

# Disproving the Conjectures from “On the Complexity of Scrypt and Proofs of Space in the Parallel Random Oracle Model”

Daniel Malinowski and Karol Żebrowski<sup>(✉)</sup>

University of Warsaw, Warsaw, Poland  
daniel.malinowski@crypto.edu.pl, k.zebrowski@mimuw.edu.pl

**Abstract.** In the paper “On the Complexity of Scrypt and Proofs of Space in the Parallel Random Oracle Model” (Eurocrypt 2016) Joël Alwen et al. focused on proving a lower bound of the complexity of a general problem that underlies both proofs of space protocols [Dziembowski et al. CRYPTO 2015] as well as data-dependent memory-hard functions like **scrypt** — a key-derivation function that is used e.g. as proofs of work in cryptocurrencies like Litecoin.

In that paper the authors introduced a sequence  $\gamma_n$  and conjectured that this sequence is upper bounded by a constant. Alwen et al. proved (among other results) that the Cumulative Memory Complexity of the hash function **scrypt** is lower bounded by  $\Omega(n^2/(\gamma_n \cdot \log^2(n)))$ . If the sequence  $\gamma_n$  is indeed bounded by a constant then this lower bound can be simplified to  $\Omega(n^2/\log^2(n))$ .

In this paper we first show that  $\gamma_n > c\sqrt{\log(n)}$  and then we strengthen our result and prove that  $\gamma_n \geq \frac{\sqrt{n}}{\text{poly}(\log(n))}$ .

Alwen et al. introduced also a weaker conjecture, that is also sufficient for their results — they introduced another sequence  $\Gamma_n$  and conjectured that it is upper bounded by a constant. We show that this conjecture is also false, namely:  $\Gamma_n \geq c\sqrt{\log(n)}$ .

## 1 Introduction

The purpose of *proofs of work* is to provide a puzzle that requires a worker to dedicate a significant amount of resources to solve it, while still remaining feasible. Originally, this technique was developed to fight spam emails — if the sender had to dedicate some nontrivial amount of resources to send a single message then sending millions of spam emails would be unprofitable. However, proofs of work gained a lot of attention only recently — they are used in cryptocurrencies to solve the problem of double spending of coins.

Originally, the resource used in proofs of work was a *time* spent on the computations, and consequently the focus was on *time complexity* of the worker. In the

---

This work was supported by the Polish National Science Centre grant 2014/13/B/ST6/03540.

view of recent hardware advances, e.g. tailored ASIC devices, *memory-hardness* appears to be a much better requirement, as memory cost is not reduced by such devices. A candidate memory-hard function **script**, introduced by Percival in [9], aims to require the evaluator to either dedicate significant amount of space for the computation or highly increase the time spent on the evaluation. A similar space-time trade-off is imposed on the worker in *proofs of space* — a concept introduced by Dziembowski et al. in [5]. In proofs of space the worker can either dedicate a specified amount of the memory to generate proofs very efficiently, or save the space and pay increased time cost every time he generates the proof.

Alwen et al. in [1] focus on proving a lower bound of the complexity of a general problem that underlies both proofs of space protocols as well as the **script** function. To prove their main results, the authors of [1] introduced two combinatorial conjectures (either of them is sufficient for their results) and assumed that they are true.

In this paper we disprove both conjectures from [1]. To give a reader intuitions and a good understanding of the definitions required for stating the conjectures we give an introduction to [1] in Sect. 1.1. We remind the parallel Random Oracle Model, the labeling and pebbling games and how to calculate the Cumulative Memory Complexity of algorithms.

## 1.1 Introduction to [1]

Alwen et al. in [1] investigate lower bounds on the time and memory complexity of an adversary algorithm  $\mathcal{A}$  whose goal is to compute labels of nodes in a directed acyclic graph. In this game (we describe it in more details in Sect. 1.1) the label of a node is a hash  $h$  of node's index and the labels of its parents<sup>1</sup>. The hash function is modeled as a random oracle, so in order to compute the label,  $\mathcal{A}$  has to keep the labels of parent nodes in the memory.

Specific instances of this problem underlie *proofs of space* protocols constructed by Dziembowski et al. in [5]. Proofs of space is an alternative concept to proofs of work, in which a prover must dedicate a significant amount of his disc space as opposed to his computing power. Proofs of space are more environmentally friendly than proofs of work, because storage does not require energy. They can be used to create e.g. greener cryptocurrencies [8].

Another application of the problem considered in [1] is an examination of a memory-hard hash function<sup>2</sup> **script** introduced by Percival in [9]. The honest evaluation of the **script** function invokes underlying hash function  $h$  (modeled as a random oracle)  $n$  times, and requires storing  $n$  labels (where  $n$  is a parameter of **script**). As Percival stated, the expectation was that even for the adversary that parallelizes the computation it holds that  $S(n) \cdot T(n) \geq n^{2-\epsilon}$ , where  $S(n)$  and  $T(n)$  denote space and time invested, respectively. However, no rigorous proof of that fact was given. Another shortcoming of Percival's analysis was measuring

<sup>1</sup> Parent of a node  $v$  is any node  $w$  s.t. an edge  $(w, v)$  exists in the graph.

<sup>2</sup> Memory-hard hash functions require large storage during evaluation. They are used as password hashing functions and in proofs of work in cryptocurrencies.

memory complexity in terms of *maximum* memory used during computation. This does not take into account that the adversary could potentially *amortize* memory usage across *multiple* invocations of `script` function for multiple different inputs. To address this issue Alwen et al. consider a *cumulative memory complexity* proposed in [3]. We briefly recall this notion in Sect. 1.1.

### Cumulative Memory Complexity in Parallel Random Oracle Model.

Alwen and Serbinenko in [3] developed a new complexity metric better suited for capturing an amortized memory hardness of a given function. The intuition behind their model is that the adversary can use specialized hardware to evaluate many instances of the function in parallel. In such a situation only the amortized cost per single evaluation is important.

