Proofs of Space: When Space is of the Essence

Abstract. Proofs of computational effort were devised to control denial of service attacks. Dwork and Naor (CRYPTO '92), for example, proposed to use such proofs to discourage spam. The idea is to couple each email message with a proof of work that demonstrates the sender performed some computational task. A proof of work can be either CPU-bound or memory-bound. In a CPU-bound proof, the prover must compute a CPU-intensive function that is easy to check by the verifier. A memory-bound proof, instead, forces the prover to access the main memory several times, effectively replacing CPU cycles with memory accesses.

In this paper we put forward a new concept dubbed *proof of space*. To compute such a proof, the prover must use a specified amount of space, i.e., we are not interested in the number of accesses to the main memory (as in memory-bound proof of work) but rather on the amount of actual memory the prover must employ to compute the proof. We give a complete and detailed algorithmic description of our model. We develop a full theoretical analysis which uses combinatorial tools from Complexity Theory (like pebbling games) which are essential in studying space lower bounds.

We remark that a similar concept has recently been described by Dziembowski et al. (Workshop held in Warsaw, 2013), however their proof-of-space paradigm is more in line with memory-bound proof of work since the prover can trade off space with computation while our definition disallow this prospect.

Keywords: Space Complexity, Proof of Work, Pebbling Game, Random Oracle Model.

1 Introduction

Space has a special meaning in Computer Science. It refers to the number of cells of the working tape used by a Turing Machine (TM). While a TM computes a function, it will make several steps (relevant to time complexity) and use a certain number of tape cells (relevant to space complexity).

In [13], Dwork and Naor proposed to employ proof of work (PoW) to discourage spam and, in general, to hinder denial of service attacks. Before any action (such as sending an email), the prover must perform some work and generate a proof of it that can be efficiently verified. Proofs of work are currently being used to implement a publicly verifiable ledger for Bitcoin, where transactions are registered and verified by a community of users to avoid the double-spending problem [26]. The work performed by the prover can be CPU-bound, in which the work represents the number of steps made by a TM, or memory-bound, in which the work represents the times a TM access the working tape. The motivation behind memory-bound PoW is that, while CPU speed may differ significantly among distinct platforms, memory latencies vary much less across machines and may prove to be more equitable and egalitarian. We stress that memory-bound function complexity measures the number of memory accesses and does not take into account the actual amount of memory employed. That is, a TM may read and mark a single cell several times to reach a certain complexity level but it will still end up using only one cell.

In this work we define the notion of *Proof of Space* (PoSpace). PoSpace forces the prover to use at least a specified amount of memory. This means, for instance, that a TM must now use a predetermined number of distinct tape cells to be able to respond to a challenge. Interestingly, we will show that our PoSpace construction is also a memory-bound PoW under the definition provided in [12, 14], while in general a PoW cannot be a PoSpace under our definition. The state of the art memory-bound PoW was described in [14] by Dwork, Goldberg, and Naor. Their scheme requires both the prover and the verifier to store a large table T but they devised ways to mitigate this problem via either hash trees or public-key signatures.

We remark that Dziembowski et al. [17] have recently and independently provided a notion of proof of space which is very similar to ours. However, their constructions are in effect a variant of memory-bound PoWs as originally defined in [1, 12] (more on this later).

We view PoSpace as a valid alternative to various flavors of PoWs. In PoSpace the spotlight is turned on the amount of space rather than on CPU cycles or memory accesses as in PoWs. In addition, PoSpace solves problems where PoW is not applicable. For instance, we believe PoSpace can be employed in forensic analysis or device attestation to confirm remotely that an embedded device has been successfully wiped. That is, a remote device could be instructed to respond to a wipe command with PoSpace as evidence that its functional memory is now overwritten.

Straw Man Solutions. Memory-bound functions were first introduced by Abadi et al. [1]. In the main construction of memory-bound PoW given in [12], both the prover and the verifier share a large random table T. The prover must compute a function by making several memory accesses to uniformly random positions in T. With proper parameters, the prover is forced to reserve an amount of memory which is proportional to S. In another construction, the authors of [12] show that the verifier does not have to store T. The idea is to sign all pairs $(i, T[i])$ and then challenge the prover on ℓ positions of T. The prover will return the ℓ values $T[i]$ along with an aggregate signature that can be checked by the verifier to ensure the prover is holding the table T.

We first remark that it is possible to harness recent advances in proof of storage schemes, such as PDP [4] and POR [29], to reduce the message complexity from $O(\ell)$ to essentially constant. This solution improves upon the one in [12] and, as long as the initialization phase is performed only once, would meet our efficiency requirements for PoSpace. However, proof of storage does not satisfy our definition of PoSpace since the running time of the verifier depends on the size of T. The only way to avoid linear dependency is to run a PDP-based scheme with spot checking [4], but then the prover either must use more space than required or won't access all the memory locations. Intuitively, the reason why a solution with proof of storage will not work rests upon the interpretation of what *proof of space* really means. Proof of storage applied to our context satisfies the notion that *"the prover can* **access** *space"*. PoSpace instead captures the stronger notion that *"the prover can* **handle** *space"*, i.e., the prover possesses, controls, and manipulates space directly. In particular, we distinguish between a prover that can only read memory and a prover that can read and write memory. This is important because, among other things, write operations cannot be parallelized within classical computer architectures. We will provide a formal definition later and make this intuition rigorous.

We also remark that the adversarial model considered in [1,12] contemplates the existence of a small but fast cache memory that must be saturated to force the prover to dispense with the cache and use traditional RAM memory. Thus, the constructions in [1,12,14] do provide a form of proof of space where the space coincides with the cache memory. But, as for proof-of-storage schemes, these schemes satisfy the weaker notion of PoSpace where the prover can only read memory.

Other Related Work. Proof of work is also known as *cryptographic puzzle* in the computer security literature. Puzzles were devised to improve on the proposal by Back [6] and employed to thwart denial of service attacks. In particular, it is important to make them hard to precompute (see [21] and references therein). Waters et al. [31] suggest to outsource the creation of puzzles to an external secure entity. Abliz and Tznati [2] introduce the concept of network-bound puzzles where clients collect tokens from remote servers before querying the service provider. They argue that network latency provides a good solution to the resource disparity problem.

All solutions above deal with proof of effort and cannot be adapted to prove possession of space in the way it is meant and defined in this paper.

Litecoin (`litecoin.org`) is a variant of Bitcoin that employs *scrypt* [27] to certify a public ledger. *scrypt* is defined as a sequential memory-hard function and originally designed as a key derivation function, but it is used as a proof of effort in litecoin to hinder the use of specialized hardware. Technically, *scrypt* is not a memory-bound function as defined in [1] since no lower bound on the memory used by the function can be guaranteed. Thus, it is not even a POSPACE. We also note it requires both the prover and the verifier to dedicate a possibly large amount of memory, while ideally only the prover should reserve and use actual memory (as in our construction to be presented later). *scrypt* does provide a form of the non-amortizability property of memory-bound functions in that no parallel RAM (PRAM) machine would asymptotically achieve a significantly lower cost compared to sequential machines.

Dziembowski et al. [17] have recently and independently suggested a notion of proof of space. While their intuition appears to be the same as ours, their constructions do not constitute a real proof of space in our sense since the prover can always trade off space for computation, i.e., the prover is free to compute more in order to save space. Thus, it is not a POSPACE according to our definition where we disallow the prover to opt for computation to save space.

General ideas behind the protocol. We cast POSPACE in the context of delegation of computation. Here, a *delegator* outsources to a *worker* the computation of a function on a certain input. Securely delegating computation is a very active area [10,11,18,20] thanks also to the popularity of cloud computing where weak devices use the cloud to compute heavy functions.

In the case of POSPACE, a function f is first selected satisfying the property that there exists a space lower bound for any TM computing it. Then, the verifier chooses a random input x and delegates to the prover the computation of $f(x)$. Functions like f are difficult to devise in the plain model. Indeed, finding an explicit construction for f would imply a separation between the class **polyL** (of problems which can be solved in polylogaritmic space) and the class **P** (while it is only proven that **polyL** $\neq$ **P**). This motivates the need for relying on the Random Oracle Model (ROM), where space lower-bounds are easy to prove. Specifically, we will turn to the class of functions derived from the *"graph labeling problem"* already used in cryptography (see for example [14–16]).

Important tools in delegation of computation are interactive proof (and argument) systems. The delegation problem, as described in [25], can be solved as follows: The worker computes $y := f(x)$ and proves using an interactive proof system that indeed $y = f(x)$. While interactive proofs with statistical soundness and non-trivial savings in verification time are unlikely to exist [8,9,19], Kilian showed [22] that proof systems for **NP** languages with only computational soundness (i.e., argument systems [7]) and *succinctness* [1] do exist. The construction in [22] relies on Merkle trees and PCP proofs: For any L in **NP**, given the **NP**-witness for the statement $x \in L$, first it generates a PCP proof π; then, it commits the proof π using the Merkle tree, where the leaves correspond to the bits of π. The verifier checks the proof by following the PCP-verifier protocol which asks for a constant number of random locations of π and the prover answers by opening the corresponding locations in the Merkle tree. We call this the PCP+MT *paradigm.*

However, in the context of POSPACE, the PCP machinery is an overkill. In fact, the prover does not have to prove the statement $f(x) = y$, but only that a space-consuming computation was carried out. Therefore, we replace the PCP verification with an ad-hoc scheme that results in a very efficient overall construction (while PCP-based constructions are notoriously impractical).

---

[1] I.e., the total amount of communication and the verification time is less than that needed to transmit and check the NP-witness.

1.1 Our Contributions

We introduce a formal definition of Proof of Space (PoSpace Definition 2) that captures the intuitive idea of proving to be able to handle (read/write) at least a specified amount of space. We also provide two protocols in the ROM satisfying this definition. These protocols are built by exploiting and generalizing the pebbling framework of Dwork et al. [14]. We call such a generalization "graph labeling *with faults*". In addition, we provide a weaker form of PoSpace and prove that it is indeed a separate notion. Most of previous work on proof of storage [4,5,29] and on memory-bound PoW, as defined in [1,12,14], can somehow be adapted to meet this weaker definition.

