Software-Based Memory Erasure with relaxed isolation requirements

Sergiu Bursuc* Reynaldo Gil-Pons* Sjouke Mauw* Rolando Trujillo-Rasua†
*University of Luxembourg †Rovira i Virgili University
{sergiu.bursuc,reynaldo.gilpons,sjouke.mauw}@uni.lu rolando.trujillo@urv.cat

***Abstract*—A Proof of Secure Erasure (PoSE) is a communication protocol where a verifier seeks evidence that a prover has erased the memory on a given device within the time frame of the protocol execution. Designers of PoSE protocols have long been aware that, if a prover can outsource the computation of the memory erasure proof to another device, then their protocols are trivially defeated. As a result, most software-based PoSE protocols in the literature assume that provers are isolated during the protocol execution, that is, provers cannot receive help from a network adversary. Our main contribution is to show that this assumption is not necessary. We introduce formal models for PoSE protocols playing against provers aided by external conspirators and develop two PoSE protocols that we prove secure in this context. We reduce the requirement of isolation to the more realistic requirement that the communication with the external conspirator is relatively slow. Software-based protocols with such relaxed isolation assumptions are especially pertinent for low-end devices, where it is too costly to deploy sophisticated protection methods.**

***Index Terms*—security protocols, formal verification, memory erasure, distant attacker**

I. Introduction

Internet of Things (IoT) devices are specially vulnerable to malware infection due to their ubiquity, connectivity and limited computational resources [32]. Once infected, an IoT device becomes both a victim and a useful weapon to launch further attacks on more advanced infrastructure and services. Detecting whether an IoT device is infected with malware is thus essential to maintaining a secure computer network. The challenge for the defender is operating within the resource constraints of IoT devices, which make them ill-suited for active security defences and health monitoring [31].

A pragmatic approach to ensure the absence of malware is secure erasure, consisting of putting a device back into a clean state by wiping out its memory. This approach was first introduced in [34] as a prerequisite for secure software update. Although memory erasure can be achieved via direct hardware manipulation, here we are interested in Secure Erasure protocols (PoSE), whereby a verifier instructs a resource-constrained device, called the prover, to erase its memory and to prove that it indeed has done so. Notice that a PoSE protocol is not a data erasure tool [25], despite both being aimed at erasing memory. The latter is meant to erase sensitive data from memory in an irreversible manner, i.e., in a way that is unrecoverable by advanced forensics techniques. The former is a lightweight communication protocol whereby a verifier attests whether a (possibly) compromised prover has filled its memory with random data.

Because provers are potentially infected with malware, PoSE protocols in general cannot rely on cryptographic secrets stored in the prover. The exception are protocols that rely on secure hardware [9], such as a Trusted Platform Module. Not all devices are manufactured with secure hardware, though. Examples are those designed for low price and low energy consumption. Hence, in this article, we don't assume secure hardware on provers and we seek a software-based solution.

In the absence of cryptographic secrets, software-based PoSE protocols have historically relied on the assumption that provers are isolated during the protocol execution, that is, provers cannot receive external help. The isolation assumption has been deemed necessary so far to prevent a malicious prover from outsourcing the erasure proof to another device. Ensuring isolation is cumbersome, though. It requires closing all communication channels between provers and potential conspirators, for example, by jamming or using a Faraday cage.

The goal of this work is to reduce the requirement of isolation to the more realistic requirement that the external conspirator (which we call a *distant attacker*) is further away from the verifier than the prover (see Figure 1 for an illustration). That is, we do not restrict the communication capabilities of the (possibly corrupt) prover with the external adversary, but their position relative to each other. In practice, this can be accomplished relying on round-trip-time measurements, to ensure that the responses received to a sequence of challenges come from the device whose memory we aim to erase. Such measurements are used in distance-bounding protocols, whose goal is to ensure the authenticity of communication with a nearby device [24]. For ensuring our distant-attacker assumption, we can therefore build upon the principles of design in distance-bounding protocols, of which there exist already various

Reynaldo Gil-Pons was supported by the Luxembourg National Research Fund, Luxembourg, under the grant AFR-PhD-14565947. Rolando Trujillo-Rasua was funded by a Ramon y Cajal grant from the Spanish Ministry of Science and Innovation and the European Union (REF: RYC2020-028954-I).

proof-of-concepts and real-life implementations. While the messages exchanged in classic distance-bounding protocols typically consist of single bits [37], recent progress in the communication infrastructure allows fast exchange of longer messages. This was proved feasible in [14], assuming that attackers are using only commercial off-the shelf hardware. Furthermore, the relay resistant protection mechanism in the EMV protocol [22] and later improvements [36] measure the round-trip-time of 32-bit packets in order to bound distance. In the context of electric vehicle charging systems, [17] also considers a bigger than binary alphabet for their messages. In our proposed protocols, we will exploit this feature in order to challenge random blocks of the device's memory, rather than bits.

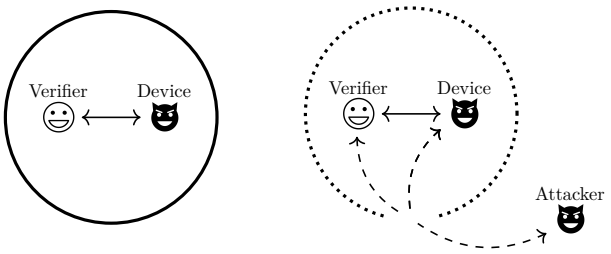

Figure 1: The isolation assumption (on the left) assumes no interference in the prover-verifier communication. The distant attacker assumption (on the right) lets the attacker interfere from far away.

Overview and contributions. To demonstrate that secure memory erasure is possible without assuming isolation, we introduce and formalize a class of PoSE protocols that employs a distance-bounding mechanism, which we call PoSE-DB, and put forward two protocols in this class together with formal security proofs against a corrupt prover that communicates with an external attacker. PoSE-DB protocols will have an interactive phase consisting of several challenge-response rounds. For each round, they will measure the round-trip-time between the challenge and response, and verify that the response is correct, aiming for two security goals. First, the prover cannot relay its messages to the distant attacker without failing the distance-bounding check, compelling the prover to compute the responses locally. Second, if the prover locally computes the responses to the verifier's challenges, then it must have erased its memory even if aided by an external attacker. The first contribution of this work is a formalization of those security requirements (Section III).

Our second contribution is the development of two PoSE-DB protocols with formal security proofs. The first protocol (Section V) is based on a straightforward idea, already used in the first secure erasure protocol proposed by Perito and Tsudik [34]: the verifier sends a random sequence of bits to the prover, who should store it in its memory; then the verifier queries random memory blocks to check that they are stored. We adapt this idea into a PoSE-DB protocol, where we query one random block per round, checking that the reply is correct and is received within the specified time bound. Our main contribution in this case is a formal security proof without resorting to the device isolation assumption. Although the protocol is simple, the proof is not trivial, and we introduce general proof methods in Section IV that help the analysis by allowing to focus on a single challenge-response round.

Sending a sequence of bits as large as the prover's memory may be undesirable in some contexts, e.g. when the communication channels have limited capacity. Instead, in our second protocol (Section VI), the verifier sends a small seed to the prover, which uses it to compute the labelling of a suitably chosen graph and store the labels of some nodes. In the interactive phase, we run several fast rounds to verify that those labels are stored. We identify a depth-robustness property for the graph that ensures the security of our proposed protocol: we show that a cheating prover needs to spend significant time to recompute any missing labels, so it will be caught by our time measurement.

Although graph-labelling techniques have been used in related works, e.g. for memory hardness [5, 6], proofs of space [21, 11, 35], and classic secure erasure [20, 27], none of them can be directly applied in our context. This is because we want to erase the full memory of the device, and we consider a stronger attacker model. In all the previous protocols, the gap between the memory filled by the prover and the full memory of the device is too high, either because they consider a different case study where this is not a concern (e.g. for proofs of space), or because the class of graphs is not strong enough (e.g. the proof in [27] can only erase $\frac{1}{32}$ of the full memory because of this reason). Furthermore, the protocols that directly target memory erasure, i.e. [20, 27], are also not adequate for our attacker model; they either assume protected memory [20] or the isolation assumption [27]. Indeed, [20] assumes at least part of a secret key not to be leaked, while in our model we allow full compromise of the memory content. In the protocol from [27], the prover needs to compute the graph labelling on the fly after being challenged to return a label, which is unsuitable for distance bounding.

We therefore propose a new class of graphs that allows our protocol to proceed in two phases, first compute and store the labels, and then reply to several rounds of challenges, thus proving the labels are stored. We prove that the resulting protocol can securely erase all but a small proportion of the prover's memory. Furthermore, our protocol provides its security guarantees in the presence of a distant attacker, without resorting to device isolation or protected memory assumptions. The undirected graph we propose (Section VII) satisfies a classic property of depth-robustness and can be computed in-place, i.e. by using only a constant memory overhead for the prover. This tightness constraint for the graph is especially important if we want to erase the full memory of the device. Our graph is tighter in this respect than other depth-robust graphs used recently in the literature [27, 21, 5, 35], and we believe this makes the graph of interest in other domains too, such

as proof of space and memory hardness.

II. Related Work

We review memory erasure and attestation protocols for resource-constrained devices that do not use secure hardware. See [30] for a review of protocols that use it. We will see that the main limitation of current software-based protocols, as highlighted in a recent survey [9], is the assumption that there is a secure communication channel between verifier and prover, called the *isolation assumption*.
Software-based memory isolation. Some protocols use software-based memory isolation techniques that continuously monitors the programs running on the device. In this class, SPEED[8] and SIMPLE[7] perform memory erasure and attestation, respectively. These protocols could, in theory, achieve our security goals. In practice, they add a significant overhead to any program running on the platform, as the software TPM continuously monitors the platform to ensure memory isolation at all times. Since memory erasure can start from any state of the device, we do not need to impose constraints on programs running on it. SPEED uses distance-bounding to ensure that only a nearby verifier can start the memory erasure procedure. We use distance-bounding in the opposite direction, where the verifier is the one aiming to ensure the prover is near.
Time-optimal memory erasure routine. SWATT is an attestation protocol [39] based on computing a checksum function over the memory of the prover within a prescribed amount of time. The idea is that, had the prover's memory been compromised, the prover would take noticeably longer to generate the correct attestation proof. Several attacks on SWATT were shown in [16], the main problem being that the checksum function and the timing constraints are rather ad hoc. This problem was later treated in [10], which proposes a formal model of software-based attestation and a generic protocol similar to SWATT, but, this time with a formal security proof. The problem here is that, if the local device is not isolated, the attacker could compute the response on a separate device and return it in time. Another problem is that the global running time of the erasure procedure is too high in order to detect small changes to the memory. The attacker could use the available time to optimize procedures and recompute the data that is missing. That is why [10] suggests the idea of a multi-round protocol, requiring shorter computation time in each round. This is an idea we develop further in this paper as part of a distance-bounding mechanism.

The first protocol for secure memory erasure based on cryptographic techniques is introduced in [34]. The paper proposes a protocol based on computing a (time-optimal) MAC over randomly generated data that fills the memory of the prover, along with an informal adversarial model and argument of security. This protocol uses a high bandwidth to transmit the random data and cannot prevent the computation to be delegated to an external attacker without isolating the device, like in [39, 10].

Graph pebbling games. A recent series of papers use graph pebbling techniques to achieve time-memory trade-offs, e.g. in proofs of secure erasure [20, 27], proofs of space, or memory-hardness results. The works aiming for memory erasure cannot be applied in our context, as discussed in the introduction. We should mention, in addition, that [27] also proposes an erasure protocol that is not based on graphs, but on hard-to-invert hash functions. In addition to assuming device isolation, the security of this protocol is proved assuming that, in the memory challenge phase, the adversary cannot query a hash function. We are not aware how this could be enforced in practice.
Distance-bounding techniques. A PoSE protocol that relies on distance bounding in a spirit similar to ours was proposed in [40]. It uses a cyclic tree automaton that occupies the full prover's memory. To obtain the erasure proof, the verifier asks the prover to transition into a new state of the automaton, based on random input chosen by the verifier, and to transmit the label of the new state. This round of exchanges is timed by the verifier to bound its distance to the prover. However, we observe that the fact that the response comes fast in each round is not sufficient to counter pre-computation attacks or an adaptive attacker. For example, depending on previous challenges, the adversary can get rid of states in the automaton that are unreachable by future verifier's challenges. This attack was overlooked because the protocol was analysed within a symbolic model, rather than a computational model as we do in this paper.

