On the Cost of Adaptivity in Graph-Based Games

Chethan Kamath*[1], Karen Klein†[2], Krzysztof Pietrzak†[2], and Michael Walter†[2]

[1]Charles University, ckamath@protonmail.com
[2]IST Austria, {kklein,pietrzak,mwalter}@ist.ac.at

January 15, 2021

Abstract

The security of cryptographic primitives and protocols against adversaries that are allowed to make adaptive choices (e.g., which parties to corrupt or which queries to make) is notoriously difficult to establish. A broad theoretical framework was introduced by Jafargholi et al. [Crypto'17] for this purpose. In this paper we initiate the study of lower bounds on loss in adaptive security for certain cryptographic protocols considered in the framework. We prove lower bounds that almost match the upper bounds (proven using the framework) for proxy re-encryption and generalized selective decryption, a security game that captures the security of certain group messaging and broadcast encryption schemes. The security games used to model these protocols involve an underlying graph that can be adaptively built by the adversary.

Some of our lower bounds only apply to a certain class of black-box reductions, which we term "oblivious". (We do however show one lower bound on proxy re-encryption that applies to general fully black-box reductions.) The fact that our lower bounds crucially rely on "obliviousness" hints to the possibility that the existing upper bounds can be improved by using more sophisticated reductions. As the main technical contribution, we introduce a two-player multi-stage game called the Builder-Pebbler Game and then analyse strategies for this game to establish bounds on success probability of its players. Finally, using oracle separation techniques, we translate these bounds into cryptographic lower bounds.

*Funded by Charles University project PRIMUS/17/SCI/9. Part of the work was done while the author was at Northeastern University, supported by the IARPA grant IARPA/2019-19-020700009.

†Funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (682815 - TOCNeT).

Contents

1 Introduction

Consider the following game played between a challenger C and an adversary A using a symmetric encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$. The challenger first samples, independently and uniformly at random, N keys $k_1, \ldots, k_N$. These correspond to users $U_1, \ldots, U_N$ respectively. The adversary A is now allowed to *adaptively* make two types of queries:

1. Ask for an encryption of k_j under the key k_i to obtain $\mathsf{Enc}_{k_i}(k_j)$, or
2. Corrupt a user U_i to obtain the key k_i.

At the end of the game, A challenges C on a user U_{i^*} and is given either the real key k_{i^*} or an independent, random key r. A wins this "real or random game" if it correctly guesses which of the two it got. If no efficient A can win with probability higher than $1/2 + \epsilon$ we say the protocol is 2ϵ secure.

The above game can be thought of as the adversary A adaptively building a "key-graph" $G = (\mathcal{V}, \mathcal{E})$, where the vertices $\mathcal{V} = \{1, \ldots, N\}$ correspond to the users and their keys, whereas the (directed) edges $\mathcal{E}$ correspond to the encryption queries that A makes: a directed edge (i, j) is added to $\mathcal{E}$ if A request the encryption of k_j under the key k_i. Note that for i^* to be a non-trivial challenge, i^* must be a sink and must *not* be reachable (in the graph-theoretic sense) from any of the corrupted vertices — otherwise, A can simply decrypt the ciphertexts along the path from any corrupted node to the challenge to learn k_{i^*}.

The above game is called *generalised selective decryption* (GSD) and it captures the security of protocols like multicast encryption [45] and continuous group key agreements [2, 1]. Thus, the question one is interested in is whether the security of this game (given that the key-graph is *acyclic*) can be based on the IND-CPA security of the underlying encryption scheme.[1] For this we need to prove a computational soundness (i.e., security) theorem of the form: if the encryption scheme is ϵ-secure then the GSD game is ϵ'-secure for some ϵ' that depends on ϵ. Ideally, the loss of security should be kept to a polynomial, i.e. $\epsilon' = \epsilon/\mathsf{poly}(N)$. Otherwise, this requires to set the security parameter of the underlying encryption scheme very large, which will lead to inefficiency. Even worse, since it is unclear if a non-trivial attack against

[1] In case the key-graph contains cycles, one must additionally assume that the encryption scheme is key-dependent message (KDM) secure [10]. Such problems are of a different flavour and we don't deal with them. As mentioned before, the GSD game is typically used to capture the security of protocols, and then the acyclicity is enforced by the protocol rules.

GSD exists that does not break the encryption scheme, a large gap in the quantitative security is counter-intuitive and points to a lack in understanding of the problem.

The simpler task of proving a soundness theorem in case the adversary is *selective*, in the sense that it commits to its queries (and thus the key-graph G) at the beginning of the GSD game, is relatively straightforward to achieve. If the graph is known ahead of time, it is easy to construct a series of $O(N)$ hybrids, each of which can be shown indistinguishable under the security of the encryption scheme (see e.g. [35]). The study of adaptive security of GSD, where the key-graph is unknown at the beginning of the game and is only gradually revealed during the query phase, was initiated in [45] and remains notoriously hard. In particular, non-trivial results are only known in settings, where the adversary is restricted to specific key-graphs (which needs to be enforced by the higher level protocol). The state of the art is represented by the general Piecewise-Guessing framework [35, 39].

1.1 Our Results

The Piecewise-Guessing Framework has found several further applications [39, 21, 1, 2, 37]. Therefore, it is a natural question whether the existing security proofs obtained in the Piecewise-Guessing Framework can be further improved. In this paper, we approach this question from the negative direction and argue that simply using existing techniques, this will not be possible for some of its applications. In particular, we show that for a certain class of black-box reductions, which we term "oblivious" (see discussion below), the upper bounds for proxy re-encryption[2] (PRE) schemes [11] given in [21] and for generalised selective-decryption (GSD) [45] given in [35] are essentially tight, as stated informally below.

Theorem 1 (Informal)**.** *Any oblivious black-box reduction from adaptive* GSD *restricted to* paths or binary trees *to the IND-CPA security of the underlying symmetric-key encryption scheme loses at least a factor that is* quasi-polynomial *in the number of users; for GSD restricted to* arbitrary directed acyclic graphs *the loss is at least* sub-exponential*.*

Theorem 2 (Informal)**.** *Any oblivious black-box reduction from adaptive PRE-CPA-security for* proxy re-encryption *restricted to* paths or binary trees *to IND-CPA security of the public-key encryption scheme and 1-(resp., for binary trees, 2-)weak key-privacy loses at least a factor that is* quasi-polynomial *in the number of users.*

The common thread to the two cryptographic protocols above is that their security game can be abstracted out by a two-player multi-stage game which we call

[2] A proxy re-encryption scheme is a public-key encryption scheme that allows the holder of a key $\mathtt{pk}$ to derive a re-encryption key for any other key $\mathtt{pk}'$. This re-encryption key lets anyone transform ciphertexts under $\mathtt{pk}$ into ciphertexts under $\mathtt{pk}'$ without having to know the underlying message. The formal definition is given in §5.2.

the "Builder-Pebbler Game". We are unable to establish lower bounds for other applications of the Piecewise-Guessing Framework (e.g., computational secret sharing, garbling circuits or constrained PRFs) as their security model is not quite captured by the Builder-Pebbler Game. The high level reason for this is that the graphs (e.g., circuit to be garbled or the access structure) in other applications of the piecewise-guessing framework (e.g., Yao's garbling or computational secret sharing) is fixed ahead of the time and the adaptivity comes from other sources (e.g., choice of garbling input or targetted user). Therefore we would require other combinatorial abstractions to establish lower bounds for them (see discussion in §6.4). We defer the discussion on the Builder-Pebbler Game to the next section (§1.2.2) and explain informally what we mean by oblivious reductions next, mostly from the perspective of GSD. We will then argue that this comprises a natural class of reductions.