We also devised a new variant of the pebbling game, i.e., *pebbling game with wildcards*, and prove an exponential tradeoff (Theorem 1) between pebbling space and time in this setting. This result is a generalization of a Theorem in [23]. While we do not claim this to be our main contribution, we find it interesting in its own right.

1.2 Structure of the paper

The paper structure is the following: in Section 2 we give some preliminary definitions and state some useful facts; in Section 3 we give the formal definition of PoSpace and provide two PoSpace protocols. In Section 4 we give the formal definition of a *weak* variant of PoSpace that captures read-only provers. We show a separation result between PoSpace and its weaker variant.

2 Notations and Preliminary Definitions

Negligible Functions. A function $\nu : \mathbb{N} \to [0,1]$ is *negligible* if and only if for each integer c there exists an integer n_c such that for all $x > n_c$, $\nu(x) < x^{-c}$. If a function f is not negligible, we say that f is *noticeable*. If ν is a negligible function then $1 - \nu$ is *overwhelming*. We denote with $\mathsf{poly}(k_1, \ldots, k_n)$ a generic polynomial function in the parameters $k_1, \ldots, k_n$.

Graph Notations. Given a directed graph $G = (V, E)$ we say that G is a *directed acyclic graph (DAG)* if and only if G has no cycles. We denote with $\Gamma^+(v)$ and $\Gamma^-(v)$ respectively the set of successors and predecessors of v in G. If $\Gamma^+(v) = \emptyset$ we say that v is an *output-node* and we denote as $\mathbf{T}(G)$ the set of all output-nodes of G. If $\Gamma^-(v) = \emptyset$, we say that v is an *input-node* and we denote with $\mathbf{S}(G)$ the set of all input-nodes of G. Given a DAG G, we will implicitly assume a topological ordering on its vertex set and a boolean encoding of the vertices respecting that ordering.

Sampling, Interactive Execution, and Space. Let $\mathcal{A}$ be a probabilistic TM. We write $y \leftarrow \mathcal{A}(x)$ to denote y sampled from the output of $\mathcal{A}$ on input x. Moreover, given $\sigma \in \{0,1\}^*$, we write $y := \mathcal{A}(x; \sigma)$ to denote the output of $\mathcal{A}$ on input x fixing the random coins to be σ. We write PPT for the class of probabilistic polynomial time algorithms. Let $\mathcal{A}$, $\mathcal{B}$ be two probabilistic interactive TMs, an interactive joint execution between $\mathcal{A}$, $\mathcal{B}$ on common input x is specified via the next message function notation: let $b_1 \leftarrow \mathcal{B}(x)$, $a_i \leftarrow \mathcal{A}(x, b_1, \ldots, b_i)$, and $b_{i+1} \leftarrow \mathcal{B}(x, a_1, \ldots, a_i)$, then $\langle \mathcal{A}, \mathcal{B} \rangle (x)$ denotes a joint execution of $\mathcal{A}$ and $\mathcal{B}$ on common input x. If the sequence of messages exchanged is $(b_1, a_1, \ldots, b_k, a_k, b_{k+1})$ we say that k is the number of *rounds* of that joint execution and we denote with $\langle \mathcal{A}, \mathcal{B} \rangle (x) = b_{k+1}$ the last output of $\mathcal{B}$ in that joint execution of $\mathcal{A}$ and $\mathcal{B}$ on common input x.

With $\mathrm{Space}(\mathcal{A}, x)$, we denote the maximal amount of space used by the TM $\mathcal{A}$ on input x without taking into account the length of the input and output. More formally, we say that $\mathcal{A}(x)$ has space s (or $\mathrm{Space}(\mathcal{A}, x) = s$) if and only if at most s locations on $\mathcal{A}$'s work tapes (excluding the input and the output tapes) are ever written by $\mathcal{A}$'s head during its computation on x. $\mathrm{Space}(\mathcal{A})$ denotes the maximum value of $\mathrm{Space}(\mathcal{A}, x)$ over all possible inputs x.

Similarly, given two probabilistic interactive TMs $\mathcal{A}$ and $\mathcal{B}$ we can define the space occupied by $\mathcal{A}$ during a joint execution on common input x as follows. Let $(b_1, a_1, \ldots, b_k, a_k, b_{k+1})$ the sequence of messages exchanged during $\langle \mathcal{A}, \mathcal{B} \rangle (x)$, then:

$$\mathrm{Space}_{\mathcal{A}}(\langle \mathcal{A}, \mathcal{B} \rangle, x) := \max_{i=1,\ldots,n} \left\{ \mathrm{Space}\left(\mathcal{A}, x, b_1, \ldots, b_i \right) \right\}$$

Finally, $\mathrm{Space}_{\mathcal{A}}(\langle \mathcal{A}, \mathcal{B} \rangle)$ denotes the maximum value over all possible inputs.

Merkle Trees. *Merkle Tree*s (MT) are classical tools in cryptography, they have been first used to build digital signature from one-time signature [24]. A MT enables a party to succinctly commit itself to a string $\boldsymbol{l} = (l_1, \ldots, l_n)$ with $l_i \in \{0, 1\}^k$. The term "succinctly" here means that the MT-commitment has size (almost) independent respect to the size of $\boldsymbol{l}$. In a later stage, when a party opens the MT-commitment, it is guaranteed that the "opening" can yield only the string $\boldsymbol{l}$ committed before (the binding property). Moreover $\boldsymbol{l}$ can be succinctly opened location-by-location: the party can open l_i for any i giving the certificate attesting the value of l_i. Here succinctly means that the "opening", i.e. the certificate, has size (almost) independent respect to the size of the string $\boldsymbol{l}$.

Define the complete binary tree T over n leaves. Without loss of generality we can suppose that n is a power of 2 by using padding if necessary. Label the i-th leaf with l_i, then, use a collision resistant hash (CRH) function $\mathsf{H}_s : \{0, 1\}^{2k} \to \{0, 1\}^k$ with random seed s to propagate the labeling $\boldsymbol{l}$ of the leaves to a labeling ϕ along the tree T up to the root according to the following rules:

1. $\phi(v) = l_i$ if v is the i-th leaf of T,
2. $\phi(v) = \mathsf{H}_s(\phi(v_1) \| \phi(v_2))$ where v_1, v_2 are the two childrens of v.

Then the *commitment* for a string $\boldsymbol{l}$ is the label $\phi(r)$ of the root of the tree: we denote it with $\mathsf{MT}_s(\boldsymbol{l})$. The commitment is succinct in fact it has size k which is independent of n.

An *opening for the i-th element in the sequence $\boldsymbol{l}$* is an ordered sequence formed by elements of $\{0, 1\}^k$ that are intended to be the label l_i of the i-th leaf of T together with the labels $\phi(v_j)$ for each v_j that is a sibling of vertices in the path from the root of the tree to the i-th leaf. An example of opening is shown in Figure 2.1.

We denote with $\mathrm{Open}_s(\boldsymbol{l}, i)$ the opening formed by l_i together with all the $\phi(v_j)$. Notice that from $\mathrm{Open}_s(\boldsymbol{l}, i)$ it is possible to compute $\phi(w)$ for each vertex w in the path from the root to the leaf at the i-th position . In particular it is efficient to compute $\mathsf{MT}_s(\boldsymbol{l})$. The opening is succinct in fact it has size $k \cdot \log n$ which is (almost) independent of n.
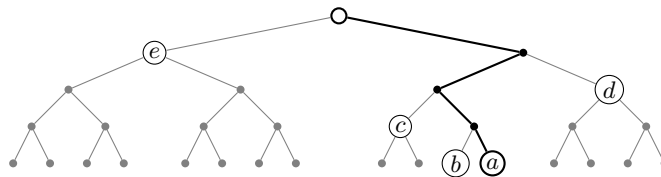

Fig. 2.1. An example of opening: $\boldsymbol{c} = (a, b, c, d, e)$

In general given an opening $\boldsymbol{c} = (c_1, \ldots, c_{\log n})$ for some position i we can behave *as if* it is $\mathsf{Open}_s(\boldsymbol{l}, i)$ and compute the label of the root according to the given $\boldsymbol{c}$ and position i. If this value is equal to $\mathsf{MT}_s(\boldsymbol{l})$ we say that $\boldsymbol{c}$ *opens the i-th position to c_1*.

Notice that, as H is a CRH function, then for any i, it is guaranteed that $\mathsf{MT}_s(\boldsymbol{l})$ can open the i-th position only to l_i.

Pebbling Games with wildcards. The following definition of *pebbling game with wildcards* is a modification of the standard definition of pebbling game that can be found for instance in [23]. Given a directed acyclic graph $G = (V, E)$ we say that a sequence $\mathcal{P} = (P_0, \ldots, P_T)$ of subsets of V is a *pebbling sequence on G with m wildcards* if and only if $P_0 = \emptyset$ and there exists a set $W \subseteq V$ of size m such that for each $i \in \{1, \ldots, T\}$ exactly one of the following holds:

- $P_i = P_{i-1} \cup \{v\}$ if $\Gamma^-(v) \subseteq P_{i-1} \cup W$ (*pebbling*) or
- $P_i \subseteq P_{i-1}$ (*unpebbling*).

If a set of vertexes Γ is such that $\Gamma \subseteq \bigcup_{i=0}^T P_i$, we say that $\mathcal{P}$ *pebbles* Γ. If $\mathcal{P}$ pebbles $\mathbf{T}(G)$ then we say that $\mathcal{P}$ is a *pebbling game on G with m wildcards*. Moreover we say that the *pebbling time* of $\mathcal{P}$ is T and the *pebbling space* of $\mathcal{P}$ is $\max_i |P_i|$.

Intuitively, a pebbled node is a node for which we have made some computations, instead W represents *complimentary* nodes, for which we have made no computations.