As summarized in Figure 2, there does not exist in the literature a software-based memory erasure/attestation protocol whose security can be formally proven without relying on the isolation assumption. Section V and Section VI in this article introduce the first two protocols of this type.

Protocol	No Isol.	No Hw.	Comm.
[39]	✗	✓	$\mathcal{O}(1)$
[28]	✓	✗	$\mathcal{O}(1)$
[34]	✗	✓	$\mathcal{O}(n)$
[33]	✓	✗	$\mathcal{O}(1)$
[11]	✗	✓	$\mathcal{O}(1)$
[27]	✗	✓	$\mathcal{O}(1)$
[26]	✗	✓	$\mathcal{O}(n)$
V	✓	✓	$\mathcal{O}(n) + \mathcal{O}(r)$
VI	✓	✓	$\mathcal{O}(1) + \mathcal{O}(r)$

Figure 2: Comparison of erasure and attestation protocols in terms of their communication complexity (Comm) and their capacity to *avoid* the use of the device isolation assumption (No Isol.) and specialised hardware (No Hw.). The value of n refers to the memory size and r is a security parameter that refers to the number of rounds during the fast phase (see Section III).

Formal security definitions. We briefly review existing security definitions related to our problem to motivate

the need for a new definition in the context of the distant attacker. The definition of memory attestation in [10] considers an experiment where a computationally unbounded adversary $\mathcal{A}$ produces a computationally bounded prover $\mathcal{P}$ which represents the state of the device to be attested. The protocol is run between a verifier $\mathcal{V}$ and $\mathcal{P}$, where $\mathcal{V}$ aims to ensure that $\mathcal{P}$ is in a desired state; it is deemed secure if a cheating prover would be caught with high probability. This model has built-in the device isolation requirement, since the local prover $\mathcal{P}$ does not interact with $\mathcal{A}$ throughout the protocol. Since we are interested in memory erasure and not attestation, in addition to strengthening the attacker model, we may also relax the requirement on the state of the prover, requiring only that it should utilize enough memory at some point in the protocol. The definition of secure memory erasure in [27] makes this relaxation on the prover state. However, as in [10], the prover $\mathcal{P}$ is assumed computationally bounded, which amounts to the isolation assumption. In order to lift this assumption, it is not sufficient to simply make $\mathcal{P}$ computationally unbounded, but we need to consider two adversaries $\mathcal{A}$ and $\mathcal{P}$ that can communicate throughout the protocol, and make $\mathcal{A}$'s memory unbounded.

A symbolic Dolev-Yao style model for secure erasure is proposed in [40]. Their main contribution is a formal proof of the necessity of the distant attacker assumption in some situations and a construction of a protocol that can be proven secure symbolically. However, the symbolic model in [40] cannot be used to quantify the size of the adversarial state, or the adversary's probability of success. Hence, it cannot be used to faithfully analyse existing PoSE protocols, which use information theoretical constructions rather than symbolic ones.

III. A Formal Model for PoSE-DB protocols

In this section, we introduce a class of memory erasure protocols that aim to resist collusion between a corrupt prover and a distant attacker, and a formal model that allows to prove their security. We call this class of protocols *Proofs of Secure Erasure with Distance Bounding* (PoSE-DB). We denote by $[n]$ the set $\{1, \ldots, n\}$, by $a \leftarrow\!\!\$\ A$ the uniformly random sampling of a from the set A and by $o \leftarrow F(x, \ldots)$ the output of an algorithm F running on given inputs. For two bitstrings $a, b \in \{0,1\}^*$, we denote by $a\|b$ their concatenation. An adversary is a probabilistic Turing machine, generally denoted by $\mathcal{A}$. It may be endowed with oracles for restricted access to some resources. In Section VI-A, we will use the random oracle methodology to model and reason about the access of the adversary to a hash function [12]. An adversary with access to a (possibly empty) list of oracles $\mathcal{O}$ is denoted by $\mathcal{A}^{\mathcal{O}}$.

A. Proof of Secure Erasure with Distance Bounding

The key feature of a PoSE-DB protocol is the use of a distance-bounding mechanism over several challenge-response rounds to prevent the prover from outsourcing the erasure proof to the distant attacker. Figure 3 depicts the generic scheme that we consider for a PoSE-DB protocol. We consider the following protocol parameters as global constants: block size (w), size of memory in blocks (m), number of rounds (r) and time threshold (Δ). The size of the memory to be erased is therefore $m \cdot w$. The time threshold is chosen at deployment so that the distant attacker assumption holds within round-trip-time bounded by Δ. In a setup phase which happens once before the protocol sessions, the verifier is instantiated with certain additional parameters necessary to run the protocol: the space used to draw initialization parameters for each session ($\mathcal{I}$, which could be a set of bitstrings or hash functions) and some auxiliary data (ρ) common for all sessions. Each PoSE-DB session then runs in three phases:

Initialization phase: the local device (playing the role of the prover) has to perform a prescribed sequence of computation steps and store its result σ in its internal memory. Such a value is meant to fill the prover's memory, leaving no room for data previously stored in the device.

Interactive phase: verifier and prover interact over a number of challenge/response rounds; the verifier measures the round-trip-time of those exchanges and stores all the challenges and their corresponding responses.

Verification phase: the verifier accepts the proof if all challenge/response pairs from the interactive phase satisfy a prescribed verification test, and if the round-trip-times are below the time threshold Δ.

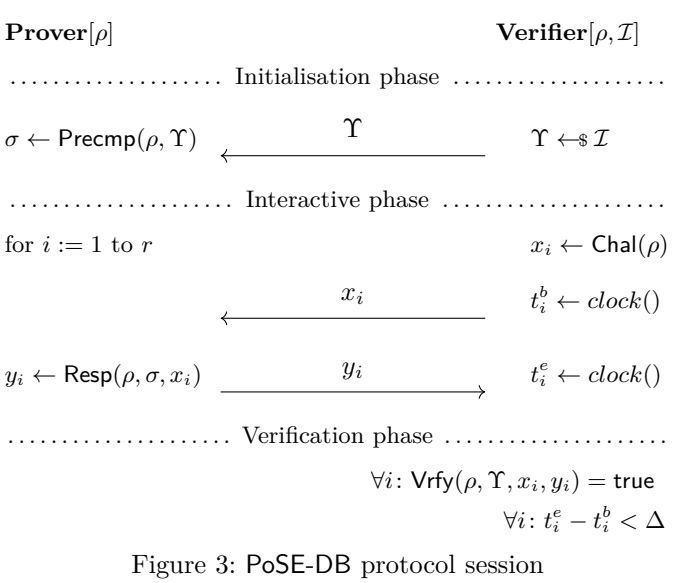

Figure 3: PoSE-DB protocol session

Notice that there are no identities exchanged during the protocol, nor pre-shared cryptographic material. Like in existing software-based memory attestation and erasure protocols, we assume the existence of an out-of-the-band authentication channel, such as visual inspection, that allows the verifier to identify the prover. In Definition 1, we formally specify PoSE-DB protocols as a set of algorithms to be executed by the prover and the verifier. We do not specify the way in which messages are exchanged or the

time verification step. These are handled by the security definition as described below, considering a Dolev-Yao model with a distant attacker that cannot act within the challenge-response round.

Definition 1. A *proof of secure erasure with distance bounding* (PoSE-DB) protocol is defined by a tuple of algorithms $(\mathsf{Setup}, \mathsf{Precmp}, \mathsf{Chal}, \mathsf{Resp}, \mathsf{Vrfy})$ and parameters (m, w, r, Δ) as illustrated in Figure 3. We have:

- $(\rho, \mathcal{I}) \leftarrow \mathsf{Setup}(m, w)$: computes some data ρ necessary to run the protocol and a parameter space $\mathcal{I}$ to be used for instantiating protocol sessions;
- $\Upsilon \leftarrow\$ \mathcal{I}$: sample data uniformly from $\mathcal{I}$ for each protocol session; some protocols may instantiate a hash function at this step;
- $\sigma \leftarrow \mathsf{Precmp}(\rho, \Upsilon)$: computes a value of size $m \cdot w$, to be stored in memory;
- $x \leftarrow \mathsf{Chal}(\rho)$: generates a uniformly random challenge;
- $y \leftarrow \mathsf{Resp}(\rho, \sigma, x)$: computes the response to the challenge. This should be a very lightweight operation consistent with the design principles of distance bounding, such as a lookup operation.
- $\mathsf{Vrfy}(\rho, \Upsilon, x, y)$: determines if y is the correct response to challenge x

Example 1. As a running example, consider the simple idea (similar to Perito and Tzudik's protocol [34]) of filling the memory of the device with random data, then challenging it to return randomly chosen blocks of that data during the interactive phase. We call this the *unconditional PoSE-DB protocol*, as its security proof in Section V does not rely on cryptographic assumptions. For such a protocol, the parameter space $\mathcal{I}$ is set to $\{0,1\}^{m \cdot w}$. This means that every protocol session will start with a random sequence of length $\{0,1\}^{m \cdot w}$, which is equal to the memory size of the prover. The specification of this protocol is as follows.

Definition 2 (The unconditional PoSE-DB protocol)**.**

- $\mathsf{Setup}(m, w)$: return $\rho = \emptyset,\ \mathcal{I} = \{0,1\}^{m \cdot w}$
- $\psi \leftarrow\$ \{0,1\}^{m \cdot w}$
- $\mathsf{Precmp}(\rho, \psi)$: parse ψ as $t_1 \| \ldots \| t_m$ with $t_i \in \{0,1\}^w$, return $\sigma = t_1 \| \ldots \| t_m$
- $\mathsf{Chal}(\rho)$: return $x \leftarrow\$ [m]$
- $\mathsf{Resp}(\rho, \sigma, x)$: return t_x, which was stored in σ
- $\mathsf{Vrfy}(\rho, \psi, x, y)$: return true iff $y = t_x$

Note that the erasure procedure itself running on the local device cannot be overwritten by σ. Hence, in practice, the device should allocate memory to store and execute the erasure procedure, and the goal should be for this procedure to introduce minimal memory overhead. This is a necessary condition in any memory erasure protocol. The erasure procedure for the unconditionally secure protocol consists in simply storing and fetching blocks from the memory. We expect its memory overhead to be minimal. For the graph-based protocol that we introduce in Section VI, the in-place property we devise for the graph in Section VII is aimed at keeping this overhead as small as possible.

B. Formalizing Secure Erasure Against Distant Attackers

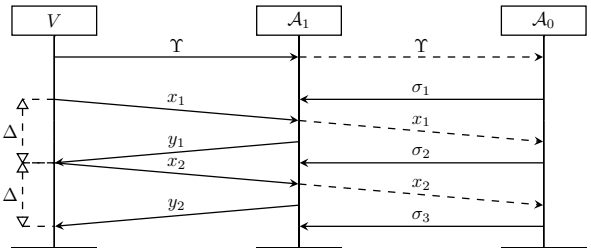

Figure 4: The y-axis denotes a timeline. Because A_0 is far, A_1 cannot relay the verifier's challenge to A_0 and wait for a response. A_0 does help A_1 in the other phases of the protocol execution.

To define secure erasure, we formally split the adversary in two: we use $\mathcal{A}_0$ to denote the distant attacker, and $\mathcal{A}_1$ to denote the local (possibly corrupt) device. Like in a Dolev-Yao model, our adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ is in full control of the network and can corrupt agents. It can eavesdrop, inject and modify messages sent to the network. One limitation for the pair of attackers is given by the physical constraints of the communication medium, which are leveraged by the distance-bounding mechanism, and the assumption that $\mathcal{A}_0$ is distant, i.e. sufficiently far from the device. In our security definition we abstract away from the distance-bounding check by not letting $\mathcal{A}_0$ act between the sending of the challenge and the receipt of the corresponding response in each round. The intuition of this abstraction is illustrated in Figure 4: because $\mathcal{A}_0$ is far and the function Resp should be computed fast, $\mathcal{A}_0$ does not have time to respond to the verifier's challenge in time. We do allow $\mathcal{A}_0$, before each round of the fast phase, to precompute some state σ_i to be used by $\mathcal{A}_1$ during the ith round of the fast phase. Assuming that σ_i is computed by $\mathcal{A}_0$ is without loss of generality, since $\mathcal{A}_0$ is unbounded and has at least as much knowledge as $\mathcal{A}_1$, which forwards all information to $\mathcal{A}_0$. Therefore, we assume that right before the challenge x_i, the attacker's available memory on the device is filled by σ_i and this is the only information that $\mathcal{A}_1$ can use to compute the response in the ith round of the fast phase.

