = n/(\lambda + 1)$ and the naïve k-pass algorithm for Argon2i and SB has absolute memory complexity $\mathsf{amc}(\mathcal{N}) = n/k$.

[23] In [MMV13] the authors give an explicit construction of a DAG which has $\mathsf{indeg}(G) = \log^2 n$ which is $(\alpha n, \beta n)$-depth robust for any α and β arbitrarily close to 1.



(a) Argon2i and SB (b) Ideal iMHF

Fig. 1. Attack Quality ($R = \bar{R} = 3000$)

Thus, our attack quality decreases with λ or k. We stress that this is not because our attacks becomes less efficient as λ and k increases, but because the $\mathcal{N}$ algorithm requires less and less memory (thus, as λ, k increase the iMHFs become increasingly less ideal). By contrast, the naïve algorithm $\mathcal{N} = \mathsf{triv}$ for our general iMHF (and for Argon2i) has $E_{\bar{R}}(\mathcal{N}) = n(n + \bar{R})$.

Customized Attack Architecture. We have outlined efficient attacks on Catena, Argon2i and the Balloon Hashing iMHFs in the theoretical Parallel Random Oracle Machine (pROM) model of computation. Because pROM is a theoretical model of computation it is not obvious a priori that our attacks translate to practically efficient attacks that could be implemented in real hardware because it can be difficult to dynamically reallocate memory between processes in an ASIC (the amount of memory used during each round of a balloon phase is significantly greater than the amount of memory used during each round of a light phase). In the full version we argue that this architecture challenge would not be a fundamental barrier to an adversary. In particular, we outline an architecture for our algorithm $\mathsf{PGenPeb}$ using Argon2i as an example.

Briefly, we execute $n^{1/4}$ instances of the iMHF in parallel. Our architecture includes $n^{1/4}$ "light phase" chips and a single "Balloon Phase" chip which is responsible for executing all of the balloon phases in a round robin fashion. Each light phase chip only needs $O(n^{3/4} \ln n)$ memory and a single instance of the compression function H. The central balloon phase chip needs to have $O(n \ln n)$ memory and $\sqrt{n}$ instances of the compression functions H.

8 Conclusions

The results in this work show that (at the very least asymptotically speaking) most candidate iMHFs fall far short of their stated goals, and even

of the weaker general upper-bound in Sect. 6. The notable exception is the construction of [AS15]. However, currently it can be viewed mainly as a theoretical result rather then a practical one due to the recursive nature of the construction and the high degree of the polylog factor complexity lower-bound (though this can partially be tightened with a slightly more fine grained security proof). Thus we are left with the central open problem of finding a practical construction of an iMHF which get as close as possible to the general upper-bound.

References

[AB16] Alwen, J., Blocki, J.: Efficiently computing data-independent memory-hard functions. Cryptology ePrint Archive, Report 2016/115 (2016). http://eprint.iacr.org/

[ABMW05] Abadi, M., Burrows, M., Manasse, M., Wobber, T.: Moderately hard, memory-bound functions. ACM Trans. Internet Technol. **5**(2), 299–327 (2005)

[ACK+16] Alwen, J., Chen, B., Kamath, C., Kolmogorov, V., Pietrzak, K., Tessaro, S.: On the complexity of scrypt and proofs of space in the parallel random oracle model. Cryptology ePrint Archive, Report 2016/100 (2016). http://eprint.iacr.org/

[Ale04] Alexander, S.: Password protection for modern operating systems. login, June 2004

[AS15] Alwen, J., Serbinenko, V.: High parallel complexity graphs and memory-hard functions. In: Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing, STOC 2015 (2015). http://eprint.iacr.org/2014/238

[ASK07] Acıiçmez, O., Schindler, W., Koç, Ç.K.: Cache based remote timing attack on the AES. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 271–286. Springer, Heidelberg (2006)

[BBBB10] Bojinov, H., Bursztein, E., Boyen, X., Boneh, D.: Kamouflage: loss-resistant password management. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 286–302. Springer, Heidelberg (2010)

[BBD13] Blocki, J., Blum, M., Datta, A.: Gotcha password hackers! In: Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security, pp. 25–34. ACM (2013)

[BD16] Blocki, J., Datta, A.: Cash: a cost asymmetric secure hash algorithm for optimal password protection. In: 29th IEEE Computer Security Foundations Symposium, CSF (2016, to appear)

[BDK15] Biryukov, A., Dinu, D., Khovratovich, D.: Fast and tradeoff-resilient memory-hard functions for cryptocurrencies and password hashing. Cryptology ePrint Archive, Report 2015/430 (2015). http://eprint.iacr.org/

[Ber] Bernstein, D.J.: Cache-Timing Attacks on AES

[Bil13] Markus, B.: Dogecoin (2013)

[BJKS03] Brainard, J.G., Juels, A., Kaliski, B., Szydlo, M.: A new two-server approach for authentication with short secrets. In: USENIX Security, vol. 3, pp. 201–214 (2003)