One of the main ingredients for the correctness of our constructions is the following generalization of the Pebbling Theorem [23]. Its proof is quite convoluted and moved to the Appendix A due to space constraints.

Theorem 1 (Pebbling Theorem). *There exist a family of directed acyclic graphs $\{G_{N,k}\}_{k,N \in \mathbb{N}}$ with constant fan-in d, N input nodes, N output nodes and $kN \log N$ nodes in total, such that:*

1. *Any pebbling sequence that pebbles Δ output nodes using at most S pebbles and m wildcards, where $|\Delta| \geqslant 4S + 2m + 1$, has pebbling time T such that*

$$T \geqslant |\Delta| \left(\frac{N - 2S - m}{2S + m + 1} \right)^k.$$

2. *There exists a pebbling sequence that pebbles the graph $G_{N,k}$ using $N + 2$ pebbles and needs time $O(kN \log N)$.*

3. *For any node $v \in V(G_{N,k})$, the incoming nodes of v and the position of v in first lexicographic topologically order of $G_{N,k}$ are computable in $O(k \log N)$.*

Graph Labeling Problem with faults. We adopt a well-known paradigm where the action of pebbling a node in a DAG G is made equivalent to the action of having calculated some labeling on it. This paradigm was introduced in [14] and also recently used in [15,16]. To satisfy the third of the pebbling rules, we make use of a Random Oracle (RO) $\mathcal{H}$ to build a labeling on G.

Definition 1 ($\mathcal{H}$-labeling with faults). *Given a DAG G and a random oracle function $\mathcal{H}$, we say that $\ell : V(G) \to \{0,1\}^*$ is a (partial) $\mathcal{H}$-labeling of G with m faults if and only if there exists a set $M \subseteq V(G)$ of size m such that for each $u \in V(G) \setminus M$*

$$\ell(u) := \mathcal{H}(u \| \ell(v_1) \| \ldots \| \ell(v_d)) \text{ where } \{v_1, \ldots, v_d\} = \Gamma^-(u). \tag{1}$$

Given a label ℓ and a node v we say that ℓ well-label v if only if the equation (1) holds for ℓ and v.

The use of a random oracle $\mathcal{H}$ provides two important benefits to our construction: First, given many instances $\mathcal{H}(x_1), \ldots, \mathcal{H}(x_k)$, the compression rate of $\mathcal{H}(x)$ is still very poor; second, in order to $\mathcal{H}$-label the graph, any TM must follow a pebble strategy. In particular, to label a node v, a TM must necessarily calculate and store the label values of all the predecessors $\ell(v_1), \ldots, \ell(v_d)$ of the node v. If the graph G needs at least S pebbles to be pebbled efficiently in a pebbling game, then a TM needs to store at least S labels (i.e., RO outputs) to compute an $\mathcal{H}$-labeling of G. This general strategy is proven sound in [14] and referred to as the *Labeling Lemma*. In our context, however, we provide m degrees of freedom and, given a partial $\mathcal{H}$-labeling ℓ of G with m faults and a $\mathcal{H}$-labeling ℓ' of G, it will likely be the case that $\ell(v) \neq \ell'(v)$ for each node v that is a descendant of a not well-labeled node. For this reason, we must state a more general version of the Labeling Lemma in [14] and its proof is provided in Appendix B.

Lemma 1 (Labeling Lemma). *Given a DAG G with degree d, a random oracle $\mathcal{H}$: $\{0,1\}^* \to \{0,1\}^k$, a $\mathcal{H}$-labeling ℓ with m faults of G and a TM $\mathcal{A}$ with advice h computing ℓ. If h is independent of $\mathcal{H}$, with overwhelming probability, there exists a pebbling sequence $\mathcal{P} = (P_1, \ldots, P_T)$ for the DAG G with m wildcards having pebbling space S such that:*

- $S \leqslant \frac{1}{k} \operatorname{Space}(\mathcal{A}) + d$,
- $T \leqslant (d + 2)\sigma$, *where σ is the number of queries of $\mathcal{A}$ to $\mathcal{H}$.*

In particular $T = O(\operatorname{Time}(\mathcal{A}))$, where $\operatorname{Time}(\mathcal{A})$ is the execution time of $\mathcal{A}$.

3 Proofs-of-Space Protocols

In this section we define the notion of PoSpace, then we give two constructions that meet the definition. We later define a second notion of a *weak* form of PoSpace and show a separation result between the two notions.

We remark that from a theoretical perspective the renowned Savitch's theorem establishes that $NSPACE(s(n)) \subseteq DSPACE(s(n)^2)$ for any function $s(n) \geqslant \log(n)$, i.e., while nondeterminism seems to provide an exponential advantage in time, it does not help much (or, maybe, at all) for space bounds (in particular, $PSPACE = NPSPACE$ while it is widely believed that $P \neq NP$). However, we must bound the adversary to be a PPT TM. Indeed, an unbounded time adversary could find a valid sequence of messages using space at most the square of the length of that sequence. Given that we seek succinct schemes, an unbounded adversary will be able to convince the verifier of a valid proof even though it uses less space than required.

In our definition below, we allow the adversary to access extra information to model the case in which the adversary may outsource storage and computation to an external provider.

Definition 2 (PoSpace). *Suppose we are given $\Sigma = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$, where Gen is a PPT TM, P and V are interactive PPT TMs, $k \in \mathbb{N}$ is a security parameter, and $\mathsf{pk} \leftarrow \mathsf{Gen}(1^k)$. Suppose that the following points hold for all $S \in \mathbb{N}$, $S > k$:*

(Completeness) $\operatorname{Space}(\mathsf{P}, \mathsf{pk}, 1^S) \geqslant S$ *and* $\Pr[\langle \mathsf{P}, \mathsf{V} \rangle (\mathsf{pk}, 1^S) = 1] = 1$;

(Succinctness) *The time and space complexity of V and the message complexity of $\langle \mathsf{P}, \mathsf{V} \rangle$, as functions of k and S, are $O(\mathsf{poly}(k) \cdot \mathsf{poly} \log S)$;*

(Soundness) *For any PPT adversary $\mathcal{A}$ and for any PPT TM with advice $\mathcal{A}'$ such that $h \leftarrow \mathcal{A}'(\mathsf{pk}, 1^S)$, the following event:*

$$\mathbf{Snd}_{\Sigma,S}^{\mathcal{A},\mathcal{A}'}(k) := \langle \mathcal{A}(h), \mathsf{V} \rangle (\mathsf{pk}, 1^S) = 1 \; \wedge \; \operatorname{Space}_{\mathcal{A}}(\langle \mathcal{A}(h), \mathsf{V} \rangle, \mathsf{pk}, 1^S) < S$$

has negligible probability (as a function of k).

Then, we say that $\Sigma = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *is a* Proof of Space (PoSpace). *To be concise, we could say informally that* $\mathcal{A}$ *wins when the event* $\exists S \in \mathbb{N} \; \exists h \leftarrow \mathcal{A}'(\mathsf{pk}) : \mathbf{Snd}_{\Sigma,S}^{\mathcal{A},\mathcal{A}'}(k)$ *occurs.*

Note that $\mathcal{A}'$ is a PPT TM that models the fact that there might be information that is hard to compute and independent from the public key but that the adversary $\mathcal{A}$ can exploit somehow to compromise PoSpace.

3.1 A 4-messages PoSpace protocol

We give a PoSpace construction in the Random Oracle Model (ROM) without any computational assumption. The protocol $\Sigma_4 = (\mathsf{Gen}, \mathsf{V}, \mathsf{P})$ is described in Figure 3.1 and it is a 4-message protocol. The protocol follows by some means Kilian's construction of argument system [22]. The verifier chooses a random α and asks the prover to build a $\mathcal{H}_\alpha$ labeling of graph $G_{N,k}$, where N depends on S. The labeling is the witness that the prover has handled at least S memory cells. The prover then commits the labeling and sends the commitment to the verifier. At this point, the verifier asks to open several random locations in the commitment and then it checks locally the integrity of the labeling. For a commitment C and for any node that the verifier has challenged, the prover sends what we call a *C-proof* for the node (defined next).

Definition 3 (C-proof). *Given a DAG G, a commitment C, and a random oracle $\mathcal{H}$, we say that a string $\boldsymbol{\pi} = (\pi_0, \dots, \pi_d)$ is a C-proof for a vertex $v \in V(G)$ w.r.t. $\mathcal{H}$ if only if given $\Gamma_G^-(v) = \{w_1, \dots, w_d\}$ the following points hold:*

1. *π_0 is a C-opening for v, let x be the value π_0 is opening to;*
2. *for each $i = 1, \dots, d$, π_i is a C-opening for w_i, let x_i be the value π_i is opening to;*
3. *$x = \mathcal{H}(v \| x_1 \| \dots \| x_d)$.*

We omit C, G, and $\mathcal{H}$, when they are clear from the context, saying that π is a proof.

In the definition of C-proof, the points 1 and 2 refer to the commitment C while point 3 ensures the integrity of the labeling. Note that the size of $\boldsymbol{\pi}$ is $O(kd \log N)$.

Notice that when TM $\mathcal{B}$ takes as input a RO $\mathcal{H}$ we intend that $\mathcal{B}$ has oracle access to $\mathcal{H}$ and the length of the input of $\mathcal{B}$ doesn't take into account the (exponentially long) length of $\mathcal{H}$.

Theorem 2. *The protocol Σ_4 in Figure 3.1 is a* PoSpace.

Here we just give the intuition behind the proof of the Theorem: For the complete proof, we refer the reader to Appendix C.

Completeness is trivial. Succinctness follows easily from point (3) of the Pebbling Theorem (Theorem 1). For Soundness, we first prove that the protocol Σ_4 is a Proof-of-Knowledge (PoK) of a partial labeling with "few" faults. By the PoK property, we can extract from a winning adversary a partial labeling with few faults. Thus, with overwhelming probability, the adversary $\mathcal{A}$ computes a partial labeling.