Oblivious black-box reductions. Oblivious reductions are a certain class of fully black-box reductions [49], and our definition is motivated by the reductions in [35]. On a high level, the behaviour of an oblivious reduction is "independent" of the adversary's behaviour throughout the simulation of the security game. To see what we mean by this, let's return to the example of GSD. A reduction (simulating some consecutive hybrids) can decide to answer an encryption query issued by the adversary either with a consistent or an inconsistent ciphertext (let's ignore the challenge ciphertext for the moment). In particular, it has total control over the number of inconcistencies in the final simulation (assuming it knows the number of queries the adversary will make). However, as the key-graph is only gradually revealed to the reduction, it doesn't know where the edge (representing the encryption query) will end up within the key-graph. We call a GSD reduction *oblivious* if it does not make use of the partial graph structure it learns during the game but rather sticks to some strategy that is independent of the history of the adversary's queries. There are several ways one could formalise this: for example, one could require the reduction as initially "committing" to which queries it will answer inconsistently. However, this does not mean that for all queries it has to commit to its decision, but rather commit to some minimal description of the edges it intends to respond inconsistently to. In order to capture as many reductions as possible (while still being able to prove lower bounds), we ended up defining them as reductions which commit to a minimal set of nodes which *covers* all inconsistent edges, i.e., a minimal *vertex cover*.[3] For example in the case of graphs of high indegree, clearly, guessing the set of sinks of inconsistent edges gives a much more succinct representation. Furthermore, in these settings we also need to assume that the reduction does not rewind the adversary, so we define oblivious reductions to be also non-rewinding. A formal definition of an oblivious GSD reduction is given in Definition 17; the corresponding definition for PREs is similar and is given in Definition 25.

[3]Technically, we do not require *minimal* vertex cover, but a weaker notion which we call "non-trivial" vertex cover (see Definition 2).

Why oblivious reductions? We note that oblivious black-box reductions are a quite natural notion and encompass some of the key reductions in the literature. Beside the reductions proposed and analysed in [35] (and its follow-up works), partitioning-based reductions, which have been successfully employed in a plethora of works [16], also roughly behave in an oblivious manner.[4] Moreover, oblivious black-box reductions encompass the currently-known techniques for establishing upper bounds for GSD and PRE, and this means that all currently-known reductions for these applications are in some sense optimal. Therefore, in order to obtain better upper bounds on the loss function λ, one needs to deviate significantly from the current proof techniques (i.e. non-oblivious or rewinding reductions for GSD and restricted PRE). Our results thus serve as a guide towards new avenues to finding better reductions by ruling out a large class of reductions – such possibilities are discussed in §6.

Beyond oblivious reductions. For proxy re-encryption on *arbitrary* directed acyclic graphs, however, we achieve a stronger *exponential* lower bound, which holds even for *non-oblivious* black-box reductions. Essential to this result is exploiting the fact that a weak key-privacy challenge must be embedded at a *node*, not an edge, which means the reduction is more restricted here.

Theorem 3 (Informal)**.** Any *(possibly non-oblivious) black-box reduction from PRE-CPA-security for* proxy re-encryption *restricted to* arbitrary DAGs *to IND-CPA security of the encryption scheme and N-weak key privacy loses a factor that is* exponential *in the number of users.*

1.2 Technical Overview

On a high level, our approach can be divided into two steps. In the first step (§1.2.2), which is purely combinatorial, we analyse a two-player multi-stage game which we call the Builder-Pebbler Game. In particular, we exploit ideas from pebbling lower bounds to establish upper bounds for the success probability of the Pebbler (who is one of two players). These upper bounds are then, in the second step (§1.2.3), translated to lower bounds on the loss in security games of concrete cryptographic protocols using oracle separation techniques to yield Theorems 1 through 3 stated in §1.1. Before explaining the two steps, we provide a summary of the overall approach so that the two steps, especially the motivation behind some of the underlying definitions, can be better appreciated.

[4] On every signature query issued by the adversary, the reduction in [16] tosses a (biased) random coin (*independent* of the history of the simulation) and depending on its outcome decides whether or not to embed the (RSA) challenge in the signature. The simulation is identical if these coin-tosses are all carried out together at the beginning of the game.

1.2.1 Our Approach

Our goal is to design adversaries that break the GSD game but where any reduction (in a specified class) to the security of the underlying encryption scheme loses a significant (superpolynomial) factor in the advantage. Since we are targeting black-box reductions, we have the luxury of constructing inefficient adversaries. Our adversaries will run in PSPACE and thus will be able to break the underlying encryption scheme. The output of our adversaries will solely depend on the distribution of inconsistent edges in the final key-graph, which we will denote as *pebbles* in the following. Clearly, in order to win the GSD game, our adversaries need to output 0 if the final key-graph is entirely consistent (i.e. contains no pebbles), and 1 if the final key-graph is entirely consistent except for the edges incident on the challenge key. Otherwise, we have complete freedom in assigning output probabilities of 0 and 1 to the remaining pebbling configurations of the final key-graph.

As we prove formally in Section 4, any reduction attempting to take advantage of our adversaries must send it's IND-CPA challenge as a response to a query and exploit the fact that the real and the random challenge will lead to different pebbling configurations of the key-graph. It's hope is that the output distribution of the adversary differs significantly between the two configurations. Note however, that when embedding the challenge in some edge (i, j) of the key-graph, all edges incident to i will, with overwhelming probability, be inconsistent independently of the challenge ciphertext, since the reduction does not know the challenge secret key and thus is unlikely to be able to send consistent responses to queries incident to i. In other words, the challenge can only be embedded into an edge where the edges incident to the source are all pebbled. This naturally leads to studying configurations that are related by valid moves in the *reversible edge-pebbbling* game: a pebble on an edge may only be added or removed if all edges incident to the source are pebbled.

We may now define the configuration graph of our key-graph G: The vertices of the configuration graph $\mathcal{P}^G$, as the name suggests, consist of all possible pebbling configurations of G. Therefore it is the power set of the edges of $G = (\mathcal{V}, \mathcal{E})$. An edge is present from a vertex $\mathcal{P}_i$ to another vertex $\mathcal{P}_j$ if $\mathcal{P}_j$ can be obtained from $\mathcal{P}_i$ using a valid pebbling move. The edges represent pairs of configurations, where the reduction may embed its IND-CPA challenge, in other words, a hybrid (from the reductions point of view). Since we consider reversible pebbling games, the edges in our configuration graphs are undirected. Therefore one can think of $\mathcal{P}^G$ as a subgraph of the Boolean hypercube on $2^{|\mathcal{E}|}$ vertices. Assuming that G has a single sink vertex t, $\mathcal{P}^G$ has two special vertices denoted $\mathcal{P}_{\text{start}} = \emptyset$ and $\mathcal{P}_{\text{target}}$ which consist of the pebbling configuration where all incoming edges to t carry a pebble. The configuration graph for C_4, the path of length 4, is given in Figure 1.(a). A more formal definition is given later in Definition 5 (§2). A path from $\mathcal{P}_{\text{start}}$ to $\mathcal{P}_{\text{target}}$ corresponds to a pebbling sequence in the reversible edge-pebbling game. Any such path can be used for a hybrid argument to prove upper bounds for the loss in security, which is what prior works did [45, 35]. In this work we are interested in ruling out the possibility of using

any of the paths (or multiple at once) to improve on these results.