$(\rho, \mathcal{I}) \leftarrow \mathsf{Setup}(m, w);\ \ \Upsilon \leftarrow\$ \mathcal{I}$
for $i := 1$ to r **do** :
  $\sigma_i \leftarrow \mathcal{A}_0(1^w, \rho, \Upsilon, \{x_j | j < i\})$
  $x_i \leftarrow \mathsf{Chal}(\rho);\ \ y_i \leftarrow \mathcal{A}_1^{\mathcal{O}}(1^w, \rho, \sigma_i, x_i)$
return $\forall i\colon \mathsf{Vrfy}(\rho, \Upsilon, x_i, y_i) = \mathsf{true}$

Figure 5: Security experiment $\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1}$.

Figure 5 formalizes the environment described above in the form of a security experiment. As $\mathcal{A}_1$ is acting within the fast phase, we may bound its number of computation steps. This is especially interesting for protocols that may rely on computational security assumptions. The only computational assumption that we will use in this paper is the random oracle assumption for one of our two protocols.

Therefore, in the security definition, if $\mathcal{A}_1$ has access to a hash function, we will restrict its use through an oracle and will count the number of calls $\mathcal{A}_1$ makes to it. This restriction is formalized by the parameter q in Definition 3 that bounds resources for an adversary. A second parameter in this definition is M, which bounds the size of the memory used by $\mathcal{A}$ on the device throughout the protocol, i.e. the maximum value of σ_i in the security experiment. We consider an attacker successful if it passes the protocol while not erasing a significant proportion of the $m \cdot w$ bits of memory on the device. Assume the portion of memory that is needed by the adversary to store malware or any other information is y. If the adversary needs to use more than $m \cdot w - y$ bits to successfully execute the protocol, then the device becomes "clean", as the memory required by the attacker is erased. Therefore, the goal of the attacker is to use at most $M = m \cdot w - y$ bits of space during the protocol execution.

Definition 3. An adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ against the memory challenge game is called (M, q)-*bounded* iff in any execution of $\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1}$ and any round i we have that $|\sigma_i| \leq M$ and $\mathcal{A}_1$ makes at most q queries to $\mathcal{O}$.

To evaluate the security of a given PoSE-DB protocol, we will consider the class of (M, q)-bounded adversaries and determine the probability of any adversary from this class to win the security game.

Definition 4 (PoSE-DB security)**.** Assume some fixed parameters (m, r, w) for the experiment from Figure 5. An adversary $(\mathcal{A}_0, \mathcal{A}_1)$ *wins the memory-challenge game* with probability ζ iff $\Pr\left[\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1} = \text{true}\right] = \zeta$, where probability is taken over the randomness used by the experiment.

There are similarities between our security definition presented above and the definition of proof of space [21, 35], the main difference being that in the latter $\mathcal{A}_1$ is isolated from $\mathcal{A}_0$ in the interactive challenge phase. We discuss more details about this relation and our modelling choices in the extended version [15].

IV. Initial results

Before specifying and analysing our two PoSE-DB protocols, we provide useful initial results on the security of PoSE-DB protocols in general. Concretely, we show that PoSE-DB security increases proportionally to the number of rounds. This will allow us to simplify the security analysis by proving a level of security for the protocol executed in a single round, and then generically deriving the corresponding security guarantees over multiple rounds. Throughout all our proofs we consider deterministic adversaries $\mathcal{A}$. This is without loss of generality: since our security definition upper-bounds the success probability of $\mathcal{A}$, we can always consider that $\mathcal{A}$ is initialized with its best random tape. In this setting, by fixing its best-case random-tape, we can consider $\mathcal{A}$ deterministic. This idea is well known and has been applied elsewhere [41]. Note that we still have randomness left in the security experiment, coming from probabilistic choices in honest algorithms and, when applied, from the random oracle.

Let Υ be an element from the initialization space $\mathcal{I}$ of a PoSE-DB protocol. For an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, denote by $\Pr_{\Upsilon \leftarrow\$ \mathcal{I}}[\mathcal{A}^r]$ the probability that $\mathcal{A}$ wins the memory game with r rounds. For a fixed Υ, we denote the corresponding winning conditional probability by $\Pr[\mathcal{A}^r \mid \Upsilon]$. We say that an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ playing the memory challenge game with fixed $\Upsilon \in \mathcal{I}$ is *uniform* if, for any sequence of challenges $x_1, \ldots, x_r$, $\mathcal{A}_0$ returns the same state σ_i in each round, i.e. $\sigma_1 = \cdots = \sigma_r$. The following lemma allows us to focus on uniform adversaries when proving the security of a PoSE-DB protocol. The main idea of the proof is that, since the challenge in each round is chosen independently, the best that $\mathcal{A}_0$ can do in any round is to choose a state σ_μ that maximizes the success probability of $\mathcal{A}_1$ for a random challenge. The proof of this lemma is in the appendix. All results we state without proof in the rest of this work have proofs in the extended version [15].

Lemma 1. For any (M, q)-bounded adversary $\mathcal{A}$ against the PoSE-DB security experiment, there is a uniform (M, q)-bounded adversary $\bar{\mathcal{A}}$ that wins the experiment with at least the same probability: $\Pr_{\Upsilon \leftarrow\$ \mathcal{I}}[\mathcal{A}^r] \leq \Pr_{\Upsilon \leftarrow\$ \mathcal{I}}[\bar{\mathcal{A}}^r]$.

Next, we show that there is a direct relationship between the winning probability of an adversary in one round and the winning probability in multiple rounds. The analysis is simplified by the fact that we can focus on uniform adversaries, according to Lemma 1. If the initialization parameters are fixed, it follows immediately that running the experiment for r rounds exponentially decreases the cheating ability of the adversary.

Lemma 2. For any uniform adversary $\mathcal{A}$ and any $\Upsilon \in \mathcal{I}$, we have $\Pr[\mathcal{A}^r \mid \Upsilon] = \Pr\left[\mathcal{A}^1 \mid \Upsilon\right]^r$.

When lifting this lemma to uniformly chosen parameters from $\mathcal{I}$, it will be necessary to account for the fact that the (one-round) adversary may be lucky on a small proportion of those random choices. The following definition and proposition tolerate this, by bounding the success probability of the adversary only for a subset of good parameters, and showing the effect after r rounds.

Definition 5. For a set $\mathcal{I}_{\mathsf{good}} \subseteq \mathcal{I}$ and a value $\zeta < 1$, we say that $\mathcal{A}$'s *winning probability is bounded by* ζ *within* $\mathcal{I}_{\mathsf{good}}$ iff $\forall \Upsilon \in \mathcal{I}_{\mathsf{good}} \colon \Pr\left[\mathcal{A}^1 \mid \Upsilon\right] \leq \zeta$.

If the proportion of cases $\mathcal{I} \setminus \mathcal{I}_{\mathsf{good}}$ in which the adversary is lucky is negligible, then for a large enough number of rounds the success probability of the adversary is also negligible, as shown by the next proposition.

Proposition 1. Given $\zeta < 1$, $\mathcal{I}_{\mathsf{good}} \subseteq \mathcal{I}$ and a uniform adversary $\mathcal{A}$ whose winning probability is bounded to ζ within $\mathcal{I}_{\mathsf{good}}$, we have $\Pr_{\Upsilon \leftarrow\$ \mathcal{I}}[\mathcal{A}^r] \leq \zeta^r + \frac{|\mathcal{I} \setminus \mathcal{I}_{\mathsf{good}}|}{|\mathcal{I}|}$.

V. The unconditional PoSE-DB protocol

This section provides the security analysis for the unconditional PoSE-DB protocol presented in Section III, Definition 2. Since this protocol does not use any hash function, the bound on the number of oracle calls by $\mathcal{A}_1$ is not relevant. Such an adversary is (M, ∞)-bounded. As a consequence, there is no computational bound on the adversary $\mathcal{A}_1$, but only a memory bound. Recall that the value $y = m \cdot w - M$ is the amount of memory that the adversary cannot erase (where it may store malware).

Theorem 1. Assume that the unconditional PoSE-DB protocol is instantiated with parameters $(m, 1, w)$. Let $\mathcal{A}$ be any adversary with measure (M, ∞). Then, there exists a set $\mathcal{I}^1_{\mathsf{good}} \subseteq \mathcal{I}$ such that $\mathcal{A}$'s winning probability is bounded to $1 - m^{-1}$ within $\mathcal{I}^1_{\mathsf{good}}$ and $\left|\mathcal{I}^1_{\mathsf{good}}\right| \geq 2^{m \cdot w} \cdot (1 - 2^{-y})$. Furthermore, if $y \geq m + w$, then there exists a set $\mathcal{I}^2_{\mathsf{good}} \subseteq \mathcal{I}$ such that $\mathcal{A}$'s winning probability is bounded to $1 - \left\lceil \frac{y-m-w+1}{w} \right\rceil m^{-1}$ within $\mathcal{I}^2_{\mathsf{good}}$ and $\left|\mathcal{I}^2_{\mathsf{good}}\right| \geq 2^{m \cdot w} \cdot (1 - (m^2 + m) \cdot 2^{-w})$.

Proof. Let σ be a bitstring of size M. Let $R_\sigma = \{\psi_1, \dots \psi_k\}$ be the set of all bitstrings such that $\mathcal{A}_0(1^w, \rho, \psi_i) = \sigma$. We will lower-bound the number of queries related to elements in R_σ for which $\mathcal{A}_1$ gives an incorrect answer. To prove the first result, we count for how many bitstrings there is at least one error. Call this set $\mathcal{I}^1_{\mathsf{good}}$. Let ψ' be the concatenation of the blocks in $\{\mathcal{A}_1(1^w, \rho, \sigma, 1), \dots, \mathcal{A}_1(1^w, \rho, \sigma, m)\}$, i.e. a bitstring of size $m \cdot w$. Since ψ' can be equal to at most one string in R_σ, $\mathcal{A}_1$ is wrong for at least one input $q \in \{1, \dots, m\}$ for at least $|R_\sigma| - 1$ of the bitstrings in R_σ. Summing up for all possible sets R and for all possible σ, we deduce that $\mathcal{A}_1$ is wrong for at least one query w.r.t. at least $2^{m \cdot w} - 2^{m \cdot w - y}$ bitstrings (as there are at most $2^{m \cdot w - y}$ different sets R_σ). We obtain $\left|\mathcal{I}^1_{\mathsf{good}}\right| \geq 2^{m \cdot w} \cdot (1 - 2^{-y})$ and:

$$\forall r \in \mathcal{I}^1_{\mathsf{good}} \colon \Pr_{q \leftarrow\$ [n]}[\mathcal{A}_1(1^w, \rho, \sigma, q) \text{ is correct } | \, \sigma] \leq 1 - m^{-1}$$

where $\sigma \leftarrow \mathcal{A}_0(1^w, \rho, \psi)$. For a proof of the second result of the theorem, see the extended version in [15]. □

From Proposition 1 and Theorem 1, we obtain:

Corollary 1. If we execute the unconditional PoSE-DB protocol for r rounds in presence of any (M, ∞)-bounded adversary, then $\Pr\left[\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1} = \mathsf{true}\right] \leq (1 - m^{-1})^r + 2^{M - m \cdot w}$. Furthermore, when $M \leq m \cdot w - m - w$ the bound improves to $\left(1 - \left\lceil \frac{m \cdot w - m - w - M + 1}{w} \right\rceil \cdot m^{-1}\right)^r + m \cdot (m + 1) \cdot 2^{-w}$.

VI. PoSE based on depth-robust graphs

While the unconditional PoSE-DB protocol offers interesting security properties, it comes at the cost of a high communication complexity, since the verifier needs to transmit a nonce of length equal to the size of the prover's memory. This section offers an alternative protocol where the prover computes the labelling of a graph in its memory, starting from a small random seed transmitted by the verifier.