We then exploit the binding property of the Merkle-Tree and show that the adversary has computed the labeling during the round (1) of the protocol. Then, thanks to the Labeling Lemma (Lemma 1), we obtain a pebbling strategy that has pebbling space and pebbling time roughly upper-bounded by the space and time complexity of the adversary $\mathcal{A}$. Since the adversary uses strictly less space than S and $\mathcal{A}$ is PPT then, by our choice of the parameters, there is a contradiction with the point (1) of the Pebbling Theorem (Theorem 1).

Generator Gen takes as **input** 1^k and outputs $\mathsf{pk} := (\{G_{N,k}\}_{N \in \mathbb{N}}, \mathcal{H}, s)$,
where $\{G_{N,k}\}_{N \in \mathbb{N}}$ is a family of graphs satisfying the Pebbling Theorem (Theorem 1), equipped with
the natural lexicographic topological ordering on its vertex set,
$\mathcal{H}$ is a RO and s is a public key for CRH H.

Common input: k, S, pk
$N := \lceil 4\gamma(d + S/k) + \gamma \rceil$, where d is the degree of $G_{N,k}$ and $\gamma \in \mathbb{R}$, $\gamma > 1$.

Verifier V
Prover P

1. Pick $\alpha \leftarrow \{0,1\}^k$
$\xrightarrow{\quad \alpha \quad}$
Let ℓ be a $\mathcal{H}_\alpha$-labeling of $G_{N,k}$,
where $\mathcal{H}_\alpha(x) := \mathcal{H}(\alpha \| x)$.

$\xleftarrow{\quad C \quad}$
Commit $C := \mathsf{MT}_s(\ell)$.

2. Pick $(v_1, \dots, v_l) \leftarrow V^l$
uniformly at random, where
$l = \lfloor k \ln^2 k \log N \rfloor$ and
$V = V(G_{N,k})$
$\xrightarrow{\quad (v_1, \dots, v_l) \quad}$
For any v_i, $\pi_i := \mathrm{Open}_s (\ell, v_i)$:
For any $v_i^j \in \Gamma^-(v_i)$:
$\pi_i^j := \mathrm{Open}_s (\ell, v_i^j)$
$\Pi_i := (\pi_i, \pi_i^1, \dots, \pi_i^d)$

$\xleftarrow{\quad (\Pi_1, \dots, \Pi_l) \quad}$
Send $(\Pi_1, \dots, \Pi_l)$

3. Check for any $i \leqslant l$ if
Π_i is a C-proof for v_i wrt
the $\mathcal{H}_\alpha$-labeling ℓ

Fig. 3.1. The 4-messages POSPACE protocol Σ_4.

3.2 A 2-messages POSPACE protocol

The idea is to apply the standard *Fiat-Shamir* paradigm to the POSPACE scheme given in Section 3.1. We use two independent random oracles $\mathcal{H}, \mathcal{L}$:

- $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ is used by the prover for the labeling of the graph;
- $\mathcal{L} : \{0,1\}^k \to V^l$ given the commitment C as input, it yields the second verifier's message (of the protocol Σ_4).

Generator Gen on **input** 1^k tosses out $\mathsf{pk}' := (\mathsf{pk}, \mathcal{L})$,
where $\mathsf{pk} \leftarrow \Sigma_4.\mathsf{Gen}(1^k)$ and $\mathcal{L}$ is a RO

Common input: k, S, pk

Verifier V
Prover P

1. Pick $\alpha \leftarrow \{0,1\}^k$
$\xrightarrow{\quad \alpha \quad}$
$C := \Sigma_4.\mathsf{P}(\mathsf{pk}, 1^S, \alpha)$
$\boldsymbol{v} := \mathcal{L}(C)$

$\xleftarrow{\quad (C, \Pi) \quad}$
$\Pi := \Sigma_4.\mathsf{P}(\mathsf{pk}, 1^S, \alpha, \boldsymbol{v})$

2. Let $\boldsymbol{v} = \mathcal{L}(C)$
Check $\Sigma_4.\mathsf{V}(\mathsf{pk}, 1^S, \alpha, C, \boldsymbol{v}, \Pi) = 1$

Fig. 3.2. The 2-messages POSPACE protocol $\Sigma_2 = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$.

Let $(\Sigma_4.\mathsf{Gen}, \Sigma_4.\mathsf{P}, \Sigma_4.\mathsf{V})$ the POSPACE defined in Figure 3.1. We define in Figure 3.2 a 2-messages POSPACE: we call $\Sigma_2 = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ this protocol.

Theorem 3. *The Protocol Σ_2 in Figure 3.2 is a POSPACE.*

For the proof of this Theorem, we refer the reader to Appendix D.

4 *Weak* Proof-of-Space

The concept of proof of space can lead to multiple interpretations. The main interpretation formalized in the POSPACE definition requires that the prover can *handle* (i.e., read/write) space. In this section we provide a definition for a weaker alternative we call *weak*-Proof-of-Space (wPOSPACE). This captures the property that the prover can just access space and formalizes what could effectively be achieved by properly adapting previous work on proof of storage [4, 5, 29] and on memory-bound PoW as defined in [1, 12] (where the adversary model contemplates the existence of cache memory).

Then we prove that wPOSPACE definition is strictly weaker than POSPACE definition by giving a protocol which is a wPOSPACE but not a POSPACE.

The definition of wPOSPACE is similar to the Definition of POSPACE (Definition 2). The only change is in the Soundness part.

Definition 4 (wPOSPACE). *Suppose we are given $\Sigma = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ where Gen is a PPT TM, P and V are interactive PPT TMs, $k \in \mathbb{N}$ is a security parameter, and $\mathsf{pk} \leftarrow \mathsf{Gen}(1^k)$. Suppose that the following points hold for all $S \in \mathbb{N}, S > k$:*

(Completeness) and (Succinctness) *the same as in the POSPACE definition,*
(weak-Soundness) *For any PPT $\mathcal{A}$ adversary such that the following event:*

$$\mathbf{wSnd}_{\Sigma,S}^{\mathcal{A}}(k) := \langle \mathcal{A}, \mathsf{V} \rangle (\mathsf{pk}, 1^S) = 1 \wedge \mathrm{Space}_{\mathcal{A}}(\langle \mathcal{A}, \mathsf{V} \rangle, \mathsf{pk}, 1^S) < S,$$

has negligible probability (as a function of k).

Then we say that Σ is a Weak Proof of Space (wPOSPACE).

Unlike in the Soundness of POSPACE, we consider an adversary that can take advantage of an unbounded space machine which is then unavailable during the protocol execution.

Theorem 4. *There exists a protocol which is a wPOSPACE but not a POSPACE.*

Proof. Consider the protocol Σ_4 in Figure 3.1 where the first message α sent by V is always the same, say 0^k. We call Σ_3 this weaker version of Σ_4.

The protocol Σ_3 is not a POSPACE. For any k and $S \in \mathbb{N}$, consider the hint h which is the $\mathcal{H}_{0^k}$-labeling of $G_{N,k}$ plus the complete Merkle-Tree of that labeling. We define an adversary that sends the commitment C that is in h and for any verifier's second message $\boldsymbol{v}$ reads the right answer from h. That adversary needs to access only h in *read-only* mode, hence without using any additional working space.

We prove that Σ_3 is a wPOSPACE in Appendix E. $\qquad\square$

References

1. Martin Abadi, Mike Burrows, Mark Manasse, and Ted Wobber. Moderately hard, memory-bound functions. *ACM Trans. Internet Technol.*, 5(2):299–327, May 2005.
2. Mehmud Abliz and Taieb Znati. A guided tour puzzle for denial of service prevention. In *ACSAC*, pages 279–288. IEEE Computer Society, 2009.
3. J. Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. *Advances in Cryptology-CRYPTO 2009*, pages 36–54, 2009.
4. Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 598–609, New York, NY, USA, 2007. ACM.
5. Giuseppe Ateniese, Seny Kamara, and Jonathan Katz. Proofs of storage from homomorphic identification protocols. In *ASIACRYPT*, pages 319–333, 2009.
6. Adam Back. Hashcash - a denial of service counter-measure. Technical report, 2002.
7. Boaz Barak and Oded Goldreich. Universal arguments and their applications. In *IEEE Conference on Computational Complexity*, pages 194–203. IEEE Computer Society, 2002.
8. R. B. Boppana, J. Hastad, and S. Zachos. Does co-np have short interactive proofs? *Inf. Process. Lett.*, 25(2):127–132, May 1987.
9. Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors. *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*. Springer, 2005.
10. Ran Canetti, Ben Riva, and Guy N. Rothblum. Refereed delegation of computation. *Information and Computation*, 226(0):16 – 36, 2013.
11. Kai-min Chung, Yael Kalai, and Salil Vadhan. Improved Delegation of Computation Using Fully Homomorphic Encryption. pages 483–501, 2010.
12. Cynthia Dwork, Andrew Goldberg, and Moni Naor. On memory-bound functions for fighting spam. *Advances in Cryptology-Crypto 2003*, 2003.
13. Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. *Advances in Cryptology-CRYPTO'92*, pages 139–147, 1993.
14. Cynthia Dwork, Moni Naor, and Hoeteck Wee. Pebbling and proofs of work. *Advances in CryptologyCRYPTO 2005*, pages 37–54, 2005.
15. Stefan Dziembowski, Tomasz Kazana, and Daniel Wichs. Key-evolution schemes resilient to space-bounded leakage. In *CRYPTO*, pages 335–353, 2011.
16. Stefan Dziembowski, Tomasz Kazana, and Daniel Wichs. One-time computable self-erasing functions. In *Proceedings of the 8th conference on Theory of cryptography*, TCC'11, pages 125–143, Berlin, Heidelberg, 2011. Springer-Verlag.
17. Stefan Dziembowski, Krzysztof Pietrzak, and Sebastian Faust. Proofs of space and a greener bitcoin. talk presented at *Workshop on Leakage, Tampering and Viruses*, Warsaw 2013, 2013.
18. Dario Fiore and Rosario Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. *Proceedings of the 2012 ACM conference on Computer and communications security - CCS '12*, page 501, 2012.
19. Oded Goldreich and Johan Hastad. On the complexity of interactive proofs with bounded communication. *Information Processing Letters*, 1998.
20. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 113–122, New York, NY, USA, 2008. ACM.
21. Ari Juels and John G. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *NDSS*. The Internet Society, 1999.
22. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, STOC '92, pages 723–732, New York, NY, USA, 1992. ACM.
23. Thomas Lengauer and Robert E. Tarjan. Asymptotically tight bounds on time-space trade-offs in a pebble game. *Journal of the ACM (JACM)*, 29(4):1087–1130, 1982.
24. Ralph C. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Security and Privacy*, pages 122–134, 1980.
25. Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.
26. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. May 2009.
27. Colin Percival. Stronger key derivation via sequential memory-hard functions. presented at BSDCan '09, 2009.
28. Nicholas Pippenger. Superconcentrators. *SIAM J. Comput.*, 6(2):298–304, 1977.