The authors of [3] consider an adversary whose goal is to compute a function  $\mathcal{H}^h$  (i.e. some function  $\mathcal{H}$  that depends on the oracle  $h$ ) with underlying hash function  $h$  modeled as a random oracle. The computation proceeds in steps and ends when the adversary computes  $\mathcal{H}^h$ . In each step the adversary gets the previous state  $\sigma_{i-1}$  (the state  $\sigma_0$  is set to the given initial state  $\sigma_{\text{init}}$ ), makes unbounded local computations and produces the next state  $\sigma_i$ . Additionally, once per step the adversary can send a *polynomial* (therefore *parallel* in the model name) set of queries to the random oracle and get back the hash values.

The *cumulative memory complexity* (CMC) of a single evaluation of  $\mathcal{H}^h$  is measured as  $\sum_i |\sigma_i|$ . CMC in parallel ROM model of  $\mathcal{H}^h$ , denoted  $\text{cmc}^{\text{PROM}}(\mathcal{H}^h)$ , is defined as minimal (over all the adversaries) expected CMC of the adversary computing  $\mathcal{H}^h$ .

**Labeling Games.** Alwen et al. in [1] proved that the hardness of `script`-like functions, as well as the security of proofs of space, rely on difficulty of the following game, called `computeLabel`.

The game is played on a single source and a single sink directed acyclic graph (DAG)  $G = (V, E)$  with subset of challenge nodes  $C \subseteq V$  and is parametrized with a hash function  $h$  (modeled as a random oracle). Each graph node, with index  $i$ , is associated with a label  $l_i$  defined recursively as a hash of index  $i$  and labels of parents of  $i$ , namely  $l_i = h(i, l_{p_1}, \dots, l_{p_d})$ , where  $p_1 < \dots < p_d$  are indices of all the parents of  $i$ . The game proceeds in  $n$  rounds, where  $n$  is a parameter of the game. At each round  $r$  a challenge  $c_r$  is drawn uniformly at random from  $C$ . The player's goal is to compute the label associated with the challenge node  $c_r$ , before moving to the next round and learning the next challenge  $c_{r+1}$ . As before, we assume the pROM model, i.e. the player can make multiple parallel random oracle calls at each step of his computation. The game ends when the last challenge is answered.

We define a CMC complexity of the `computeLabel` game as the expected value of CMC of the best adversary playing the game. The second result of Alwen et al., described in Sect. 1.1, applies to the CMC complexity of the `computeLabel` game played on a simple path graph, which underlies the `script` function.

**Pebbling Games and “entangled” Pebbling Games.** A standard pebbling game, similarly to the `computeLabel`, is played on a single source and a single sink DAG  $G = (V, E)$ . At each step the player can put or remove a pebble from a node of  $G$  according to the following two rules: a new pebble can be placed on any node  $v$  for which all parents of  $v$  have pebbles on them (in particular, a pebble can always be placed on the source), and pebbles can always be removed. The game ends when a pebble is placed on the sink of  $G$ . In the parallel pebbling game the player at each step places at the same time as many pebbles as he wants (as long as he follows the rules) and then he removes any number of pebbles of his choice.

The *cumulative complexity* (CC) of the strategy for the (parallel) pebbling game is defined as  $\sum_i |S_i|$  where  $S_i$  is a set of pebbled nodes at the end of  $i$ -th step. The CC of a graph  $G$  is defined as CC of the best pebbling strategy for  $G$ .

For a graph  $G = (V, E)$  one could consider a pebbling analogue of the `computeLabel` game, called `pebble` in [1]. At each round a challenge  $c_r$  is sampled uniformly at random from  $C \subseteq V$ , and the goal of the player is to pebble the challenge node (following the same rules as in the parallel pebbling game), before advancing to the next round and learning the challenge  $c_{r+1}$ .

It is easy to see that any pebbling strategy in the `pebble` game can be adapted as a strategy in the `computeLabel` game for a *restricted* adversary who stores in memory only the labels, i.e. random oracle outputs. One could consider a slightly strengthened model in which the adversary can store specific functions of the labels (but not yet *arbitrary* ones). For example, consider an adversary playing the `computeLabel` game who stores in memory the XOR of labels  $x := l_i \oplus l_j$ . Later, e.g. in the next round, he could compute  $l_i$  and use it together with  $x$  to recover  $l_j$  (or the other way around). This way he could potentially improve the complexity in terms of CMC.

A pebbling abstraction of such an adversary is an adversary playing the *entangled pebbling game*, a new class of randomized pebbling game introduced in [1]. In this game, for a set  $\mathcal{V} \subseteq V$  and some integer  $0 \leq t < |\mathcal{V}|$  a player who has individual pebbles on all the nodes in  $\mathcal{V}$  is allowed to place an *entangled pebble*  $\langle \mathcal{V} \rangle_t$  on  $\mathcal{V}$  that weights  $|\mathcal{V}| - t$ . The meaning of such an entangled pebble is that when the player has both  $\langle \mathcal{V} \rangle_t$  and individual pebbles on any  $t$  nodes from  $\mathcal{V}$  then he can at once put individual pebbles on all the nodes in  $\mathcal{V}$ . The pebbles used to “disentangle”  $\mathcal{V}$  might be a result of disentangling another entangled pebble. Note that the entangled pebble is a generalization of a normal pebble where  $\langle v \rangle_0$  corresponds to the individual pebble on vertex  $v$ . The initial pebbling in this game consists of a number of entangled pebbles.