A. Graph-based notation and PoSE scheme

We first introduce the general notion of graph-labelling and discuss its use in the literature. We let H be the set of all functions from $\{0,1\}^\kappa$ to $\{0,1\}^w$, for a suitably large κ. Each new session of the protocol will rely on a hash function h instantiated randomly from this set. Elements from the set H are also called random oracles, since we will make the random oracle assumption for h drawn from this set. Given a directed acyclic graph (DAG) G, the list of successors and predecessors of the node v in G are respectively $\Gamma^+(v)$ and $\Gamma^-(v)$. If $\Gamma^-(v) = \emptyset$ then v is an input node. For a node v of G, we let $\mathsf{lip}(v, G)$ be the length of the longest path in G that ends in v. If $R \subseteq V(G)$, we denote by $G \setminus R$ the graph obtained after removing the nodes in R and keeping all edges between the remaining nodes.

Definition 6. Given a hash function h and a DAG G, $l \colon V(G) \to \{0,1\}^w$ is an h-labeling of G if and only if for each $v \in V(G)$:

$$l(v) \coloneqq h(v \| l(v_1) \| \dots \| l(v_d)) \text{ where } (v_1, \dots, v_d) = \Gamma^-(v)$$

If a node v has no predecessor, then $l(v) = h(v)$. We also let $\ell^-(v) = v || \ell(v_1) || \dots || \ell(v_d)$.

Graph labelling as defined above is a well established technique used in protocols for proofs of space [21, 11, 35], memory hardness [5, 6, 2, 3, 4] and classic secure erasure [20, 27]. The typical structure of these protocols is as follows:

- the verifier sends a nonce to the prover, which is used as a seed to determine the hash function h;
- the prover uses h to compute the labelling of an agreed-upon graph G, and stores a subset of the resulting labels, denoted $O(G)$;
- the verifier challenges the prover to reply with the labels of several randomly chosen vertices in the graph, and accepts the proof if these labels are correct.

Let m be the size of $O(G)$. The general goal of this technique is to ensure that any cheating prover that uses less than m words of memory to compute the responses can only do so correctly by paying a noticeable amount of computational time. This has been achieved by using depth-robust graphs [23, 35], which are graphs that contain *at least one long path*, even if a significant proportion of nodes have been removed. Intuitively, in the corresponding protocol, this means that the label for at least one of the verifier challenges will be hard to compute if the prover has cheated.

Given a DAG G_m, a hash function $h \in H$ (which we model as a random oracle) and a list of nodes $O(G_m) \subseteq V(G_m)$ of size m, we define our remote memory erasure protocol as follows:

- $\mathsf{Setup}(m, w)$: return $\rho = (G_m, O(G_m)),\ \mathcal{I} = H$
- $h \leftarrow\!\!\$\ H$
- $\mathsf{Precmp}(\rho, h)$: compute the labels of the nodes in $O(G_m)$, and output the concatenation of these labels $\sigma = l(o_1)\|\ldots\|l(o_m)$ where $(o_1, \ldots, o_m) = O(G_m)$
- $\mathsf{Chal}(\rho)$: return a random vertex in $O(G_m)$
- $\mathsf{Resp}(\rho, \sigma, x)$: responds with $l(x)$, which was stored in σ
- $\mathsf{Vrfy}(\rho, h, x, y)$: return true iff $y = l(x)$

In order to prove our graph based PoSE-DB secure, there are several issues we need to address. First, as we need fast responses, our verifier can query only one random challenge node per round. This means that it is not sufficient to have a single long path in the graph in order to catch a cheating prover; we will thus strengthen the depth-robustness property to require *at least a certain number of long paths* to be present. Another constraint, particularly relevant to memory erasure, is that we should be able to compute the graph labelling with minimal memory overhead, so that as much memory as possible can be erased from the device. We call this property *in-place*, since intuitively it means that the set of labels to be stored should be computed in almost the same amount of space as their total size. To our knowledge, no graph in the literature exists that satisfies these two properties. We design one in Section VII.

Definition 7. A graph G can be labelled **in-place** with respect to a list of nodes $O(G) \subseteq V(G)$ if and only if there is an algorithm that outputs the list of labels for all nodes in $O(G)$ using at most $|O(G)| \cdot w + \mathcal{O}(w)$ bits of memory.

A second issue that we address lies in the tightness of the security bound, i.e. how big is the gap between the memory erased by a cheating prover and the memory it is supposed to erase. In our case study, this gap could be used to store malware, so it should be as small as possible. We improve upon previous security bounds for protocols based on graph labelling, first by performing a fine-grained and formal security analysis, and second by identifying a restricted class of adversaries, that simplifies the proofs while also further improving the security bound. We relate this class to previous restrictions in this area and argue that it is a strictly more general notion, resulting in weaker restrictions for the adversary and therefore stronger security guarantees.

B. Depth-robustness is sufficient for security

We show that a variation of the classic graph depth-robustness property [23, 35] is sufficient to prove security for the PoSE scheme from Section VI-A. As explained above, while depth-robustness in all previous works requires the existence of one single long path after some nodes have been removed, we require the existence of several long paths. The computation required by a cheating prover in this context is proportional to the depth of the longest remaining path. While in most previous works the length is linear in the size of the graph, we will tolerate graphs with slightly shorter paths, i.e. of sublinear size. On the one hand, this is useful in practice, since it will allow us to construct a class of depth-robust graphs that can be labelled in-place efficiently. On the other hand, we show that this does not affect security, as long as the computational power of the adversary during the interactive phase is constrained in proportion to the prescribed path length, which can be done by the time measurements we described in Section III.

Definition 8. Let G be a DAG and let $O(G) \subseteq V(G)$ with $|O(G)| = \mu$. We say G is (μ, γ)-**depth-robust** with respect to $O(G)$ iff for every set $R \subset V(G)$ such that $|R| < \mu$, there exists a subset of nodes $O' \subseteq O(G)$ with $|O'| \geq \mu - |R|$ such that for every node $v \in O'$ there is a path in $G \setminus R$ of length at least γ that ends in v, i.e. $\mathsf{llp}(v, G \setminus R) \geq \gamma$.

The security bounds obtained using such a depth-robust graph in the graph-based PoSE scheme are presented in Corollary 2, which can be obtained by a direct application of Theorem 2 and Proposition 1. For this protocol, the oracle $\mathcal{O}$ in Figure 5 gives to $\mathcal{A}_1$ oracle access to the hash function h resulting from the initialization phase. The first part of Corollary 2 shows that we obtain better security bounds if we consider a notion of graph-restricted adversaries, which will be discussed in the next subsection. Note that the parameter m for the number of memory blocks in the experiment and the bound $q < \gamma$ on the number of oracle calls by $\mathcal{A}_1$ are related to the pair (m, γ) determined by the depth-robustness of the graph.

Corollary 2. Assume that the graph-based PoSE scheme is instantiated with parameters (m, r, w) and an (m, γ)-**depth-robust** graph G. Then, for any (M, q)-bounded adversary $(\mathcal{A}_0, \mathcal{A}_1)$, with $q < \gamma$, we have:

$$\Pr_h[\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1} = \mathsf{true}] \leq (M'/m)^r + 2^{-w_0} \text{ where:}$$

- if $\mathcal{A}$ is *graph-restricted*: $w_0 = w$ and $M' = \lceil M/w \rceil$
- else: $w_0 = w - \log(m) - \log(q)$ and $M' = \lceil M/w_0 \rceil$

Intuitively, the previous result shows that the attacker's success probability is proportional to M: the smaller the state it uses to reply to our challenges, the higher the probability that it will fail to pass the verification test. This result is not far from optimal, as this bound can actually be achieved by an adversary that stores $\frac{M}{w}$ of the labels in $O(G)$.

For example, if we would like to erase a malware of size at least 6 KB from a device with total memory 100 KB, then the parameters would be $w = 256, m = 100 \cdot 2^{13}/256 = 3200, M = (100 - 5) \cdot 2^{13} = 778240$. If we wanted to be certain that any graph-restricted attacker can pass the protocol without erasing malware with probability at most 10^{-3}, then we need to execute the protocol for 112 rounds.

C. Graph-restricted adversary

Several works studying protocols based on graph labelling related to ours consider restricted classes of ad-

versaries in order to obtain tight security bounds and reductions to graph pebbling [19, 21, 5]. In all these notions, the adversary can only make oracle calls corresponding to valid labels in the graph and, in addition, the adversary is restricted in the type of computation that it can apply to get a state to be stored in memory, e.g. it can only store labels [19, 21] or so-called entangles labels [5]. The notion that we consider in Definition 9 is more general, considering the full class of graph-restricted adversaries that can perform any computation to obtain the state to be stored.

Definition 9. We say that $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ is *graph-restricted to G* if and only if all oracle calls done by $\mathcal{A}_1$ correspond to valid labels (equal to $\ell^-(v)$ for some node v), and its responses to the verifier challenges are always correct.

A graph-restricted adversary may compute arbitrary functions (for example compression, cut labels into pieces, entangle them with new algorithms, etc), but it doesn't do any guessing while making oracle queries nor when responding challenges. The assumption is that the adversary knows what it is doing, i.e. knows that an oracle query would be useless or that a particular response to the challenge would be wrong. The notable difference with respect to the previous classes of graph-playing adversaries, e.g. the pebbling adversary from [21] or the entangled pebbling adversary from [5], is that we have no a priori restriction on how the labels of values are processed and stored. We discuss these notions in more detail in the extended version [15].

D. Security proof: first step

The following theorem will be the main ingredient for proving Corollary 2. We split its proof in two steps. The first step is presented in this subsection and is independent of the adversarial class. The second step is simpler for graph-restricted adversaries. We present the proof for that case in the next subsection and for the case of general adversaries in appendix. Our proof strategy builds upon the proofs from [5] and [35]. Relying on our new notion of depth-robustness, we combine ideas from both proofs, improve on their security bounds, and adapt them to graph-restricted adversaries and to the case where we only ask one challenge from the prover.

Theorem 2. Assume that the graph-based PoSE scheme is instantiated with parameters $(m, 1, w)$ and with a (m, γ)-**depth-robust** graph G. Let $\mathcal{A}$ be any (M, q)-*bounded* adversary, with $q < \gamma$. There exists a set of random oracles $H_{\mathsf{good}} \subseteq H$ such that $\mathcal{A}$'s winning probability is bounded by $\frac{M'}{m}$ within H_{good}, where:

- $M' = \lceil \frac{M}{w} \rceil$ and $|H_{\mathsf{good}}| \geq |H| \cdot (1 - 2^{-w})$ if $\mathcal{A}$ is graph-restricted
- else: $M' = \left\lceil \frac{M}{w_0} \right\rceil$ and $|H_{\mathsf{good}}| \geq |H| \cdot (1 - 2^{-w_0})$, where $w_0 = w - \log(m) - \log(q)$.

Consider an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ against the memory challenge game for our PoSE instantiated with a graph G. Let $\sigma = \mathcal{A}_0(1^w, \rho, h)$. Let $O(G) = \{o_1, \ldots, o_m\}$ be the set of all challenge vertices that can be given to $\mathcal{A}_1$ during the experiment. For this protocol the adversary $\mathcal{A}_1$ has access to the random oracle h through $\mathcal{O}$, which we make clear in the notation onwards by calling the oracle function $\mathcal{O}_h$. A query Q from $\mathcal{A}_1$ to $\mathcal{O}_h$ is good if $\exists v \colon Q = \ell^-(v)$. For every $i \in \{1, \ldots, m\}$, considering the execution of $\mathcal{A}_1^{\mathcal{O}_h}(1^w, \rho, \sigma, o_i)$, we let:

- $Q_{i,j}$ be the input to the j-th oracle call made to $\mathcal{O}_h$ by $\mathcal{A}_1$ in this execution;
- t_i be the total number of oracle calls made by $\mathcal{A}_1$ in this execution;
- Q_{i,t_i+1} be the output of $\mathcal{A}_1$ on this execution.

Definition 10 (Blue node)**.** We say that a node $v \in V(G)$ is blue if and only if there exists $v' \in V(G)$, $i \in \{1, \ldots, m\}$ and $j \geq 1$ such that:

1) $v \in \Gamma^-(v')$ and $Q_{i,j} = \ell^-(v')$
2) $\forall i' \in \{1, \ldots, m\} \forall j' < j.\ Q_{i',j'} \neq \ell^-(v)$

We say that v is a *blue node* from iteration j, and that $Q_{i,j}$ is the query associated to v. Denote by B_j all blue nodes in iteration j, and by B the set of all blue nodes. A bitstring is a *pre-label* if it is equal to $\ell^-(v)$ for some v, and a *possible pre-label* if it is the concatenation of a node v and $w \cdot |\Gamma^-(v)|$ bits.