29. Hovav Shacham and Brent Waters. Compact proofs of retrievability. In *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '08, pages 90–107, Berlin, Heidelberg, 2008. Springer-Verlag.
30. Martin Tompa. Time-space tradeoffs for computing functions, using connectivity properties of their circuits. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, STOC '78, pages 196–204, New York, NY, USA, 1978. ACM.
31. Brent Waters, Ari Juels, J. Alex Halderman, and Edward W. Felten. New client puzzle outsourcing techniques for dos resistance. In *Proceedings of the 11th ACM conference on Computer and communications security*, CCS '04, pages 246–256, New York, NY, USA, 2004. ACM.

A   Proof of the Pebbling Theorem (Theorem 1)

This section is devoted to the proof of the Pebbling Theorem. For the sake of convenience we re-write the statement of that theorem here. For the definitions about pebbling games, pebbling space and pebbling time we refer the reader to Section 2.

Pebbling Theorem (Theorem 1). *There exist a family of directed acyclic graphs $\{G_{N,k}\}_{k,N\in\mathbb{N}}$ with constant fan-in d, N input nodes, N output nodes and $kN\log N$ nodes in total, such that:*

1. *Any pebbling sequence that pebbles Δ output nodes using at most S pebbles and m wildcards, where $|\Delta| \geqslant 4S + 2m + 1$, has pebbling time T such that*

$$T \geqslant |\Delta| \left(\frac{N - 2S - m}{2S + m + 1} \right)^k .$$

2. *There exists a pebbling sequence that pebbles the graph $G_{N,k}$ using $N + 2$ pebbles and needs time $O(kN \log N)$.*
3. *For any node $v \in V(G_{N,k})$, the incoming nodes of v and the position of v in first lexicographic topologically order of $G_{N,k}$ are computable in $O(k \log N)$.*

The family of graphs $G_{N,k}$ we build is made up by DAGs that are k layered stack of superconcentrators [28] with N inputs and N outputs.

Definition 5 (superconcentrator). *We say that a directed acyclic graph $G = (V, E)$ is an N-superconcentrator if and only if*

1. $|\mathbf{S}(G)| = |\mathbf{T}(G)| = N$,
2. *for each $S \subseteq \mathbf{S}(G)$ and for each $T \subseteq \mathbf{T}(G)$ such that $|S| = |T| = \ell$ there exists ℓ vertex-disjoint paths from S to T, i.e. paths such that no vertex in V is in two of them.*

Definition 6 (stack of superconcentrators). *Given an N-superconcentrator G we define the graph $G^{\times k}$, the stack of k copies of G, inductively as follows: $G^{\times 1} = G$ and $G^{\times k}$ is obtained from $G^{\times(k-1)}$ and G first renaming the vertexes of G so that they do not appear in the vertexes of $G^{\times(k-1)}$ obtaining a graph G' and then identifying $\mathbf{T}(G^{\times(k-1)})$ with $\mathbf{S}(G')$.*

Not any k layered stack of superconcentrators satisfy points 2 and 3 of the Pebbling Theorem. To obtain those properties we use the following well known superconcentrator: the *Butterfly Graph B_N* built by putting together two FFT graphs with N inputs and outputs [30]. An example of such construction (for $N = 16$) is given in Figure A.1.

It is easy to see that the family of graphs $G_{N,k} := B_N^{\times k}$ satisfy points 2 and 3 of the Pebbling Theorem and it is a k layered stack of N-superconcentrators. It remains to prove only point 1.

For superconcentrator graphs we can prove a tradeoff between pebbling space and pebbling time, even for pebbling games with wildcards. This result is based on a variation of the *"Basic Lower Bound Argument"* (BLBA) by Tompa [30].
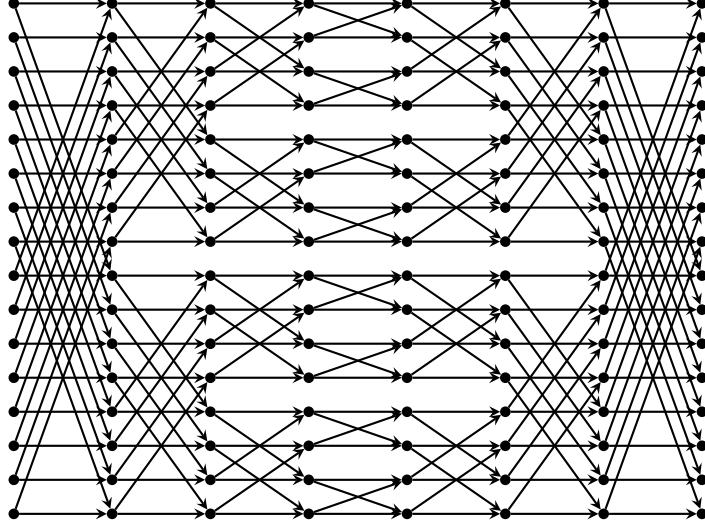
Fig. A.1. An example of 16-superconcentrator: the Butterfly Graph

Lemma 2 (BLBA with wildcards). *Suppose we are given an N-superconcentrator G, a set $M \subseteq \mathbf{T}(G)$ and a pebbling sequence $\mathcal{P} = (P_0, \ldots, P_\ell)$ with m wildcards that pebbles M. Let A be the set of all elements of $\mathbf{S}(G)$ pebbled and unpebbled at some point in $\mathcal{P}$. If $|M| \geqslant |P_0| + |P_\ell| + m + 1$ then $|A| \geqslant N - |P_0| - |P_\ell| - m$.*

Proof. By contradiction suppose that $|A| < N - |P_0| - |P_\ell| - m$, this means that $|A^c| \geqslant |P_0| + |P_\ell| + m + 1$ and every element in A^c, by definition, is not pebbled and unpebbled. Take any $B \subseteq A^c$ such that $|B| = |M|$ then, as G is an N-superconcentrator, we have $\pi_1, \ldots, \pi_{|M|}$ vertex disjoint paths from B to M. Let W a set of m fault for $\mathcal{P}$. By construction $|M| > |P_0 \cup P_\ell \cup W|$ so we have some path π from some element v of B to some element w of M such that $\pi \cap (P_0 \cup P_\ell \cup W) = \emptyset$. By definition of A^c, and hence of B, v is not pebbled and unpebbled and $v \notin P_0 \cup P_\ell \cup W$, so $v \notin P_i$ for each $i \in [\ell]$. As $\pi \cap (P_0 \cap W) = \emptyset$ then we must have that $\pi \cap P_i = \emptyset$ for each $i \in [\ell]$. But then the vertex w is not in $\bigcup_{i \in [\ell]} P_i$ contradicting the fact that $\mathcal{P}$ pebbles M. $\qquad\square$

Theorem 5 (Pebbling Theorem (point 1)). *Let $G^{\times k}$ be a stack of k copies of an N-superconcentrator G, $\Delta \subseteq \mathbf{T}(G^{\times k})$ and $\mathcal{P}$ a pebbling sequence for $G^{\times k}$ with m wildcards that pebbles Δ. Let S the pebbling space of $\mathcal{P}$ and T the pebbling time of $\mathcal{P}$. If $|\Delta| \geqslant 4S + 2m + 1$ then*

$$T \geqslant |\Delta| \left(\frac{N - 2S - m}{2S + m + 1} \right)^k .$$

Proof. Let $G_1, \ldots, G_k$ be the k copies of G that form $G^{\times k}$. Suppose we want to pebble a set $M \subseteq \mathbf{T}(G_i)$ such that $|M| \geqslant 2S + m + 1$ and let $A_1, \ldots, A_n$ be disjoint subsetes of M such that A_1 are the first $2S + m + 1$ elements of M to be pebbled in $\mathcal{P}$, A_2 are the second $2S + m + 1$ elements of M to be pebbled in $\mathcal{P}$ and so on. Clearly we have that the number of these sets is $n = \left\lfloor \frac{|M|}{2S+m+1} \right\rfloor$. By Lemma 2 we have that for each A_j there exists a set $\beta(A_j)$ in $\mathbf{S}(G_i)$, whose elements are pebbled and unpebbled, of size at least $N - 2S - m$. So we have that the total number of elements in $\mathbf{S}(G_i)$ that have been pebbled and unpebbled is at least $(N - 2S - m) \left\lfloor \frac{|M|}{2S+m+1} \right\rfloor$. Notice now that each $\beta(A_j) \in \mathbf{T}(G_{i-1})$ so we can repeat the argument above to it. Provided that for each j, $|\beta(A_j)| \geqslant N - 2S - m \geqslant 2S + m + 1$ and this is implied by the fact that $N \geqslant |\Delta| \geqslant 4S + 2m + 1$. So starting from $\mathbf{T}(G_k) = \mathbf{T}(G^{\times k})$ we can prove by

induction, going back to G_1, that at least

$$|\Delta| \left(\frac{N - 2S - m}{2S + m + 1} \right)^k$$

actions of pebbling and unpebbling take place on $\mathbf{S}(G_1) = \mathbf{S}(G^{\times k})$. As each action is an elementary step in the pebbling game $\mathcal{P}$ we have that the result follows. □