Alwen et al. show a clever trick using polynomial interpolation which allows to translate an entangled pebble  $\langle \mathcal{V} \rangle_t$  to an encoding of length  $w \cdot (|\mathcal{V}| - t)$  such that given any  $t$  labels of nodes from  $\mathcal{V}$  it is possible to recover the remaining ones (here  $w$  is the length of a single label). An adversary allowed to use only such encodings in the `computeLabel` game is called an *entangled adversary*.

It is not obvious if relaxing the restriction by allowing the adversary to use entangled pebbles improves his cumulative complexity of the `pebble` game.

However, Alwen et al. show an example of a graph for which entangled pebbling is strictly more powerful (see [1] Appendix A).

**Alwen et al. Conjectures and Their Implications.** The authors of [1] define a sequence  $\gamma_n$  (we define it in Definition 7, Sect. 2) and prove that:

1. For any DAG  $G = (V, E)$  with  $|V| = n$ , with high probability over the choice of the random hash function  $h: \{0, 1\}^* \rightarrow \{0, 1\}^w$ , the pROM *time* complexity to play `computeLabel` on  $G$ , for any number of challenges, starting with any initial 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  $\gamma_n \cdot k$ .
2. The pROM CMC of `computeLabel` for  $L_n$  (a simple graph underlying `script`, that consists of a single path from a source to a sink) is  $\Omega\left(\frac{n^2}{\gamma_n \cdot \log^2(n)}\right)$ .

Alwen et al. conjecture that the sequence  $\gamma_n$  is upper bounded by a constant (see Conjecture 13 in [1] or Conjecture 1, Sect. 2). They use this conjecture to boost their results proved for a restricted class of adversaries, so called entangled adversaries (see Sect. 1.1), to hold for *arbitrary* adversaries in pROM.

Under this conjecture, the first result would solve the main open problem from the work of Dziembowski et al. [5] on proofs of space. The second one would imply a near-quadratic lower bound on CMC of evaluating `script` for arbitrary pROM adversaries.

The authors of [1] also prove the same results using a different sequence  $\Gamma_n$  instead of  $\gamma_n$ . It is easy to show that for each  $n$  it holds  $\Gamma_n \leq \gamma_n$ . Therefore the results would hold assuming only a weaker conjecture — that the sequence  $\Gamma_n$  is upper bounded by a constant (see Conjecture 16 in [1] or Conjecture 2, Sect. 2). However, the authors of [1] concentrate on the stronger conjecture, because the sequence  $\gamma_n$  is more convenient to work with.

## 1.2 Our Results

As stated before, in this paper we disprove the Conjectures 13 and 16 from [1] (Conjectures 1 and 2 Sect. 2). To do it, we first show how to construct a *transcript* (see Sect. 2) from a graph and we prove that the properties of such graph-derived transcripts are connected to the clique number and the (fractional) chromatic number of that graph.

We disprove Conjecture 1 first using the Mycielski construction [7] (from that we get  $\gamma_n \geq \sqrt{\log(n)/2}$ ) and then we strengthen our result using random graphs (we get  $\gamma_n \geq \sqrt{n}/\text{poly}(\log(n))$ ). We disprove Conjecture 2 using Kneser graphs [6] (we get  $\Gamma_n \geq c\sqrt{\log(n)}$ ).

## 1.3 Related Work

In recent work [2] Alwen et al. proved that CMC complexity in pROM model of `script` is  $\Omega(n^2w)$ , where  $w$  and  $n$  are the output length and number of invocations

of the underlying hash function, respectively. That bound clearly improves the result  $\Omega\left(\frac{n^2}{\gamma_n \cdot \log^2(n)}\right)$  from [1] and is tight because CMC complexity of `script` computed as prescribed is  $O(n^2w)$ .

However, the techniques used in [2] do not use the notion of entangled pebbling games and are strictly tailored for `script` function. Thus the approach from [1] might be of more general use.

## 2 Preliminaries

In this section we recall the conjectures from [1] and definitions that are necessary to state them. We also give the intuitions behind the notions introduced by Alwen et al. They are, however, not necessary to understand our results from Sect. 3.

**Definition 1.** For  $n \in \mathbb{N}$  an  $n$ -transcript  $T$  is a set of implications of the form  $\tau_j = (i_1, i_2, \dots, i_k \rightarrow i_0)$  for  $k, i_0, i_1, \dots, i_k \in [n] = \{1, 2, \dots, n\}$ .

The idea behind the notion of a transcript is as follows. Consider an adversary  $\mathcal{A}$  playing the `computeLabel` game on a graph  $G$  having some fixed initial state  $\sigma_{\text{init}}$ . Here  $\mathcal{A}$  is unrestricted, which in particular means that  $\sigma_{\text{init}}$  can contain any information, not only labels of the vertices. We fix the random oracle  $h$  as well. We include the implication  $\tau_j = (i_1, i_2, \dots, i_k \rightarrow i_0)$  into the transcript describing  $\mathcal{A}^h(\sigma_{\text{init}})$  if for some sequence of challenges  $c_1, \dots, c_m$  at some round in the game:

- the labels  $l_{i_1}, \dots, l_{i_k}$  are all the labels that appeared as inputs or outputs of the oracle so far,
- the label  $l_{i_0}$  did not appear as an input or an output of the oracle before, and
- $\mathcal{A}$  makes a query to the random oracle using  $l_{i_0}$  as one of the inputs.

