On the Complexity of Scrypt and Proofs of Space in the Parallel Random Oracle Model

Joël Alwen[1], Binyi Chen[2], Chethan Kamath[1], Vladimir Kolmogorov[1], Krzysztof Pietrzak[1], and Stefano Tessaro[2]

[1] IST Austria
[2] UC Santa Barbara

Abstract. We study the time- and memory-complexities of the problem of computing labels of (multiple) randomly selected challenge-nodes in a directed acyclic graph. The w-bit label of a node is the hash of the labels of its parents, and the hash function is modeled as a random oracle. Specific instances of this problem underlie both proofs of space [Dziembowski et al. CRYPTO'15] as well as popular memory-hard functions like `scrypt`. As our main tool, we introduce the new notion of a *probabilistic parallel entangled pebbling game*, a new type of combinatorial pebbling game on a graph, which is closely related to the labeling game on the same graph.

As a first application of our framework, we prove that for `scrypt`, when the underlying hash function is invoked n times, the cumulative memory complexity (CMC) (a notion recently introduced by Alwen and Serbinenko (STOC'15) to capture amortized memory-hardness for parallel adversaries) is at least $\Omega(w \cdot (n/\log(n))^2)$. This bound holds for adversaries that can store many natural functions of the labels (e.g., linear combinations), but still not arbitrary functions thereof.

We then introduce and study a combinatorial quantity, and show how a sufficiently small upper bound on it (which we conjecture) extends our CMC bound for `scrypt` to hold against *arbitrary* adversaries.

We also show that such an upper bound solves the main open problem for proofs-of-space protocols: namely, establishing that the *time complexity* of computing the label of a random node in a graph on n nodes (given an initial kw-bit state) reduces tightly to the time complexity for black pebbling on the same graph (given an initial k-node pebbling).

1 Introduction

The common denominator of *password hashing* (e.g., as in PKCS#5 [13]) and *proofs of work* [7,12] is the requirement for a certain computation to be sufficiently expensive, while still remaining feasible. In this context, "expensive" has traditionally meant high *time complexity*, but recent hardware advances have shown this requirement to be too weak, with fairly inexpensive tailored-made ASIC devices for Bitcoin mining and password cracking gaining increasingly widespread usage.

In view of this, a much better requirement is *memory-hardness*, i.e., the *product* of the memory (a.k.a. space) *and* the time required to solve the task at hand (this is known as the *space-time* (ST) complexity) should be large. The ST complexity is widely considered to be a good estimate of the product of the area and the time (AT) complexity of a circuit solving the task [16,5,3], and thus increasing ST complexity appears to incur a higher dollar cost for building custom circuits compared to simply increasing the required raw computing power alone. Motivated by this observation, Percival [16] developed `scrypt`, a candidate memory-hard function for password hashing and key derivation which has been well received in practice (e.g., it underlies the Proof of Work protocols of LiteCoin [14], one of the currently most prevalent cryptocurrencies in terms of market capitalization [1]). This has made memory-hardness one of the main desiderata in candidates for the recent password-hashing competition, including its winner, Argon2 [4]. Dziembowski *et al* [9] introduce the concept of *proofs of space* (PoSpace), where the worker (or miner) can either dedicate a large amount of storage space, and then generate proofs extremely efficiently, or otherwise must pay a large time cost for every proof generated. The PoSpace protocol has also found its way into a recent proposal for digital currency [15].

Our contributions, in a nutshell. Cryptanalytic attacks [5,3,6,**?**] targeting candidate memory-hard functions [11,2,4,**?**] have motivated the need for developing constructions with *provable security guarantees*. With the exception of [3], most candidate memory-hard functions come without security proofs and those that do (e.g. [16,11,**?**]) only consider a severely restricted class of algorithms and complexity notions, as we discuss below. A primary goal of this paper is to advance the foundations of memory-hardness, and we make progress along several fronts.

We develop a new class of probabilistic pebbling games on graphs – called *entangled pebbling games* – which are used to prove results on the memory-hardness of tasks such as computing `scrypt` for large non-trivial classes of adversaries. Moreover, we show how to boost these results to hold against *arbitrary* adversaries in the parallel random oracle model (pROM) [3] under the conjecture that a new combinatorial quantity which we introduce is (sufficiently) bounded.

A second application of the techniques introduced in this paper considers Proofs of Space. We show that time lower bounds on the pebbling complexity of a graph imply time lower bounds in the pROM model agains *any* adversary. The quantitative bounds we get depend on the combinatorial value we introduce, and assuming our conjecture, are basically tight. This solves, modulo the conjecture, the main problem left open in the Proofs of Space paper [9].

Sequentially memory-hard functions. Recall that `scrypt`[3] uses a hash function $\mathsf{h} : \{0,1\}^* \to \{0,1\}^w$ (e.g., SHA-256), and proceeds in two phases, given an

[3] In fact, what we describe here is only a subset of the whole `scrypt` function, called ROMix. ROMix is the actual core of the `scrypt` function, and we will use the generic name "scrypt" for in the following. ROMix (with some minor modification and extensions) also underlies one of the two variants of the winner Argon [4] of the recent password hashing competition https://password-hashing.net/, namely the data-dependent variant Argon2d.

input X. It first computes $X_i = \mathsf{h}^i(X)$ for all $i \in [n]$, and with $S_0 = X_n$, it then computes $S_1, \ldots, S_n$ where

$$S_i = \mathsf{h}(S_{i-1} \oplus X_{\mathsf{int}(S_{i-1})})$$

where $\mathsf{int}(S)$ reduces an w-bit string S to an integer in $[n]$. The final output is S_n. Note that is possible to evaluate `scrypt` on input X using $n \cdot w$ bits of memory and in time linear in n, by keeping the values $X_1, \ldots, X_n$ stored in memory once they are computed. However, the crucial point is that there is no apparent way to save memory – for example, to compute S_i, we need to know $X_{\mathsf{int}(S_{i-1})}$, and under the assumption that $\mathsf{int}(S_{i-1})$ is (roughly) uniformly random in $[n]$, an evaluator without memory needs to do linear work (in n) to recover this value before continuing with the execution. This gives a constant-memory, $O(n^2)$ time algorithm to evaluate `scrypt`. In fact, as stated by Percival [16], the actual hope is that no matter how much time $T(n)$ and how much memory $S(n)$ an adversarial evaluator invests, we always have $S(n) \cdot T(n) \geq n^{2-\epsilon}$ for all $\epsilon > 0$, *even if the evaluator can parallelize its computation arbitrarily.*

Percival's analysis of `scrypt` assumes that h is a random oracle. The analysis is limited in two ways: (1) It only considers adversaries which can only store random oracle outputs in their memory. (2) The bound measures memory complexity in terms of the *maximum* memory resources $S(n)$. The latter is undesirable, since the ultimate goal of an adversary performing a brute-force attack is to evaluate `scrypt` on *as many inputs as possible*, and if the large memory usage is limited to a small fraction of the computing time, a much higher *amortized complexity* can be achieved.

Alwen and Serbinenko (AS) [3] recently addressed these shortcomings, and delivered provably sequentially memory-hard functions in the so-called *parallel random oracle model* (pROM), developing new and better complexity metrics tailored to capturing amortized hardness. While their work falls short of delivering guarantees for `scrypt`-like functions, it serves as an important starting point for our work, and we give a brief overview.

From sequential memory-hardness to pebbling. AS consider adversaries attempting to evaluate a function $\mathcal{H}^{\mathsf{h}}$ (which makes calls to some underlying hash function h, modeled as a random oracle). These adversaries proceed in rounds: in each round i, the adversary can make an *unbounded* number of *parallel* queries to h, and then pass on a state σ_i to the next round. The complexity of the adversary is captured by its *cumulative memory complexity* (CMC) given by $\sum_i |\sigma_i|$. One then denotes as $\mathsf{cmc}^{\mathsf{pROM}}(\mathcal{H})$ the expected CMC of the best adversary where the expectation is over the choice of RO h and coins of the adversary. We stress that CMC exhibits some very important features: *First*, a lower bound appears to yield a reasonable lower bound on the AT complexity metric. *Second,* In contrast to the ST complexity the CMC of a task also gives us a lower-bound on the electricity consumption of performing the task. This is because storing data in volatile memory for, say, the time it takes to evaluate h consumes a significant amount of electricity. Thus CMC tells us something not only about the dollar cost of building a custom circuit for computing a task but also about the dollar

cost of actually running it. While the former can be amortized over the life of the device, the later represents a recurring fee.

AS study sequentially memory-hard functions naturally defined by a single-source and single-sink directed acyclic graph (DAG) $G = (V, E)$.The label of a vertex $i \in V$ with parents $\{p_1, \ldots, p_d\}$ (i.e., $(p_j, v) \in E$ for $i = 1, \ldots, d$) is defined as $\ell_i = \mathsf{h}(i, \ell_{p_1}, \ldots, \ell_{p_d})$. Note that the labels of all vertices can be recursively computed starting with the sources. The function $\mathsf{label}(G, \mathsf{h})$ is now simply the label ℓ_v of the sink v. There is a natural connection between $\mathsf{cmc}^{\mathsf{pROM}}(\mathsf{label}(G, \mathsf{h}))$ for a randomly chosen h and the *cumulative pebbling complexity* (CC) of the graph G.[4] CC is defined in a game where one can place pebbles on the vertices of V, according to the following rules: In every step of the game, new pebbles can be placed on any vertex for which all parents of v have pebbles on them (in particular, pebbles can always be placed on sources), and pebbles can always be removed. The game is won when a pebble has been placed on the sink. The CC of a strategy for pebbling G is defined as $\sum_i |S_i|$, where S_i is the set of vertices on which a pebble is placed at the end of the i^{th} step, and the CC of G – denoted $\mathsf{cc}(G)$ – is the CC of the best strategy.

Indeed, $\mathsf{cc}(G)$ captures the CMC of *restricted* pROM adversaries computing $\mathsf{label}(G, \mathsf{h})$ for which every state σ_i *only consists of random oracle outputs*, i.e., of vertex labels. A pebble on v is equivalent to the fact that σ_i contains ℓ_v. However, a full-fledged pROM adversary *has no reason to be restricted to such a strategy* – it could for example store as part of its state σ_i a particular encoding of the information accumulated so far. Nonetheless, AS show that (up to a negligible extent) such additional freedom does *not* help in computing $\mathsf{label}(G, \mathsf{h})$. They complement this with an efficiently constructible class of constant-degree DAGs G_n on n vertices such that $\mathsf{cc}(G_n) = \Omega(n^2/\mathrm{polylog}(n))$.

Unfortunately however, the framework of [3] does not extend to functions like `scrypt`, as they are *data dependent*, i.e., the values which need to be input to h are determined *at run-time*. While this makes the design far more intuitive, AS's techniques crucially rely on the relationship between intermediate values in the computation being laid out a priori in a *data-independent* fashion.

Our contributions. This paper validates the security of `scrypt`-like functions with two types of results – results for *restricted* adversaries, as well as results for arbitrary adversaries under a combinatorial conjecture. Our results also have direct implications on proofs of space, but we postpone this discussion to ease presentation.

1) Probabilistic pebbling games. We introduce a generalization pebble of pebbling games on a DAG $G = (V, E)$ with dynamic challenges uniformly sampled from a set $C \subseteq V$. With the same pebbling rules as before, we now proceed over n rounds, and at every round, a challenge c_i is drawn uniformly at random

[4] A similar connection, for a weaker pebbling game, was first exploited to construct functions for which evaluation requires many cache memory in [8] and more recently to build one-time computable functions [10] as well as in the security proofs the memory-hard functions in [11,**?**].

from C. The player's goal is to place a pebble on c_i, before moving to the next round, and learning the next challenge c_{i+1}. The game terminates when the last challenge has been covered by a pebble. One can similarly associate with G a *labeling game* computeLabel in the pROM, where the goal is instead to compute the *label* ℓ_{c_i} of c_i, rather than placing a pebble on it. For instance, the computation of `scrypt` is tightly connected to the computeLabel played on the line graph L_n with vertices $[n] = \{1, 2, \ldots, n\}$, edges $\{(i, i+1) : i \in [n-1]\}$, and challenges $C = [n]$ (as detailed in Section 2.5). The labels to be computed in this game are those needed to advance the computation in the second half of the `scrypt` computation, and the challenges (in the actual `scrypt` function) are computed from hash-function outputs.

In fact, it is not hard to see that in computeLabel for some graph G a pROM adversary that only stores random-oracle generated outputs can easily be turned into a player for the pebble for graph G. This is particular true for $G = L_n$, and thus lower bounding the CC of an adversary playing pebble on L_n also yields a lower bound on the CMC of computing (the second half of) `scrypt`. Our first result provides such a lower bound.