[BK15] Biryukov, A., Khovratovich, D.: Tradeoff cryptanalysis of memory-hard functions. Cryptology ePrint Archive, Report 2015/227 (2015). http://eprint.iacr.org/

[BKPS13] Blocki, J., Komanduri, S., Procaccia, A., Sheffet, O.: Optimizing password composition policies. In: Proceedings of the Fourteenth ACM Conference on Electronic Commerce, pp. 105–122. ACM (2013)

[BL13] Bernstein, D.J., Lange, T.: Non-uniform cracks in the concrete: the power of free precomputation. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 321–340. Springer, Heidelberg (2013)

[BM06] Bonneau, J., Mironov, I.: Cache-collision timing attacks against AES. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 201–215. Springer, Heidelberg (2006)

[Bon12] Bonneau, J.: The science of guessing: analyzing an anonymized corpus of 70 million passwords. In: 2012 IEEE Symposium on Security and Privacy (SP), pp. 538–552. IEEE (2012)

[CGBS16] Corrigan-Gibbs, H., Boneh, D., Schechter, S.: Balloon hashing: provably space-hard hash functions with data-independent access patterns. Cryptology ePrint Archive, Report 2016/027 (2016). http://eprint.iacr.org/

[Cha11] Lee, C.: Litecoin (2011)

[CHS06] Canetti, R., Halevi, S., Steiner, M.: Mitigating dictionary attacks on password-protected local storage. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 160–179. Springer, Heidelberg (2006)

[CLN12] Camenisch, J., Lysyanskaya, A., Neven, G.: Practical yet universally composable two-server password-authenticated secret sharing. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 525–536. ACM (2012)

[DC08] Daher, W., Canetti, R.: Posh: a generalized captcha with security applications. In: Proceedings of the 1st ACM workshop on Workshop on AISec, pp. 1–10. ACM (2008)

[DGN03] Dwork, C., Goldberg, A.V., Naor, M.: On memory-bound functions for fighting spam. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 426–444. Springer, Heidelberg (2003)

[DNW05] Dwork, C., Naor, M., Wee, H.: Pebbling and proofs of work. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 37–54. Springer, Heidelberg (2005)

[EGS75] Erdoes, P., Graham, R.L., Szemeredi, E.: On sparse graphs with dense long paths. Technical report, Stanford, CA, USA (1975)

[FH10] Florêncio, D., Herley, C.: Where do security policies come from? In: Proceedings of SOUPS, p. 10 (2010)

[FLW13] Forler, C., Lucks, S., Wenzel, J.: Catena: A memory-consuming password scrambler. IACR Cryptology ePrint Archive 2013:525 (2013)

[JR13] Juels, A., Rivest, R.L.: Honeywords: making password-cracking detectable. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security. ACM (2013)

[Kal00] Kaliski, B.: Pkcs# 5: Password-based cryptography specification version 2.0 (2000)

[KSK+11] Komanduri, S., Shay, R., Kelley, P.G., Mazurek, M.L., Bauer, L., Christin, N., Cranor, L.F., Egelman, S.: Of passwords and people: measuring the effect of password-composition policies. In: Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems, pp. 2595–2604. ACM (2011)

[LT82] Lengauer, T., Tarjan, R.E.: Asymptotically tight bounds on time-space trade-offs in a pebble game. J. ACM (JACM) **29**(4), 1087–1130 (1982)

[Man96] Manber, U.: A simple scheme to make passwords based on one-way functions much harder to crack. Comput. Secur. **15**(2), 171–176 (1996)

[MMV13] Mahmoody, M., Moran, T., Vadhan, S.P.: Publicly verifiable proofs of sequential work. In: Kleinberg, R.D. (ed.) Innovations in Theoretical Computer Science, ITCS 2013, Berkeley, CA, USA, 9–12 January 2013, pp. 373–388. ACM (2013)

[MT79] Morris, R., Thompson, K.: Password security: a case history. Commun. ACM **22**(11), 594–597 (1979)

[NB+15] Narayanan, A., Bonneau, J., Felten, E.W., Miller, A., Goldfeder, S.: Bitcoin and Cryptocurrency Technology (manuscript) (2015). Accessed 6 Aug 2015

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

[PHC] Password hashing competition. https://password-hashing.net/

[PM] Provos, N., Mazieres, D.: Bcrypt algorithm

[RTSS09] Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, pp. 199–212. ACM, New York (2009)

[SS09] Scarfone, K., Souppaya, M.: Nist special publication 800–118: Guide to enterprise password management (draft), April 2009

[Tho79] Thompson, C.D.: Area-time complexity for VLSI. In: Fischer, M.J., DeMillo, R.A., Lynch, N.A., Burkhard, W.A., Aho, A.V. (eds.) Proceedings of the 11h Annual ACM Symposium on Theory of Computing, 30 April - 2 May 1979, Atlanta, Georgia, USA, pp. 81–88. ACM, New York (1979)

[Val77] Valiant, L.G.: Graph-theoretic arguments in low-level complexity. In: Gruska, J. (ed.) MFCS 1977. LNCS, vol. 53, pp. 162–176. Springer, Heidelberg (1977)