B   Proof of the Labeling Lemma (Lemma 1)

Before proceeding with the proof of the Lemma we recall the definitions of min-entropy and conditional average min-entropy as stated in [3]. For any random variable X we define the min-entropy of X as

$$\mathbf{H}_\infty(X) := -\log \left(\max_{\mathcal{B} \; TM} \Pr\left[\mathcal{B}() = X\right] \right)$$

and we define the conditional average min-entropy of X given an other random variable Y as

$$\widetilde{\mathbf{H}}_\infty(Y|X) := -\log \left(\max_{\mathcal{B} \; TM} \Pr\left[\mathcal{B}(X) = Y\right] \right).$$

It's easy to show [3] the following chaining rules for conditional average min-entropy:

$$\widetilde{\mathbf{H}}_\infty(Y|X) \geqslant \mathbf{H}_\infty(Y) - \mathbf{H}_\infty(X).$$

Lemma 1.   *Given a DAG G with degree d, a random oracle $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$, a $\mathcal{H}$-labeling ℓ with m faults of G and a TM $\mathcal{A}$ with advice h computing ℓ. If h is independent of $\mathcal{H}$, with overwhelming probability, there exists a pebbling sequence $\mathcal{P} = (P_1, \dots, P_T)$ for the DAG G with m wildcards having pebbling space S such that:*

- *$S \leqslant \frac{1}{k} \mathrm{Space}(\mathcal{A}) + d$,*
- *$T \leqslant (d+2)\sigma$, where σ is the number of queries of $\mathcal{A}$ to $\mathcal{H}$.*

In particular $T = O(\mathrm{Time}(\mathcal{A}))$, where $\mathrm{Time}(\mathcal{A})$ is the execution time of $\mathcal{A}$.

Proof. Let W be the set of not well-labeled nodes by the partial labeling ℓ, in particular $|W| = m$. We look at the ordered set of queries $\mathcal{Q} = \{Q_1, \dots, Q_\tau\}$ of $\mathcal{A}$ to $\mathcal{H}$ of the form $Q_i := \left(w_i \| \ell(w_i^1) \| \dots \| \ell(w_i^d)\right)$ where $\forall i \; w_i \in V(G)$ and $\forall j \; w_i^j \in \Gamma^-(w_i)$. Notice that by costruction $\tau \leqslant \sigma$.

Let $S_i = S_{i-1} \cup \{w_i\} \cup \Gamma^-(w_i)$. As ℓ is a partial $\mathcal{H}$-labeling and $\mathcal{H}$ a RO we have that with overwhelming probability $\Gamma^-(w_i) \subseteq S_{i-1} \cup W \cup \mathbf{S}(G)$. Hence the sequence $(S_1, \dots, S_\sigma)$ can be interpolated to a pebbling sequence adding source vertexes when needed. This would make a pebbling sequence $\tilde{P}$ of length at most $(d+1)\tau$. Interpolating $\tilde{\mathcal{P}}$ with suitable unpebbling moves after each S_i, i.e. removing all unnecessary pebbles following the informal rule "unpebble as soon as possible", we can obtain a pebbling sequence $\mathcal{P}$ using minimal space. This is the desired pebbling sequence $\mathcal{P} = (P_1, \dots, P_T)$ with as set of wildcards W. Notice that by construction $T = (d+2)\tau \leqslant (d+2)\sigma$, hence we have only to focus on bounding the pebbling space of $\mathcal{P}$.

Let $\tilde{S} = \mathrm{Space}(\mathcal{A})$, suppose for simplicity that k divides $\tilde{S}$ and, by contradiction, suppose that there are $\tilde{S}/k + d + 1$ pebbles at some time t^* in $\mathcal{P}$. This time should correspond to a query to $\mathcal{H}$. After the unpebble operation, by the rule "unpebble as soon as possible", there are left at least $\tilde{S}/k + 1$ pebbles at time $t^* + 1$. We look at the labels of the nodes of G having at time

$t^* + 1$ one of the $\tilde{S}/k + 1$ pebbles: by the use of a RO $\mathcal{H}$ their labels correspond to an uniformly random string $L := \ell_1 \| \ldots \| \ell_{\tilde{S}/k+1} \in \{0,1\}^{\tilde{S}+k}$.

Let M be the content of the memory of $\mathcal{A}$ at time $t^* + 1$. Note that M is a random variable in $\{0,1\}^{\tilde{S}-1}$. By the chaining rule of the conditional average min-entropy we have that

$$\widetilde{\mathbf{H}}_\infty(L|M,h) = \widetilde{\mathbf{H}}_\infty(L|M) > \mathbf{H}_\infty(L) - S = k, \tag{2}$$

where the first equality comes from the independence of $\mathcal{H}$ from h.

We want to upperbound $\widetilde{\mathbf{H}}_\infty(L|M,h)$. To do this we define $\mathcal{B}$ a TM that on input M and h executes $\mathcal{A}$ from the state where she was at time $t^* + 1$, writing in the $\mathcal{A}$'s working tape the string M and on the input tape the hint h and simulating the random oracle $\mathcal{H}$. Notice that, as we unpebble as soon as possible then, for any $1 \leqslant i \leqslant S/k + 1$, the followings are tue:

1. $\mathcal{A}$ is going to query (as substring of queries to $\mathcal{H}$) the value ℓ_i with overwhelming probability. In fact if this is not the case then in $\mathcal{P}$ it would be already deleted[2]. The overwhelming probability comes from the fact that $\mathcal{A}$ could guess ℓ_i. Hence $\mathcal{B}$ can collect all of those labels $\ell_1, \ldots, \ell_{\tilde{S}/k+1}$ by looking at the queries made by $\mathcal{A}$ to $\mathcal{H}$. Moreover $\mathcal{B}$ can reply to all the queries to $\mathcal{H}$ uniformly choosing strings in $\{0,1\}^k$ and storing those values to reply accordingly in the future. This simulation is possible as h is independent from $\mathcal{H}$.

2. $\mathcal{A}$ won't query $\mathcal{H}$ with a string which corresponds to a successor of ℓ_i before having tossed out ℓ_i as a query to $\mathcal{H}$ with overwhelming probability. Otherwise in $\mathcal{P}$ it would already have been deleted. The overwhelming probability comes from the fact that there could be a collision in the RO $\mathcal{H}$. Hence the simulation of $\mathcal{H}$ stores the labels $\ell_1, \ldots, \ell_{\tilde{S}/k+1}$ before using them as replies to $\mathcal{A}$, therefore the simulation is indistinguishable from the $\mathcal{A}$'s point of view. Eventually $\mathcal{B}$ outputs all the labels collected.

The existence of $\mathcal{B}$ implies that $-\log(1 - \nu(k)) \geqslant \widetilde{\mathbf{H}}_\infty(L|M,h)$, where $\nu(k)$ is an negligible function. This give an immediate contradiction with equation (2). Hence the pebbling space of $\mathcal{P}$ is at most $\tilde{S}/k + d$. □

C The protocol Σ_4 in Figure 3.1 is a PoSPACE (Theorem 2)

Completeness and *Succinctness* follow immediately from points (2) and (3) of Theorem 1. Hence we will focus on the *Soundness*. We divide the proof in two parts. First, in the PoK Lemma, we prove that the Protocol Σ_4 is a PoK for a partial labeling with "few" faults, and that the partial labeling has been computed during the round (1), i.e. after the verifier V sent the message α and before the adversary sent the commitment message C. Then we sum up the PoK Lemma with the Labeling Lemma and the Pebbling Theorem to reach a contradiction assuming that there exists an winning PPT adversary.

We stress out that the knowledge extractor doesn't need to be efficient since we do not use any computational assumption.

Lemma 3 (PoK Lemma). *Given the protocol $\Sigma_4 = (\mathsf{Gen}, \mathsf{V}, \mathsf{P})$, a PPT adversary $\mathcal{A}$ and a PPT Turing Machine with advice $\mathcal{A}'$, a space parameter S and a security parameter k such that $\mathsf{pk} \leftarrow \mathsf{Gen}(1^k)$ and $h \leftarrow \mathcal{A}'(\mathsf{pk}, 1^S)$. If $\Pr\left[\mathbf{Snd}_{\Sigma_4,S}^{\mathcal{A},\mathcal{A}'}(k)\right]$ is noticeable, then there exists a randomness $\tilde{\rho}$ and a first message of V $\tilde{\alpha}$ such that $\mathcal{A}(h)$ receiving as first message $\tilde{\alpha}$ and fed with the randomness $\tilde{\rho}$ during the round (1) of the protocol Σ_4 calculate a $\mathcal{H}_{\tilde{\alpha}}$-labeling ℓ of $G_{N,k}$ with m faults such that the followings hold:*

---

[2] This is the reason why we apply this information-theoretical argument to time $t^* + 1$ and not to time t^*. In fact it would be the case that at time t^* we are pebbling a vertex w but none of the vertexes in $\Gamma^-(w)$ would be used again.

- the hint h is independent from $\mathcal{H}_{\tilde{\alpha}}$,
- $m = O\left(\frac{N}{\ln k}\right)$ and
- for any well-labeled node u in ℓ we have $(\tilde{\alpha}\|u\|\ell(u_1)\|\ldots\|\ell(u_d)) \in \mathcal{Q}$,

where $\mathcal{Q}$ is the set of queries to $\mathcal{H}$ made by $\mathcal{A}$ when fed with randomness $\tilde{\rho}$ until the end of round (1) and $\{u_1, \ldots, u_d\} = \Gamma^-(u)$.