> **Theorem 1.** For any constant $\delta > 0$, the CC of an adversary playing pebble on the line graph L_n with challenges $[n]$ is $\Omega_\delta(n^2/\log^2(n))$ with probability $1 - \delta$ over the choice of all challenges.[5]

To appreciate this result, it should be noted that it inherently relies on the choice of the challenges being *independent* of the adversary playing the game – indeed, *if the challenges are known a priori*, techniques from [3] directly give a strategy with CC $O(n^{1.5})$ for the above game. Also this result already improves on Percival's analysis (which, implicitly, places similar restrictions on class of pROM algorithms considered), as Theorem 1 uses the CC of the (simple) pebbling of a graph, and thus it actually generalized to a lower bound on the *amortized* complexity of computing multiple `scrypt` instances in the pROM.[6]

2) Entangled pebbling. The above result is an important first step – to the best of our knowledge all known evaluation attacks against memory-hard functions indeed *only store hash labels directly or not at all* and thus fit into this model – but we ask the question whether the model can be strengthened. For example, an adversary could store the XOR $\ell_i \oplus \ell_j$ of two labels (which only takes w bits) and depending on possible futures of the game, recover both labels given any one of them. As we will see, this *can* help. As a middle ground between capturing pROM security for arbitrary adversaries and the above pebbling adversaries, we introduce a new class of pebbling games, called *entanglement pebbling games*, which constitutes a combinatorial abstraction for such adversaries.

In such games, an adversary can place on a set $\mathcal{Y} \subseteq V$ an "entangled pebble" $\langle \mathcal{Y} \rangle_t$ for some integer $0 \leq t \leq |\mathcal{Y}|$. The understanding here is that placing an

[5] The subscript δ in Ω_δ denotes that the hidden constant depends on δ.

[6] This follows from a special case of the Lemma in [3] showing that CC of a graph is equal to the sum of the CCs the graphs disconnected components.

individual pebble on any t vertices $v \in \mathcal{Y}$ – which we see as a special case of $\langle v \rangle_0$ entangled pebble – is equivalent to having individual pebbles on all vertices in $\mathcal{Y}$. The key point is that keeping an entangled pebble $\langle \mathcal{Y} \rangle_t$ costs *only* $|\mathcal{Y}| - t$, and depending on challenges, we may take different choices as to which t pebbles we use to "disentangle" $\langle \mathcal{Y} \rangle_t$. Also, note that in order to *create* such an entangled pebble, on all elements of $\mathcal{Y}$ there must be either an individual pebble, or such pebble can easily be obtained by disentangling existing entangled pebbles.

In the pROM labeling game, an entangled pebble $\langle \mathcal{Y} \rangle_t$ corresponds to an encoding of length $w \cdot (|\mathcal{Y}| - t)$ of the w-bit labels $\{\ell_i \ : \ i \in \mathcal{Y}\}$ such that given any t of those labels, we can recover all the remaining ones. Such an encoding can be obtained as follows: Fix $2d - t$ elements $x_1, \ldots, x_{2d-t}$ in the finite field $\mathbb{F}_{2^w}$. Let $\mathcal{Y} = \{y_1, \ldots, y_d\}$, and consider the (unique) degree $d - 1$ polynomial $p(.)$ over the finite field $\mathbb{F}_{2^w}$ (whose element are represented as w-bit strings) such that

$$\forall i \in [d] \ : \ p(x_i) = \ell_{y_i} \ .$$

The encoding now simply contains $\{p(x_{d+1}), \ldots, p(x_{2d-t})\}$, i.e., the evaluation of this polynomial on $d - t$ points. Note that given this encoding and any t labels $\ell_i, i \in \mathcal{Y}$, we have the evaluation of $p(.)$ on d points, and thus can reconstruct $p(.)$. Once we know $p(.)$, we can compute all the labels $\ell_{y_i} = p(i)$ in $\mathcal{Y}$.

In general, we prove (in the full version) that entangled pebbling is *strictly more powerful* (in terms of minimizing the expected CC) than regular pebbling. Fortunately, we will also show that for the probabilistic pebbling game on the line graph L_n entangled pebbling cannot outperform regular ones.

> **Theorem 2.** For any constant $\delta > 0$, the CC of an entangled pebbling adversary playing pebble on graph L_n is $\Omega_\delta(n^2/\log^2(n))$ with probability $1 - \delta$ over the choice of all challenges.

Interestingly, the proof is a simple adaptation of the proof of for the non-entangled case. This result can again be interpreted as providing a guarantee in the label game in the pROM for L_n for the class of adversaries that can be abstracted by entangled pebbling strategies.

3) Arbitrary Adversaries. So far we have only discussed (entangled) pebbling lower bounds, which then imply lower bounds for restricted adversaries in the pROM model. In Section 4 we consider security against arbitrary adversaries. Our main results there show that there is a tight connection between the complexity of playing $\mathsf{computeLabel}$ and a combinatorial quantity γ_n that we introduce. We show two results. The first lower-bounds the *time* complexity of playing $\mathsf{computeLabel}$ for *any* graph G while the second lower-bounds the *CMC* of playing $\mathsf{computeLabel}$ for L_n (and thus $\mathtt{scrypt}$).

1. For any DAG $G = (V, E)$ with $|V| = n$, with high probability over the choice of the random hash function h, the pROM *time* complexity to play $\mathsf{computeLabel}$ for graph G, for any number of challenges, using h and when starting with any state of size $k \cdot w$ is (roughly) at least the time complexity needed to play pebble on G with the same number of challenges and starting with an initial pebbling of size roughly $\gamma_n \cdot k$.

$\mathsf{pebble}(G, C, m, \mathtt{T}, P_{\mathsf{init}})$: The m-round parallel pebbling game for DAG $G = (V, E)$, challenge set $C \subseteq V$ and initial pebbling configuration $P_{\mathsf{init}} \subseteq V$ is played between a challenger and a pebbler $\mathtt{T}$.

1. Initialise $\mathsf{cnt} := 0$, $\mathsf{round} := 0$, $P_{\mathsf{cnt}} := P_{\mathsf{init}}$ and $\mathsf{cost} := 0$.
2. A challenge $c \leftarrow C$ is chosen uniformly from C and passed to $\mathtt{T}$.
3. $\mathsf{cost} := \mathsf{cost} + |P_{\mathsf{cnt}}|$.
4. $\mathtt{T}$ choses a new pebbling configuration $P_{\mathsf{cnt}+1}$ which must satisfy
$$\forall i \in P_{\mathsf{cnt}+1} \setminus P_{\mathsf{cnt}} \ : \ \mathsf{parent}(i) \in P_{\mathsf{cnt}} \tag{1}$$
5. $\mathsf{cnt} := \mathsf{cnt} + 1$.
6. If $c \notin P_{\mathsf{cnt}}$ go to step 3. *c not yet pebbled*
7. $\mathsf{round} := \mathsf{round} + 1$. If $\mathsf{round} < m$ go to step 2, otherwise if $\mathsf{round} = m$ the experiment is over, the output is the final count cnt and the cumulative cost cost.

Fig. 1: Description of the m-round, probabilistic parallel pebbling game

2. The pROM CMC for pebble for L_n is $\Omega(n^2/\log^2(n) \cdot \gamma_n)$.

At this point, we do not have any non-trivial upper bound on γ_n but we *conjecture* that γ_n grows very small (if at all) as a function of n. The best lower bound we have is $\gamma_5 > 3/2$. Note that γ does not need to be constant in n – we would get non-trivial statements even if γ_n were to *grow* moderately as a function of n, i.e. $\gamma_n = \mathrm{polylog}(n)$ or $\gamma_n = n^\epsilon$ for some small $\epsilon > 0$.

Therefore, assuming our conjecture on γ_n, the first result in fact solves the main open problem from the work of Dziembowski et al [9] on proofs of space. The second result yields, in particular, a near-quadratic lower bound on the CMC of evaluating `scrypt` for arbitrary pROM adversaries.

2 Pebbling, Entanglement, and the pROM

In this section, we first present both a notion of parallel pebbling of graphs with probabilistic challenges, and then extend this to our new notion of entangled pebbling games. Next, we discuss some generic relations between entangled and regular pebbling, before finally turning to defining the parallel random-oracle model (pROM), and associated complexity metrics.

Throughout, we use the following notation for common sets $\mathbb{N} := \{0, 1, 2, \ldots\}$, $\mathbb{N}^+ := \mathbb{N} \setminus \{0\}$, $\mathbb{N}_{\leq c} := \{0, 1, \ldots, c\}$ and $[c] := \{1, 2, \ldots, c\}$. For a distribution $\mathcal{D}$ we write $x \in \mathcal{D}$ to denote sampling x according to $\mathcal{D}$ in a random experiment.

2.1 Probabilistic Graph Pebbling

Throughout, let $G = (V, E)$ denote a directed acyclic graph (DAG) with vertex set $V = [n]$. For a vertex $i \in V$, we denote by $\mathsf{parent}(i) = \{j \in V \ : \ (j, i) \in E\}$

the parents of i. The *m-round, probabilistic parallel pebbling game* between a player $\mathtt{T}$ on a graph $G = (V, E)$ with challenge nodes $C \subseteq V$ is defined in Figure 1. The *cumulative black pebbling complexity* is defined as

$$\begin{aligned}\mathsf{cc}(G, C, m, \mathtt{T}, P_{\mathsf{init}}) &:= \mathop{\mathbb{E}}_{\mathsf{pebble}(G,C,m,\mathtt{T},P_{\mathsf{init}})}[\mathsf{cost}] \\ \mathsf{cc}(G, C, m, k) &:= \min_{\substack{\mathtt{T}, P_{\mathsf{init}} \subseteq V \\ |P_{\mathsf{init}}| \leq k}} \{\mathsf{cc}(G, C, m, \mathtt{T}, P_{\mathsf{init}})\}\end{aligned}$$

Similarly, the *time cost* is defined as

$$\begin{aligned}\mathsf{time}(G, C, m, \mathtt{T}, P_{\mathsf{init}}) &:= \mathop{\mathbb{E}}_{\mathsf{pebble}(G,C,m,\mathtt{T},P_{\mathsf{init}})}[\mathsf{cnt}] \\ \mathsf{time}(G, C, m, k) &:= \min_{\substack{\mathtt{T}, P_{\mathsf{init}} \subseteq V \\ |P_{\mathsf{init}}| \leq k}} \{\mathsf{time}(G, C, m, \mathtt{T}, P_{\mathsf{init}})\}\end{aligned}$$

The above notions consider the expected cost of a pebbling, thus even if, say $\mathsf{cc}(G, C, m, k)$, is very large, this could be due to the fact that for a tiny fraction of challenge sequences the complexity is very high, while for all other sequences it is very low. To get more robust security notions, we will define a more fine-grained notion which will guarantee that the complexity is high on all but some ϵ fraction on the runs.

$$\begin{aligned}\mathsf{cc}_\epsilon(G, C, m, \mathtt{T}, P_{\mathsf{init}}) &:= \inf\left\{\gamma \;\middle|\; \mathop{\mathbb{P}}_{\mathsf{pebble}(G,C,m,\mathtt{T},P_{\mathsf{init}})}[\mathsf{cost} \geq \gamma] \geq 1 - \epsilon\right\} \\ \mathsf{cc}_\epsilon(G, C, m, k) &:= \min_{\substack{\mathtt{T}, P_{\mathsf{init}} \subseteq V \\ |P_{\mathsf{init}}| \leq k}} \{\mathsf{cc}_\epsilon(G, C, m, \mathtt{T}, P_{\mathsf{init}}\} \\ \mathsf{time}_\epsilon(G, C, m, \mathtt{T}, P_{\mathsf{init}}) &:= \inf\left\{\gamma \;\middle|\; \mathop{\mathbb{P}}_{\mathsf{pebble}(G,C,m,\mathtt{T},P_{\mathsf{init}})}[\mathsf{cnt} \geq \gamma] \geq 1 - \epsilon\right\} \\ \mathsf{time}_\epsilon(G, C, m, k) &:= \min_{\substack{\mathtt{T}, P_{\mathsf{init}} \subseteq V \\ |P_{\mathsf{init}}| \leq k}} \{\mathsf{time}_\epsilon(G, C, m, \mathtt{T}, P_{\mathsf{init}}\}\end{aligned}$$

In general, we cannot upper bound cc in terms of cc_ϵ if $\epsilon > 0$ (same for time in terms of time_ϵ), but in the other direction it is easy to show that

$$\mathsf{cc}(G, C, m, \mathtt{T}, P_{\mathsf{init}}) \geq \mathsf{cc}_\epsilon(G, C, m, \mathtt{T}, P_{\mathsf{init}})(1 - \epsilon)$$

2.2 Entangled Graph Pebbling

In the above pebbling game, a node is always either pebbled or not and there is only one type of pebble which we will hence forth refer to as a "black" pebble. We will now introduce a more general game, where $\mathtt{T}$ can put "entangled" pebbles.