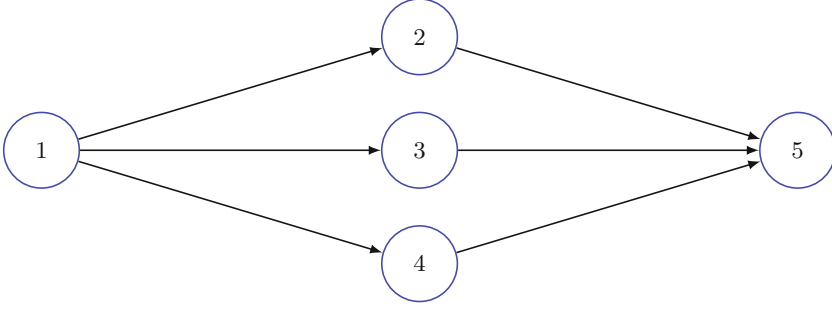
Intuitively, this means that we are able to “extract” the label  $l_{i_0}$  (without querying the oracle  $h$  for it) from  $\sigma_{\text{init}}$  and the labels  $l_{i_1}, \dots, l_{i_k}$ , by invoking  $\mathcal{A}^h(\sigma_{\text{init}})$ .

For example, consider a DAG  $G$  from Fig. 1. Suppose  $\mathcal{A}$  so far queried only for the label of the vertex 1 i.e.  $l_1 = h(1)$  and for the label of the vertex 3 i.e.  $l_3 = h(3, l_1)$ . At this round he makes a query for a label of the vertex 5 i.e.  $l_5 = h(5, l_2, l_3, l_4)$ .  $\mathcal{A}$  had to extract  $l_2$  and  $l_4$  from  $\sigma_{\text{init}}$  and  $l_1, l_3$ , so in this case we would include into the transcript the implications  $(1, 3 \rightarrow 2)$  and  $(1, 3 \rightarrow 4)$ .

**Definition 2.** A set  $U \subseteq [n]$  satisfies an  $n$ -transcript  $T$ , if for some  $s \leq n$  there exists a sequence  $U_0, \dots, U_s$ , s.t.:

- $U_0 = U$ ,
- For each  $j = 1, 2, \dots, s$  there exists  $\tau_j = (i_1, i_2, \dots, i_k \rightarrow i_0) \in T$  s.t.  $U_j = U_{j-1} \cup \{i_0\}$  and  $i_1, i_2, \dots, i_k \in U_{j-1}$ ,
- $U_s = [n]$ .

**Definition 3.** For an  $n$ -transcript  $T$  we define  $ex(T) = n - \min_U |U|$  where the minimum is taken over all sets  $U \subseteq [n]$  that satisfy  $T$ .



**Fig. 1.** An example graph used to illustrate definitions from Sect. 2.

Let the transcript  $T$  describe an adversary  $\mathcal{A}$  playing the `computeLabel` game on a graph  $G$  with an initial state  $\sigma_{\text{init}}$  (as explained before). Intuitively,  $ex(T)$  is the maximal number of labels that can be extracted from  $\sigma_{\text{init}}$  and some label set  $\{l_i | i \in U\}$  (where  $U$  corresponds to the set from Definition 3) by invoking  $\mathcal{A}^h(\sigma_{\text{init}})$  on several different challenge sequences, in an optimal way.

**Definition 4.** *The entangled set  $S = \langle q_1, q_2, \dots, q_t \rangle_m$  for  $0 \leq m \leq t - 1$  is an object that given  $m$  different numbers from  $\{q_1, q_2, \dots, q_t\}$  returns all the numbers  $\{q_1, q_2, \dots, q_t\}$ . The weight of  $S$  is defined as  $w(S) = t - m$ . The weight of the family  $F = \{S_1, \dots, S_r\}$  of entangled sets is the sum of weights of the entangled sets  $w(F) = w(S_1) + \dots + w(S_r)$ . We will write  $S^*$  to denote the (real) set  $\{q_1, q_2, \dots, q_t\}$ .*

**Definition 5.** *We say that the  $n$ -transcript  $T$  is covered by the family of entangled sets  $F = \{S_1, \dots, S_r\}$  if for every implication  $\tau = (i_1, i_2, \dots, i_k \rightarrow i_0) \in T$  there exists a sequence of sets  $V_0, \dots, V_s$  s.t.*

- $V_0 = \{i_1, i_2, \dots, i_k\}$ ,
- For each  $j = 1, 2, \dots, s$  there exists an entangled set  $S_j = \langle q_1, q_2, \dots, q_t \rangle_m \in F$  s.t.  $V_j = V_{j-1} \cup \{q_1, q_2, \dots, q_t\}$  and  $|\{q_1, q_2, \dots, q_t\} \cap V_{j-1}| \geq m$ ,
- $i_0 \in V_s$ .

**Definition 6.** *The weight  $w(T)$  of the  $n$ -transcript  $T$  is the smallest weight  $w(F)$  of a family  $F$  of entangled sets that covers  $T$ .*

The intuition behind the notion of the transcript weight  $w(T)$  is as follows. Let  $T$  describe an unrestricted adversary  $\mathcal{A}$  playing the `computeLabel` game with an initial state  $\sigma_{\text{init}}$  on a graph  $G$ . Then there exists an entangled pebbling adversary (see Sect. 1.1)  $\mathcal{A}'$  playing the `pebble` game on the same graph  $G$  with an initial pebbling state  $\sigma'_{\text{init}}$  of weight  $w(T)$  who is able to mimic the adversary  $\mathcal{A}$  in the following sense: whenever  $\mathcal{A}$  makes a query to compute some label  $l_i = h(i, l_{p_1}, \dots, l_{p_t})$ ,  $\mathcal{A}'$  puts a (normal) pebble on the vertex  $i$ . Note that the

initial state of  $\mathcal{A}'$  may contain entangled pebbles which cannot be translated to standard pebbles.