The first point above implies $\ell^-(v') = v' \| y_1 \| \ldots \| y_m$ and $\ell(v) \in \{y_1, \ldots, y_m\}$. The second point will ensure that we can extract the labels of blue nodes for free, i.e. without querying them to the random oracle, by running $\mathcal{A}_1$ in parallel for all possible challenges on the state σ computed by $\mathcal{A}_0$. Let $T_i = t_i$ if the response to the challenge o_i by $\mathcal{A}_1$ is correct, or infinite otherwise. The strategy for the proof of Theorem 2 will be the following:

- *First step:* prove that the success probability of the adversary is smaller than the fraction between the number of blue nodes, whose label it has stored, and the total number of labels it is supposed to store.
- *Second step:* prove an upper bound on the number of blue nodes; for graph-restricted adversaries, this will be $\lceil \frac{M}{w} \rceil$, matching the value M' from Theorem 2.

First, we show that to answer a challenge correctly, A_1 needs to recompute the labels of the longest path to any node that is not blue:

Lemma 3. $\forall i \in \{1, \ldots, m\} \colon T_i \geq \mathsf{llp}(o_i, G \setminus B)$, i.e. the time it takes to compute the label of o_i is at least the length of the longest path $\mathsf{llp}(o_i, G \setminus B)$.

Furthermore, if the graph is (m, γ)-**depth-robust**, such paths are with high probability longer than γ if M is significantly smaller than $m \cdot w$.

Lemma 4. If the graph G is (m, γ)-**depth-robust**, then:

$$\Pr_{i \leftarrow \$ [m]}[T_i \geq \gamma] \geq 1 - |B| m^{-1}$$

Given the previous results, we can bound the probability of success of the adversary in proportion to the number of blue nodes, as shown in the following lemma. This way we conclude the first step of the proof of Theorem 2.

Lemma 5. Given an (M, q)-*bounded* adversary $\mathcal{A}$, with $q < \gamma$, and a random oracle $h \in H$ such that $|B| \leq \bar{B}$, then its probability of success is bounded by $\bar{B} \cdot m^{-1}$.

Proof. From Lemma 4 we deduce that the probability that $T_i \geq \gamma$ is at least $1 - |B|m^{-1} \geq 1 - \bar{B}m^{-1}$. But given that $q < \gamma$, it follows that the adversary cannot reply to these challenges on time, which implies that its probability of success is bounded by $\bar{B}m^{-1}$, as claimed. □

E. Security proof: second step for graph-restricted $\mathcal{A}$

Next we show that, with high probability over the choice of the random oracle, we obtain the desired upper bound on the number of blue nodes. That is, for a big proportion of (good) random oracles, $|B|$ is smaller than the number of blocks stored by $\mathcal{A}_0$ in σ. This will help us bound the prediction ability of a cheating adversary that did not store enough memory. Recall that H is the set of all random oracles from $\{0,1\}^{\kappa}$ to $\{0,1\}^{w}$. We will rely on the following well-known result. Intuitively, relying on the adversary $\mathcal{A}$ and its set of blue nodes, we will construct an encoder and a decoder to which we will be able to apply Lemma 6 to derive the desired bounds.

Lemma 6 (adapted from Fact 8.1 in [18])**.** If there is a deterministic encoding procedure $Enc\colon H \to \{0,1\}^s$ and a decoding procedure $Dec\colon \{0,1\}^s \to H$ such that $\Pr_{x \leftarrow\$ H}[Dec(Enc(x)) = x] \geq \delta$, then $s \geq \log|H| + \log\delta$.

Definition 11 (Good oracle)**.** We say that $h \in H$ is a good oracle for a *graph-restricted* $\mathcal{A}$ if $|B| \leq \lceil \frac{M}{w} \rceil$.

Let $\mathcal{S}$ be the oracle machine in Figure 6. It executes $\mathcal{A}_1$ in parallel for all possible challenges. Notice that $\mathcal{S}$ executes basically the same operations as $\mathcal{A}_1$ and does not make repeated queries to the oracle. $\mathcal{S}$ terminates when it has finished processing all parallel executions of $\mathcal{A}_1$. In each round, $\mathcal{S}$ processes the respective oracle calls of each instance of $\mathcal{A}$, and outputs all calls to the environment at the end of the round. In the next lemma $\mathcal{S}$ will be used by the encoder and decoder that we construct, which will observe its outputs and process all the oracle calls of $\mathcal{S}$. The encoder will answer the oracle calls by looking directly at h, while the decoder will obtain the desired values from the encoded state. The fact that $\mathcal{A}_1$ is graph-restricted helps the decoder determine blue nodes directly.

Lemma 7. Let $\mathcal{A}$ be an attacker with parameters $(m, 1, w)$ and $H_{\mathsf{good}} \subseteq H$ be the corresponding set of good oracles. Then $|H_{\mathsf{good}}| \geq (1 - 2^{-w}) \cdot |H|$.

Proof. We show there exist an encoder and a decoder algorithm that using $\mathcal{A}$ are able to compress a random function from H, as long as the size of B is greater than $\lceil \frac{M}{w} \rceil$, i.e. the function is not in H_{good}. Then we apply Lemma 6 to obtain an upper bound for the number of functions for which this is possible.

run $\mathcal{A}_1^{\mathcal{O}_h}(1^w, \rho, \sigma, o_1), \ldots, \mathcal{A}_1^{\mathcal{O}_h}(1^w, \rho, \sigma, o_m)$ in parallel :
for $j = 1$ to $\max\{t_1, \ldots, t_n\} + 1$: Let L be an empty list
 for $i = 1$ to m :
  if $\mathcal{A}_1^{\mathcal{O}_h}(1^w, \rho, \sigma, o_i)$ has not finished :
   run $\mathcal{A}_1^{\mathcal{O}_h}(1^w, \rho, \sigma, o_i)$until the j-th query (or output) Q_{ij}
   - if Q_{ij} is an output, add $(0, i, Q_{ij})$ to L
   - if Q_{ij} is repeated, answer with the previous response
   - if Q_{ij} is a possible pre-label of v, add $(1, v, Q_{ij})$ to L
   - else ask $\mathcal{O}_h(Q_{ij})$, relay the answer to $\mathcal{A}_1$ and store it
 for each (t, v, q) in L : (ordered in inverse topological
  output (t, v, q) order w.r.t. G)
  if $t = 1$ ask $\mathcal{O}_h(q)$, relay the answer to $\mathcal{A}_1$ and store it

Figure 6: Procedure for $\mathcal{S}_{\mathcal{A}}^{\mathcal{O}_h}(1^w, \rho, \sigma)$

The encoder works as follows: for a function h, first run $\mathcal{A}_0$ to obtain σ. Second, run $\mathcal{S}$ with input σ and keep track of its outputs. As the adversary is graph-restricted, all these outputs are either tuples that correspond to labels of challenge nodes, or pre-labels. In both cases, store the blue nodes in order. Once S finishes, if the number of blue nodes stored is less than or equal $\lceil \frac{M}{w} \rceil$, output nothing. Else, output σ, the responses c to all oracles calls made by $\mathcal{S}$ in order (except the ones associated with blue nodes), and all the remaining oracle values c' (not asked by S, nor labels of blue nodes) in lexicographic order. Note that the labels of blue nodes are not stored explicitly but encoded in σ.

The decoder works as follows: given σ, c, c', it will output the whole function table for h. First, it executes $\mathcal{S}$ with input σ. When $\mathcal{S}$ makes an output, as the adversary is graph-restricted, the decoder can deduce the labels of the blue nodes associated to it (if any), and store these values. When $\mathcal{S}$ makes an oracle call, if the response to the oracle call is known (because it is the label of a node that was stored before, or is a repeated call), then respond with that value. Else respond with the next value from c. When S finishes, read all the remaining values in the input c'. At this point the decoder knows the whole table for h, and outputs it.

The size of the encoding is exactly $M + \log|H| - |B| \cdot w$, and it is correct with probability δ, which is the proportion of functions h such as the number of blue nodes is more than $\lceil \frac{M}{w} \rceil$ (oracles not in H_{good}). From Lemma 6 we deduce $M + \log|H| - |B| \cdot w \geq \log|H| + \log(\delta)$ which, together with $|B| > \lceil \frac{M}{w} \rceil$, implies $\delta \leq 2^{-w}$. We conclude that $|H_{\mathsf{good}}| = (1 - \delta) \cdot |H| \geq (1 - 2^{-w}) \cdot |H|$. □

Putting together Lemma 7, Definition 11 for good oracles and Lemma 4, we obtain immediately the statement of Theorem 2 for graph-restricted adversaries. The proof for general adversaries is similar and presented in appendix. The main difference is that the decoder will need some

additional advice to detect where the blue nodes are, which will imply worse bounds when applying Lemma 6.

VII. Depth-robust graphs that can be labelled in-place

In this section we look for a family of **depth-robust** graphs w.r.t. a subset of nodes that can be labelled **in-place** w.r.t. the same subset of nodes. Although some classes of graphs from previous works, e.g. [27, 11], satisfy pebbling-hardness notions related to depth-robustness and can be labelled in place, they don't satisfy the strong depth-robustness property that we require. We therefore propose a new construction. Our construction is based on the graph in [38], where a simple and recursively constructible depth-robust graph w.r.t. edge deletions was proposed. The recent results in [13] showed that it is possible in general to transform an edge depth-robust graph into a vertex depth-robust graph where only one single long path is guaranteed to exist after node deletions, but not several as required by our notion. Inspired by these results, although using a different (and ad-hoc) transformation on the graph from [38], we designed a new class of graphs with the properties by the protocol in Section VI. The main result of this section is the following:

Theorem 3. There exists a family of DAGs $\{G_n\}$ where G_{n+1} has $\mathcal{O}(2^n \cdot n^2)$ nodes and a subset of nodes $O(G_{n+1}) \subset V(G_{n+1})$ of size 2^n such that:

1) G_{n+1} is $(2^n, 2^n)$-**depth-robust** w.r.t. $O(G_{n+1})$;
2) G_{n+1} can be labelled **in-place** w.r.t. $O(G_{n+1})$.

If Γ is a graph, we say that a graph G is a fresh copy of Γ if it is isomorphic and disjoint, and denote it by $G \cong \Gamma$. Let L_n be a list of 2^n nodes. For any DAG G, let $\mathsf{In}(G)$, $\mathsf{Out}(G)$ its input and output nodes respectively. If Y and Z are two disjoint lists of nodes of the same size, denote by $X = Y : Z$ the concatenation of both lists. If $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ are lists of nodes of size n, then $X \to Y = \{(x_i, y_i) \mid 1 \leq i \leq n\}$.

In our construction we use graph with high connectivity, so-called connectors.

Definition 12. A directed acyclic graph with bounded indegree, n inputs, and n outputs is an n-connector if for any $1 \leq k \leq n$ and any sequences $(s'_1, \ldots, s'_k)$ of inputs and $(t'_1, \ldots, t'_k)$ of outputs there are k vertex-disjoint paths connecting each s'_i to the corresponding t'_i.

Next, we introduce notation for connecting two lists of nodes using a connector graph. Let H_n be a 2^n-connector. Given two lists of nodes I and O of size 2^n, and a fresh copy of H_n named C, we denote by $I \xrightarrow{H} O$ the graph with nodes $I \cup O \cup V(C)$ and edges $E(C) \cup (I \to \mathsf{In}(C)) \cup (\mathsf{Out}(C) \to O)$.