A *t-entangled pebble*, denoted $\langle \mathcal{Y} \rangle_t$, is defined by a subset of nodes $\mathcal{Y} \subseteq [n]$ together with an integer $t \in \mathbb{N}_{\leq |\mathcal{Y}|}$. Having black pebble on all nodes $\mathcal{Y}$ now corresponds to the special case $\langle \mathcal{Y} \rangle_0$. Entangled pebbles $\langle \mathcal{Y} \rangle_t$ now have the following behaviour. Once any subset of $\mathcal{Y}$ of size (at least) t contains black pebbles then all $v \in \mathcal{Y}$ immediatly receive a black pebble (regardless of whether

their parents already contained black pebbles or not). We define the *weight* of an entangled pebble as:

$$|\langle \mathcal{Y} \rangle_t|_{\updownarrow} := |\mathcal{Y}| - t.$$

More generally, an *(entangled) pebbling configuration* is defined as a set $P = \{\langle \mathcal{Y}_1 \rangle_{t_1}, \ldots, \langle \mathcal{Y}_z \rangle_{t_s}\}$ of entangled pebbles and its weight is

$$|P|_{\updownarrow} := \sum_{i \in [s]} |\langle \mathcal{Y}_i \rangle_{t_i}|_{\updownarrow}.$$

The rule governing how a pebbling configuration P_{cnt} can be updated to configuration $P_{\mathsf{cnt}+1}$ – which previously was the simple property eq.(1) – are now a bit more involved. To describe them formally we need the following definition.

Definition 1 (Closure). *The* closure *of an entangled pebbling configuration* $P = \{\langle \mathcal{Y}_1 \rangle_{t_1}, \ldots, \langle \mathcal{Y}_s \rangle_{t_s}\}$ *– denoted* $\mathsf{closure}(S)$ *– is defined recursively as follows: initialise* $\Lambda = \emptyset$ *and then*

$$\textit{while } \exists j \in [s] \ : \ (\mathcal{Y}_j \not\subseteq \Lambda) \wedge (\Lambda \cap \mathcal{Y}_j \geq t_j) \textit{ set } \Lambda := \Lambda \cup \mathcal{Y}_j$$

once Λ *cannot be further extended using the rule above we define* $\mathsf{closure}(S) = \Lambda$.

Note that $\mathsf{closure}(S)$ is non-empty iff there's at least one set of t-entangled pebbles $\langle \mathcal{Y} \rangle_t$ in P with $t = 0$. Equipped with this notion we can now specify how a given pebbling configuration can be updated.

Definition 2 (Valid Update). *Let* $P = \{\langle \mathcal{Y}_1 \rangle_{t_1}, \ldots, \langle \mathcal{Y}_m \rangle_{t_s}\}$ *be an entangled pebbling configuration. Further,*

- *Let* $\mathcal{V}_1 := \mathsf{closure}(P)$.
- *Let* $\mathcal{V}_2 := \{i \ : \ \mathsf{parent}(i) \subseteq \mathcal{V}_1\}$. *These are the nodes that can be pebbled using the black pebbling rules (eq.1).*

Now $P' = \{\langle \mathcal{Y}'_1 \rangle_{t'_1}, \ldots, \langle \mathcal{Y}'_{s'} \rangle_{t'_{s'}}\}$ *is a* valid update *of* P *if for every* $\langle \mathcal{Y}'_{j'} \rangle_{t'_{j'}}$ *one of the two conditions is satisfied*

1. $\mathcal{Y}'_{j'} \subseteq (\mathcal{V}_1 \cup \mathcal{V}_2)$.
2. $\exists i$ *with* $\mathcal{Y}'_{j'} = \mathcal{Y}_i$ *and* $t'_j \geq t_i$. *That is,* $\langle \mathcal{Y}'_{j'} \rangle_{t'_{j'}}$ *is an entangled pebble* $\langle \mathcal{Y}_i \rangle_{t_i}$ *that is already in* P*, but where we potentially have increased the threshold from* t_i *to* $t'_{j'}$.

The entangled pebbling game $\mathsf{pebble}^{\updownarrow}(G, C, m, \mathtt{T})$ is now defined like the game $\mathsf{pebble}(G, C, m, \mathtt{T})$ above, except that $\mathtt{T}$ is allowed to choose entangled pebblings. We give it in Figure 2. The *cumulative entangled pebbling complexity* and the *entangled time complexity* of this game are defined analogously to those of the simple pebbling game – we just replace cc with $\mathsf{cc}^{\updownarrow}$ and time with $\mathsf{time}^{\updownarrow}$ in our notation to account for entanglement being considered. In the full version, we show that entanglement can indeed improve the cumulative complexity with respect to unentangled pebbling. However, in the next section, we will show that this is not true with respect to *time* complexity.

$\mathsf{pebble}^{\updownarrow}(G, C, m, \mathtt{T}, P_{\mathsf{init}})$: The m-round parallel, entangled pebbling game for DAG $G = (V, E)$, challenge set $C \subseteq V$ and initial entagled pebbling configuration P_{init}

1. Initialise $\mathsf{cnt} := 0$, $\mathsf{round} := 0$, $P_{\mathsf{cnt}} := P_{\mathsf{init}}$ and $\mathsf{cost} := 0$.
2. A challenge $c \leftarrow C$ is chosen uniformly from C and passed to $\mathtt{T}$.
3. $\mathsf{cost} := \mathsf{cost} + |P_{\mathsf{cnt}}|_{\updownarrow}$.
4. $\mathtt{T}$ choses a new pebbling configuration $P_{\mathsf{cnt}+1}$ which must be a valid update of P_{cnt}.
5. $\mathsf{cnt} := \mathsf{cnt} + 1$.
6. If $c \notin \mathsf{closure}(\mathsf{P}_{\mathsf{cnt}})$ go to step 3. *c not yet pebbled*
7. $\mathsf{round} := \mathsf{round} + 1$. If $\mathsf{round} < m$ go to step 2. Otherwise if $\mathsf{round} = m$ end the experiment and output the final count cnt and cumulative cost cost.

Fig. 2: The entangled pebbling game $\mathsf{pebble}^{\updownarrow}(G, C, m, \mathtt{T})$.

2.3 Entanglement Does not Improve Time Complexity

We show that in terms of *time* complexity, entangled pebbling are no more efficient than normal pebbles.

Lemma 3 (Entangled Time = Simple Time). *For any $G, C, m, \mathtt{T}^{\updownarrow}, {P_{\mathsf{init}}}^{\updownarrow}$ and $\epsilon \geq 0$ there exist a $\mathtt{T}, P_{\mathsf{init}}$ such that $|P_{\mathsf{init}}| \leq |{P_{\mathsf{init}}}^{\updownarrow}|_{\updownarrow}$ and*

$$\mathsf{time}(G, C, m, \mathtt{T}, P_{\mathsf{init}}) \leq \mathsf{time}^{\updownarrow}(G, C, m, \mathtt{T}^{\updownarrow}, {P_{\mathsf{init}}}^{\updownarrow}) \tag{2}$$

$$\mathsf{time}_{\epsilon}(G, C, m, \mathtt{T}, P_{\mathsf{init}}) \leq \mathsf{time}^{\updownarrow}_{\epsilon}(G, C, m, \mathtt{T}^{\updownarrow}, {P_{\mathsf{init}}}^{\updownarrow}) \tag{3}$$

in particular

$$\mathsf{time}^{\updownarrow}(G, C, m, k) = \mathsf{time}(G, C, m, k) \qquad \mathsf{time}^{\updownarrow}_{\epsilon}(G, C, m, k) = \mathsf{time}_{\epsilon}(G, C, m, k) \tag{4}$$

Proof. The $\geq$ directions in eq.(4) follows directly from the fact that a black pebbling is a special case of an entangled pebbling. The $\leq$ direction follows from eq.(2) and eq.(3). Below we prove eq.(2), the proof for eq.(3) is almost analogous.

We say that a player $\mathtt{A}_{\mathsf{greedy}}$ for a normal or entangled pebbling is "greedy", if its strategy is simply to pebble everything possible in every round and never remove pebbles. Clearly, $\mathtt{A}_{\mathsf{greedy}}$ is optimal for time complexity, i.e.,

$$\forall G, C, m, P_{\mathsf{init}} : \min_{\mathtt{T}} \mathsf{time}(G, C, m, \mathtt{T}, P_{\mathsf{init}}) = \mathsf{time}(G, C, m, \mathtt{A}_{\mathsf{greedy}}, P_{\mathsf{init}}) \tag{5}$$

$$\forall G, C, m, {P_{\mathsf{init}}}^{\updownarrow} : \min_{\mathtt{T}} \mathsf{time}^{\updownarrow}(G, C, m, \mathtt{T}, {P_{\mathsf{init}}}^{\updownarrow}) = \mathsf{time}^{\updownarrow}(G, C, m, \mathtt{A}_{\mathsf{greedy}}, {P_{\mathsf{init}}}^{\updownarrow}) \tag{6}$$

We next describe how to derive an initial black pebbling ${P_{\mathsf{init}}}^*$ from an entangled pebbling ${P_{\mathsf{init}}}^{\updownarrow}$ of cost $|{P_{\mathsf{init}}}^*| \leq |{P_{\mathsf{init}}}^{\updownarrow}|_{\updownarrow}$ such that

$$\mathsf{time}(G, C, m, \mathtt{A}_{\mathsf{greedy}}, {P_{\mathsf{init}}}^*) \leq \mathsf{time}^{\updownarrow}(G, C, m, \mathtt{A}_{\mathsf{greedy}}, {P_{\mathsf{init}}}^{\updownarrow}) \tag{7}$$

Note that this then proves eq.(2) (with $\mathtt{A}_{\mathsf{greedy}}, {P_{\mathsf{init}}}^*$ being $\mathtt{T}, P_{\mathsf{init}}$ in the statement of the lemma) as

$$\mathsf{time}^{\updownarrow}(G, C, m, \mathtt{T}^{\updownarrow}, {P_{\mathsf{init}}}^{\updownarrow}) \geq \mathsf{time}^{\updownarrow}(G, C, m, \mathtt{A}_{\mathsf{greedy}}, {P_{\mathsf{init}}}^{\updownarrow}) \tag{8}$$

$$\geq \mathsf{time}(G, C, m, \mathtt{A}_{\mathsf{greedy}}, {P_{\mathsf{init}}}^*) \tag{9}$$

It remains to prove eq.(7). For every share $\langle \mathcal{Y} \rangle_t \in {P_{\mathsf{init}}}^{\updownarrow}$ we observe which $|\mathcal{Y}| - t$ pebbles are the last ones to become available[7] in the random experiment $\mathsf{pebble}^{\updownarrow}(G, C, m, \mathtt{T}^{\updownarrow}, {P_{\mathsf{init}}}^{\updownarrow})$, and we add these pebbles to P_{init} if they're not already in there.

Note that then $|P_{\mathsf{init}}| \leq |{P_{\mathsf{init}}}^{\updownarrow}|_{\updownarrow}$ as required. Moreover eq.(7) holds as at any timestep, the nodes available in $\mathsf{pebble}^{\updownarrow}(G, C, m, \mathtt{A}_{\mathsf{greedy}}, {P_{\mathsf{init}}}^{\updownarrow})$ are nodes already pebbled in $\mathsf{pebble}(G, C, m, \mathtt{A}_{\mathsf{greedy}}, {P_{\mathsf{init}}}^*)$ at the same timestep. □

2.4 The Parallel Random Oracle Model (pROM)

We turn to an analogue of the above pebbling games n the parallel random oracle model (pROM) [3]. In particular, let $G = (V, E)$ be a DAG with a dedicated set $C \subseteq V$ of challenge edges, we identify the vertices with $V = [n]$. A labelling $\ell_1, \ldots, \ell_n$ of G's verticies using a hash functiotn $\mathsf{h} : \{0,1\}^* \to \{0,1\}^w$ is defined as follows. Let $\mathsf{parent}(i) = \{j \in V : (j, i) \in E\}$ denote the parents of i, then

$$\ell_i = \mathsf{h}(i, \ell_{p_1}, \ldots, \ell_{p_d}) \quad \text{where} \quad (p_1, \ldots, p_d) = \mathsf{parent}(i) \tag{10}$$

Note that if i is a source, then its label is simply $\ell_i = \mathsf{h}(i)$.

We consider a game $\mathsf{computeLabel}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$ where an algorithm $\mathtt{A}$ must m times consecutively compute the label of a node chosen at random from C. $\mathtt{A}$ gets an initial state $\sigma_0 = \sigma_{\mathsf{init}}$. The *cumulative memory complexity* is defined as follows.

$$\mathsf{cmc}^{\mathsf{pROM}}(\mathsf{G}, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h}) = \mathop{\mathbb{E}}_{\mathsf{computeLabel}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})}[\mathsf{cost}]$$

$$\mathsf{cmc}^{\mathsf{pROM}}(G, C, m, \sigma_{\mathsf{init}}) = \min_{\mathtt{A}} \mathop{\mathbb{E}}_{\mathsf{h} \leftarrow \mathcal{H}} \mathsf{cmc}^{\mathsf{pROM}}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$$

The *time complexity* of a given adversary is

$$\mathsf{time}^{\mathsf{pROM}}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h}) = \mathop{\mathbb{E}}_{\mathsf{computeLabel}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})}[\mathsf{cnt}]$$

We will also consider this notion against the best adversaries from some restricted class of adversaries, in this case we put the class as subscript, like

$$\mathsf{cmc}^{\mathsf{pROM}}_{\mathcal{A}}(G, C, m, \sigma_{\mathsf{init}}) = \min_{\mathtt{A} \in \mathcal{A}} \mathop{\mathbb{E}}_{\mathsf{h} \leftarrow \mathcal{H}} \mathsf{cmc}^{\mathsf{pROM}}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$$

[7] A pebble is available if it's in the closure of the current entangled pebbling configuration, also note that $\mathtt{A}_{\mathsf{greedy}}$'s strategy is deterministic and independent of the challenges it gets, so the "last nodes to become available" is well defined.