For example, consider a DAG  $G$  from Fig. 1. Suppose  $\sigma_{\text{init}}$  consists of  $l_2$ , a xor of the first half of  $l_1$  with the first half of  $l_4$  and a xor of the second half of  $l_3$  with the second part of  $l_4$ . Then  $\mathcal{A}$  can extract  $l_2$  just from  $\sigma_{\text{init}}$  and  $l_4$  from  $\sigma_{\text{init}}, l_1, l_3$ . So  $T = \{(\rightarrow 2), (1, 3 \rightarrow 4)\}$ . In this case the initial pebbling state  $\sigma'_{\text{init}}$  could be equal to  $\{\langle 2 \rangle_0, \langle 1, 3, 4 \rangle_2\}$ . Then  $\mathcal{A}'$  can use  $\langle 2 \rangle_0$  when  $\mathcal{A}$  extracts  $l_2$  from  $\sigma_{\text{init}}$  and use  $\langle 1, 3, 4 \rangle_2$  when  $\mathcal{A}$  extracts  $l_4$  from  $\sigma_{\text{init}}, l_1, l_3$ .

**Definition 7.** We define a sequence  $\gamma_n$  as:

$$\gamma_n = \max_{n\text{-transcript } T} \frac{w(T)}{ex(T)}$$

**Conjecture 1 (Conjecture 13 from [1]).** *There exists a constant  $C$  s.t. for all natural  $n$  we have  $\gamma_n < C$ .*

**Definition 8.** Let  $l_i$  for  $i = 1, 2, \dots, n$  be independent random labels chosen uniformly from  $\{0, 1\}^w$ . We say that the state  $\sigma \in \{0, 1\}^*$ , that might depend on those labels, satisfies the transcript  $T$  if for every implication  $(i_1, i_2, \dots, i_k \rightarrow i_0) \in T$  the label  $l_{i_0}$  is a function of a tuple  $(l_{i_1}, \dots, l_{i_k}, \sigma)$ . Let  $\sigma_w$  denote the shortest state that satisfies  $T$ . Then  $shannon(T) = \inf_w \frac{|\sigma_w|}{w}$ .

The value  $shannon(T)$  is the length of the shortest state  $\sigma$ , divided by a label length  $w$ , that for any implication  $(i_1, i_2, \dots, i_k \rightarrow i_0) \in T$  allows to extract  $l_{i_0}$  given labels  $l_{i_1}, \dots, l_{i_k}$ .

For example, if  $T = \{(1 \rightarrow 2), (2 \rightarrow 1)\}$  then a state  $\sigma = l_1 \oplus l_2$  allows to recover  $l_2$  given  $l_1$ , and to recover  $l_1$  given  $l_2$ . So in this case  $shannon(T) \leq 1$ .

**Definition 9.** We define a sequence  $\Gamma_n$  as:

$$\Gamma_n = \max_{n\text{-transcript } T} \frac{w(T)}{shannon(T)}$$

**Conjecture 2 (Conjecture 16 from [1]).** *There exists a constant  $C$  s.t. for all natural  $n$  we have  $\Gamma_n < C$ .*

It is easy to prove that for each transcript  $T$  we have  $ex(T) \leq shannon(T) \leq w(T)$ , so always  $\gamma_n \geq \Gamma_n$ . To see the first inequality let  $U = \{i_1, \dots, i_k\} \subseteq [n]$  be the smallest set that satisfies  $T$  and  $\sigma_w$  be a state that satisfies  $T$ . By Definition 2 we can expand the set  $U$  to the whole set  $[n]$  using implications from  $T$  and by Definition 8 for each implication  $\tau = (j_1, j_2, \dots, j_m \rightarrow j_0) \in T$  we can extract the label  $l_{j_0}$  from  $\sigma_w$  and the labels  $l_{j_1}, l_{j_2}, \dots, l_{j_m}$ . Therefore  $(l_1, \dots, l_n)$  is a function of  $(\sigma_w, l_{i_1}, \dots, l_{i_k})$  and

$$\begin{aligned} |\sigma_w| &\geq H(\sigma_w) \geq H(\sigma_w | l_{i_1}, \dots, l_{i_k}) = H(\sigma_w, l_{i_1}, \dots, l_{i_k}) - H(l_{i_1}, \dots, l_{i_k}) \geq \\ &\geq H(l_1, \dots, l_n) - H(l_{i_1}, \dots, l_{i_k}) = (n - k) \cdot w = ex(T) \cdot w. \end{aligned}$$



This ends the proof of the first inequality. The second inequality follows from the fact that the family of entangled sets  $F$  covering  $T$  can be thought of as a special case of a state  $\sigma$ , because on the trick used in [1] to translate an entangled pebble to the encoding of length proportional to the pebble weight (see Sect. 1.1).

Therefore the Conjecture 2 is weaker than the Conjecture 1. As stated before, both Conjecture 1 and Conjecture 2 are sufficient for the main result of [1].

### 3 Our Results

We disprove Conjecture 1 in Sect. 3.1 and Conjecture 2 in Sect. 3.2.

#### 3.1 Disproving Conjecture 1

Let  $G$  denote an undirected simple<sup>3</sup> graph with vertex set equal to  $[n]$ . We call such a graph an  $n$ -graph.

**Definition 10.** Let  $G$  be an  $n$ -graph. By  $T(G)$  we denote an  $n$ -transcript  $T(G) = \{\tau_1, \dots, \tau_n\}$  where  $\tau_i = (i_1, i_2, \dots, i_k \rightarrow i)$  and  $i_1, i_2, \dots, i_k$  are all the vertices in  $[n] \setminus \{i\}$  that are not adjacent to the vertex  $i$ .

**Lemma 1.** Let  $G$  be an  $n$ -graph. Then  $ex(T(G)) = \omega(G)$  where  $\omega(G)$  is the clique number of  $G$  i.e. the size of the largest clique in  $G$ .