Now we are ready to construct our graph family. For each graph G_n from our family, we also define a list of so-called base nodes, which will serve for connecting G_n with other graphs in order to construct the graph G_{n+1}. We define G_0 to be the graph formed of a single node. Then, G_{n+1} is constructed from two copies of G_n and a copy of H_n, which we connect with a set of additional edges. First, every base node in the first copy of G_n is connected through an edge to a corresponding input node in the copy of H_n. Second, the copy of H_n is connected to the second copy of G_n through a recursive edge construction described below and illustrated in Figure 7. Formally, we have:

- G_0: $V(G_0) = \{v\}$, $E(G_0) = \emptyset$ and $\mathsf{Base}(G_0) = (v)$.
- G_n contains three components (L, C, R), denoted by $\mathsf{Left}(G_n)$, $\mathsf{Center}(G_n)$, $\mathsf{Right}(G_n)$, where $L \cong R \cong G_{n-1}$, and $C \cong H_{n-1}$. We let $\mathsf{Base}(G_n) = \mathsf{Base}(L) : \mathsf{Base}(R)$ denote the base vertices.
- The operator $\triangleright$ denotes a recursively defined subgraph, $X \triangleright G_0 = X \to \mathsf{Base}(G_0)$, and:

  $$(X_1 : X_2) \triangleright Y = (X_1 \triangleright \mathsf{Left}(Y)) \cup (X_2 \xrightarrow{H} \mathsf{In}(\mathsf{Center}(Y)))$$

  The vertices of G_n are:

  $$V(G_n) = V(L) \cup V(C) \cup V(R) \cup V(\mathsf{Out}(C) \triangleright R)$$

  The edges of G_n include the edges in the components L, C, R, plus two new sets of edges: the first from $\mathsf{Base}(L)$ to $\mathsf{In}(C)$, and the second from $\mathsf{Out}(C)$ to R:

  $$E(X) = E(L) \cup E(C) \cup E(R) \cup (\mathsf{Base}(L) \to \mathsf{In}(C)) \cup E(\mathsf{Out}(C) \triangleright R)$$

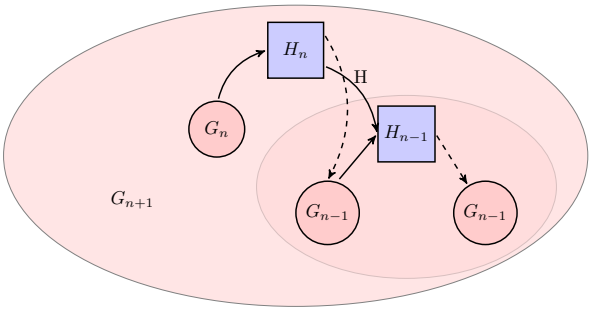

Figure 7: G_{n+1}, continuous arrows represent simple edges $\to$, H arrows represent connector subgraphs $\xrightarrow{H}$ and dashed arrows represent the operator $\triangleright$.

The next result is essential for our construction. It states that if some nodes are removed from a connector with the objective of disconnecting input/output pairs, then it is always possible to do so by just removing inputs or outputs. For a set of nodes R, let $\mathsf{Con}(R)$ be all pairs of nodes (v, w), where v is an input and w is an output, such that all paths from v to w contain at least one node in R.

Lemma 8. Let C_n be an n-connector and $R \subset V(C_n)$ be a subset of its nodes with $|R| < n$. Then there exists a subset of nodes R' containing only input and output nodes, such that $|R'| \leq |R|$ and $\mathsf{Con}(R) \subseteq \mathsf{Con}(R')$.

Proof. Consider a bipartite graph B_R with n nodes on each side such that two nodes are connected iff the corresponding pair is in $\mathsf{Con}(R)$. Consider a maximum matching M and the minimum vertex cover VC in this graph. From König's

theorem [29] we deduce that $|M| = |VC|$. Moreover, from the connector property of C_n we deduce that $|M| \leq |R|$: otherwise, using the connector property between the sets of nodes in the matching, we would find $|R| + 1$ disjoint paths from the input to the output, and those cannot be covered by the nodes in R. Then, $|VC| \leq |R|$. Consider the set $R' = VC$. If these nodes are removed from C_n, consider the bipartite graph $B_{R'}$ (analogous to B_R). Clearly B_R is a subgraph of $B_{R'}$ by the vertex cover property. This implies that $\mathsf{Con}(R) \subseteq \mathsf{Con}(R')$, as claimed. □

To prove the **depth-robustness** of G_n, we will prove first that after removing some nodes, there remains a long path containing nodes from $\mathsf{Base}(G_n)$.

Theorem 4. Consider G_n and any subset of nodes $R \subset V(G_n)$, with $|R| < 2^n$. There is a path L in $G_n \setminus R$ such that $|V(L) \cap \mathsf{Base}(G_n)| \geq 2^n - |R|$.

proof sketch. First, by using Lemma 8 we reduce the problem by showing that if there is a counterexample set R, then there is also a counterexample set R', but for which no node removed is strictly inside any connector graph, i.e. they must be part of the input or output. To conclude, we prove by induction a stronger statement, in which not only the path L exists, but the nodes in $V(L) \cap \mathsf{Base}(\mathsf{Right}(\mathsf{Left}^k(G_n)))$ also must be reachable from the nodes in $\mathsf{In}(\mathsf{Center}(\mathsf{Left}^k(G_n)))$ for $0 \leq k \leq n$. This property makes possible to glue together paths while proving the induction hypothesis. For the full proof see Section IX-C. □

Corollary 3. G_{n+1} is $(2^n, 2^n)$-**depth-robust** w.r.t. $\mathsf{Base}(\mathsf{Right}(G_{n+1}))$

Proof. Consider a set R of size less than 2^n and the path L given by the previous theorem (the theorem would be applicable even if $2^n \leq |R| < 2^{n+1}$, but we are only using a restricted result). Then $V(L) \cap \mathsf{Base}(G_{n+1}) \geq 2^{n+1} - |R|$. Let $M = V(L) \cap \mathsf{Base}(\mathsf{Right}(G_{n+1}))$. Then $|M| \geq 2^n - |R|$ given:

$$\begin{aligned} 2^n + |M| &= |\mathsf{Base}(\mathsf{Left}(G_{n+1}))| + |M| \\ &\geq |V(L) \cap \mathsf{Base}(G_{n+1})| \geq 2^{n+1} - |R| \end{aligned}$$

We conclude that the $2^n - |R|$ rightmost elements (with respect to L) in M contain each at least $2^{n+1} - |R| - (2^n - |R|) = 2^n$ predecessors in L, as claimed. □

As connectors we use the butterfly family of graphs [13], which can be easily labelled **in-place**, and allow the next result to hold:

Lemma 9. G_n can be labelled **in-place** w.r.t. $\mathsf{Base}(G_{n+1})$, and w.r.t. $\mathsf{Base}(\mathsf{Right}(G_{n+1}))$, in $\mathcal{O}(2^n \cdot n^2)$ time.

proof sketch. We apply induction on both statements at the same time. For the induction step of the first statement, notice that to label G_{n+1} with respect to its Base, we can first label $\mathsf{Left}(G_{n+1}) \cong G_n$ using $2^n \cdot w + \mathcal{O}(w)$ memory. Then, while keeping the previous labels stored, compute the labels of $\mathsf{In}(\mathsf{Center}(G_{n+1}))$ using $2^n \cdot w + \mathcal{O}(w)$ extra memory. From these we can compute **in-place** the labels of the nodes in $\mathsf{Center}(G_{n+1})$ and later $\mathsf{Out}(\mathsf{Center}(G_{n+1})) \triangleright \mathsf{Right}(G_{n+1})$, as these subgraphs only contain copies of butterfly graphs, which can be easily labelled **in-place**. Then, by the induction hypothesis, compute the labels of $\mathsf{Base}(\mathsf{Right}(G_{n+1}))$ using the same memory as before. The proof for the induction step for the second statement is the same, except that the labels of $\mathsf{Base}(\mathsf{Left}(G_{n+1}))$ are not stored, so this memory is overwritten afterwards. □

The proposed protocol for memory erasure depends on a graph and a subset of its nodes $O(G)$. In particular, the memory size in words needs to be the same as the $O(G)$. Thus, the graphs constructed previously can only be used if the memory size is a power of two, which may not be true in practice. The next results solve this issue: they show how to construct **depth-robust** graphs that can be labelled **in-place** with respect to a subset of nodes of arbitrary size.

Theorem 5. Let m be any memory size, and let n be the smallest integer such that $2^{n+1} \geq m$. Then there exists an $(m, 2^n)$-**depth-robust** graph G w.r.t. $O(G)$ that can be labelled **in-place** w.r.t. $O(G)$.

Proof. Take $G = C_1 \cup C_2$ where $C_1 \cong C_2 \cong G_{n+1}$ (as defined in our construction). By Corollary 3 and Lemma 11, G is $(2^{n+1}, 2^n)$-**depth-robust** w.r.t. $O(C_1) \cup O(C_2)$. Let O' be a subset of $O(C_2)$ with size $m - 2^n$, and let $O(G) = O(C_1) \cup O'$. Then by Lemma 12 G is $(m, 2^n)$-**depth-robust** w.r.t. $O(G)$. Furthermore, by Lemma 9 and Lemma 13 G can be labelled **in-place** w.r.t. $O(G)$, as claimed. □

A brief discussion on performance and feasibility. We provide a prototype implementation of the algorithm from Section VI with the goal of showing that our depth-robust graphs can indeed be labelled in-place. Our prototype implementation in two languages (python and C) can be found at [1]. To estimate memory overhead, we compiled our prototype to a 32-bit architecture, obtaining a program of size 3.4KB including the space necessary to store the hash values used for graph-labelling. To estimate computational time, we ran our prototype in a standard desktop computer and simulated the erasure of 32KB of memory, which took 0.25s. As far as communication overhead is concerned, we did not directly test the speed of the fast phase. But, taking into account the results obtained in a similar setting, that of performing distance-bounding via Bluetooth communication [1], we estimate that 128 rounds of the fast phase should take around 10 seconds. While recognizing the limitations of our performance analysis, we believe these preliminary values are promising and leave for future work a throughout performance analysis of software-based erasure protocols, including ours.

[1] https://gitlab.uni.lu/regil/memory-erasure

VIII. Conclusions

We presented the first two provable secure PoSE protocols capable of deterring provers from outsourcing the erasure proof to an external conspirator. One protocol is simple to implement but suffers from high communication complexity; the other one asks the prover to label a depth-robust graph from a random seed. Because the labelling algorithm is implemented by a resource-constrained device, we introduced a class of graphs with depth-robust properties that can be labelled in-place using hash functions. Both protocols were proven secure within a formal model. Notably, we proved security bounds for both protocols guaranteeing that all the prover's memory, except for a small part, is erased. Those security bounds are tighter against a restricted, yet plausible, adversary. Hence, future work directed to closing such gap is needed.

References

[1] Aysajan Abidin et al. "Secure, Accurate, and Practical Narrow-Band Ranging System". *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021).

[2] Joël Alwen, Jeremiah Blocki, and Ben Harsha. "Practical Graphs for Optimal Side-Channel Resistant Memory-Hard Functions". *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017.

[3] Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. "Depth-Robust Graphs and Their Cumulative Memory Complexity". *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017.

[4] Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. "Sustained Space Complexity". *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018.

[5] Joël Alwen et al. "On the Complexity of Scrypt and Proofs of Space in the Parallel Random Oracle Model". *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2016.

[6] Joël Alwen et al. "Scrypt Is Maximally Memory-Hard". *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017.

[7] Mahmoud Ammar, Bruno Crispo, and Gene Tsudik. "Simple: A Remote Attestation Approach for Resource Constrained Iot Devices". *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2020.

[8] Mahmoud Ammar et al. "Speed: Secure Provable Erasure for Class-1 Iot Devices". *Eighth ACM Conference on Data and Application Security and Privacy*. 2018.

[9] Sigurd Frej Joel Jørgensen Ankergaard, Edlira Dushku, and Nicola Dragoni. "State-of-the-Art Software-Based Remote Attestation: Opportunities and Open Issues for Internet of Things". *Sensors* 21.5 (2021).

[10] Frederik Armknecht et al. "A Security Framework for the Analysis and Design of Software Attestation". *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. 2013.

[11] Giuseppe Ateniese et al. "Proofs of Space: When Space Is of the Essence". *International Conference on Security and Cryptography for Networks*. Springer, 2014.

[12] Mihir Bellare and Phillip Rogaway. "Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols". *Proceedings of the 1st ACM Conference on Computer and Communications Security*. 1993.

[13] Jeremiah Blocki and Mike Cinkoske. "A New Connection Between Node and Edge Depth Robust Graphs". *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2021.

[14] Ioana Boureanu et al. "Security Analysis and Implementation of Relay-Resistant Contactless Payments". *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020.

[15] Sergiu Bursuc et al. *Software-Based Memory Erasure with relaxed isolation requirements: Extended Version*. 2024. arXiv: 2401.06626 [cs.CR].

[16] Claude Castelluccia et al. "On the Difficulty of Software-Based Attestation of Embedded Devices". *Proceedings of the 16th ACM Conference on Computer and Communications Security*. 2009.