$\underline{\mathsf{computeLabel}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h} : \{0,1\}^* \to \{0,1\}^w) :}$

1. Initialise $\mathsf{cnt} := 0$, $\mathsf{round} := 0$, $\sigma_{\mathsf{cnt}} := \sigma_{\mathsf{init}}$ and $\mathsf{cost} := 0$.
2. A challenge $c \leftarrow C$ is chosen uniformly from C.
3. $(q_1, \ldots, q_s, \ell) \leftarrow \mathtt{A}(c, \sigma_{\mathsf{cnt}})$ $\mathtt{A}$ *choses parallel* h *queries and (optionally) a guess for* ℓ_c
4. $\mathsf{cost} := \mathsf{cost} + |\sigma_{\mathsf{cnt}}| + s \cdot w$.
5. $(\sigma_{\mathsf{cnt}+1}) \leftarrow \mathtt{A}(c, \sigma_{\mathsf{cnt}}, \mathsf{h}(q_1), \ldots, \mathsf{h}(q_s))$ $\mathtt{A}$ *outputs next state*
6. $\mathsf{cnt} := \mathsf{cnt} + 1$
7. If $\ell = \perp$ (no guess in this round) go to step 3.
8. If $\ell \neq \ell_c$ (wrong guess) set $\mathsf{cost} = \infty$ and abort.
9. $\mathsf{round} := \mathsf{round} + 1$. If $\mathsf{round} = m$ end the experiment. Otherwise go to step 2.
10. $\mathsf{round} := \mathsf{round} + 1$. If $\mathsf{round} < m$ go to step 2. Otherwise if $\mathsf{round} = m$ end the experiment and output the final count cnt and cumulative cost cost.

Fig. 3: The labeling game $\mathsf{computeLabel}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$.

As for pebbling, also here we will consider the more meaningful ϵ variants of these notions

$$\mathsf{cmc}^{\mathtt{pROM}}_{\epsilon}(\mathsf{G}, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h}) = \inf\left\{\gamma \;\middle|\; \mathop{\mathbb{P}}_{\mathsf{computeLabel}(G,C,m,\mathtt{A},\sigma_{\mathsf{init}},\mathsf{h})}[\mathsf{cost} \geq \gamma] \geq 1 - \epsilon\right\}$$

$$\mathsf{cmc}^{\mathtt{pROM}}_{\epsilon}(G, C, m, \sigma_{\mathsf{init}}) = \min_{\mathtt{A}} \mathop{\mathbb{E}}_{\mathsf{h} \leftarrow \mathcal{H}} \mathsf{cmc}^{\mathtt{pROM}}_{\epsilon}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$$

$$\mathsf{time}^{\mathtt{pROM}}_{\epsilon}(\mathsf{G}, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h}) = \inf\left\{\gamma \;\middle|\; \mathop{\mathbb{P}}_{\mathsf{computeLabel}(G,C,m,\mathtt{A},\sigma_{\mathsf{init}},\mathsf{h})}[\mathsf{cnt} \geq \gamma] \geq 1 - \epsilon\right\}$$

2.5 `scrypt` and the computeLabel Game

We informally discuss the relation between evaluating `scrypt` in the pROM and the computeLabel game for the line graph (described below) and, and explain why we will focus on the latter. A similar discussion can be made for Argon2d.

First, recall that `scrypt` uses a hash function $\mathsf{h} : \{0,1\}^* \to \{0,1\}^w$, and proceeds in two phases, given an input X. In the first phase it computes $X_i = \mathsf{h}^i(X)$ for all $i \in [n]$,[8] and in the second phase, setting $S_0 = X_n$, it computes $S_1, \ldots, S_n$ defined recursively to be

$$S_i = \mathsf{h}(S_{i-1} \oplus X_{\mathsf{int}(S_{i-1})})$$

where $\mathsf{int}(S)$ reduces a w-bit string S to an integer in $[n]$ such that if S is uniform random then $\mathsf{int}(S)$ is (close to) uniform over $[n]$. The final output of $\mathtt{scrypt}^h_n(X) = S_n$. To show that `scrypt` is memory-hard, we need to lower-bound the CMC required to compute it in the pROM.

We argue that to obtain this bound it suffices to restrict our attention to the minimal final value of cost in $\mathsf{cmc}^{\mathtt{pROM}}(L_n, [n], n)$ where $L_n = (V, E)$ is the

[8] Here $\mathsf{h}^i(X)$ denotes iteratively applying h i times to the input X.

line graph where $V = [n]$ and $E = \{(i, i+1) \ : \ i \in [n-1]\}$. Intuitively this is rather easy to see. Clearly any algorithm which hopes to evaluate `scrypt` with more than negligble probability must, at some point, compute all X_i values and all S_j values since guessing them is almost impossible. Moreover until S_{i-1} has been computed the value of $\mathsf{int}(S_{i-1})$ – i.e. the challenge label needed to compute S_i – is uniform random and independent, just like the distribution of i^{th} challenge $c \leftarrow C$ in the $\mathsf{computeLabel}$ game. In other words once an algorithm has computed the values $X_1, \ldots, X_n$ computing the values of $S_1, \ldots, S_n$ corresponds exactly to playing the $\mathsf{computeLabel}$ game on graph L_n with challenge set $[n]$ for n rounds. The initial state is exactly the state given to the algorithm as input in the step where it first computes X_n. It is immediate that, when restricted to strategies which don't simply guess relevant outputs of h, then any strategy for computing the values $S_1, \ldots, S_n$ corresponds to a strategy for playing $\mathsf{computeLabel}(L_n, [n], n)$.

In summary, once $\mathtt{A}$ has finished the first phase of evaluating `scrypt`, the second phase essentially corresponds to playing the $\mathsf{computeLabel}$ game on the graph L_n with challenge set $[n]$ for n rounds. The initial state σ_{init} in $\mathsf{computeLabel}$ is the state given to $\mathtt{A}$ as input in the first step of round 1 (i.e. in the step when $\mathtt{A}$ first computes X_n). It is now immediate that (when restricted to strategies which don't simply guess relevant outputs of h) then any strategy $\mathtt{A}$ for computing the second phase of `scrypt` is essentially a strategy for playing $\mathsf{computeLabel}(L_n, [n], n)$. Clearly the total CMC of $\mathtt{A}$ when computing both phases of `scrypt` is at least the CMC of computing just the second. Thus our lowerbound on $\mathsf{cmc}^{\mathtt{pROM}}(L_n, [n], n)$ in Theorem 15 also gives us a lower bound on the CMC of $\mathtt{scrypt}_n$. (The proof is rather tedious, and omitted from this version of the paper.)

Simple Algorithms. Theorem 15 below will make no restrictions on the algorithm playing $\mathsf{computeLabel}$, at the cost of relying on γ_n, for which we only conjecture an upper bound. We do not need such conjectures if we restrict our attention to *simple algorithms* from the class $\mathcal{A}_{SA}$: A simple algorithms $\mathtt{A} \in \mathcal{A}_{SA}$ is one which either stores a value X_i directly in its intermediary states[9] or stores nothing about the value of X_i at all. (They are however permitted to store arbitrary other information in their states.) For example a simple algorithm may not store, say, $X_i \oplus X_j$ or just the first 20 bits of X_i. We note that, to the best of our knowledge, all algorithms in the literature for computing `scrypt` (or any memory-hard function for that matter) are indeed of this form. For simple algorithms, then we obtain an unconditional lower-bound on the CMC of `scrypt` by using Theorem 4 below, which only consider pebbling games.

Much as in the more general case above, for the set of algorithms $\mathcal{A}_{SA}$ we can now draw a parallel between computing phase two of `scrypt` in the pROM and playing the game pebble on the graph L_n with challenge set $[n]$ for n rounds. Therefore Theorem 4 immediatly gives us a lower-bound on the CMC of $\mathtt{scrypt}_n$ for all algorithms in $\mathcal{A}_{SA}$.

[9] or at least an equivalent encoding of X_i

Entangled Adversaries. In fact we can even relax our restrictions on algorithms computing `scrypt` to the class $\mathcal{A}_{EA}$ of *entangled* algorithms while still obtaining an unconditional lower-bound on the CMC of `scrypt`. In addition to what is permitted for simple algorithms we also allow storing "entangled" information about the values of $X_1, \ldots, X_n$ of the following form. For any subset $L \subseteq [n]$ and integer $t \in [|L|]$ an algorithm can store an encoding of $X_L = \{X_i\}_{i \in L}$ such that if it obtains any t values in L then it can immediatly output all remaining $|L| - t$ values in L with no further information or queries to h. One such encoding uses polynomial interpolation as described in the introduction. Indeed, this motivates our definition of entangled pebbles above.

As shown in the full version, the class $\mathcal{A}_{EA}$ is (in general) strictly more powerful $\mathcal{A}_{SA}$ when it comes to minimizing CMC. Thus we obtain a more general unconditional lower-bound on the CMC of `scrypt` using Theorem 9 which lower-bounds $\mathsf{cc}^{\updownarrow}(L_n, [n], n, n)$, the entangled cumulative pebbling complexity of L_n.

3 Pebbling Lower Bounds for the Line Graph

In this section, we prove lower bounds for the cumulative complexity of the n-round probabilistic pebbling game on the line graph L_n with challenges from $[n]$. We will start with the case without entanglement (i.e., dealing only with black pebbles) which captures the essence of our proof, and then below, extend our proof approach to the entangled case.

Theorem 4 (Pebbling Complexity of the Line Graph). *For all* $0 \leq k \leq n$, *and constant* $\delta > 0$,

$$\mathsf{cc}_{\delta}(L_n, C = [n], n, k) = \Omega_{\delta}\left(\frac{n^2}{\log^2(n)}\right) .$$

We note in passing that the above theorem can be extended to handle a different number of challenges $t \neq n$, as it will be clear in the proof. We dispense with the more general theorem, and stick with the simpler statement for the common case $t = n$ motivated by `scrypt`. The notation Ω_{δ} indicates that the constant hidden in the Ω depends on δ.

In fact, we also note that our proof allows for more concrete statements as a function of δ, which may be constant. However, not surprisingly, the bound becomes weaker the smaller δ is, but note that if we are only interested in the expectation $\mathsf{cc}(L_n, C = [n], n, k)$, then applying the result with $\delta = O(1)$ (e.g., $\frac{1}{2}$) is sufficient to obtain a lower bound of $\Omega\left(\frac{n^2}{\log^2 n}\right)$.

Proof intuition – the expectation game. Before we turn to the formal proof, we give some high-level intuition. It turns out that most of the proof is going to in fact lower bound the cc of a much simpler game, where the goal is far simpler than covering challenges from $[n]$ with a pebble. In fact, the game will be completely deterministic.

The key observation is that every time a new challenge c_i is drawn, and the player has reached a certain pebbling configuration P, then there is a well-defined expected number $\Phi(P)$ of steps the adversary needs to take *at least* in order to cover the random challenge. We refer to $\Phi(P)$ as the *potential* of P. In particular, the best strategy is the greedy one, which looks at the largest $j = j(c_i) \leq c_i$ on which a pebble is placed, i.e., $j \in P$, and then needs to output a valid sequence of *at least* $c_i - j$ further pebbling configurations, such that the last configuration contains c_i. Note if $j = c_i$, we still need to perform one step to output a valid configuration. Therefore, $\Phi(P)$ is the expected value of $\max(1, c_i - j(c_i))$. We will consider a new game – called the *expectation game* – which has the property that at the beginning of every stage, the challenger just computes $\Phi(P)$, and expects the player $\mathtt{T}$ to take $\Phi(P)$ legal steps until $\mathtt{T}$ can move to the next stage.

Note that these steps can be totally arbitrary – there is no actual challenge any more to cover. Still, we will be interested in lower bounding the *cumulative complexity* of such a strategy for the expectation game, and it is not obvious how $\mathtt{T}$ can keep the cc low. Indeed:

- If the potential is high, say $\Phi(P) = \Omega(n)$, then this means that linearly many steps must be taken to move to the next stage, and since every configuration contains at least one pebble, we pay a cumulative cost of $\Omega(n)$ for the present stage.
- Conversely, if the potential $\Phi(P)$ is *low* (e.g., $O(1)$), then we can expect to be faster. However we will show that this implies that there are *many* pebbles in P (at least $\Omega(n/\Phi(P))$), and thus one can expect high cumulative cost again, i.e,, linear $\Omega(n)$.

However, there is a catch – the above statements refer to the *initial configurations*. The fact that we have many pebbles at the beginning of a stage and at its end, does not mean we have many pebbles *throughout the whole stage.* Even though the strategy $\mathtt{T}$ is forced to pay $\Phi(P)$ steps, the strategy may try to drop as many pebbles as possible for a while, and then adding them back again. *Excluding that this can happen is the crux of our proof.* We will indeed show that for the expectation game, any strategy incurs cumulative complexity $\Omega(n^2/\log^2(n))$ roughly. The core of the analysis will be understanding the behavior of the potential function throughout a stage.