*Proof.* First we show that  $ex(T(G)) \geq \omega(G)$ . Let  $V$  be the largest clique in  $G$  and  $U = [n] \setminus V$ . Then  $|U| = n - \omega(G)$  and  $U$  satisfies  $T(G)$ . That is because we can add elements of  $V$  to  $U$ , in any order, using implications from  $T(G)$ . Formally, let  $i_0 \in [n] \setminus U = V$ . Then  $\tau_{i_0} = (i_1, i_2, \dots, i_k \rightarrow i_0) \in T(G)$  where  $i_1, i_2, \dots, i_k$  are the vertices not adjacent to  $i_0$ . But  $V$  is a clique, so all the vertices  $i_1, i_2, \dots, i_k$  are contained in  $U$ , so we can use  $\tau_{i_0}$  and by that add  $i_0$  to  $U$ . We can do the same for all  $i \in [n] \setminus U = V$  and at the end we get the whole set  $[n]$ . Therefore the set  $U$  satisfies  $T(G)$ , so  $ex(T(G)) \geq n - |U| = \omega(G)$ .

Now we show that  $ex(T(G)) \leq \omega(G)$ . Let  $U$  be the smallest set that satisfies  $T(G)$ . Assume by contradiction, that  $|U| < n - \omega(G)$ . Then  $V = [n] \setminus U$  is not a clique (as  $|V| > \omega(G)$ ) — there exist  $i_0 \neq j_0 \in V$  that are not adjacent. As  $U$  satisfies  $T(G)$ , we have to add both  $i_0$  and  $j_0$  to  $U$ . But the only implication in  $T(G)$  with  $i_0$  on the right side has  $j_0$  on the left side, and the only implication in  $T(G)$  with  $j_0$  on the right side has  $i_0$  on the left side. In other words  $i_0$  depends on  $j_0$  and  $j_0$  depends on  $i_0$ . Therefore we cannot add either of  $i_0, j_0$  to  $U$  because the other element would have to be added first. Consequently  $U$  does not satisfy  $T(G)$ . We have a contradiction —  $U$  cannot be smaller than  $n - \omega(G)$ .

**Lemma 2.** Let  $G$  be an  $n$ -graph. Then  $\sqrt{\chi(G)} \leq w(T(G)) \leq \chi(G)$  where  $\chi(G)$  is the chromatic number of  $G$  i.e. the smallest number of colors that has to be used to properly color the vertices of  $G$ .

<sup>3</sup> A simple graph is a graph containing no graph loops or multiple edges.

*Proof.* First we prove the second inequality. Let  $\mathcal{C}: [n] \rightarrow [\chi(G)]$  be the proper coloring of  $G$ . Let  $\mathcal{C}^{-1}(i) = \{q_1^i, q_2^i, \dots, q_{p_i}^i\}$ ,  $S_i = \langle q_1^i, q_2^i, \dots, q_{p_i}^i \rangle_{p_i-1}$  and  $F = \{S_1, S_2, \dots, S_{\chi(G)}\}$ . Then  $w(S_i) = 1$ ,  $w(F) = \chi(G)$  and  $T(G)$  is covered by  $F$ . The proof of this claim is easy. Let  $\tau_{i_0} = (i_1, i_2, \dots, i_k \rightarrow i_0) \in T(G)$ . Then the vertices  $i_1, i_2, \dots, i_k$  are all the vertices in  $G$  that are not adjacent to  $i_0$ . Let  $j = \mathcal{C}(i_0)$  be the color of the vertex  $i_0$ . All the other vertices with the color  $j$  are not adjacent to  $i_0$ , which means that  $S_j^* \cap \{i_1, i_2, \dots, i_k\} = S_j^* \setminus \{i_0\}$ . So we can use the entangled set  $S_j$  to get a vertex  $i_0$ .

Now we prove the first inequality. Assume that  $T(G)$  is covered by the family of entangled sets  $F = \{S_1, S_2, \dots, S_r\}$ . It is enough to show a proper coloring of vertices of  $G$  using  $w(F)^2$  colors. First we show a coloring using  $r$  colors in which every vertex has less than  $w(F)$  same color neighbors. Then, using a greedy algorithm, we can change it to a proper coloring using  $r \cdot w(F) \leq w(F)^2$  colors.

Let  $i_0 \in [n]$  be any vertex in  $G$  and  $\tau_{i_0} = (i_1, i_2, \dots, i_k \rightarrow i_0) \in T(G)$ . Let  $V_0, V_1, \dots, V_s$  be any shortest sequence of sets as in Definition 5. Let  $S_{j_i} = \langle q_1^{j_i}, q_2^{j_i}, \dots, q_{t_{j_i}}^{j_i} \rangle_{m_{j_i}}$  be the entangled set opened at the step number  $i = 1, 2, \dots, s$ , i.e.  $V_i = V_{i-1} \cup S_{j_i}^*$  and  $|V_{i-1} \cap S_{j_i}^*| \geq m_{j_i}$ . We assign the color number  $j_s$  to the vertex  $i_0$ . Obviously  $j_s \in [r]$  and the coloring is unambiguous as there is exactly one implication in  $T(G)$  with the element  $i_0$  on the right side. Additionally, as the sequence  $V_0, V_1, \dots, V_s$  is the shortest, we know that  $i_0 \in S_{j_s}^*$  and all the indices  $j_i$  are different.

Now we show that for any vertex  $i_0 \in [n]$  there are less than  $w(F)$  other vertices that are adjacent to  $i_0$  and have the same color  $j_s$ . Let  $N(i_0) \subseteq [n] \setminus \{i_0\}$  denotes the set of neighbors of the vertex  $i_0$ . We know that all the vertices with the color  $j_s$  are contained in the set  $S_{j_s}^*$ . So it is enough to show, that  $|S_{j_s}^* \cap N(i_0)| < w(F)$ .