[17] Mauro Conti et al. "Evexchange: A Relay Attack on Electric Vehicle Charging System". *Computer Security–ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark*. Springer, 2022.

[18] Anindya De, Luca Trevisan, and Madhur Tulsiani. "Time Space Tradeoffs for Attacks against One-Way Functions and PRGs". *Annual Cryptology Conference*. Springer, 2010.

[19] Cynthia Dwork, Moni Naor, and Hoeteck Wee. "Pebbling and Proofs of Work". *Annual International Cryptology Conference*. Springer, 2005.

[20] Stefan Dziembowski, Tomasz Kazana, and Daniel Wichs. "One-Time Computable Self-Erasing Functions". *Theory of Cryptography Conference*. Springer, 2011.

[21] Stefan Dziembowski et al. "Proofs of Space". *Annual Cryptology Conference*. Springer, 2015.

[22] E. M. V. EMVCo. "Contactless Specifications for Payment Systems". *Book C-2, Kernel* 2 (2021).

[23] Paul Erdos, Ronald L. Graham, and Endre Szemerédi. "On Sparse Graphs with Dense Long Paths". *Comp. and Math. with Appl* 1 (1975).

[24] Reynaldo Gil Pons et al. "Is Eve Nearby? Analysing Protocols under the Distant-Attacker Assumption". *IEEE Computer Security Foundations Symposium, August 7-10, 2022, Haifa, Israel*. 2022.

[25] Peter Gutmann. "Secure Deletion of Data from Magnetic and Solid-State Memory". *6th USENIX Security Symposium (USENIX Security 96)*. San Jose, CA: USENIX Association, July 1996.

[26] Ghassan O. Karame and Wenting Li. "Secure Erasure and Code Update in Legacy Sensors". *International Conference on Trust and Trustworthy Computing*. Springer, 2015.

[27] Nikolaos P. Karvelas and Aggelos Kiayias. "Efficient Proofs of Secure Erasure". *International Conference on Security and Cryptography for Networks*. Springer, 2014.

[28] Chongkyung Kil et al. "Remote Attestation to Dynamic System Properties: Towards Providing Complete System Integrity Evidence". *IEEE/IFIP International Conference on Dependable Systems & Networks*. 2009.

[29] Dénes Konig. "Graphs and Matrices". *Matematikai és Fizikai Lapok* 38 (1931).

[30] Boyu Kuang et al. "A Survey of Remote Attestation in Internet of Things: Attacks, Countermeasures, and Prospects". *Computers & Security* 112 (2022).

[31] Francesca Meneghello et al. "IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices". *IEEE Internet of Things Journal* 6.5 (2019).

[32] Lily Hay Newman. *An Elaborate Hack Shows How Much Damage IoT Bugs Can Do*. https://www.wired.com/story/elaborate-hack-shows-damage-iot-bugs-can-do/. 2010.

[33] Bryan Parno, Jonathan M. McCune, and Adrian Perrig. "Bootstrapping Trust in Commodity Computers". *IEEE Symposium on Security and Privacy*. IEEE, 2010.

[34] Daniele Perito and Gene Tsudik. "Secure Code Update for Embedded Devices via Proofs of Secure Erasure". *European Symposium on Research in Computer Security*. Springer, 2010.

[35] Krzysztof Pietrzak. "Proofs of Catalytic Space". *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*. Ed. by Avrim Blum. Vol. 124. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[36] Andreea-Ina Radu et al. "Practical EMV Relay Protection". *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022.

[37] Kasper Bonne Rasmussen and Srdjan Capkun. "Realization of RF Distance Bounding." *USENIX Security Symposium*. 2010.

[38] Georg Schnitger. "On Depth-Reduction and Grates". *24th Annual Symposium on Foundations of Computer Science (Sfcs 1983)*. IEEE, 1983.

[39] Arvind Seshadri et al. "SWATT: Software-based Attestation for Embedded Devices". *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*. IEEE, 2004.

[40] Rolando Trujillo-Rasua. "Secure Memory Erasure in the Presence of Man-in-the-Middle Attackers". *Journal of Information Security and Applications* 57 (2019).

[41] Dominique Unruh. "Random Oracles and Auxiliary Input". *Annual International Cryptology Conference*. Springer, 2007.

IX. Appendix

A. Reduction to uniform adversaries and to one round

Lemma 1. For any (M, q)-bounded adversary $\mathcal{A}$ against the PoSE-DB security experiment, there is a uniform (M, q)-bounded adversary $\bar{\mathcal{A}}$ that wins the experiment with at least the same probability: $\Pr_{\Upsilon \leftarrow\$ \mathcal{I}}[\mathcal{A}^r] \leq \Pr_{\Upsilon \leftarrow\$ \mathcal{I}}[\bar{\mathcal{A}}^r]$.

Proof. It is sufficient to prove that, for any fixed $\Upsilon \in \mathcal{I}$, we have $\Pr[\mathcal{A}^r \mid \Upsilon] \leq \Pr\left[\bar{\mathcal{A}}^r \;\middle|\; \Upsilon\right]$. In this case, since $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ is deterministic and Υ is fixed, for every i the stateσ_{i+1} returned by $\mathcal{A}_0$ is uniquely determined by the sequence of challenges $x_1, \ldots, x_i$ in the experiment $\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1}$. Let Σ be the set of all possible states σ_i that can be output in any round by $\mathcal{A}_0$ over all possible sequences of challenges. We denote by $\Pr[\mathcal{A}_1 \mid \Upsilon, \sigma]$ the probability that $\mathcal{A}_1$ answers a single challenge correctly when given the state σ as input:

$$\Pr[\mathcal{A}_1 \mid \Upsilon, \sigma] = \Pr\left[\mathsf{Vrfy}(\rho, \Upsilon, x, y) \;\middle|\; \begin{smallmatrix} x \leftarrow \mathsf{Chal}(\rho) \\ y \leftarrow \mathcal{A}_1^{\mathcal{O}_h}(1^w, \rho, \sigma, x) \end{smallmatrix}\right]$$

where the probability is taken over the randomness used by the challenge generation algorithm. We define σ_μ to be the state in Σ for which the probability of winning given a random challenge is maximal $\sigma_\mu = \arg\max_{\sigma \in \Sigma} \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma]$. Let $\bar{\mathcal{A}} = (\bar{\mathcal{A}}_0, \mathcal{A}_1)$, where $\bar{\mathcal{A}}_0$ always returns σ_μ, independent of the set of challenges it obtains as input. Note that $\bar{\mathcal{A}}$ is (M_0, q)-bounded, since the state returned by $\bar{\mathcal{A}}_0$ is among the possible states returned by $\mathcal{A}_0$, and the adversary $\mathcal{A}_1$ is the same. We will now show that $\bar{\mathcal{A}}$ achieves at least the same probability of success as $\mathcal{A}$. If $\bar{x}_k$ is a sequence of challenges $\{x_1, x_2, \ldots, x_k\}$ we denote by:

- $\Pr\left[\mathcal{A}^k \;\middle|\; \Upsilon, \bar{x}_k\right]$ the probability that the adversary $\mathcal{A}$ gets a correct answer in the first k rounds given that the first k challenges are $\bar{x}_k$. As $\mathcal{A}$ is deterministic, this probability is either 0 or 1.
- $\Pr[\mathcal{A}^t(\bar{x}_k) \mid \Upsilon, \bar{x}_k]$ the probability that the adversary $\mathcal{A}$ gets a correct answer in t successive rounds starting from round $k+1$ given that the challenges in the first k rounds are $\bar{x}_k$.
- $\Pr[\bar{x}_k \mid \Upsilon]$ the probability, taken over the randomness used by the challenge algorithm, that the first k challenges are $\bar{x}_k$.

For a vector of challenges $\bar{x}_k$ we have $\Pr\big[\mathcal{A}^1(\bar{x}_k) \mid \Upsilon, \bar{x}_k\big] = \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma, \bar{x}_k]$ where $\sigma \leftarrow \mathcal{A}_0(\bar{x}_k)$ and:

$$\Pr[\mathcal{A}_1 \mid \Upsilon, \sigma, \bar{x}_k] = \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma] \leq \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma_\mu]$$

where the equality comes from the independence between A_1 and $\bar{x}_k$ given Υ, σ; the inequality follows by definition of σ_μ. Using the inequality above, we prove by induction that, for any $k \geq 0$, we have $\Pr\big[\mathcal{A}^{k+1} \mid \Upsilon\big] \leq \Pr\big[\bar{\mathcal{A}}^{k+1} \mid \Upsilon\big]$. The case $k = 0$ is trivial. For $k > 1$, we obtain:

$$\begin{aligned}
&\Pr\big[\mathcal{A}^{k+1} \mid \Upsilon\big] \\
&= \sum_{\bar{x}_k} \Pr[\bar{x}_k \mid \Upsilon] \cdot \Pr\big[\mathcal{A}^k \mid \Upsilon, \bar{x}_k\big] \cdot \Pr\big[\mathcal{A}^1(\bar{x}_k) \mid \Upsilon, \bar{x}_k\big] \\
&\leq \sum_{\bar{x}_k} \Pr[\bar{x}_k \mid \Upsilon] \cdot \Pr\big[\mathcal{A}^k \mid \Upsilon, \bar{x}_k\big] \cdot \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma_\mu] \\
&= \Pr\big[\mathcal{A}^k \mid \Upsilon\big] \cdot \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma_\mu] \\
&\leq \Pr\big[\bar{\mathcal{A}}^k \mid \Upsilon\big] \cdot \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma_\mu] = \Pr\big[\bar{\mathcal{A}}^{k+1} \mid \Upsilon\big]
\end{aligned}$$

□

Lemma 2. For any uniform adversary $\mathcal{A}$ and any $\Upsilon \in \mathcal{I}$, we have $\Pr[\mathcal{A}^r \mid \Upsilon] = \Pr\big[\mathcal{A}^1 \mid \Upsilon\big]^r$.

Proof.

$$\Pr[\mathcal{A}^r \mid \Upsilon] = \Pr[\mathcal{A}^r \mid \Upsilon, \sigma \leftarrow \mathcal{A}_0] = \Pr[\mathcal{A} \mid \Upsilon]^r \qquad \square$$

Proposition 1. Given $\zeta < 1$, $\mathcal{I}_{\mathsf{good}} \subseteq \mathcal{I}$ and a uniform adversary $\mathcal{A}$ whose winning probability is bounded to ζ within $\mathcal{I}_{\mathsf{good}}$, we have $\Pr_{\Upsilon \leftarrow \$\mathcal{I}}[\mathcal{A}^r] \leq \zeta^r + \frac{|\mathcal{I} \setminus \mathcal{I}_{\mathsf{good}}|}{|\mathcal{I}|}$.

Proof. From Lemma 2, we have:

$$\Pr_{\Upsilon \leftarrow \$\mathcal{I}}[\mathcal{A}^r] = \frac{1}{|\mathcal{I}|} \sum_{\Upsilon \in \mathcal{I}} \Pr[\mathcal{A}^r \mid \Upsilon] = \frac{1}{|\mathcal{I}|} \sum_{\Upsilon \in \mathcal{I}} \Pr[\mathcal{A} \mid \Upsilon]^r$$

To reduce notation in what follows, we let $p_\Upsilon = \Pr[\mathcal{A} \mid \Upsilon]$ and omit $\Upsilon \in \mathcal{I}$ when we sum over all Υ in $\mathcal{I}$. We have:

$$\begin{aligned}
&\sum_{\Upsilon} \Pr[\mathcal{A} \mid \Upsilon]^r = \sum_{\Upsilon} p_\Upsilon^r = \sum_{\Upsilon, p_\Upsilon > \zeta} p_\Upsilon^r + \sum_{\Upsilon, p_\Upsilon \leq \zeta} p_\Upsilon^r \\
&\leq \sum_{\Upsilon,\ p_\Upsilon > \zeta} 1 + \sum_{\Upsilon} \zeta^r \leq \sum_{\substack{\Upsilon,\ p_\Upsilon > \zeta \\ \Upsilon \in \mathcal{I}_{\mathsf{good}}}} 1 + \sum_{\substack{\Upsilon,\ p_\Upsilon > \zeta \\ \Upsilon \notin / I_{\mathsf{good}}}} 1 + \sum_{\Upsilon} \zeta^r \\
&\leq 0 + |\mathcal{I} \setminus \mathcal{I}_{\mathsf{good}}| + |\mathcal{I}| \cdot \zeta^r
\end{aligned}$$

where for the last inequality we use that $\mathcal{A}$'s winning probability is bounded to ζ within $\mathcal{I}_{\mathsf{good}}$ to deduce that the first sum is empty. Combining the two results above, we deduce $\Pr_{\Upsilon \leftarrow \$\mathcal{I}}[\mathcal{A}^r] \leq \zeta^r + \frac{|\mathcal{I} \setminus \mathcal{I}_{\mathsf{good}}|}{|\mathcal{I}|}$ as claimed. □

B. Reduction of PoSE security to depth-robustness

Lemma 3. $\forall i \in \{1, \ldots, m\} \colon T_i \geq \mathsf{llp}(o_i, G \setminus B)$, i.e. the time it takes to compute the label of o_i is at least the length of the longest path $\mathsf{llp}(o_i, G \setminus B)$.