Now, we can expect that a low-cc strategy $\mathtt{T}$ for the original parallel pebbling game on L_n gives us one for the expectation game too – after all, for every challenge, the strategy $\mathtt{T}$ needs to perform roughly $\Phi(P)$ steps from the initial pebbling configuration when learning the challenge. This is *almost* correct, but again, there is a small catch. The issue is that $\Phi(P)$ is only an expectation, yet we want to have the guarantee that we go for $\Phi(P)$ steps with sufficiently high probability (this is particularly crucial if we want to prove a statement which is parameterized by δ). However, this is fairly simple (if somewhat tedious) to overcome – the idea is that we partition the n challenges into n/λ groups of λ challenges. For every such group, we look at the initial configuration P when learning the first of the next λ challenges, and note that with sufficiently high probability (roughly $e^{-\Omega(\lambda^2)}$ by a Chernoff bound) there will be one challenge

(among these λ ones) which is at least (say) $\Phi(P)/2$ away from the closest pebble. This allows us to reduce a strategy for the n-challenge pebbling game on L_n to a strategy for the (n/λ)-round expectation game. The value of λ can be chosen small enough not to affect the overall analysis.

Proof (Theorem 4). As the first step in the proof, we are going to reduce playing the game $\mathsf{pebble}(L_n, C = [n], n, \mathtt{T}, P_{\mathsf{init}})$, for an arbitrary player $\mathtt{T}$ and initial pebbling configuration P_{init} ($|P_{\mathsf{init}}| \leq k$), to a simpler (and somewhat different) pebbling game, which we refer to as the *expectation game*.

To this end, we introduce first the concept of a *potential function* $\Phi : 2^{[n]} \to \mathbb{N}$. The *potential* of a pebbling configuration $P = \{\ell_1, \ell_2, \ldots, \ell_m\} \subseteq [n]$ is

$$\begin{aligned}\Phi(P) &:= \tfrac{m}{n} + \tfrac{1}{n}\sum_{i=0}^{m}(1 + \ldots + (l_{i+1} - l_i - 1)) \\ &= \tfrac{m}{n} + \tfrac{1}{2n}\sum_{i=0}^{m}(\ell_{i+1} - \ell_i)\cdot(\ell_{i+1} - \ell_i - 1) = \tfrac{1}{2n}\sum_{i=0}^{m}(\ell_{i+1} - \ell_i)^2 - \tfrac{n+1-2m}{2n}\end{aligned}$$

Here $m = |P|$ and we let $\ell_0 = 0$ and $\ell_{m+1} = n + 1$ as notational placeholders. Indeed, $\Phi(P)$ is the expected number of moves required (by an optimal strategy) to pebble a random challenge starting from the pebbling configuration P, where the expectation is over the choice of the random challenge. (Note in particular it is required to pay at least one move even if a pebble is already on the challenge node.) In other words, $\Phi(P)$ is exactly $\mathsf{time}(L_n, [n], 1, \mathtt{T}^*, P)$ for the optimal strategy $\mathtt{T}^*$.

Now we are ready to introduce the *expectation game* which has no challenge. At the beginnning of every stage, the challenger only computes $\Phi(P)$, and expects the player $\mathtt{T}$ to take $\Phi(P)$ steps until he can move to the next stage. The game $\mathsf{expect}(n, t, \mathtt{T}, P_{\mathsf{init}})$ is played by a pebbler $\mathtt{T}$ as depicted in Figure 4.

In the following, for a (randomized) pebbler $\mathtt{T}$ and initial configuration P_{init}, we write $\mathsf{expect}_{n,t}(\mathtt{T}, P_{\mathsf{init}})$ for the output of the expectation game; note the output only depends on the randomness of pebbler $\mathtt{T}$ and configuration P_{init}. We similarly define the *cumulative complexity of the expectation game*

$$\mathsf{cc}_\delta(\mathsf{expect}_{n,t}(\mathtt{T}, P_{\mathsf{init}})) := \inf\left\{\gamma \;\middle|\; \mathop{\mathbb{P}}_{\mathsf{expect}(n,t,\mathtt{T},P_{\mathsf{init}})}[\mathsf{cost} \geq \gamma] \geq 1 - \epsilon\right\}$$

$$\mathsf{cc}_\delta(\mathsf{expect}_{n,t,k}) := \min_{\substack{\mathtt{T}, P_{\mathsf{init}} \subseteq V \\ |P_{\mathsf{init}}| \leq k}} \left\{\mathsf{cc}_\delta(\mathsf{expect}_{n,t}(\mathtt{T}, P_{\mathsf{init}}))\right\}$$

The expectation game $\mathsf{expect}_{n,t,k}$ has an important feature: because the randomness is only over the pebbler's coins, these coins can be fixed to their optimal choice without making the overall cc worse. This implies that $\mathsf{cc}_\delta(\mathsf{expect}_{n,t,k}) = \mathsf{cc}_0(\mathsf{expect}_{n,t,k})$ for all $\delta \geq 0$. In particular, we use the shorthand $\mathsf{cc}(\mathsf{expect}_{n,t,k})$ for the latter.

The remainder of the proof consists of the following two lemmas. Below, we combine these two lemmas in the final statement, before turning to their proofs.

$\underline{\mathsf{expect}}(n, t, \mathtt{T}, P_{\mathsf{init}})$: The t-round expectation game of parameter n and an initial pebbling configuration $P_{\mathsf{init}} \subseteq V$ is played by challenger and player $\mathtt{T}$ as follows.

1. Initialize $\mathsf{cnt} := 0$, $\mathsf{round} := 0$, $P_{\mathsf{cnt}} := P_{\mathsf{init}}$ and $\mathsf{cost} := |P_{\mathsf{init}}|$.
2. Player $\mathtt{T}$ submits a sequence of *non-empty* pebbling configurations $(P_{\mathsf{round},1}, \ldots, P_{\mathsf{round},t_{\mathsf{round}}}) \subset [n]^{\times t_{\mathsf{round}}}$,
3. Let $P_{\mathsf{round},0} := P_{\mathsf{cnt}}$. Check if $t_{\mathsf{round}} \geq \Phi(P_{\mathsf{cnt}})$ and $\forall i \in [t_{\mathsf{round}}]$

$$\forall v \in P_{\mathsf{round},i} \setminus P_{\mathsf{round},i-1} \ : \ \mathsf{parent}(v) \in P_{\mathsf{round},i-1} \ .$$

   If check fails, output $\mathsf{cnt} = \mathsf{cost} = \infty$ and halt.
4. $\mathsf{cnt} := \mathsf{cnt} + t_{\mathsf{round}}$.
5. $\mathsf{cost} := \mathsf{cost} + \sum_{j=1}^{t_{\mathsf{round}}} |P_{\mathsf{round},j}|$.
6. $P_{\mathsf{cnt}} := P_{\mathsf{round},t_{\mathsf{round}}}$.
7. $\mathsf{round} := \mathsf{round} + 1$. If $\mathsf{round} < t$ go to step 2, otherwise if $\mathsf{round} = t$ the experiment is over, the output is the final count cnt and the cumulative cost cost.

Fig. 4: The Expectation Game

(The proof of Lemma 5 is deferred to the full version for lack of space, and relies on the intuition given above.)

Lemma 5 (Reduction to the Expectation Game). *For all n, t, k, λ, and any $\delta > 3\mu(t, \lambda)$, we have*

$$\mathsf{cc}(\mathsf{expect}_{n,t,k}) = \mathsf{cc}_{\delta - 3\mu(t,\lambda)}(\mathsf{expect}_{n,t,k}) \leq 2 \cdot \mathsf{cc}_{\delta}(L_n, C = [n], t \cdot \lambda, k) \ ,$$

where $\mu(t, \lambda) = t \cdot e^{-\lambda^2/8}$.

To give some intuition about the bound, note that in general, for every $\delta' \leq \delta$, we have $\mathsf{cc}_{\delta'}(\mathsf{expect}_{n,t,k}) \leq \mathsf{cc}_{\delta}(\mathsf{expect}_{n,t,k})$. This is because if a c is such that for all $\mathtt{T}$ and P_{init} we have $\mathbb{P}(\mathsf{expect}_{n,t}(\mathtt{T}, P_{\mathsf{init}}) \geq c) \geq 1 - \delta'$, then also $\mathbb{P}(\mathsf{expect}_{n,t}(\mathtt{T}, P_{\mathsf{init}}) \geq c) \geq 1 - \delta$. Thus the set from which we are taking the supremum only grows bigger as δ increases. In the specific case of Lemma 5, the $3\mu(t, \lambda)$ offset captures the loss of our reduction.

Lemma 6 (CC Complexity of the Expectation Game). *For all $t, 0 \leq k \leq n$ and $\epsilon > 0$, we have*

$$\mathsf{cc}(\mathsf{expect}_{n,t,k}) \geq \left\lfloor \frac{\epsilon t}{2} \right\rfloor \cdot \frac{n^{1-\epsilon}}{6} \ .$$

To conclude the proof before turning to the proofs of the above two lemmas, we choose t, λ such that $t \cdot \lambda = n$, and $\mu(t, \lambda) = t \cdot e^{-\lambda^2/8} < \delta/3$. We also set $\epsilon = 0.5 \log\log(n)/\log(n)$, and note that in this case $n^{1-\epsilon} = n/\sqrt{\log(n)}$. In

particular, we can set $\lambda = O(\sqrt{\log t})$, and can choose e.g. $t = n/\sqrt{\log n}$. Then, by Lemma 6,

$$\mathsf{cc}(\mathsf{expect}_{n,t,k}) \geq \left\lfloor \frac{\epsilon t}{2} \right\rfloor \cdot \frac{n^{1-\epsilon}}{6} = \Omega\left(\frac{n^2}{\log^2(n)}\right) .$$

This concludes the proof of Theorem 4.

Proof (Proof of Lemma 6). First we observe if a pebbling configuration P has potential Φ, the size $|P|$ of the pebbling configuration (i.e., the number of vertices on which a pebble is placed) will be at least $\frac{n}{6\cdot\Phi}$. We give a formal proof for completeness.[10].

Lemma 7. *For every* non-empty *pebbling configuration* $P \subseteq [n]$*, we have*

$$\Phi(P) \cdot |P| \geq \frac{n}{6} .$$

Proof. Let $m = |P| \geq 1$, by definition of *potential*:

$$\Phi(P) = \frac{1}{2n}\sum_{i=0}^{m}(\ell_{i+1} - \ell_i)^2 - \frac{n+1-2m}{2n} ,$$

where $\ell_0 = 0$ and $\ell_{m+1} = n+1$ are notational placeholders. Since $\Phi(P) \geq 1$ and $m \geq 1$, we have $\frac{n+1-2m}{2n} \leq \frac{1}{2} \leq \frac{1}{2}\cdot\Phi(P)$. Therefore

$$\Phi(P) \geq \frac{2}{3}\cdot\frac{1}{2n}\sum_{i=0}^{m}(\ell_{i+1} - \ell_i)^2 ,$$

since $m \geq \frac{m+1}{2}$, multiply the left side by m and the right side by $\frac{m+1}{2}$, we have

$$\Phi(P)\cdot m \geq \frac{2}{3}\left(\frac{1}{2n}\sum_{i=0}^{m}(\ell_{i+1} - \ell_i)^2\right)\cdot\frac{m+1}{2} = \frac{1}{6n}\left(\sum_{i=0}^{m}(\ell_{i+1} - \ell_i)^2\right)\cdot(m+1)$$

Therefore $\Phi(P)\cdot m \geq \frac{n}{6}$ follows, since by Cauchy-Schwarz Inequality we have

$$\left(\sum_{i=0}^{m}(\ell_{i+1} - \ell_i)^2\right)\cdot(m+1) \geq \left(\sum_{i=0}^{m}(\ell_{i+1} - \ell_i)\right)^2 \geq n^2 .\square$$

Also, the following claim provides an important property of the potential function.

Lemma 8. *In one iteration, the potential can decrease by at most one.*

[10] Note that the contra-positive is not necessarily true. A simple counter-example is when pebbles are placed on vertices $[0, n/2]$ of C_n (that is, $|P| = \mathrm{O}(n)$). The expected number of moves in this case is still $\Omega(n)$.