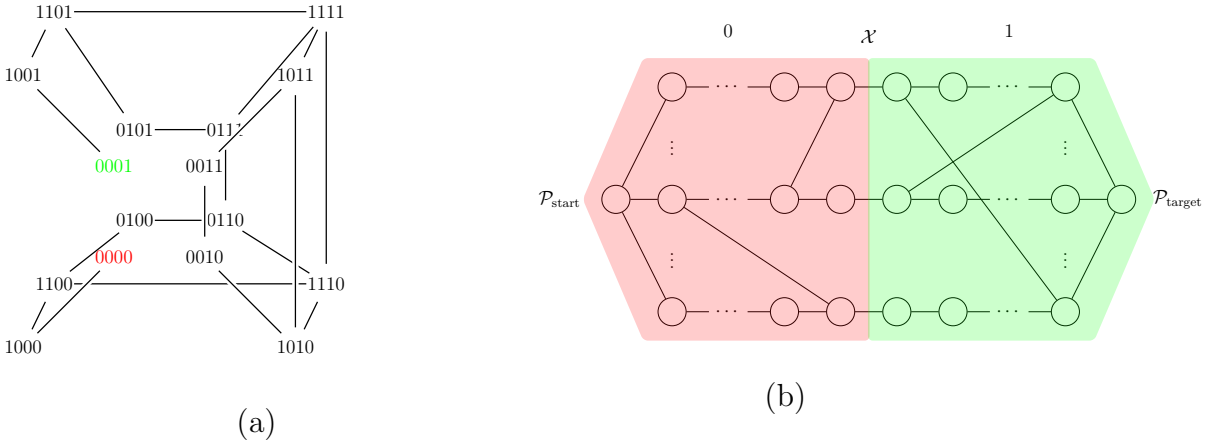

Figure 1: (a) Configuration graph for paths of length 4, $C_4 = ([5], \{(1,2),(2,3),(3,4),(4,5)\})$. The labels of the vertices encode the pebbling status of the corresponding edge: e.g., the vertex labelled 0000 is completely unpebbled whereas the vertex labelled 1000 has a pebble only on the first edge $(1,2)$. The special vertices for $\mathcal{P}^{C_4}$ are $\mathcal{P}_{\text{start}} = 0000$ (red) and $\mathcal{P}_{\text{target}} = 0001$ (green). (b) A schematic diagram showing a configuration graph and its cut. The vertices in the configuration graph consist of all possible pebbling configurations of the graph and therefore the power set of the edges. An edge exists between a configuration $\mathcal{P}_i$ and $\mathcal{P}_j$ if $\mathcal{P}_j$ can be obtained from $\mathcal{P}_i$ via *one* valid pebbling move.

Pebbling lower bounds: Barriers to better cryptographic upper bounds. In our approach, we will show that in *any* sequence of hybrids there exist "bottleneck" configurations related to pebbling lower bounds. These bottleneck configurations define a cut for the configuration graph $\mathcal{P}^G$. Looking ahead, our adversaries will concentrate all their advantage on these cuts and we will show that it is hard for any reduction to guess the pebbled edges of the corresponding pebbling configurations.

For example, let's consider the pebbling lower bound for binary trees. It is known that the number of pebbles that are needed to *node*-pebble a complete binary tree of N vertices is at least $\log N$ (see [51] for example), and the argument can be easily adapted for the case of edge pebbling as follows. Consider a pebbling sequence for a complete binary tree. At the beginning of the sequence *none* of the $N/2$ paths from the root to the leaves carries a pebble; whereas at the end of the sequence – at which point both the edges incident on the root must carry a pebble – *all* the paths from the root to the leaves carry a pebble. Furthermore, by the rules of edge pebbling, only new pebbles on edges going out of the sources, i.e., the leaves, decrease the number of paths that carry a pebble. So any pebbling move can only decrease the number of paths that carry a pebble by one. Therefore there have to exist two consecutive configurations in the pebbling sequence such that in the first configuration there exists a path that does not carry a pebble but in the next configuration every path carries

a pebble. At this point, for each node on the path (except the leaf) there must be at least one pebble on the graph to pebble all paths going through this node via the other in-going edge, and therefore there exists a pebbling configuration where there are at least $\log N$ pebbles. Such pairs of configurations will serve as the cut for the case of binary trees. An illustration can be found in Figure 2 below.

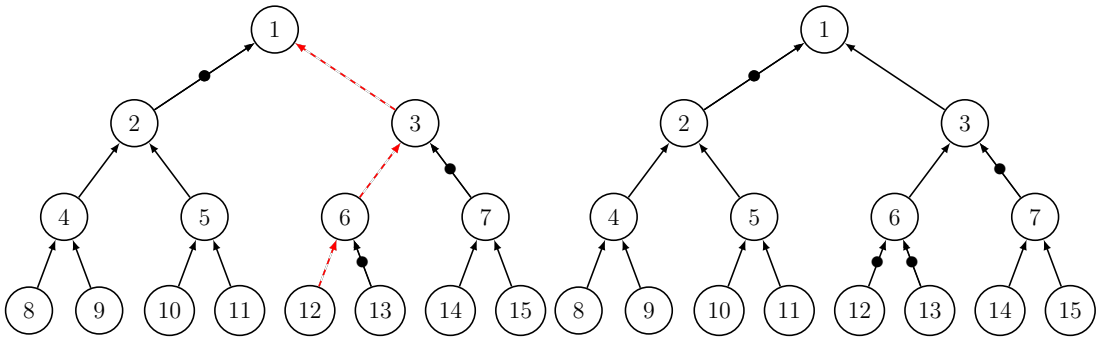

Figure 2: The pebbling configurations used to argue lower bounds for edge-pebbling of a perfect binary tree B_3 of depth 3. In the left configuration there exists a path from a leaf to the root that is *not* covered by a pebble (highlighted by the dashed red path). In the right configuration *all* the paths in B_3 are covered by pebbles. The cut is defined at such two configurations.

From pebbling lower bounds to cryptographic lower bounds via Builder-Pebbler Game. The immediate idea would be to translate pebbling lower bounds directly to cryptographic lower bounds. But pebbling lower bounds apply to fixed graphs. Therefore we are missing a component that captures the dynamic nature of the security games, like that of GSD, which involves (the adversary) choosing a graph G randomly from a class of graphs $\mathcal{G}$. To remedy this, we introduce a two-player multi-stage game that we call the Builder-Pebbler Game and then show that pebbling lower bounds can be used to upper bound the probability of success of the Pebbler (Step I: §1.2.2), one of the players. Then we will use oracle separation techniques to translate these upper bounds into cryptographic lower bounds (Step II: §1.2.3).

1.2.2 Step I: Combinatorial Upper Bounds

We start off with an informal description of the Builder-Pebbler Game, a two-player game that will abstract out the combinatorial aspect of establishing lower bounds for cryptographic protocols that are modelled by multi-user games where the adversary adaptively builds a graph structure among the set of users, as in GSD (formal definition in §3). The game is played between a Pebbler and a Builder, and intuitively, Pebblers play the role of reduction algorithms whereas Builders correspond to adversaries in security games.

Builder-Pebbler Game. For a parameter $N \in \mathbb{N}$, the Builder-Pebbler Game is played between a Builder and a Pebbler in rounds. The game starts with an empty DAG $G = (\mathcal{V} = [1, N], \mathcal{E} = \emptyset)$ and an empty pebbling configuration $\mathcal{P}$, and in each round the following happens: the Builder first picks an edge $e \in [1, N]^2 \setminus \mathcal{E}$ and adds it to the DAG and the Pebbler then decides whether or not to place a pebble on e. This way the Builder and the Pebbler will construct a graph G and a pebbling configuration $\mathcal{P}$ on this graph. The Builder can stop the game at any point by choosing a sink in G as the challenge. This results in a *challenge* DAG $G^* = (\mathcal{V}^*, \mathcal{E}^*)$, the subgraph of G that is induced by all nodes from which the challenge is reachable. The Pebbler wins if it ends up with a "good" pebbling configuration $\mathcal{P}$, a property that is determined by the graph G. Otherwise, the Builder is declared the winner. In case the strategies are randomised, we call the probability with which the Builder (resp., the Pebbler) wins the game as *Builder's (resp., Pebbler's) advantage*, and denote it by $\beta = \beta(N)$ (resp., $\pi = \pi(N)$). We also consider restricted games where the graph G (resp., G^*) comes from a class of graphs $\mathcal{G}$ (resp., $\mathcal{G}^*$). Therefore, in summary, one can think of the game as the Builder building a graph and the Pebbler placing pebbles on this graph with the aim of getting into a good configuration and the Builder preventing this from happening.[5]

Defining good configurations via cuts of configuration graph. Although the Builder-Pebbler Game is meaningful for any notion of "good" configuration, we are interested in a particular definition that is essential in establishing our cryptographic lower bounds: we will set the good configurations as the ones that belong to bottleneck configurations in the configuration graph of G. We prove that it will be difficult for oblivious Pebblers to get into such configurations and therefore they will serve as the winning condition for our Builder-Pebbler Game.