We know that  $V_0$  is exactly the set  $[n] \setminus N(i_0) \setminus \{i_0\}$ . So  $S_{j_s}^* \cap N(i_0) \subseteq V_s \setminus V_0 \setminus \{i_0\}$ . On the other hand we have  $|V_i \setminus V_{i-1}| \leq w(S_{j_i}) = t_{j_i} - m_{j_i}$  as we add at most  $t_{j_i}$  elements but only if there were already  $m_{j_i}$  elements present. We now have:

$$|S_{j_s}^* \cap N(i_0)| < |V_s \setminus V_0| = \sum_{i=1}^s |V_i \setminus V_{i-1}| \leq \sum_{i=1}^s w(S_{j_i}) \leq \sum_{i=1}^r w(S_i) = w(F).$$

Now we change the coloring into a proper coloring using colors from the set  $[r] \times [w(F)]$ . We use a greedy algorithm. For each vertex  $i_0$ , with color  $j_s$ , we assign it the color  $(j_s, k)$  where  $k$  is any number from  $[w(F)]$  s.t. the color  $(j_s, k)$  was not assigned earlier to any neighbor of  $i_0$ . We can always find such  $k$  because there are less than  $w(F)$  neighbors of  $i_0$  which previously had the color  $j_s$ .

We have constructed a proper coloring of vertices of  $G$  using at most  $w(F)^2$  colors, so  $w(F)^2 \geq \chi(G)$ .

To prove that  $\gamma_n$  is unbounded we use graphs that have big chromatic number but small clique number. We first give an example of explicit graphs using Mycielski construction [7] that satisfy these conditions. In Sect. 3.1 we use random graphs to get a stronger result.

The Mycielski construction generates graph  $\mu(G)$  from a given graph  $G$ . The construction has the following properties:

- $|V(\mu(G))| = 2 \cdot |V(G)| + 1$ ,
- $\chi(\mu(G)) = \chi(G) + 1$ ,
- $\omega(\mu(G)) = \max(2, \omega(G))$ .

The proof of these properties can be found in [7].

**Corollary 1.** *Using the Mycielski construction [7], starting from  $M_2$  equal to a single edge, we can create a graph  $M_k$  that has  $n = 3 \cdot 2^{k-2} - 1 < 2^k$  vertices,  $\omega(M_k) = 2$  and  $\chi(M_k) = k$ . That means that  $ex(T(M_k)) = 2$  and  $w(T(M_k)) \geq \sqrt{\chi(M_k)} = \sqrt{k}$ . So  $\gamma_n \geq \frac{w(T(M_k))}{ex(T(M_k))} = \frac{\sqrt{k}}{2} > \frac{\sqrt{\log(n)}}{2}$  is unbounded therefore the Conjecture 1 is false.*

**Stronger Result.** In this section we show that  $\gamma_n \geq \frac{\sqrt{n}}{\text{poly}(\log(n))}$ .

Let  $G(n, p)$  denote a random  $n$ -graph s.t. each edge is present with probability  $p$ .

**Lemma 3.** *We have:*

- $\mathbb{P}(\omega(G(n, 1/2)) \geq M) \leq \binom{n}{M} \cdot 2^{-\binom{M}{2}}$ ,
- $\lim_{n \rightarrow \infty} \mathbb{P}(\omega(G(n, 1/2)) \geq \log(n)^2) = 0$ .

*Proof.* The first part of the lemma follows from the union bound — there are  $\binom{n}{M}$  candidate sets to be a clique of size  $M$ , each of them is a clique with probability  $2^{-\binom{M}{2}}$ .

The second part of the lemma follows from the first part:

$$\begin{aligned} \mathbb{P}(\omega(G(n, 1/2)) \geq \log(n)^2) &\leq \binom{n}{\log(n)^2} \cdot 2^{-\binom{\log(n)^2}{2}} \leq n^{\log(n)^2} \cdot 2^{-\log(n)^4/4} = \\ &= 2^{\log(n)^3 - \log(n)^4/4}, \end{aligned}$$

So  $\lim_{n \rightarrow \infty} \mathbb{P}(\omega(G(n, 1/2)) \geq \log(n)^2) = 0$ .

**Lemma 4.** *There exists  $d > 0$  s.t.  $\lim_{n \rightarrow \infty} \mathbb{P}(\chi(G(n, 1/2)) \leq d \frac{n}{\log n}) = 0$ .*

Proof of Lemma 4 can be found in [4].

**Theorem 1.** *There exists  $c > 0$  s.t.  $\gamma_n \geq c \frac{\sqrt{n}}{\log(n)^{5/2}}$ .*

*Proof.* From the previous lemmas we know that there exists  $d > 0$  s.t. with probability  $1 - o(1)$  a random graph  $G \leftarrow G(n, 1/2)$  has the following properties:

- $\omega(G) < \log(n)^2$ ,
- $\chi(G) > d \frac{n}{\log n}$ .

$$\text{So } \gamma_n \geq \frac{w(T(G))}{ex(T(G))} \geq \frac{\sqrt{\chi(G)}}{\omega(G)} > \sqrt{d} \frac{\sqrt{n}}{\log(n)^{5/2}} = c \frac{\sqrt{n}}{\log(n)^{5/2}}.$$

### 3.2 Disproving Conjecture 2

**Definition 11.** Let  $G$  be an  $n$ -graph. The  $b$ -fold coloring of  $G$  is an assignment of sets of size  $b$  (i.e.  $b$  colors) to each vertex of  $G$  s.t. adjacent vertices receive disjoint sets. The  $a : b$  coloring is a  $b$ -fold coloring using  $a$  colors.  $\chi_b(G)$  is the smallest number  $a$  of colors s.t.  $a : b$  coloring exists. The fractional chromatic number of  $G$  is  $\chi_F(G) = \inf_b \frac{\chi_b(G)}{b}$ .

Remark: For each  $n$ -graph  $G$  there exists an  $a : b$  coloring s.t.  $\chi_F(G) = a/b$  (see e.g. [6]).