Proof. Consider an arbitrary configuration $P = \{\ell_1, \ell_2, \ldots, \ell_m\} \subseteq [n]$. The best that a pebbling algorithm can do to decrease the potential is to place new pebbles next to *all* the current pebbles – let's call the new configuration P'. That is,

$$P' = \{\ell_1, \ell_1 + 1, \ell_2, \ell_2 + 1, \ldots, \ell_m, \ell_m + 1\} \subseteq [n].$$

The potential of the new configuration is

$$\Phi(P') = \frac{1}{2n}\left(\ell_1^2 + \sum_{i=1}^{m} 1 + (\ell_{i+1} - (\ell_i + 1))^2\right) - \frac{n + 1 - 2|P'|}{2n} \tag{11}$$

$$= \frac{1}{2n}\left(m + \sum_{i=0}^{m}\left((\ell_{i+1} - \ell_i)^2 - 2(\ell_{i+1} - \ell_i) + 1\right)\right) - \frac{n + 1 - 2|P'|}{2n} \tag{12}$$

$$\geq \frac{1}{2n}\left(m + \sum_{i=0}^{m}\left((\ell_{i+1} - \ell_i)^2 - 2(\ell_{i+1} - \ell_i) + 1\right)\right) - \frac{n + 1 - 2m}{2n} \tag{13}$$

$$\geq \Phi(P) + \frac{m}{n} - \frac{1}{n}\sum_{i=0}^{m}(\ell_{i+1} - \ell_i) \geq \Phi(P) - 1 \tag{14}$$

where the first inequality holds because $|P'| \geq m$. □

Assume without loss of generality the pebbler $\mathtt{T}$ is legal and deterministic. Consider a particular round $i \in [t]$ of the expectation game. Let P and P' denote the initial and final pebbling configurations in the i-th round, and let us denote by $\phi_i = \Phi(P)$ the potential of the initial configuration in round i. Depending on the value of $\Phi(P')$, we classify the pebbling sequence from P to P' into three different categories:

Type 1: $\Phi(P') > \phi_i \cdot n^{\epsilon/2}$; or
Type 2: $\Phi(P') \leq \phi_i \cdot n^{\epsilon/2}$ – we have two sub-cases:
- Type 2a: the potential was *always* less than $\phi_i \cdot n^{\epsilon}$ for all the intermediate pebbling configurations from P to P'; or
- Type 2b: the potential went above $\phi_i \cdot n^{\epsilon}$ for some intermediate configuration.

With each type, we associate a cost that the pebbling algorithm has to pay, which lower bounds the contribution to the cumulative complexity of the pebbling configurations generated during this stage. The pebbling algorithm can carry out pebbling of Type 1 for *free*[11] – however, the latter two have accompanying costs.

– For pebbling sequences of Type 2a, the corresponding cumulative cost is at least $\phi_i \cdot \frac{n}{6 \cdot \phi_i n^{\epsilon}} = \frac{1}{6} n^{1-\epsilon}$ since by Lemma 7, the size of the pebbling configuration is never less than $\frac{n}{6\phi_i n^{\epsilon}}$ during all intermediate iterations and in stage i valid pebbler must produce at least ϕ_i configurations.

[11] The cost might be greater than zero, but setting it to zero doesn't affect the lower bound.

– For sequences of Type 2b, by Lemma 8, it follows that in a Type 2b sequence it takes at least $\phi_i(n^\epsilon - n^{\epsilon/2})$ steps to decrease the potential from $\phi \cdot n^\epsilon$ to $\phi_i \cdot n^{\epsilon/2}$, and the size of the pebbling configuration is at least $\frac{n}{6\phi_i n^\epsilon}$ in every intermediate step by Lemma 7. Therefore, the cumulative cost is at least

$$\phi_i(n^\epsilon - n^{\epsilon/2}) \cdot \frac{n}{6\phi_i n^\epsilon} \geq \frac{n}{6} - \frac{n^{1-\epsilon/2}}{6} \geq \frac{1}{6} n^{1-\epsilon} ,$$

where the last inequality follows for sufficiently large n.

To conclude the proof, we partition the $t \geq \lceil 2/\epsilon \rceil$ rounds into groups of consecutive $\lceil 2/\epsilon \rceil$ phases. We observe that any group *must* contain at least one pebbling sequence of Type 2: otherwise, with ϕ being the potential at the beginning of the first of theses $2/\epsilon$ phases, the potential at the end would be strictly larger than

$$\phi n^{\frac{\epsilon}{2} \cdot \frac{2}{\epsilon}} \geq \phi \cdot n > n/2$$

which cannot be, as the potential can be at most $\frac{n}{2}$. By the above, however, the cumulative complexity of each group of phases is at least $\frac{n^{1-\epsilon}}{6}$, and thus we get

$$cc(\mathsf{expect}_{n,t,k}) \geq \left\lfloor \frac{\epsilon t}{2} \right\rfloor \cdot \frac{n^{1-\epsilon}}{6} , \tag{15}$$

which concludes the proof of Lemma 6. □

As the second result, we show that the above theorem also holds for the entangled case.

Theorem 9 (Entangled Pebbling Complexity of the Line Graph). *For all $0 \leq k \leq n$ and constant $\delta > 0$,*

$$\mathsf{cc}^{\updownarrow}_{\delta}(L_n, C = [n], n, k) = \Omega\left(\frac{n^2}{\log^2 n}\right) .$$

Luckily, it will not be necessary to repeat the whole proof. We will give now a proof sketch showing that in essence, the proof follows by repeating the same format and arguments as the one for Theorem 4, using Lemma 3 as a tool.

Proof (Sketch). One can prove the theorem following exactly the same framework of Theorem 4, with a few differences. First off, we define a natural entangled version of the expectation game where, in addition to allowing entanglement in a pebbling configuration, we define the potential as

$$\Phi^{\updownarrow}(P) = \mathsf{time}^{\updownarrow}(L_n, C = [n], 1, \mathsf{T}^{*,\updownarrow}, P) ,$$

i.e., the expected time complexity for *one* challenge of an optimal entangled strategy $\mathsf{T}^{*,\updownarrow}$ starting from the (entangled) pebbling configuration P.

First off, a proof similar to the one of Lemma 5, based on a Chernoff bound, can be used to show that if we separate challenges in t chunks of λ challenges

each, and we look at the configuration P at the beginning of each of the t chunks, then there exists at least one challenge (out of λ) which requires spending time $\Phi^{\updownarrow}(P)$ to be covered, except with small probability.

A lower bound on the cumulative complexity of the (entangled) expectaton game follows exactly the same lines as the proof as Lemma 6. This is because the following two facts (which correspond to the two lemmas in the proof of Lemma 6) are true also in the setting with entanglement:

- First off, for every P and $\mathtt{T}^{*,\updownarrow}$ such that $\Phi^{\updownarrow}(P) = \mathsf{time}^{\updownarrow}(L_n, C = [n], 1, \mathtt{T}^{*,\updownarrow}, P)$, Lemma 3 guarantees that there exist a (regular) pebbling strategy $\mathtt{T}'$ and a (regular) pebbling configuration P' such that $|P|_{\updownarrow} \geq |P'|$ and

$$\begin{aligned}\Phi^{\updownarrow}(P) &= \mathsf{time}^{\updownarrow}(L_n, C = [n], 1, \mathtt{T}^{*,\updownarrow}, P) \\ &\geq \mathsf{time}(L_n, C = [n], 1, \mathtt{T}', P') \geq \Phi(P') \ .\end{aligned}$$

  Therefore, by Lemma 7,

$$|P|_{\updownarrow} \cdot \Phi^{\updownarrow}(P) \geq |P'| \cdot \Phi(P') \geq \frac{n}{6} \ . \tag{16}$$

- Second, the potential can decrease by at most one when making an arbitrary step from one configuration P to one configuration P'. This is by definition – assume it were not the case, and $\Phi^{\updownarrow}(P') < \Phi^{\updownarrow}(P) - 1$. Then, there exists a strategy to cover a random challenge starting from P which first moves to P' in one step, and then applies the optimal strategy achieving expected time $\Phi^{\updownarrow}(P')$. The expected number of steps taken by this strategy is smaller than $\Phi^{\updownarrow}(P)$, contradicting the fact that $\Phi^{\updownarrow}(P)$ is the optimal number of steps required by any strategy. □

4 From Pebbling to pROM

4.1 Trancscipts and Traces

Below we define the notion of a trace and transcript, which will allow us to relate the $\mathsf{computeLabel}$ and $\mathsf{pebble}^{\updownarrow}$ experiments. For any possible sequence of challenges $\boldsymbol{c} \in C^m$, let $\mathsf{cnt}_{\boldsymbol{c}}$ denote the number of steps (i.e., the variable cnt) made in the $\mathsf{computeLabel}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$ experiment conditioned on the m challenges being $\boldsymbol{c}$ (note that once $\boldsymbol{c}$ is fixed, the entire experiment is deterministic, so $\mathsf{cnt}_{\boldsymbol{c}}$ is well defined). Let $\tau_{\boldsymbol{c}} = q_1|q_2|\ldots|q_{\mathsf{cnt}_{\boldsymbol{c}}}$ be the *trace* of the computation: here $q_1 \subset [n]$ means that the first batch of parallel queries are the queries required to output the labels $\{\ell_i, i \in q_1\}$, etc..

For example, for the Graph in Figure 5, $\tau_7 = 2|4,5|7$ corresponds to a first query $\ell_2 = \mathsf{h}(2)$, then two parallel queries $\ell_4 = \mathsf{h}(4, \ell_1), \ell_5 = \mathsf{h}(5, \ell_2)$, and then the final query computing the label of the challenge $\ell_7 = \mathsf{h}(7, \ell_4, \ell_5, \ell_6)$.

A trace as a pebbling. We can think of a trace as a parallel pebbling, e.g., $\tau_7 = 2|4,5|7$ means we pebble node 2 in the first step, nodes $4,5$ in the second, and 7 in the last step. We say that an initial (entangled) pebbling configuration P_{init} is consistent with a trace τ, if starting from P_{init}, τ is a valid pebbling sequence. E.g., consider again the traces $\tau_7 = 2|4,5|7, \tau_8 = 3|6|8$ for the graph in Figure 5, then $P_{\mathsf{init}} = \{1,5,6\}$ is consistent with τ_7 and τ_8, and it's the smallest initial pebbling having this property. In the entangled case, $P_{\mathsf{init}}^{\updownarrow} = \{\langle 1\rangle_0, \langle 5,6\rangle_1\}$ is consistent with τ_7, τ_8. Note that in the entangled case we only need a pebbling configuration of weight 2, whereas the smallest pebbling configuration for the standard pebbling game has weight 3. In fact, there are traces where the gap between the smallest normal and entangled pebbling configuration consistent with all the traces can differ by a factor $\Theta(n)$.

Turning a trace into a transcript. We define the implications $T_{\boldsymbol{c}}$ of a trace $\tau_{\boldsymbol{c}} = q_1|q_2|\ldots|q_{\mathsf{cnt}_{\boldsymbol{c}}}$ as follows. For $i = 1,\ldots,\mathsf{cnt}_{\boldsymbol{c}}$, we add the implication $(v_i) \to (f_i)$, where $v_i \subset [n]$ denotes all the vertices whose labels have appeared either as inputs or outputs in the experiment so far, and f_i denotes the labels contained in the inputs from this round which have never appeared before (if the guess for the challenge label in this round is non-empty, i.e., $\ell \neq \perp$, then we include ℓ in f_i).

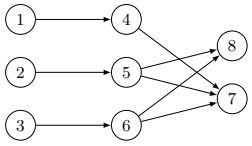

Fig. 5

Example 10. Consider the graph from Figure 5 with $m = 1$ and challenge set $C = \{7,8\}$, and traces

$$\tau_7 = 2|4,5|7 \quad \text{and} \quad \tau_8 = 3|6|8$$

We have

$$T_7 = \{(2) \to 1, (1,2,4,5) \to 6\} \quad T_8 = \{(3,6) \to 5\} \tag{17}$$

Where e.g. $(2) \to 1$ is in there as the first query is $\ell_2 = \mathsf{h}(2)$, and the second query is $\ell_4 = \mathsf{h}(4,\ell_1)$ and in parallel $\ell_5 = \mathsf{h}(5,\ell_2)$. At this point we so far only observed the label $v_2 = \{\ell_2\}$, so the label $f_2 = \{\ell_1\}$ used as input in this query is fresh, which means we add the implication $(2) \to 1$.

Above we formalised how to extract a transcript $T_{\boldsymbol{c}}$ from $(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$, with

$$T(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h}) = \cup_{\boldsymbol{c}\in C^m} T_{\boldsymbol{c}}$$

we denote the union of all $T_{\boldsymbol{c}}$'s.

4.2 Extractability, Coverability and a Conjecture

In this section we introduce the notion of extractability and coverability of a transcript. Below we first give some intuition what these notions have to do with the $\mathsf{computeLabel}$ and $\mathsf{pebble}^{\updownarrow}$ experiments.

Extractability intuition. Consider the experiment $\mathsf{computeLabel}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$. We can invoke $\mathtt{A}$ on some particular challenge sequence $\boldsymbol{c} \in C^m$, and if at some point $\mathtt{A}$ makes a query whose input contains a label ℓ_i which has not appeared before, we can "extract" this value from $(\mathtt{A}, \sigma_{\mathsf{init}})$ without actually querying h for it. More generally, we can run $\mathtt{A}$ on several challenge sequences scheduling queries in a way that will maximise the number of labels that can be extracted from $(\mathtt{A}, \sigma_{\mathsf{init}})$. To compute this number, we don't need to know the entire input/output behaviour of $\mathtt{A}$ for all possible challenge sequences, but the transcript $T = T(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$ is sufficient. Recall that T contains implication like $(1, 5, 6) \to 3$, which means that for some challenge sequence, there's some point in the experiment where $\mathtt{A}$ has already seen the labels ℓ_1, ℓ_5, ℓ_6, and at this point makes a query whose input contains a label ℓ_3 (that has not been observed before). Thus, given σ_{init} and ℓ_1, ℓ_5, ℓ_6 we can learn ℓ_3.