Combinatorial Upper Bounds in the Builder-Pebbler Game. We bound the advantage of oblivious Pebblers from above for classes of graphs that are interesting from a practical point of view, and this includes paths, binary trees and complete graphs. The notion of obliviousness for Pebblers is naturally derived from the one for reductions, see discussion above and Definition 10.

Theorem 4. *(Informal) Any oblivious Pebbler in the Builder-Pebbler Game restricted to* paths or binary trees *has advantage at most inverse* quasi-polynomial *in the size*

[5]This is reminiscent of Maker-Breaker games [31], a class of positional games (which includes Shannon Switching Game, Tic-Tac-Toe and Hex) which are played between a Maker, who is trying to end up with a (winning) position and a Breaker, whose goal is to prevent the Maker from getting into such (winning) positions. One fundamental difference between Maker-Breaker Games and the Builder-Pebbler Game is that in Maker-Breaker games one usually considers optimal (deterministic) strategies, whereas we consider randomised strategies for Builder-Pebbler Game. (Another way of looking at this is that our "board" is dynamic.) Another difference is the asymmetry in the nature of moves.

of the graph.

Theorem 5. *(Informal) Any oblivious Pebbler in the Builder-Pebbler Game restricted to* complete (directed acyclic) graphs *has advantage at most inverse* sub-exponential *in the size of the graph.*

The upper bounds in Theorems 4 and 5 are (almost) tight since a random Pebbler yields (almost) matching lower bounds.

A notable quirk in our proof is that all our Builder strategies will also be "oblivious", where oblivious is defined different for Builders than for Pebblers: it means that the query strategy is independent of the Pebbler's responses.[6] The reason we restrict ourselves to such Builders is mostly for our convenience: looking ahead, it means that we can ensure that the reductions in our cryptographic applications cannot exploit the querying behaviour of the adversary to gain a larger advantage, rather they must rely solely on the final output bit. Extending our results to non-oblivious reductions/Pebblers might require to consider also non-oblivious adversaries/Builders, but then the separation argument in the next section might become even more subtle.

1.2.3 Step II: From Combinatorial Upper Bounds to Cryptographic Lower Bounds

For translating upper bounds established in Step I into loss in security games of concrete cryptographic protocols, we adapt ideas from oracle separations. We explain the ideas involved below, using GSD as a running example.

Oracle separations. Oracle separations are traditionally used to rule out reduction of one primitive to another. The approach suggested in [34] to rule out black-box constructions of a primitive P from another primitive Q is to come up with an oracle O relative to which Q exists, but every black-box implementation of P is broken. As pointed out in [49], this approach actually rules out a larger class of reductions (so-called "relativized" reduction). Our setting is slightly different from above in the sense that we are interested in a primitive Q that is used in an adaptive "multi-user" setting P, and we would like to bound the loss in security (i.e., λ) incurred in reducing P to Q. Moreover, we are interested (for a start) in ruling out just fully black-box constructions. Our idea is to design an oracle O that consists of two parts: an idealised implementation of the primitive Q and an oracle A_P that breaks the multi-user protocol P. The key is to argue that the implementation of Q remains "secure" even in the presence of A_P[7], and to argue this we rely on the combinatorial upper bounds established in Step I. To be precise, we show that the existence of

[6] One could think of the Builder playing the role of "nature" (who also adopts a strategy that is oblivious of the opposing player) in Papadimitrou's *Games Against Nature* [46].

[7] This is reminiscent of the so-called two-oracle paradigm [33] which is also used to rule out fully black-box reduction of one primitive from another.

a reduction R which, when given oracle access to an adversary A_P that breaks the protocol P, breaks the primitive Q with a loss at most λ implies the existence of a corresponding Pebbler P in the Builder-Pebbler Game that has advantage at least $\pi = \Omega(1/\lambda)$. Since the loss in security incurred by R and advantage of P are tightly related (to within an additive factor), any reduction R that beats the claimed loss in security violates the upper bounds we established on the Builder-Pebbler Game. In the remainder of the section, we explain the working of the adversary A_P.

The threshold adversary. Suppose that the protocol is implemented on a class of graphs $\mathcal{G}$. Intuitively, the adversary A_P "simulates" an oblivious Builder in the Builder-Pebbler Game played on graphs in $\mathcal{G}$. That is, it chooses a graph $G \in \mathcal{G}$ uniformly at random and then proceeds to query its edges one at a time (and at the end of the security game challenges on a random component). In order for it to be a valid adversary it must break the implementation of P (with a non-negligible probability). To this end, we design it such that it concentrates all its distinguishing advantage in a cut in the configuration graph of G. This requires A_P to distinguish pebbled edges from non-pebbled edges and inferring the exact pebbling configuration that it receives from R in the simulation. Therefore it must be powerful enough to brute force the implementation of Q (this can be carried out, e.g., in PSPACE). Once it knows the final pebbling configuration, it outputs 0 if it lies (topologically) to the "left" of the cut and 1 for every configuration to the "right" of the cut: see Figure 1.(b). That is, it "thresholds" at the cut and therefore we call A_P a *threshold adversary*. For the reduction R to have any chance of solving its own challenge c, this challenge must be embedded in a configuration in this cut. However, intuitively this means that R will implicitly win the Builder-Pebbler Game. We show this formally by arguing that for every reduction R there exists a Pebbler strategy P with similar success probability. This step crucially relies on the fact that the primitive Q is ideally implemented. It follows that an upper bound on the success probability of the Pebbler in the Builder-Pebbler Game translates to a lower bound on the loss in security for the reduction R (i.e., λ).

Example: GSD. For concreteness, let's consider GSD on binary trees, in which case Q is a symmetric-key encryption scheme and P corresponds to the GSD game described in the introduction but restricted to being played on binary trees. The ideal implementation of Q in the separation consists of two oracles $(\mathsf{Enc}, \mathsf{Dec})$. Here Enc is a random expanding function (expanding by a factor of 6, say) which is injective with overwhelming probability; Dec is defined to be "consistent" with Enc. This means that (i) it is difficult for any query-bounded algorithm to guess any values (e.g., key or ciphertext) that it has not queried before; and (ii) an all-powerful adversary can tell whether ciphertexts correspond to pebbled (consistent) or unpebbled (inconsistent) edges (there are some technicalities that are addressed later in §4.4). A_P is the all-powerful threshold adversary: i.e., it plays the GSD game on the binary tree (picking

edges at random) and for every ciphertext obtained it checks whether it corresponds to a pebbled ciphertext or not. At the end of the GSD game, it ends up with a pebbling configuration for the binary tree, and outputs 0 if the configuration is on the "left" of the cut and 1 otherwise. For any reduction R to take advantage of A_P, it must embed the challenge ciphertext c^* in the cut, and if it manages this with a loss that is significantly less than quasi-polynomial in N, it would imply the existence of a Pebbler that is successful with a probability greater than inverse quasi-polynomial, a contradiction to Theorem 4.