Proof. Let

$$\mathcal{Q}' := \left\{\alpha \in \{0,1\}^k \mid \mathcal{A}'(\mathsf{pk}, 1^S) \text{ made at least a RO query to } \mathcal{H} \text{ with prefix } \alpha\right\}.$$

We say that a first message α is *good* iff $\Pr\left[\langle\mathcal{A}(h), \mathsf{V}\rangle(\mathsf{pk}, 1^S) = 1 \mid \mathsf{V} \text{ sends } \alpha \text{ as first message}\right]$ is noticeable. By Markov's inequality the probability that an uniformly random first message α is good is noticeable. On the other hand the probability that an uniformly random first message α is in $\mathcal{Q}'$ is bounded by $\mathsf{poly}(k, S)/2^k$, so there exists $\tilde{\alpha}$ that is both good and not in $\mathcal{Q}'$. Since $\tilde{\alpha} \notin \mathcal{Q}'$ then $h \leftarrow \mathcal{A}'(\mathsf{pk}, 1^S)$ is independent from $\mathcal{H}_{\tilde{\alpha}}$. Let $V = V(G_{N,k})$ the vertex set of $G_{N,k}$ and let $W_C(\boldsymbol{v})$ the following event:

> "$\mathcal{A}(h, \mathsf{pk}, \tilde{\alpha})$ sends the commitment C at the end of round (1)
> and makes the verifier V to accept
> when challenged with the vector $\boldsymbol{v} \in V^l$ during round (2)".

Remind that l is the length of the message as defined in the protocol Σ_4.

A commitment string sent by $\mathcal{A}(h, \mathsf{pk}, \tilde{\alpha})$ to the verifier V at the end of round (1) is a *good commitment* iff $\Pr[W_C(\boldsymbol{v})]$ is noticeable. We have that with noticeable probability $\mathcal{A}(h, \mathsf{pk}, \tilde{\alpha})$ sends a good commitment at the end of round (1).

It might be the case that $\mathcal{A}(h, \mathsf{pk}, \tilde{\alpha})$ instead of computing the labeling through the RO queries guesses the correct one. However the $\mathcal{A}$'s guess is correct with probability less than 2^{-k}. Hence with noticeable probability during round (1):

- $\mathcal{A}(h)$ is not making any guess and
- $\mathcal{A}(h)$ sends a good commitment.

So there exists a randomness $\tilde{\rho}$ such that $\mathcal{A}(h, \mathsf{pk}, \tilde{\alpha})$ using as source of randomness $\tilde{\rho}$ has the above two properties. Let $\tilde{\mathcal{A}}$ be such adversary, i.e. $\mathcal{A}$ where we have fixed the randomness in round (1) to $\tilde{\rho}$ and that receives as first message $\tilde{\alpha}$. Let C be the good commitment sent by $\tilde{\mathcal{A}}$. Now we focus on round (2). Let

$$E := \left\{u \in V \mid \Pr\left[\tilde{\mathcal{A}} \text{ challenged on } \boldsymbol{v} \in V^l \text{ sends a } C\text{-proof for } u \ \wedge \ u \in \boldsymbol{v}\right] < r(k)\right\},$$

where $r(k) \in \mathsf{poly}(k)$ some function we will decide later.

There exists a procedure to compute the set E (recall that the extraction process doesn't need to be efficient). Notice that by construction the nodes of $V \setminus E$ can be C-opened with noticeable probability, so they C-open to fixed labels with noticeable probability, otherwise an efficient sampling procedure can break the binding property of the underlying Merkle-Tree.

For each $u \in V \setminus E$ there exists a randomness ρ_u such that $\tilde{\mathcal{A}}$ using as source of randomness ρ_u during round (2) (i.e. $\mathcal{A}(h)$ globally using as randomness first $\tilde{\rho}$ and then ρ_u) sends to the verifier V a C-proof for u that C-opens u to the most probable label.

If we collect all those values we obtain a $\mathcal{H}_{\tilde{\alpha}}$-labeling with at most $|E|$ faults.

16

We now prove the second item of the lemma, i.e. the upper-bound for $m = |E|$. We have the following chain of inequalities:

$$\Pr_{\boldsymbol{v}\leftarrow V^l}[W_C(\boldsymbol{v})] = \Pr_{\boldsymbol{v}\leftarrow V^l}[W_C(\boldsymbol{v}) \wedge \forall v_i \in \boldsymbol{v}\ v_i \notin E] + \Pr_{\boldsymbol{v}\leftarrow V^l}[W_C(\boldsymbol{v}) \wedge \exists v_i \in \boldsymbol{v}\ v_i \in E] \leqslant$$

$$\leqslant \Pr_{\boldsymbol{v}\leftarrow V^\ell}[\forall v_i \in \boldsymbol{v}\ v_i \notin E] + \sum_{u\in E}\Pr_{\boldsymbol{v}\leftarrow V^l}[W_C(\boldsymbol{v}) \wedge u \in \boldsymbol{v}] \leqslant$$

$$\leqslant \left(1 - \frac{m}{|V|}\right)^l + \sum_{u\in E}\Pr_{\boldsymbol{v}\leftarrow V^l}\left[\tilde{\mathcal{A}} \text{ challenged by } \mathsf{V} \text{ on } \boldsymbol{v} \text{ sends a } C\text{-proof for } u\ \wedge\ u \in \boldsymbol{v}\right] \leqslant$$

$$< \left(1 - \frac{m}{|V|}\right)^l + |V|r(k).$$

There exists a constant $c \in \mathbb{N}$ such that for $k \gg 1$, $\Pr[W_C(\boldsymbol{v})] \geqslant k^{-c}$, hence if we choose $r(k) = (|V|2k^c)^{-1}$ we obtain that

$$e^{-\frac{m}{|V|}\cdot l} > \left(1 - \frac{m}{|V|}\right)^l > \frac{1}{2k^c}.$$

This implies

$$\frac{m}{|V|} \cdot l < \ln 2 + c\ln k.$$

Notice that, in the graph family $\{G_{N,k}\}$, $|V| = kN\log N$, hence, as $l = k\ln^2 k\log N$ we obtain the required upper-bound for m.

We now prove the third item of the Lemma, i.e. for any assignment ρ of the randomness of $\mathcal{A}$ for the round (1) such that (a) the commitment C is a good commitment and (b) $\mathcal{A}$ has not made any guess, we have that for any $u \in V \setminus E$, $Q_u := (\tilde{\alpha}\|u\|\ell(u_1)\|\ldots\|\ell(u_d)) \in \mathcal{Q}$, where $\mathcal{Q}$ is the set of queries made by $\mathcal{A}$ to $\mathcal{H}$ until the end of round (1) and $\{u_1, \ldots, u_d\} = \Gamma^-(u)$.

If that would not be the case we can define a reduction that breaks the binding property of the Merkle Tree. The probability that a uniformly random assignment of the randomness of $\mathcal{A}$ has the properties (a) and (b) and a uniformly random first message of V is good and not in $\mathcal{Q}'$ is noticeable: let $p(k) \in \mathsf{poly}(k)$ a lower-bound for that probability.

By contradiction suppose there exists an u such that the query $Q_u \notin \mathcal{Q}$. By the choice of $\tilde{\alpha}$ $Q_u \notin \mathcal{Q}'$ hence it must have be done in the round (2) of the protocol.

Suppose for the moment that the reduction know the values $\ell(u_1), \ldots, \ell(u_d)$, then

(a) it can choose an assignment of the randomness ρ of $\mathcal{A}$ for the round (1) and a first message $\alpha \in \{0,1\}^k$ both uniformly at random. Then execute $\mathcal{A}(h, \mathsf{pk}, 1^S, \alpha; \rho)$ obtaining as output a commitment C;

(b) V challenge $\mathcal{A}$ in round (2) with an uniformly random vector $\boldsymbol{v} \in V^l$ containing u as a coordinate. $\mathcal{A}$ will make the query Q_u to $\mathcal{H}$ and the reduction respond to it with a uniformly random string in $\{0,1\}^k$;

(c) rewind $\mathcal{A}$ to the point corresponding to the end of round (1) of the protocol, then repeat again point (b) of the reduction.

$\mathcal{A}$ will C-open the node u in points (b) and (c) of the reduction with two different values with a probability at least $p(k) \cdot (r(k)^2 - 2^{-k})$.

In general the reduction doesn't know the values $\ell(u_1), \ldots, \ell(u_d)$ and it can't figure out which of the queries made by $\mathcal{A}$ in the round (2) of the protocol is Q_u. But $\mathcal{A}$ is PPT so we can find a $q(k) \in \mathsf{poly}(k, S)$ that upper-bounds the number of queries made by $\mathcal{A}$ to $\mathcal{H}$ during round (2).

$\mathcal{A}$ will C-opens the node u, by guessing the moment Q_u is queried, with two different values with a probability at least $q(k)^{-1} \cdot p(k) \cdot (r(k)^2 - 2^{-k})$. Hence this reduction breaks the binding property of the Merkle Tree. $\qquad\square$

Now that we have set all the preliminary results the proof of Theorem 2 follows in a few lines. By contradiction, suppose there exist an adversary $\mathcal{A}$ for the protocol Σ_4.

By applying Lemma 3 we extract a partial $\mathcal{H}_{\tilde{\alpha}}$-labeling, ensure that $\mathcal{A}(h)$ computed that labeling during round (1) and the hint h is independent from $\mathcal{H}_{\tilde{\alpha}}$. Hence satisfying the hypothesis of Lemma 1.

This gives us, with overwhelming probability, a pebbling sequence $\mathcal{P}$ of $G_{N,k}$ with m wildcards having pebbling space $S' = \frac{S}{k} + d$, $m = O(\frac{N}{\ln k})$ and the pebbling time of $\mathcal{P}$ is a lower bound for the execution time of $\mathcal{A}$ during round (1). We want to apply Theorem 1. Hence we want to fix N such that

$$N - l \geqslant 4S' + 2m + 1, \tag{3}$$

where l denotes the number of wildcards that are in $\mathbf{T}(G_{N,k})$.

Given $c \in (0, 1)$ and $\epsilon > 0$ we have that $(2 + \epsilon)m + l \leqslant cN$. If we find N such that

$$N \geqslant 4S' + cN + 1,$$

hence $N \geqslant 4\gamma S' + \gamma$, where $\gamma = 1/(1 - c)$, then the inequality (3) will follow. This is why in the Protocol Σ_4 we have set $N := \lceil 4\gamma S' + \gamma \rceil$.