We denote with $ex(T)$ the maximum number of labels that can be extracted from T. If the labels are uniformly random values in $\{0,1\}^w$, then it follows that σ_{init} will almost certainly not be much smaller than $ex(T) \cdot w$, as otherwise we could compress $w \cdot ex(T)$ uniformly random bits (i.e., the extracted labels) to a string which is shorter than their length, but uniformly random values are not compressible.

Coverability intuition. In the following, we say that an entangled pebbling experiment $\mathsf{pebble}^{\updownarrow}(G, C, m, \mathsf{P}, P_{\mathsf{init}}{}^{\updownarrow})$ mimics the $\mathsf{computeLabel}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$ experiment if for every challenge sequence the following is true: whenever $\mathtt{A}$ makes a query to compute some label $\ell_i = h(i, \ell_{p_1}, \ldots, \ell_{p_t})$, P puts a (normal) pebble on i. For this $P_{\mathsf{init}}{}^{\updownarrow}$ must contain (entangled) pebbles that allow to cover every implication in T (as defined above), e.g., if $(1, 5, 6) \to 3 \in T$, then from the initial pebbling $P_{\mathsf{init}}{}^{\updownarrow}$ together with the pebbles $\langle 1\rangle_0, \langle 5\rangle_0, \langle 6\rangle_0$ seen so far it must be possible derive $\langle 3\rangle_0$, i.e., $\langle 3\rangle_0 \in \mathsf{closure}(P_{\mathsf{init}}{}^{\updownarrow} \cup \langle 1\rangle_0, \langle 5\rangle_0, \langle 6\rangle_0\})$. We say that such an initial state $P_{\mathsf{init}}{}^{\updownarrow}$ covers T. We're interested in the maximum possible ratio of $\max_{T \text{ over } [n]} \min_{P_{\mathsf{init}}{}^{\updownarrow}, P_{\mathsf{init}}{}^{\updownarrow} \text{ covers } T} |P_{\mathsf{init}}{}^{\updownarrow}|_{\updownarrow} / ex(T)$, which we'll denote with γ_n, thus, if any T is k extractable, it can be covered by an initial pebbling $P_{\mathsf{init}}{}^{\updownarrow}$ of weight $\gamma_n \cdot k$. The best current lower bound we have on γ_n is 1.5, we conjecture that γ_n is small, polylog(n) or even constant. We will prove in § 4.3 that pebbling time complexity implies pROM time complexity for any graph, and in § 4.4 that CC complexity implies cumulative complexity in the pROM model for the scrypt graph. The loss in our reductions will depend on γ_n. Assuming $\gamma_n = \Theta(1)$ we get the best bounds one can hope for, but already $\gamma_n \in o(n)$ would give the first non-trivial bounds on pROM complexity.

Definitions. Let $n \in \mathbb{N}$. An "implication" $(\mathcal{X}) \to z$ given by a value $z \in [n]$ and a subset $\mathcal{X} \subset [n] \setminus z$ means that "knowing $\mathcal{X}$ gives z for free". We use $(\mathcal{X}) \to \mathcal{Z}$ as a shortcut for the set of implications $\{(\mathcal{X}) \to z \ : \ z \in \mathcal{Z}\}$.

A transcript is a set of of implications. Consider a transcript $T = \{\alpha_1, \ldots, \alpha_\ell\}$, each α_i being an implication. We say that a transcript T is k $(0 \leq k \leq n)$ extractable if there exists an extractor E that makes at most $n - k$ queries in the following game:

- At any time E can query for a value in $[n]$.
- Assume E has values $\mathcal{L} \subset [n]$ and there exists an implication $(\mathcal{X}) \to z \in T$ where $\mathcal{X} \subset \mathcal{L}$, then E gets the value z "for free".
- The game is over when E has received all of $[n]$.

Every (even an empty) transcript T is 0 extractable as E can always simply ignore T and query for $1, 2, \ldots, n$. Let

$$ex(T) = \max_k(T \text{ is } k\text{-extractable})$$

Example 11. Let $n = 5$ and consider the transcript

$$T = \{(1,2) \to 3, (2,3) \to 1, (3,4) \to 2, (1) \to 4\} \tag{18}$$

This transcript is 2 but not 3 extractable. To see 2 extractability consider the E which first asks for 1, then gets 4 for free (due to $(1) \to 4$), next E asks for 2 and gets 3 for free (due to $(1,2) \to 3$).

A set S of entangled pebbles covers an implication $(\mathcal{X}) \to z$ if $z \in \mathsf{closure}(S \cup \langle \mathcal{X} \rangle_0)$, with $\mathsf{closure}$ as defined in Definition 1.

Definition 12 (k-coverable). *We say that a transcript T is k-coverable if there exists a set of entangled pebbles S of total weight k such that every implication in T is covered by S. With $cw(T)$ we denote the minimum weight of an S covering T:*

$$cw(T) = \min_{S \ that \ covers \ T} |S|_{\updownarrow}$$

Note that every transcript is trivially n coverable by using the pebble $\langle 1, \ldots, n \rangle_0$ of weight n which covers every possible implication. For the 2 extractable transcript from Example 11, a set of pebbles of total weight 2 covering it is

$$S = \{\langle 1,2,3 \rangle_2, \langle 1,4 \rangle_1\} \tag{19}$$

For example $(3,4) \to 2$ is covered as $2 \in \mathsf{closure}(\langle 1,2,3 \rangle_2, \langle 1,4 \rangle_1, \langle 3,4 \rangle_0) = \{1,2,3,4\}$: we first can set $\Gamma = \{3,4\}$ (using $\langle 3,4 \rangle_0$), then $\Gamma = \{1,3,4\}$ using $\langle 1,4 \rangle_1$, and then $\Gamma = \{1,2,3,4\}$ using $\langle 1,2,3 \rangle_2$.

We will be interested in the size of the smallest cover for a transcript T. One could conjecture that every k-extractable transcript is k-coverable. Unfortunately this is not true, consider the transcript

$$T^* = \{(2,5) \to 1, (1,3) \to 2, (2,4) \to 3, (3,5) \to 4, (1,4) \to 5\} \tag{20}$$

We have $ex(T^*) = 2$ (e.g. via query $2, 4, 5$ and extract $1, 3$ using $(2, 5) \to 1, (2, 4) \to 3$), but it's not 2-coverable (a cover of weight 3 is e.g. $\{\langle 5, 1\rangle_1\}, \langle 2, 3, 4\rangle_1\}$). With γ_n we denote the highest coverability vs extractability ration that a transcript over $[n]$ can have:

Conjecture 13. Let

$$\gamma_n = \max_{T \text{ over } [n]} \min_{S \text{ that covers } T} \frac{|S|_{\updownarrow}}{ex(T)} = \max_{T \text{ over } [n]} \frac{cw(T)}{ex(T)}$$

then (weak conjecture) $\gamma_n \in \text{polylog}(n)$, or even (strong conjecture) $\gamma_n \in \Theta(1)$.

By the example eq.(20) above, γ_n is at least $\gamma_n \geq \gamma_5 \geq 3/2$. We will update the full version of this paper as we get aware on progress on (dis)proving this conjecture. In the full version we also introduce another parameter $shannon(w)$, which can give better lower bounds on the size of a state required to realize a given transcript in terms of Shannon entropy.

4.3 Bounding pROM Time Using Pebbling Time

We are ultimately interested in proving lower bounds on time and cumulative complexity in the parallel ROM model. We first show that pebbling time complexity implies time complexity in the pROM model, the reduction is optimal up to a factor γ_n. Under conjecture 13, this basically answers the main open problem left in the Proofs of Space paper [9]. In the theorem below we need the label length w to in the order of $m \log(n)$ to get a lower bound on $|\sigma_{\mathsf{init}}|$. For the proofs of space application, where $m = 1$, this is a very weak requirement, but for scrypt, where $m = n$, this means we require rather long labels (the number of queries q will be $\leq n^2$, so the $\log(q)$ term can be ignored).

Theorem 14. *Consider any $G = (V, E), C \subseteq V, m \in \mathbb{N}, \epsilon \geq 0$ and algorithm $\mathtt{A}$. Let $n = |V|$ and γ_n be as in Conjecture 13. Let $\mathcal{H}$ contain all functions $\{0,1\}^* \to \{0,1\}^w$, then with probability $1 - 2^{-\Delta}$ over the choice of $\mathsf{h} \leftarrow \mathcal{H}$ the following holds for every $\sigma_{\mathsf{init}} \in \{0,1\}^*$. Let q be an upper bound on the total number of h queries made by $\mathtt{A}$ and let*

$$k = \frac{|\sigma_{\mathsf{init}}| + \Delta}{(w - m\log(n) - \log(q))}$$

(so $|\sigma_{\mathsf{init}}| \approx k \cdot w$ for sufficiently large w), then

$$\mathsf{time}^{\mathsf{pROM}}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h}) \geq \mathsf{time}(G, C, m, \lceil k \cdot \gamma_n \rceil)$$

and for every $1 > \epsilon \geq 0$

$$\mathsf{time}^{\mathsf{pROM}}_{\epsilon}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h}) \geq \mathsf{time}_{\epsilon}(G, C, m, \lceil k \cdot \gamma_n \rceil)$$

In other words, if the initial state is roughly $k \cdot w$ bits large (i.e., it's sufficient to store k labels), then the pROM time complexity is as large as the pebbling time complexity of $\mathsf{pebble}(G, C, m)$ for any initial pebbling of size $k \cdot \gamma_n$. Note that the above theorem is basically tight up to the factor γ_n: consider an experiment $\mathsf{time}(G, C, m, \mathsf{P}, P_{\mathsf{init}})$, then we can come up with a state σ_{init} of size $k \cdot w$, namely $\sigma_{\mathsf{init}} = \{\ell_i, i \in P_{\mathsf{init}}\}$, and define $\mathtt{A}$ to mimic P, which then implies

$$\mathsf{time}_{\epsilon}^{\mathtt{pROM}}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h}) = \mathsf{time}_{\epsilon}(G, C, m, \mathsf{P}, P_{\mathsf{init}}) \quad \text{with} \quad |\sigma_{\mathsf{init}}| = k \cdot w$$

in particular, if we let $\mathsf{P}, P_{\mathsf{init}}$ be the strategy and initial pebbling of size k minimising time complexity we get

$$\mathsf{time}_{\epsilon}^{\mathtt{pROM}}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h}) \geq \mathsf{time}_{\epsilon}(G, C, m, k) \quad \text{with} \quad |\sigma_{\mathsf{init}}| = k \cdot w$$

Wlog. we will assume that $\mathtt{A}$ is deterministic (if $\mathtt{A}$ is probabilistic we can always fix some "optimal" coins). Below we prove two claims which imply Theorem 14.

Claim. With probability $1 - 2^{-\Delta}$ over the choice of $\mathsf{h} \leftarrow \mathcal{H}$; If the transcript $T(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$ is k-extractable, then

$$|\sigma_{\mathsf{init}}| \geq k \cdot (w - m \log(n) - \log(q)) - \Delta \tag{21}$$

where q is an upper bound on the total number of h queries made by $\mathtt{A}$.

Proof. Let L be an upper bound on the length of queries made by $\mathtt{A}$, so we can assume that the input domain of h is finite, i.e., $\mathsf{h} : \{0,1\}^{\leq L} \to \{0,1\}^w$. Let $|\mathsf{h}| = 2^L \cdot w$ denote the size of h's function table.

Let $\ell_{i_1}, \ldots, \ell_{i_k}$ be the indices of the k labels (these must not be unique) that can be "extracted", and let h^- denote the function table of h, but where the rows are in a different order (to be defined), and the rows corresponding to the queries that output the labels to be extracted are missing, so $|\mathsf{h}| - |\mathsf{h}^-| = k \cdot w$.

Given the state σ_{init}, the function table of h^- and some extra information α discussed below, we can reconstruct the entire function table of h. As this table is uniform, and a uniform string of length s cannot be compressed below $s - \Delta$ bits except with probability $2^{-\Delta}$, we get that with probability $1 - 2^{-\Delta}$ eq.(21) must hold, i.e.,

$$|\sigma_{\mathsf{init}}| + |\mathsf{h}^-| + |\alpha| \geq |\mathsf{h}| - \Delta$$

as $|\mathsf{h}| - |\mathsf{h}^-| = k \cdot w$ we get

$$|\sigma_{\mathsf{init}}| \geq k \cdot w - |\alpha| - \Delta$$

It remains to define α and the order in which the values in h^- are stored. For every label to be extracted, we specify on what challenge sequence to run the adversary $\mathtt{A}$, and where exactly in this execution the label we want to extract appears (as part of a query made by $\mathtt{A}$). This requires up to $m \log(n) + \log(q)$ bits for every label to be extracted, so

$$|\alpha| \leq k \cdot (m \cdot \log(n) + \log(q))$$

The first part of h^- now contains the outputs of h in the order in which they are requested by the extraction procedure just outlined (if a query is made twice, then we have to remember it and not simply use the next entry in h^-). Let us stress thaw we only store the w bit long outputs, not the inputs, this is not a problem as we learn the corresponding inputs during the extraction procedure. The entries of h which are not used in this process and are not extracted labels, make up the 2nd part of the h^- table. As we know for which inputs we're still missing the outputs, also here we just have to store the w bit long outputs such that the inputs are the still missing inputs in lexicographic order.