1.3 Related Work

1.3.1 Adaptive Security.

The security of multi-party computation in the context of adaptive corruption has been well studied. It is known that a protocol that is proven secure against static (i.e., non-adaptive) adversaries may turn out insecure once the adversary is allowed adaptive corruption [13]. On the other hand, in the (programmable) random oracle model it *is* possible to compile a selective protocol into an adaptively-secure one through non-committing encryption [43].

The notion of generalised selective decryption (GSD) was introduced by Panjwani [45] to study adaptive corruption in less general settings. His motivation was to better understand the problem of selective decommitment [17] (which is also known as selective opening in some works [7]) and the closely-related problem of selective decryption. The problem was further studied by Fuchsbauer et al. [20] who gave a quasi-polynomial reduction when the GSD game is restricted to trees.

In parallel, the study of adaptive security in the setting of circuit garbling was undertaken in the works of Bellare et al. [6], Hemenway et al. [32] and Jafargholi and Wichs [36]. The latter two works are especially relevant since they established a relationship between adaptive security and a particular notion of pebbling. It is also worth noting that the study of adaptive security of garbled RAM was carried out in [24, 23].

The above two series of works culminated in the Piecewise-Guessing Framework of Jafargholi et al. [35] who managed to abstract out the ideas therein and give even more fine-grained reductions. In addition to capturing the results from [36, 20, 22], they applied the framework to obtain new results for adaptive secret sharing. The framework was further applied to argue adaptive security for attribute-based encryption schemes [39], proxy re-encryption schemes [21], continuous group key-agreement [1, 2] and non-interactive zero-knowledge [37].

1.3.2 Black-Box Separations.

The study of limitations of black-box reductions was initiated in the seminal work of Impagliazzo and Rudich [34]. They used the notion of *oracle separations* to rule out

black-box reductions of key agreement to symmetric-key primitives. This approach turned out quite useful and has been further exploited to rule out black-box constructions of a variety of cryptographic primitives from one another (e.g., [50, 52]). A fine-grained study of the notion of black-box reductions and oracle separations was later carried out by Reingold et al. [49].

In addition to ruling out reductions, oracle techniques have also been used to study the efficiency of a construction of one primitive from another [26, 25, 38]. This has been applied to the case of adaptive security as well. Perhaps the works most relevant to ours is that of Lewko and Waters [41], who showed that the security of adaptively-secure hierarchical identity-based encryption must degrade exponentially in depth, and Fuchsbauer et al. [22], who showed that certain types of constrained PRFs must incur an exponential loss (in the size of the input) in adaptive security. Both these works employ the more recent meta-reduction technique [12, 27, 47], which is of different flavour from oracle separations. Our work is thus similar in spirit to [26, 25, 38].

1.3.3 Graph Pebbling.

The notion of graph pebbing, first introduced in the 70's to study programming languages, turned out quite useful in computational complexity theory to study the relationship between space and time; in recent years, pebbling has found applications in cryptography as well [18, 19, 3]. The notion of node pebbling first appeared (albeit implicitly) in [48], whereas the notion of *reversible* node pebbling was introduced by Bennett to study reversible computation [8]. The notion of edge pebbling used in this work is defined in [35]. The lower bound on the reversible node pebbling complexity of paths was established by Chung et al. [14] and an alternative proof can be found in [40]. As for the lower bound on the node pebbling complexity for binary trees, a proof can be found in [51]. We refer the reader to the textbook by Savage [51] or the excellent survey by Nordström [44] for more details on pebbling.

2 Notation and Definitions

We use the notation $[N] = \{1, \ldots, N\}$ and $[N]_0 = \{0\} \cup [N]$. Let $N \in \mathbb{N}$ and $G = (\mathcal{V}, \mathcal{E})$ define a directed acyclic graph (DAG) with vertex set $\mathcal{V} = [N]$, edge set $\mathcal{E} \subset [N] \times [N]$, and a unique sink $\mathcal{T}$. For a subset $\mathcal{S} \subset [N]$ of nodes, let $\text{in}(\mathcal{S})$ denote the set of ingoing edges and $\text{parents}(\mathcal{S})$ denote the set of parent nodes of nodes in $\mathcal{S}$. For a set of n edges $\mathcal{P} = \{(v_i, w_i)\}_{i=1}^n$, let $\mathcal{V}(\mathcal{P}) := \bigcup_{i=1}^n \{v_i, w_i\}$ denote the set of nodes that have an incident edge in $\mathcal{P}$. The edge set $\mathcal{P}$ is called *disjoint*, if they do not share a node, i.e. if $|\mathcal{V}(\mathcal{P})| = |\bigcup_{i=1}^n \{v_i, w_i\}| = 2n$. We denote by $E(G)$ the number of edges in G and by $D(G)$ the maximal number of disjoint edges in G.

Definition 1 (cuts, cut-sets, frontiers)**.** Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected graph. A *cut* $\mathcal{S}$ of G is a subset of the nodes $\mathcal{V}$. For two nodes $v_1, v_2 \in \mathcal{V}$ an *s-t-cut that*

separates v_1 *and* v_2 is a cut $\mathcal{S}$ such that $v_1 \in \mathcal{S}$ and $v_2 \notin \mathcal{S}$. The *cut-set* of a cut $\mathcal{S}$ is the set of edges with one endpoint in $\mathcal{S}$ and the other outside of $\mathcal{S}$. We call the *frontier of a cut* $\mathcal{S}$ the set of all nodes in $\mathcal{S}$ that have an incident edge in the cut-set of $\mathcal{S}$.

Definition 2 (Vertex Covers)**.** Let $G = (\mathcal{V}, \mathcal{E})$ be a directed or undirected graph and $\mathcal{P} \subset \mathcal{E}$ be a subset of edges. A *vertex cover* of $\mathcal{P}$ is a subset $\mathcal{S}$ of $[N]$ such that for each edge $(i, j) \in \mathcal{P}$ either the source i or the sink j lies in $\mathcal{S}$. We define a *non-trivial* vertex cover to be a vertex cover $\mathcal{S}$ such that $\mathcal{S} \subset \mathcal{V}(\mathcal{P})$. We denote the size of a minimal vertex cover of $\mathcal{P}$ by

$$\mathsf{VC}(\mathcal{P}) := \min\{|\mathcal{S}| : \mathcal{S} \subset [N] \text{ covers } \mathcal{P}\}.$$

A *pebbling configuration* on the graph $\mathcal{G}$ is a set $\mathcal{P} \subset \mathcal{E}$ defining the subset of pebbled edges. Let $|\mathcal{P}|$ denote the number of pebbles in the configuration and $\mathcal{V}(\mathcal{P})$ the set of nodes involved in the pebbling. We define the *complexity* of a pebbling configuration $\mathcal{P}$ as the size of a minimal vertex cover of $\mathcal{P}$. For a pebbling sequence $\boldsymbol{\mathcal{P}} = (\mathcal{P}_0, \ldots, \mathcal{P}_L)$, we define $\mathsf{VC}(\boldsymbol{\mathcal{P}}) := \max_{i \in [L]_0} \mathsf{VC}(\mathcal{P}_i)$.

Let $\mathcal{P}_{\text{start}}$ denote the unique configuration with $|\mathcal{P}_{\text{start}}| = \mathsf{VC}(\mathcal{P}_{\text{start}}) = 0$, i.e., $\mathcal{P}_{\text{start}} = \emptyset$, and $\mathcal{P}_{\text{target}} = \text{in}(\mathcal{T}) = \{(i, \mathcal{T}) \in \mathcal{E}\}$ denote the configuration where only all the edges incident on the sink $\mathcal{T}$ are pebbled. We will consider sequences of pebbling configurations $\boldsymbol{\mathcal{P}} = (\mathcal{P}_{\text{start}}, \ldots, \mathcal{P}_{\text{target}})$ where subsequent configurations have to follow certain pebbling rules.