Proof. Fix i. If the response to the challenge o_i is not correct, then the claim is trivial (T_i is infinite). The case $o_i \in B$ is also trivial. Assume $o_i \notin B$ and that the answer to the challenge o_i is correct. Consider the longest path ending in o_i, let it be $(v_1, v_2, \ldots, v_{k-1}, v_k = o_i)$, where $k = \mathsf{llp}(o_i, G \setminus B)$. As none of the nodes in this path are blue, all their labels were computed by asking the oracle for the corresponding value. Let f_j be the smallest index such that $Q_{x,f_j} = \ell^-(v_j)$ for some x. As $v_k = o_i$, then $t_i \geq f_k$.

We prove that $\forall j \colon 1 \leq j < k$ we have $f_j < f_{j+1}$. Each inequality can be proved using the same argument, so we prove only $f_1 < f_2$. As v_1 is not blue and v_2 is a successor of v_1, then the query $\ell^-(v_2)$ in round f_2 (which contains the label of v_1) must have happened after the query $\ell^-(v_1)$ in round f_1. This implies $f_2 > f_1$, as needed.

As $\forall i \colon f_i \geq 1$ then $f_k \geq k \implies T_i \geq k$. □

Lemma 4. If the graph G is (m, γ)-**depth-robust**, then:

$$\Pr_{i \leftarrow \$[m]}[T_i \geq \gamma] \geq 1 - |B| m^{-1}$$

Proof. We can assume $|B| < m$, since the result trivially holds otherwise. Therefore, from the definition of **depth-robustness**, there exist a set $O' \subseteq \{o_i\} \setminus B$ with $|O'| \geq m - |B|$ s.t. $\forall o_i \in O' \colon \mathsf{llp}(o_i, G \setminus B) \geq \gamma$. From Lemma 3, we deduce $\forall o_i \in O' \colon T_i \geq \gamma$. Therefore, we deduce

$$\Pr_{i \leftarrow \$[m]}[T_i \geq \gamma] \geq \Pr_{i \leftarrow \$[m]}[o_i \in O'] \geq 1 - |B| m^{-1} \qquad \square$$

The next definition and lemma hold for general adversaries.

Definition 13 (Good oracle)**.** We say that $h \in H$ is a good oracle for an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ if we have $|B| \leq \left\lceil \frac{M}{w - \log(m \cdot q)} \right\rceil$.

Lemma 10. If $\mathcal{A}$ has parameters $(m, 1, w)$ and $H_{\mathsf{good}} \subseteq H$ be the corresponding set of good oracles. Then $|H_{\mathsf{good}}| \geq |H| \cdot (1 - 2^{-w + \log(m \cdot q)})$.

Proof. The proof follows closely the one of Lemma 7, but must be extended for general adversaries.

As before, the encoder will output 0 if $h \in H_{\mathsf{good}}$. Else, it will output σ, p, c, c', where the new value p is a list of indices, where each index a_i corresponds to the i-th output of $\mathcal{S}$, and indicates that it contains the first appearance of the label of a blue node. Each of these pairs can be encoded using $\log(m \cdot q)$ bits.

The decoder will also execute $\mathcal{S}$ as before. The difference is that when $\mathcal{S}$ outputs a value, only if the corresponding index is in p, it will extract and store the labels of the blue nodes associated with that output. At the end of execution, the decoder will have stored all labels of blue nodes.

As the size of p is at most the number of blue nodes, the size of the encoding is at most $M + |B| \cdot \log(m \cdot q) + \log|H| - |B| \cdot w$, and it is correct with probability $\delta = 1 - |H_{\mathsf{good}}|/|H|$. From Lemma 6 we deduce:

$$\begin{aligned}
&M + \log|H| - |B| \cdot (w - \log(m \cdot q)) \geq \log|H| + \log(\delta) \\
&\implies \delta \leq 2^{-w + \log(m \cdot q)} \quad \text{from } |B| > \left\lceil \frac{M}{w - \log(m \cdot q)} \right\rceil
\end{aligned}$$

Hence $|H_{\mathsf{good}}| = (1 - \delta) \cdot |H| \geq (1 - 2^{-w + \log(m \cdot q)}) \cdot |H|$. □

C. Graph construction

Theorem 4. Consider G_n and any subset of nodes $R \subset V(G_n)$, with $|R| < 2^n$. There is a path L in $G_n \setminus R$ such that $|V(L) \cap \mathsf{Base}(G_n)| \geq 2^n - |R|$.

Proof. Assume the contrary, there exists a set R such that there is no path containing $2^n - |R|$ nodes from $\mathsf{Base}(G_n)$. Consider the partition $B \cup R_1 \cup \ldots \cup R_k$ of the vertices in R such that the nodes in each partition R_i belong to exactly one subgraph H_i (defined by some $\xrightarrow{H}$, or the Center of some component) and $B = R \cap \mathsf{Base}(G_n)$. If R'_i is the set of nodes guaranteed to exist by Lemma 8, then removing R'_i instead of R_i doesn't increase the connectivity between nodes in $\mathsf{Base}(G_n)$. We deduce that if the result is true for R', then after removing the nodes in $R' = B \cup R'_1 \cup \ldots \cup R'_k$ from G_n, there is a path containing $2^n - |R'| \geq 2^n - |R|$ nodes from $\mathsf{Base}(G_n)$. The vertices in R' are either in the input or output of some H_i, or in $\mathsf{Base}(G_n)$.

It remains to prove that if the nodes in R' are removed, there is a path with the required features. We prove this by induction in a stronger statement: in $G_n \setminus R'$ there exists a path L with the required properties such that the nodes in $V(L) \cap \mathsf{Base}(\mathsf{Right}(\mathsf{Left}^k(G_n)))$ are reachable from $\mathsf{In}(\mathsf{Center}(\mathsf{Left}^k(G_n)))$ for $0 \leq k < n$.

The base cases G_0 and G_1 can be checked manually. For the induction step assume in R' there are l, r, c nodes from $\mathsf{Left}(G_{n+1})$, $\mathsf{Right}(G_{n+1})$ and $\mathsf{Center}(G_{n+1}) \cup (\mathsf{Out}(\mathsf{Center}(G_{n+1})) \triangleright \mathsf{Right}(G_{n+1}) \setminus \mathsf{Right}(G_{n+1}))$, respectively. By the induction hypothesis, there is a path L_l in $\mathsf{Left}(G_{n+1})$ with the required properties, i.e. $|V(L_l) \cap \mathsf{Base}(\mathsf{Left}(G_{n+1}))| \geq 2^n - l$. There is also L_r in $\mathsf{Right}(G_{n+1})$.

We will finish the proof by case analysis. If $r + c \geq 2^n$, then $2^n - l \geq 2^{n+1} - |R'|$ and $L = L_l$ has the required properties. Else, $r + c < 2^n$. Let $c_i = |R' \cap \mathsf{In}(\mathsf{Center}(G_{n+1}))|$.

Next we prove that there is a list of nodes $F \subseteq V(L_r) \cap \mathsf{Base}(\mathsf{Right}(G_{n+1}))$ such that $|F| \geq 2^n - r - c + c_i$ and all nodes in F are reachable from at least one node in $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$.

Consider a node $v \in V(L_r) \cap \mathsf{Base}(\mathsf{Right}(G_{n+1}))$. Let k be the largest such that $v \in \mathsf{Left}^k(\mathsf{Right}(G_{n+1}))$, and $G_v = \mathsf{Left}^k(\mathsf{Right}(G_{n+1}))$. If $k = n$, then the only way that this node is not reachable from $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$ is if its predecessor in that set belongs to R'. If $k < n$, then by the induction hypothesis, given that $v \in \mathsf{Base}(\mathsf{Right}(G_v))$, then v is reachable from a node in $\mathsf{In}(\mathsf{Center}(G_v))$, call this node w. The only way that v is not reachable from $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$ through w is if all paths in the connector $\bar{H}$ (which is isomorphic to H_{n-k-1}) from this set to w are covered by nodes in R'. As w is not covered, this is only possible if all nodes from $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$ connected to $\mathsf{In}(\bar{H})$ are in R', which are exactly 2^{n-k-1} nodes. But there are at most 2^{n-k-1} nodes in $\mathsf{Base}(\mathsf{Right}(G_v))$. Applying the previous deduction for all v, we deduce that the number of nodes in $V(L_r) \cap \mathsf{Base}(\mathsf{Right}(G_{n+1}))$ not reachable from $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$ is at most the number of nodes that were removed from $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$. As this number is at most $c - c_i$, the claim follows.

Consider now the path P_F determined by the list of nodes F. All these nodes are reachable from $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$. Furthermore, as $c_i \leq c + r < 2^n$, these nodes are also reachable from some node in $\mathsf{In}(\mathsf{Center}(G_{n+1}))$. We will concatenate this path with a suitable subset of L_l. Let $D \subseteq L_l \cap \mathsf{Base}(\mathsf{Left}(G_{n+1}))$ be the set of nodes that are connected to a node in $\mathsf{In}(\mathsf{Center}(G_{n+1}))$. Then $|D| \geq 2^n - l - c_i$. Consider the path P_D determined by the list of nodes D. This path can be extended with P_F and the resulting path contains at least $2^n - l - c_i + 2^n - r - c + c_i = 2^{n+1} - |R'|$ nodes in $\mathsf{Base}(G_{n+1})$, as was needed. □

Lemma 11. Let G be (a, b)-**depth-robust** w.r.t. $O(G)$. Then the graph defined by the disjoint union of two copies of G, G_1 and G_2, is $(2a, b)$-**depth-robust** w.r.t. $O(G_1) \cup O(G_2)$.

Proof. Let R be a set of nodes with $|R| < 2 \cdot a$. Let $r_1 = |R \cap V(G_1)|$ and $r_2 = |R \cap V(G_2)|$. If $r_1 < a$ then by the depth robustness of G_1 there are at least $a - r_1$ nodes v in $G_1 \setminus R$ with $\mathsf{llp} v > b$. We conclude by case analysis:

- if $r_2 \geq a \implies r_1 < a$, from which the result follows given that $a - r_2 \geq 2 \cdot a - r_1 - r_2$
- if $r_1 \geq a$ the result follows by symmetry with the previous case
- else $r_2 < a \wedge r_1 < a$, and we can find $a - r_2$ paths in G_2 and $a - r_1$ paths in G_1 with length greater than b, ending in different vertices in $O(G_1) \cup O(G_2)$. □

Lemma 12. Let G be (a, b)-**depth-robust** w.r.t. $O(G)$. Let O' be a subset of $O(G)$ of size a'. Then G is (a', b)-**depth-robust** w.r.t. O'.

Proof. Let R' be a set of nodes with $|R'| < a'$. Let $R = R' \cup (O(G) \setminus O')$. Notice that $|R| < a$. Then the nodes with longs paths guaranteed by the depth robustness of G in $G \setminus R$ will also be nodes with long paths in $G \setminus R'$. Moreover, by definition of R these nodes are in O' and there are at least $a - |R| = a' - |R'|$ such nodes, as needed. □

Lemma 13. Let G be a graph that can be labelled **in-place** w.r.t. $O(G)$, and G' the graph defined by the disjoint union of two copies of G, G_1 and G_2. If O' is a subset of $O(G_2)$, then G' can be labelled **in-place** w.r.t. $O(G_1) \cup O'$.

Proof. First compute the labels of $O(G_2)$ **in-place**. Then keep storing only the labels from O'. With the space left, compute the labels of $O(G_1)$ **in-place**. □