Now with this N we can apply Theorem 1 obtaining that the pebbling time of $\mathcal{P}$ and hence the execution time T of $\mathcal{A}$ is such that

$$T \geqslant (N - l) \left(\frac{N - 2S' - m}{2S' + m + 1} \right)^k \geqslant c'(1 + \epsilon)^k, \tag{4}$$

where c' is a constant that is a lower bound for $N - l$ and the $(1 + \epsilon)^k$ comes from the fact that, eventually,

$$N - 2S' - m \geqslant (1 + \epsilon)(2S' + m + 1).$$

Equation (4) shows that the execution time of $\mathcal{A}$ is exponential in k and it is not possible as, by hypothesis, $\mathcal{A}$ is PPT. $\qquad\square$

D The Protocol Σ_2 in Figure 3.2 is a PoSpace (Theorem 3)

Completeness and *Succinctness* follow by the fact that Σ_4 is a PoSpace as proved in Theorem 2. To prove *Soundness* we reduce an adversary $\mathcal{A}$ for the scheme Σ_2 to an adversary $\mathcal{B}$ for the scheme Σ_4 (of Figure 3.2). Suppose, by contradiction, there exists an adversary $\mathcal{A}$, a hint $h \leftarrow \mathcal{A}(\mathsf{pk}')$ and a value S such that for k large enough:

$$\Pr\left[\mathbf{Snd}_{\Sigma_2, S}^{\mathcal{A}, \mathcal{A}'}(k) \right] \geqslant k^{-c}.$$

Let $q(k)$ a polynomial that upper-bounds the number of queries made by $\mathcal{A}$ to the RO $\mathcal{L}$. For each $0 \leqslant i < q(k)$, let Q_i be the i-th query made by $\mathcal{A}$ to the oracle $\mathcal{L}$. Without loss of generality we can assume that all the Q_i are different and $\mathcal{A}$ verifies her output certificate before sending it: if she hasn't find a valid certificate she outputs the empty string. So either there exist an index m such that $(C, \Pi) \leftarrow \mathcal{A}(h, \mathsf{pk}', \alpha; \rho)$ with $Q_m = C$ and $\Sigma_4.\mathsf{V}\left(\mathsf{pk}, 1^S, \alpha, C, \mathcal{L}(C), \Pi \right) = 1$ or $\mathcal{A}(h, \mathsf{pk}', \alpha; \rho)$ outputs the empty string. Moreover:

$$\Pr[(C, \Pi) \leftarrow \mathcal{A}(h, \mathsf{pk}', \alpha) \wedge Q_m = C] \geqslant k^{-c} \cdot q(x)^{-1}.$$

Notice that the probability above says that the outputs of $\mathcal{A}$ will be the same value asked by $\mathcal{A}$ to $\mathcal{L}$ at m-th call, however, the value Q_m is a random variable that may depends on all the randomness used until the m-th call. We divide the randomness used by $\mathcal{A}$ into two pieces ρ_1 and ρ_2: ρ_1 is the randomness used by $\mathcal{A}$ before she has made the m-th query to the oracle $\mathcal{L}$, ρ_2 is the remaining randomness.

By an averaging argument there exists a setting of ρ_1 such that the following equation holds

$$\Pr_{\rho_2, \mathcal{H}, \mathcal{L}} \left[(C, \Pi) := \mathcal{A}(h, \mathsf{pk}', \alpha; \rho_1 \| \rho_2) \wedge Q_m = C \right] \geqslant k^{-c} \cdot q(x)^{-1} \tag{5}$$

Now the value Q_m is a random variable that depends only by randomness of the challenge α, the ROs $\mathcal{H}$ and $\mathcal{L}$ and ρ_1. So, by repeating many times the TM $\mathcal{A}(h, \mathsf{pk}', \alpha; \rho_1 \| \rho_2)$ for a uniformly random ρ_2 using the same oracles $\mathcal{H}, \mathcal{L}$ we'll get the same value Q_m as m-th query to $\mathcal{L}$. We define the PPT TM with advice $\mathcal{B}'$ that on input pk and advice ρ_1, m outputs the hint $\tilde{h} := (\mathcal{A}'(\mathsf{pk}, \mathcal{L}), \mathcal{Q}', m, \rho_1)$ where $\mathcal{Q}'$ is the set of RO queries made by $\mathcal{A}'$ to $\mathcal{H}$ and $\mathcal{L}$.

Define $\mathcal{B}(\tilde{h}, \mathsf{pk}, 1^S)$:

  - **Wait** for the message α;
  - Compute $\mathcal{A}(\tilde{h}, \mathsf{pk}, \alpha; \rho_1)$ replying the RO query of $\mathcal{A}$ respectively using $\mathcal{H}$ or $\mathcal{L}$ when needed;
    until she makes the m-th RO query to $\mathcal{L}$, let Q_m be such query;
  - **Pause** the machine $\mathcal{A}$;
  - **if** $Q_m \in \mathcal{Q}'$ **abort**.
  - **Send** Q_m to $\Sigma_4.\mathsf{V}$;

  - **Wait** for the message $\boldsymbol{v}$ from $\Sigma_4.\mathsf{V}$;
  - choose ρ_2 uniformly at random;
  - **Resume** $\mathcal{A}$ reply the query Q_m with $\boldsymbol{v}$ and use as randomness ρ_2;
  - **If** $\mathcal{A}$ outputs the empty string **send** the empty string **to** $\Sigma_4.\mathsf{V}$;
    else let (C, Π) be the output of $\mathcal{A}$;
  - **if** $C = Q_m$ **then send** Π **to** $\Sigma_4.\mathsf{V}$
    else send the empty string **to** $\Sigma_4.\mathsf{V}$.

First we show that the value Q_m has high min-entropy. Since $|\mathcal{Q}'| \in \mathsf{poly}(k, S)$ this implies, with overwhelming probability, that $\mathcal{B}$ won't abort.

By the equation (5), there exists a $\mathcal{H}_\alpha$-labeling ℓ with $m = O(N/\ln k)$ faults such that $Q_m = \mathsf{MT}_s(\ell)$. Let $\mathcal{R}'$ be the set of RO replies made by $\mathcal{H}$ and $\mathcal{L}$ to the RO queries of $\mathcal{A}'$. As α is uniformly random in $\{0, 1\}^k$ then

$$\widetilde{\mathbf{H}}_\infty(\ell \mid \mathcal{R}', \alpha) \geqslant (|V(G_{N,k})| - m - 1) \cdot k.$$

In fact only with negligible probability in k the set $\mathcal{Q}'$ contains queries to $\mathcal{H}$ with prefix α. The min-entropy of Q_m is $\omega(\log k)$ because MT is collision resistant. The probability that Q_m is in $\mathcal{Q}'$ is then negligible.

When $\mathcal{B}$ doesn't abort the simulation of the protocol Σ_2 is perfect. Also notice that $\mathcal{B}$ needs space $S' = S + O(\log(\mathsf{poly}(k)))$ where S is from the emulations of $\mathcal{A}$ and $O(\log \mathsf{poly}(k))$ for the auxiliary indexes. Moreover if the event in equation (5) holds then $\mathcal{B}$ won't send to $\Sigma_4.\mathsf{V}$ the empty string.

Hence we proved that the above adversary $\mathcal{B}$ accessing the hint function $\tilde{h}$ and using space S' for k large enough is such that:

$$\Pr \left[\mathbf{Snd}_{\Sigma_4, S'}^{\mathcal{B}, \mathcal{B}'}(k) \right] \geqslant k^{-c} \cdot q(k)^{-1} - \mathsf{negl}(k).$$

Hence we obtain the contradiction that $\mathcal{B}$ is a winning adversary for the protocol Σ_4 we already proved is a POSPACE (Theorem 2). $\qquad\square$

E Σ_3 is a Weak Proof of Space Protocol (Theorem 4)

The structure of the proof is the same of the one for Theorem 2. We give a particular case of the PoK Lemma (Lemma 3) where the hint h is the empty string and $\tilde{\alpha}$ is 0^k. For sake of clarity we restate the PoK Lemma for this particular setting.

Lemma 4. *Given the protocol $\Sigma_3 = (\mathsf{Gen}, \mathsf{V}, \mathsf{P})$, a* PPT *adversary $\mathcal{A}$, a space parameter S and a security parameter k such that $\mathsf{pk} \leftarrow \mathsf{Gen}(1^k)$. If $\Pr\left[\mathbf{wSnd}_{\Sigma_3,S}^{\mathcal{A}}(k)\right]$ is noticeable, then there exists a randomness $\tilde{\rho}$ such that $\mathcal{A}$ fed with the randomness $\tilde{\rho}$ during the round (1) of the protocol Σ_3 calculate a $\mathcal{H}_{0^k}$-labeling ℓ of $G_{N,k}$ with m faults such that the following points hold:*

– *$m = O\left(\frac{N}{\ln k}\right)$ and*
– *for any well-labeled node u in ℓ we have $(\tilde{\alpha}\|u\|\ell(u_1)\|\ldots\|\ell(u_d)) \in \mathcal{Q}$,*

where $\mathcal{Q}$ is the set of queries to $\mathcal{H}$ made by $\mathcal{A}$ when fed with randomness $\tilde{\rho}$ until the end of round (1) and $\{u_1, \ldots, u_d\} = \Gamma^-(u)$.

By contradiction, suppose there exist an adversary $\mathcal{A}$ for the protocol Σ_3.

By applying Lemma 4 we extract a partial $\mathcal{H}_{0^k}$-labeling, ensure that $\mathcal{A}$ computed that labeling during round (1). Hence satisfying the hypothesis of Lemma 1.

This gives us, with overwhelming probability, a pebbling sequence $\mathcal{P}$ of $G_{N,k}$ with m wildcards having pebbling space $S' = \frac{S}{k} + d$, $m = O(\frac{N}{\ln k})$ and the pebbling time of $\mathcal{P}$ is a lower bound for the execution time of $\mathcal{A}$ before round (2). From now on, the proof follows exactly the same line of the proof of Theorem 2. $\qquad\square$