Reversible Pebbling. We consider the reversible edge-pebbling game from [35].

Definition 3 (Edge-Pebbling)**.** An edge pebbling of a DAG $G = (\mathcal{V}, \mathcal{E})$ with *unique sink* $\mathcal{T}$ is a pebbling sequence $\boldsymbol{\mathcal{P}} = (\mathcal{P}_0, \ldots, \mathcal{P}_\ell)$ with $\mathcal{P}_0 = \mathcal{P}_{\text{start}}$ and $\mathcal{P}_\ell = \mathcal{P}_{\text{target}}$, such that for all $i \in [\ell]$ there is a unique $(u, v) \in \mathcal{E}$ such that:

- $\mathcal{P}_i = \mathcal{P}_{i-1} \cup \{(u, v)\}$ or $\mathcal{P}_i = \mathcal{P}_{i-1} \setminus \{(u, v)\}$,
- $\text{in}(u) \subset \mathcal{P}_{i-1}$.

For some applications, we will actually consider the classical *reversible node-pebbling* as in [8], where a node is deemed pebbled whenever all ingoing edges are pebbled and two subsequent pebbling configurations differ by a node. Note that any node-pebbling sequence induces an edge-pebbling sequence, so we view node-pebbling as a more restricted version of edge pebbling.

Definition 4 (Node-Pebbling)**.** A node pebbling of a DAG $G = (\mathcal{V}, \mathcal{E})$ with *unique sink* $\mathcal{T}$ is a pebbling sequence $\boldsymbol{\mathcal{P}} = (\mathcal{P}_0, \ldots, \mathcal{P}_\ell)$ with $\mathcal{P}_0 = \mathcal{P}_{\text{start}}$ and $\mathcal{P}_\ell = \mathcal{P}_{\text{target}}$, such that for all $i \in [\ell]$ there is a unique $v \in [N]$ such that:

- $\mathcal{P}_i = \mathcal{P}_{i-1} \cup \text{in}(v)$ or $\mathcal{P}_i = \mathcal{P}_{i-1} \setminus \text{in}(v)$,

- for all $u \in \text{parents}(v)$: $\text{in}(u) \subset \mathcal{P}_{i-1}$.

Definition 5 (Configuration Graph)**.** Let $G = (\mathcal{V}, \mathcal{E})$ be some graph. We define the associated *configuration graph* $\mathcal{P}^G$ as the graph that has as its vertex set all $2^{|\mathcal{E}|}$ possible pebbling configurations of G. The edge set will contain an edge between two vertices, if the transisition between the two vertices is an allowed pebbling move according to the pebbling game rules.

Note that the configuration graph depends on the pebbling game. If we consider reversible pebbling as in Definitions 3 and 4, the graph is undirected.

3 Builder-Pebbler Game

In this work, we consider security games for multi-user schemes where an adversary can adaptively do the following actions:

- query for information between pairs of users,
- corrupt users and gain secret information associated to these users,
- issue a distinguishing challenge query associated to a target user of its choice,
- guess a bit $b \in \{0, 1\}$.

We consider such games as games on graphs, where users represent the nodes of the graph and edges are defined by the adversary's pairwise queries. If the pairwise information depends asymmetrically on the two users, then this is represented by the direction of the corresponding edge and after the game one can extract a directed graph structure from the transcript of the game. Here, we only consider the case of directed *acyclic* graphs, i.e., where the adversary is forbidden to query cycles. Furthermore, to avoid trivial winning strategies, the adversary must not query a challenge on a node which is reachable from a corrupt node.
To prove a scheme secure under such an adaptive game based on standard assumptions (e.g., the security of some involved primitive), a common approach is to construct a reduction that has black-box access to an adversary against the scheme and tries to use the advantage of this adversary to break the basic assumption. To this aim, the reduction has to simulate the game to the adversary and at the same time embed some challenge c on the basic assumption into its answers so that the adversary's output varies depending on this embedded challenge. Hence, the reduction might not answer all queries correctly but rather "fakes" some of the edges; such wrong answers will be represented as *pebbled* edges in the graph. However, if the reduction answers all queries connected to the challenge node independent of the challenge user's secrets, then the edge queries do not help the adversary to distinguish its challenge and its advantage in this game can be at most the advantage it has in an

alternative security game where no edge queries are possible. Indistinguishability in such a weaker scenario usually follows trivially by some basic assumption.

Thus, we are interested in games that can be abstracted by the following two-player game.

Definition 6 (N- and $(N, \mathcal{G}, \mathcal{G}^*)$-Builder-Pebbler Game)**.** For a parameter $N \in \mathbb{N}$, the N-Builder-Pebbler Game is played between two players, called Builder and Pebbler, in at most $N \cdot (N-1)/2$ rounds. The game starts with an empty DAG $G = ([1, N], \mathcal{E} = \emptyset)$ and an empty set $\mathcal{P} = \emptyset$. In each round:

1. the Builder first picks an edge $e \in [1, N]^2 \setminus \mathcal{E}$ and adds it to G (i.e., $\mathcal{E} := \mathcal{E} \cup \{e\}$); the Builder is restricted to only query edges that do not form cycles; and
2. the Pebbler then either places a pebble on e (i.e., $\mathcal{P} := \mathcal{P} \cup \{e\}$) or not (i.e., $\mathcal{P}$ remains the same).

the Builder can stop the game at any point by choosing a sink in G as the challenge. This results in a *challenge* DAG $G^* = (\mathcal{V}^*, \mathcal{E}^*)$, the subgraph of G that is induced by all nodes from which the challenge is reachable.
In an $(N, \mathcal{G}, \mathcal{G}^*)$-Builder-Pebbler Game, the Builder is restricted to building $G \in \mathcal{G}$ and setting a challenge DAG $G^* \in \mathcal{G}^*$, for classes of graphs $\mathcal{G}$ and $\mathcal{G}^*$.

Definition 7 (Winning Condition and Advantage for $(N, \mathcal{G}, \mathcal{G}^*)$-Builder-Pebbler Game)**.** Consider an $(N, \mathcal{G}, \mathcal{G}^*)$-Builder-Pebbler Game and let G, $G^* = (\mathcal{V}^*, \mathcal{E}^*)$ and $\mathcal{P}$ be as in Definition 6. We model the winning condition for the game through a function X that maps a graph to a collection of subsets of its own edges. We say that the Pebbler wins the $(N, \mathcal{G}, \mathcal{G}^*)$-Builder-Pebbler Game under winning condition X if the following two conditions are satisfied:

1. only edges in $\mathcal{E}^*$ are pebbled, i.e., $\mathcal{P} \subseteq \mathcal{E}^*$; and
2. the pebbling satisfies the winning condition, i.e., $\mathcal{P} \in X(G^*)$.

Otherwise, the Builder is declared the winner. In case the strategies are randomised, we call the probability with which the Builder (resp., Pebbler) wins the game the *Builder's (resp., Pebbler's) advantage*, and denote it by $\beta = \beta(N)$ (resp., $\pi = \pi(N)$). Since there are no draws, we have $\beta + \pi = 1$.

Remark 1. The corresponding definitions for N-Builder-Pebbler Game can be obtained by simply ignoring the restrictions to $\mathcal{G}$ and $\mathcal{G}^*$.

In our setting we will be interested in functions X that output sets of vertices that represent the frontier of a cut in the configuration graph of the input.

Definition 8 (Cut Function)**.** For a family $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of graphs, a function $X : \mathcal{G} \mapsto 2^{\mathcal{E}}$ is called a *cut function* if $X(G)$ is the frontier of an s-t-cut of the configuration graph $\mathcal{P}^G$ that separates $\mathcal{P}_{\text{start}}$ from $\mathcal{P}_{\text{target}}$ for any input $G \in \mathcal{G}$.