**Lemma 5.** Let  $G$  be an  $n$ -graph. Then  $\chi_F(G) \geq \text{shannon}(T(G))$ .

*Proof.* Let  $\mathcal{C} : [n] \rightarrow 2^{[a]}$  be a fixed  $b$ -fold coloring of  $G$  s.t.  $\chi_F(G) = a/b$ . Let  $l_i \in \{0, 1\}^b$  for  $i = 1, 2, \dots, n$  be random labels of the vertices of  $G$  and let  $l_i[r]$  denote the  $r$ -th bit of  $l_i$ . We will construct a state  $\sigma_b$  of length  $a$  that satisfies  $T(G)$ .

Let  $\mathcal{C}(i) = \{j_1^i, j_2^i, \dots, j_b^i\}$  where  $j_1^i < j_2^i < \dots < j_b^i$ . We say that the bit  $l_i[r]$  has the color  $j_r^i$ . Let  $\sigma_b[c]$  be the xor of all the bits  $l_i[r]$  (where  $i \in [n], r \in [b]$ ) which have the color  $c$ , for  $c = 1, 2, \dots, a$ .

It should be easy to see that  $\sigma_b$  satisfies  $T(G)$ . That is because for  $\tau = (i_1, i_2, \dots, i_k \rightarrow i) \in T(G)$  and  $r \in [b]$  we can calculate  $l_i[r]$ . Let  $c$  be the color of the bit  $l_i[r]$ . Then  $l_i[r] = \sigma_b[c] \oplus l_{j_1}[r_1] \oplus \dots \oplus l_{j_s}[r_s]$  where  $l_i[r], l_{j_1}[r_1], l_{j_2}[r_2], \dots, l_{j_s}[r_s]$  are all the bits of color  $c$ . Vertices  $i, j_1, j_2, \dots, j_s$  have common color, therefore  $i$  is not adjacent to any of  $j_1, j_2, \dots, j_s$ . Thus by the definition of  $T(G)$  we know that all the numbers  $j_1, j_2, \dots, j_s$  are present on the left side of the implication  $\tau$ . Now we can read the bits  $l_{j_1}[r_1], l_{j_2}[r_2], \dots, l_{j_s}[r_s]$  from the labels  $l_{j_1}, l_{j_2}, \dots, l_{j_s}$  and calculate  $l_i[r]$ . This can be done for any  $\tau \in T(G)$  and any  $r \in [b]$ .

To prove that  $\Gamma_n$  is unbounded, we use graphs that have big chromatic numbers, but small fractional chromatic numbers. An example of graphs with these properties are Kneser graphs [6]. The vertices of the Kneser graph  $K_{a:b}$  for  $a \geq b$  are all  $b$ -element subsets of the set  $[a]$ . Two vertices are adjacent if their corresponding subsets are disjoint. The Kneser graph  $K_{a:b}$  for  $a \geq 2b$  has the following properties:

- $|V(K_{a:b})| = \binom{a}{b}$ ,
- $\chi(K_{a:b}) = a - 2b + 2$ ,
- $\chi_F(K_{a:b}) = a/b$ .

The proofs of these properties can be found in [6].

**Corollary 2.** Let  $K_{a:b}$  be a Kneser graph [6] for  $a := 3b$  and  $n := \binom{a}{b}$ . We have:

$$\Gamma_n \geq \frac{w(T(K_{a:b}))}{\text{shannon}(T(K_{a:b}))} \geq \frac{\sqrt{\chi(K_{a:b})}}{\chi_F(K_{a:b})} = \frac{\sqrt{a - 2b + 2}}{a/b} = \frac{\sqrt{b + 2}}{3} = \Omega(b^{0.5})$$

so  $\Gamma_n$  is unbounded.

In this example  $n = \binom{3b}{b} < 2^{3b}$  therefore we have proved that  $\Gamma_n$  is greater than  $\Omega(\sqrt{\log(n)})$ .

## References

1. Alwen, J., Chen, B., Kamath, C., Kolmogorov, V., Pietrzak, K., Tessaro, S.: On the complexity of script and proofs of space in the parallel random oracle model. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 358–387. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-49896-5\\_13](https://doi.org/10.1007/978-3-662-49896-5_13)
2. Alwen, J., Chen, B., Pietrzak, K., Reyzin, L., Tessaro, S.: Script is maximally memory-hard. In: Coron, J.S., Nielsen, J. (eds.) Advances in Cryptology - EUROCRYPT 2017. EUROCRYPT 2017. LNCS, vol. 10212, pp. 33–62. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56617-7\\_2](https://doi.org/10.1007/978-3-319-56617-7_2)
3. Alwen, J., Serbinenko, V.: High parallel complexity graphs and memory-hard functions. In: STOC (2015)
4. Bollobás, B.: The chromatic number of random graphs (1988)
5. Dziembowski, S., Faust, S., Kolmogorov, V., Pietrzak, K.: Proofs of space. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 585–605. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48000-7\\_29](https://doi.org/10.1007/978-3-662-48000-7_29)
6. Ullman, D.H., Scheinerman, E.R.: Fractional graph theory: A rational approach to the theory of graphs (2013)
7. Mycielski, J.: Sur le coloriage des graphs. Colloquium Math. **3**(2), 161–162 (1955)
8. Park, S., Kwon, A., Alwen, J., Fuchsbauer, G., Gaži, P., Pietrzak, K.: SpaceMint: A Cryptocurrency Based on Proofs of Space. Cryptology ePrint Archive, Report 2015/528 (2015). <http://eprint.iacr.org/2015/528>
9. Percival, C.: Stronger key derivation via sequential memory-hard functions (2009)

Information Theoretic Security

10th International Conference, ICITS 2017, Hong Kong,  
China, November 29 – December 2, 2017, Proceedings

Shikata, J. (Ed.)

2017, XII, 235 p. 31 illus., Softcover

ISBN: 978-3-319-72088-3