Let us mention that if $\mathtt{A}$ behaved nice in the sense that all its queries are on inputs which are actually required to compute the corresponding labels, then we would only need $\log(n)$ bits extra information per label, namely the indices $i_1, \ldots, i_k$. But as $\mathtt{A}$ can behave arbitrarily, we can't tell when $\mathtt{A}$ actually uses real labels as inputs or some junk, and thus must exactly specify where the real labels to be extracted show up.

Claim. If the transcript $T = T(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$ is k-extractable (i.e., $ex(T) = k$), then

$$\mathsf{time}^{\mathtt{pROM}}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h}) \geq \mathsf{time}(G, C, m, \lceil k \cdot \gamma_n \rceil) \tag{22}$$

and for any $1 > \epsilon \geq 0$

$$\mathsf{time}^{\mathtt{pROM}}_{\epsilon}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h}) \geq \mathsf{time}_{\epsilon}(G, C, m, \lceil k \cdot \gamma_n \rceil) \tag{23}$$

Proof. We will only prove the first statement eq.(22). As T is k-extractable, there exist $(\mathsf{P}, P^{\updownarrow})$ where $P^{\updownarrow}$ is of weight $\leq \lceil k \cdot \gamma_n \rceil$ such that

$$\mathsf{time}^{\updownarrow}(G, C, m, \mathsf{P}, P^{\updownarrow}) = \mathsf{time}^{\mathtt{pROM}}(G, C, m, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$$

The claim now follows as

$$\mathsf{time}^{\updownarrow}(G, C, m, \mathsf{P}, P^{\updownarrow}) \geq \mathsf{time}^{\updownarrow}(G, C, m, \lceil k \cdot \gamma_n \rceil) = \mathsf{time}(G, C, m, \lceil k \cdot \gamma_n \rceil)$$

where the first inequality follows by definition (recall that $|P^{\updownarrow}|_{\updownarrow} \leq \lceil k \cdot \gamma_n \rceil$) and the second by Lemma 3 which states that for time complexity, entangled pebblings are not better than normal ones.

Theorem 14 follow directly from the two claims above.

4.4 The CMC of the Line Graph

Throughout this section $L_n = (V, E), V = [n], E = \{(i, i+1) \ : \ i \in [n-1]\}$ denotes the path of length n, and the set of challenge nodes $C = [n]$ contains all verticies. In Section 3 we showed that – with overwhelming probability over the choice of a function $\mathsf{h} : \{0,1\}^* \to \{0,1\}^w$ – the cumulative parallel entangled pebbling complexity for pebbling n challenges on a path of length n is

$$\mathsf{cc}^{\updownarrow}(L_n, C = [n], n, n) = \Omega\left(n^2/\log^2(n)\right)$$

this then implies a lower bound on the cumulative memory complexity in the pROM against the class $\mathcal{A}^{\updownarrow}$ of adversaries which are only allowed to store "encoding" of labels.

$$\mathsf{cmc}^{\mathtt{pROM}}_{\mathcal{A}^{\updownarrow}}(L_n, C = [n], n, n) = \Omega\left(w \cdot n^2/\log^2(n)\right)$$

This strengthens previous lower bounds which only proved lower bounds for CC complexity, which then implied security against pROM adversaries that could only store plain labels. In the full version, we show that $\mathsf{cc}^{\updownarrow}$ can be strictly lower than cc, thus, at least for some graphs, the ability to store encodings, not just plain labels, can decrease the complexity.

In this section we show a lower bound on $\mathsf{cmc}^{\mathtt{pROM}}(\mathsf{G}, C, m)$, i.e., without making any restrictions on the algorithm. Our bound will again depend on the parameter γ_n from Conjecture 13. We only sketch the proof as it basically follows the proof of Theorem 4.

Theorem 15. *For any $n \in \mathbb{N}$, let $L_n = (V = [n], E = \{(i, i+1) \ : \ i \in [n-1]\})$ be the line of length n and γ_n be as in Conjecture 13, and the label length $w = \Omega(n \log n)$, then*

$$\mathsf{cmc}^{\mathtt{pROM}}(L_n, C = [n], n, \sigma_{\mathsf{init}}) = \Omega\left(w \cdot n^2/\log^2(n) \cdot \gamma_n\right)$$

and for every $\epsilon > 0$

$$\mathsf{cmc}^{\mathtt{pROM}}_{\epsilon}(L_n, C = [n], n, \sigma_{\mathsf{init}}) = \Omega_{\epsilon}\left(w \cdot n^2/\log^2(n) \cdot \gamma_n\right)$$

Proof (sketch). We consider the experiment $\mathsf{computeLabel}(L_n, C, n, \mathtt{A}, \sigma_{\mathsf{init}}, \mathsf{h})$ for the $\mathtt{A}$ achieving the minimal $\mathsf{cmc}^{\mathtt{pROM}}$ complexity if h is chosen at random (we can assume $\mathtt{A}$ is deterministic). Let $(\mathsf{P}, P_{\mathsf{init}})$ be such that $\mathsf{pebble}^{\updownarrow}(L_n, C, n, \mathsf{P}, P_{\mathsf{init}})$ mimics (as defined above) this experiment. By Theorem 9, $\mathsf{cc}^{\updownarrow}(L_n, C = [n], n, n) = \Omega\left(n^2/\log^2(n)\right)$, unfortunately – unlike for time complexity – we don't see how this would directly imply a lower bound on $\mathsf{cmc}^{\mathtt{pROM}}$.

Fortunately, although Theorems 4 and 9 are about CC complexity, the proof is based on time complexity: At any timepoint the "potential" of the current state lower bounds the time required to pebble a random challenge, and if the potential is small, then the state has to be large (cf. eq.(16)).

For any $0 \leq i \leq n$ and $\boldsymbol{c} \in C^i$ let $\sigma_{\boldsymbol{c}}$ denote the state in the experiment $\mathsf{computeLabel}(L_n, C, n, \mathtt{A}, \sigma_{\mathsf{init}} = \emptyset, \mathsf{h})$ right after the i'th label has been computed by $\mathtt{A}$ and conditioned on the first i challenges being $\boldsymbol{c}$ (as $\mathtt{A}$ is deterministic and we fixed the first i challenges, $\sigma_{\boldsymbol{c}}$ is well defined).

At this point, the remaining experiment is $\mathsf{computeLabel}(L_n, C, n-i, \mathtt{A}, \sigma_{\boldsymbol{c}}, \mathsf{h})$. Similarly, we let $P_{\boldsymbol{c}}$ denote the pebbling in the "mimicing" $\mathsf{pebble}^{\updownarrow}(L_n, C, n - i, \mathsf{P}, P_{\boldsymbol{c}})$ experiment after P has pebbled the challenge nodes $\boldsymbol{c}$. Let $P'_{\boldsymbol{c}}$ be the entangled pebbling of the smallest possible weight such that there exists a P' such that $\mathsf{pebble}^{\updownarrow}(L_n, C, n - i, \mathsf{P}, P_{\boldsymbol{c}})$ and $\mathsf{pebble}^{\updownarrow}(L_n, C, n - i, \mathsf{P}', P'_{\boldsymbol{c}})$ make the same queries on all possible challenges.

The expected *time* complexity to pebble the $i+1$'th challenge in $\mathsf{pebble}^{\updownarrow}(L_n, C, n-i, \mathsf{P}', P'_{\boldsymbol{c}})$ – and thus also in $\mathsf{computeLabel}(L_n, C, n - i, \mathtt{A}, \sigma_{\boldsymbol{c}}, \mathsf{h})$ – is at least

$n/6|P'_{\boldsymbol{c}}|_{\updownarrow}$ by eq.(16). And by Theorem 14, we can lower bound the size of the state $\sigma_{\boldsymbol{c}}$ as (assuming w is sufficiently large)

$$|\sigma_{\boldsymbol{c}}| \geq \Omega(w \cdot |P'_{\boldsymbol{c}}|_{\updownarrow}/\gamma_n)$$

The CC cost of computing the next $(i+1)$th label in $\mathsf{computeLabel}(L_n, C, n-i, \mathtt{A}, \sigma_{\boldsymbol{c}}, \mathsf{h})$ – *if we assume that the state remains roughly around its initial size* $|\sigma_{\boldsymbol{c}}|$ *until the challenge is pebbled* – is roughly (cf. the intuition for the expectation game given in Section 3)

$$\frac{n}{2 \cdot |P'_{\boldsymbol{c}}|_{\updownarrow}} \cdot |\sigma_{\boldsymbol{c}}| = \Omega\left(\frac{n}{|P'_{\boldsymbol{c}}|_{\updownarrow}} \cdot \frac{w \cdot |P'_{\boldsymbol{c}}|_{\updownarrow}}{\gamma_n}\right) = \Omega\left(\frac{n \cdot w}{\gamma_n}\right)$$

As there are n challenges, this would give an $\Omega(w \cdot n^2/\gamma_n)$ bound on the overall CC complexity. Of course the above assumption that the state size never decreases is not true in general, an adversary case always chose to drop most of the pebbles once the challenge is known.

Note that in the above argument we don't actually use the size $|\sigma_{\boldsymbol{c}}|$ of the current state, but only argue using the potential of the lightest pebbling $P'_{\boldsymbol{c}}$ necessary to mimic the remaining experiment. Following the same argument as in Theorem 4 (in particular, using Lemma 8) one can show that for a $1/\log(n)$ fraction of the challenges, the potential says within a $\log(n)$ factor of its initial sizes. This argument will lose us a $1/\log^2(n)$ factor in the CC complexity, giving the claimed $\Omega\left(w \cdot n^2/\log^2(n) \cdot \gamma_n\right)$ bound.

Acknowledgments

Joël Alwen, Chethan Kamath, and Krzysztof Pietrzak's research is partially supported by an ERC starting grant (259668-PSPC). Vladimir Kolmogorov is partially supported by an ERC consolidator grant (616160-DOICV). Binyi Chen was partially supported by NSF grants CNS-1423566 and CNS-1514526, and a gift from the Gareatis Foundation. Stefano Tessaro was partially supported by NSF grants CNS-1423566, CNS-1528178, a Hellman Fellowship, and the Glen and Susanne Culler Chair.

This work was done in part while the authors were visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant CNS-1523467.

References

1. Crypto-Currency Market Capitalizations. http://coinmarketcap.com/. Accessed: 2015-10-07.
2. Leonardo C. Almeida, Ewerton R. Andrade, Paulo S. L. M. Barreto, and Marcos A. Simplicio Jr. Lyra: Password-based key derivation with tunable memory and processing costs. Cryptology ePrint Archive, Report 2014/030, 2014. http://eprint.iacr.org/2014/030.

3. Joël Alwen and Vladimir Serbinenko. High parallel complexity graphs and memory-hard functions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 595–603. ACM Press, June 2015.
4. Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Fast and tradeoff-resilient memory-hard functions for cryptocurrencies and password hashing. Cryptology ePrint Archive, Report 2015/430, 2015. http://eprint.iacr.org/2015/430.
5. Alex Biryukov and Dmitry Khovratovich. Tradeoff cryptanalysis of memory-hard functions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 633–657. Springer, Heidelberg, November / December 2015.
6. Donghoon Chang, Arpan Jati, Sweta Mishra, and Somitra Kumar Sanadhya. Time memory tradeoff analysis of graphs in password hashing constructions. pages 256–266, 2014.
7. Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 139–147. Springer, Heidelberg, August 1993.
8. Cynthia Dwork, Moni Naor, and Hoeteck Wee. Pebbling and proofs of work. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 37–54. Springer Berlin Heidelberg, 2005.
9. Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 585–605. Springer, Heidelberg, August 2015.
10. Stefan Dziembowski, Tomasz Kazana, and Daniel Wichs. One-time computable self-erasing functions. In Yuval Ishai, editor, *Theory of Cryptography*, volume 6597 of *Lecture Notes in Computer Science*, pages 125–143. Springer Berlin Heidelberg, 2011.
11. Christian Forler, Stefan Lucks, and Jakob Wenzel. Catena: A memory-consuming password scrambler. Cryptology ePrint Archive, Report 2013/525, 2013. http://eprint.iacr.org/2013/525.
12. Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In Bart Preneel, editor, *Secure Information Networks: Communications and Multimedia Security, IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS '99), September 20-21, 1999, Leuven, Belgium*, volume 152 of *IFIP Conference Proceedings*, pages 258–272. Kluwer, 1999.
13. B. Kaliski. Pkcs #5: Password-based cryptography specification version 2.0, 2000.
14. Lee, Charles. Litecoin, 2011.
15. Sunoo Park, Krzysztof Pietrzak, Albert Kwon, Joël Alwen, Georg Fuchsbauer, and Peter Gaži. Spacemint: A cryptocurrency based on proofs of space. Cryptology ePrint Archive, Report 2015/528, 2015. http://eprint.iacr.org/2015/528.
16. Colin Percival. Stronger key derivation via sequential memory-hard functions. 2009.