Oblivious player strategies. In this work we restrict player strategies for both the Builder and Pebbler to be oblivious. Note that the notion of obliviousness differs for Builder and Pebbler, i.e. they are oblivious in their own way.[8]

Definition 9 (Oblivious Builder Strategy)**.** We say that a Builder's strategy in the $(N, \mathcal{G}, \mathcal{G}^*)$-Builder-Pebbler Game is *oblivious* if its choice of graph $G \in \mathcal{G}$ and order of edge queries are *independent* of (i.e. oblivious to) the Pebbler's strategy.

This restriction on the Builder ensures that the Builder-Pebbler Game game is not trivial for the cut functions we are interested in. If this was not in place, it is easy to come up with Builder strategies in which any Pebbler has advantage 0. However, such strategies are not interesting in our setting, since they would allow reductions to exploit the query behaviour or choice of graph to gain advantage in their own security game. This restriction does not weaken our result, since we are constructing lower bounds for reductions.

Definition 10 (Oblivious Pebbler Strategies)**.** We say that a Pebbler's strategy is *oblivious* if it fixes a subset of vertices $\mathcal{S} \subseteq [1, N]$ at the beginning of the game, and at the end of the game $\mathcal{S}$ is always a non-trivial vertex cover of the pebbling $\mathcal{P}$.

Most of our results also require the Pebbler to be oblivious, but in a different way: it may use the a priori knowledge of the graph structure during the query phase. We capture this by requiring the Pebbler to commit to a non-trivial vertex cover of the pebbling configuration. This allows to compress pebbling configurations based on the graph structure: if the Pebbler knows that the graph contains nodes with high degree and it aims to pebble all (or some) of the incident edges of such a node, it may guess this node ahead of time and then adjust its query responses assuming the guess is correct. In the known upper bounds for the applications we consider this is used to compress the amount of information that needs to be guessed ahead of time. The fact that the vertex cover is required to be non-trivial ensures that this restriction is also non-trivial: otherwise, the Pebbler may simply output the entire set $[N]$. On the other hand, using a minimal vertex cover seems too strong, since we do not actually require it to prove our bounds.

In contrast to obliviousness of the Builder, the restriction to oblivious Pebblers in our lower bounds (with the exception of Corollary 7) does weaken our results, but we do not know how to circumvent it. We do not even dare to venture a guess if better upper bounds can be obtained by non-oblivious reductions/Pebblers or if our lower bounds can be improved. In any case, this restriction is weak enough to still capture all known reductions.

[8] We considered changing the name of oblivousness of at least one of the players to make the distinction clearer, but did not find another suitable term, since both concepts capture a kind of obliviousness. So we stick with this defintion and simply hope it does not cause too much confusion.

4 Lower Bounds for Edge Pebbling

In this section we consider edge pebbling (Definition 3) and cryptographic applications that consider this pebbling strategy for establishing upper bounds on loss in security (e.g., GSD). In the first step, we consider the Builder-Pebbler Game for different classes of graphs like paths (§4.1), binary trees (§4.2) and arbitrary graphs (§4.3), and establish upper bounds on the advantage $\pi(N)$ of the Pebbler. In all the cases, we show that for appropriately-chosen cut functions, an *oblivious* Pebbler has at most a negligible chance of winning against a random Builder. Then in §4.4 we translate these upper bounds into cryptographic lower bounds by showing that an "oblivious" reduction in the GSD game must lose at least a factor $\lambda(N) := \Omega(1/\pi(N))$. We defer the formal definition of an oblivious GSD reduction to §4.4, but loosely speaking such reductions are *oblivious* to the adversary's behaviour in the sense that they do not make use of the partial graph structure they get to know throughout the game but rather stick to some simulation strategy that is independent of the history of the adversary's queries (like, e.g., in the piecewise guessing framework).

4.1 Combinatorial Upper Bound for Paths

For the Builder-Pebbler Game restricted to paths of length N (i.e., the class $\mathcal{C}_N$), we show that any oblivious Pebbler playing the Builder-Pebbler Game against a random Builder with a definition of cut that is closely related to pebbling lower bounds for paths (see §4.1.1) has advantage at most quasi-polynomial in N (§4.1.2). We exploit the observation that whenever the Pebbler behaves obliviously and the Builder queries edges uniformly at random, the nodes from the vertex cover will be uniformly distributed on the path. Our result matches the best known Pebbler strategy (of simply guessing the nodes in the cut) that has an advantage $\pi \geq 1/N^{\log N}$ up to constant factors in the exponent.

4.1.1 Pebbling Characteristics of Paths.

To define a suitable cut in the configuration graph, we use a known lower bound on the number of pebbles needed to reversibly pebble a path [14]: For any $k \geq 1$ and every pebbling sequence $\boldsymbol{\mathcal{P}}_k = (\mathcal{P}_{\text{start}}, \ldots, \mathcal{P}_k)$, where $(2^k, 2^k + 1) \in \mathcal{P}_k$, it must hold $|\boldsymbol{\mathcal{P}}_k| := \max\{|\mathcal{P}| \mid \mathcal{P} \in \boldsymbol{\mathcal{P}}_k\} \geq k + 1$. One can prove this by induction: First, note that pebbling the second edge $(2, 3)$ requires 2 pebbles. Now assume the claim is true for $k - 1$ with $k > 1$. Clearly, any valid pebbling sequence $\boldsymbol{\mathcal{P}}_k$ must contain a configuration where the 2^{k-1}th edge is pebbled for the first time, i.e., the 2^{k-1}th edge is pebbled and all subsequent edges are unpebbled. Assume $|\boldsymbol{\mathcal{P}}_k| \leq k$ and consider the following two cases: Either the 2^{k-1}th edge remains pebbled until the 2^kth edge is pebbled, which would immediately imply a pebbling strategy to pebble the 2^{k-1}th edge using only $k - 1$ pebbles – a contradiction. Or the pebble on the 2^{k-1}th edge is removed while there is at least one pebble on some subsequent edge (to guarantee

progress), which would imply that the pebble on the 2^{k-1}th edge can be removed using only $k-2$ additional pebbles – again a contradiction due to the reversible pebbling rules. This proves the claim. The above lower bound is indeed tight and a matching reversible pebbling strategy can be found, for example, in [8].

In particular, for all valid edge pebbling sequences $\boldsymbol{\mathcal{P}} = (\mathcal{P}_{\text{start}}, \ldots, \mathcal{P}_{\text{target}})$ of a path on $N = 2^n + 1$ nodes, with $\mathcal{P}_{\text{start}} = \emptyset$ and $\mathcal{P}_{\text{target}}$ being the configuration where only the last edge is pebbled, there must exist a pebbling configuration $\mathcal{P} \in \boldsymbol{\mathcal{P}}$ such that $|\mathcal{P}| = \lfloor \log(N) \rfloor + 1$. Thus, we define a cut in the configuration graph as follows:

Definition 11 (Good pebbling configurations, cuts and cut function for paths)**.** We call a pebbling configuration $\mathcal{P}$ for a path $C = C_N$ of on N nodes *good* if it contains $\lfloor \log N \rfloor$ pebbles and there exists a valid pebbling sequence $\boldsymbol{\mathcal{P}} = (\mathcal{P}_{\text{start}}, \ldots, \mathcal{P})$ such that $|\boldsymbol{\mathcal{P}}| = \lfloor \log N \rfloor$. We define a *cut set* $\mathcal{X}$ in the configuration graph $\mathcal{P}^C$ as the set of all edges consisting of a good pebbling configuration and a configuration which can be obtained from this good configuration by *adding* one pebble (following the pebbling rules). The cut function $X_{\mathcal{C}}$ is defined as in Definition 8 as the frontier of this cut.

Remark 2. A complete characterisation of such reachable configurations is given in [42]. Let the pebbles in a configuration $\mathcal{P}$ be $\{(v_i, v_i + 1)\}_{i \in [1, \log(N)]}$ for $v_i \in [0, N-1]$. Then $\mathcal{P}$ is reachable if and only if for every $i \in [1, \log(N)]$, $\mathcal{P}$ has a pebble in the range $\{(v_i - 2^i, v_i - 2^i + 1), \ldots, (v_i - 1, v_i)\}$.

4.1.2 The Upper Bound.

Since we consider oblivious Pebbler strategies, this means that a successful Pebbler must choose a vertex cover $\mathcal{S} \subset [N]$ such that each node in $\mathcal{S}$ is either source or sink of a pebbled edge in $\mathcal{P}$. If the adversary queries a uniformly random path on $[N]$, then $\mathcal{S}$ will be a uniformly random subset of nodes. Obviously, we must have $(\log N)/2 \leq |S| \leq 2 \log N$. In the following Lemma we bound the probability that a uniformly random subset $\mathcal{S}$ of nodes of some fixed size $s \in [(\log N)/2, 2 \log N]$ is a vertex cover of a good configuration $\mathcal{P}$ and $\mathcal{S}$ is a subset of the nodes $V(\mathcal{P})$ involved in $\mathcal{P}$.

Lemma 1. *Let $\mathcal{S} \subset [N]$ be a uniformly random subset of size $s \in [(\log N)/2, 2 \log N]$, $\sigma = \min\{s, \log N\}$, and P be the set of good pebbling configurations on paths on N nodes. Then*

$$\mathsf{Pr}[\exists \mathcal{P} \in P : \ \mathcal{S} \textit{ covers } \mathcal{P} \ \wedge \ \mathcal{S} \subset V(\mathcal{P})] \leq \frac{s^{2s}}{N^{s-\sigma} 2^{\sigma(\sigma+1)/2}} \leq \frac{N^{\log \log N}}{N^{\log(N)/8}}.$$

Proof. We call $\mathcal{S}$ *good* if it covers a good pebbling configuration $\mathcal{P}$ and $\mathcal{S} \subset V(\mathcal{P})$. First, we count the number of subsets of size s which are good. To this aim, note that since we consider reversible pebbling, a configuration $\mathcal{P}$ with $|\mathcal{P}| = \log N$ is good if

and only if all pebbles can be removed without the need of any additional pebbles.
Now, assume $\mathcal{S}$ covers a good pebbling configuration $\mathcal{P}$ and $\mathcal{S} \subset V(\mathcal{P})$. If $s \geq \log N$, then it must be the case that there are $\bar{s} \geq s - \log N$ pairs of nodes in $\mathcal{S}$ such that both nodes cover the same edge in $\mathcal{P}$, respectively, and one node from each pair can be removed from $\mathcal{S}$ such that the remaining set $\mathcal{S}' \subset S$ still covers $\mathcal{P}$. Let $\bar{s}$ be maximal with this property, hence $\mathcal{S}'$ a minimal vertex cover of $\mathcal{P}$; we denote its size by $s' = s - \bar{s}$. Clearly $s' \leq \log N$, and there must exist $\log N - s'$ nodes in $\mathcal{S}'$ which each cover two edges and the pairs of consecutive edges are pairwise disjoint.
Considering the edges in $\mathcal{P}$ to be pebbled, one pebble of each such pair of consecutive pebbles can be removed trivially from the graph. These $\log N - s'$ pebbles can now be used to remove further pebbles. Note, in general, using k pebbles, one can remove a pebble at distance at most 2^k from its predecessor. This in particular implies that the set $\mathcal{S}'$ must contain a pair of nodes u_1, v_1 that have distance at most $2^{\log N - s'}$ in the path. After removing the pebble incident on node v_1, we have one more pebble at our disposal to remove a further pebble incident on a node in $\mathcal{S}' \setminus \{v_1\}$ at distance $\leq 2^{\log N - s' + 1}$ of its predecessor in $\mathcal{S}' \setminus \{v_1\}$ on the path. Pursuing this idea, in the kth step, there must be a node at distance $\leq 2^{k + \log N - s'}$ from its preceding node on the path. In total, there are $\log N - s'$ gaps between nodes in $\mathcal{S}'$, where one gap is of size $\in [2^{\log N - s'}]$, another one is of size $\in [2^{\log N - s' + 1}]$, another one of size $\in [2^{\log N - s' + 2}]$, and so on, up to size $\in [N/2]$.[9]
In total, for the s gaps on the path between the nodes in $\mathcal{S}$ it holds: $s - \sigma \leq \bar{s}$ gaps must be of size 1, the remaining $\sigma - s'$ gaps of size 1 are in particular one gap of size $\in [2^{\log N - \sigma}]$, one of size $\in [2^{\log N - \sigma + 1}]$, and so on, up to size $\in [2^{\log N - s' - 1}]$. For the remaining s' gaps, as stated above, there must be one gap of size $\in [2^{\log N - s'}]$, one of size $\in [2^{\log N - s' + 1}]$, up to size $\in [N/2]$. Thus, independent of $\bar{s}$, there must be $s - \sigma$ gaps of size 1 and σ gaps of sizes $\in [2^{\log N - \sigma + k - 1}]$ for $k \in [\sigma]$, respectively.
Thus, we can upper bound the number of good subsets of size s as the number of possible subsets of s nodes having the required gap sizes on the path as discussed. Of course, the s gaps do not need to be in order, so we get an upper bound on the number of different good subsets $\mathcal{S}$ by

$$\#\text{ good subsets } \leq s! \cdot \prod_{k=0}^{\sigma-1} 2^{\log N - \sigma + k} \leq s^s \cdot 2^{\sum_{k=\log N - \sigma}^{\log(N)-1} k} = s^s \cdot 2^{\sigma(2\log N - \sigma - 1)/2}.$$

On the other hand, the total number of subsets of s nodes is $\binom{N}{s}$. Thus, we can upper bound the probability of $\mathcal{S}$ being good by

$$\mathsf{Pr}[\mathcal{S} \text{ is good}] \leq \frac{s^s \cdot 2^{\sigma(2\log N - \sigma - 1)/2}}{\binom{N}{s}} \leq \frac{s^s \cdot 2^{\sigma \log N - \sigma(\sigma+1)/2} \cdot s^s}{N^s} \leq \frac{s^{2s}}{N^{s-\sigma} 2^{\sigma(\sigma+1)/2}}.$$

[9] Note, we also consider the distance between the source node and the first node in $\mathcal{S}$ on the path as a gap.

This upper bound is maximal when $s = (\log N)/2$, where it attains

$$\frac{((\log N)/2)^{\log N}}{2^{(\log N)/2\cdot((\log N)/2+1)/2}} \leq \frac{N^{\log\log N}}{N^{\log(N)/8}}.$$

On the other hand, the probability of $\mathcal{S}$ being good is 0 whenever it has size $<$ $(\log N)/2$ or $> 2\log N$. The claim follows. □

Lemma 1 immediately allows us to prove the following upper bound on the advantage $\pi(N)$ of an oblivious Pebbler whenever we restrict the Builder-Pebbler Game to paths.

Theorem 6 (Combinatorial Upper Bound for Paths)**.** *The advantage of any oblivious Pebbler against a random Builder in the $(N, \mathcal{C}_N, \mathcal{C}_N)$-Builder-Pebbler Game with the winning condition $X_{\mathcal{C}}$ defined as in Definition 11 is at most*

$$\pi \leq 1/N^{\log(N)/8-\log\log(N)}.$$

Proof. Since the Pebbler is oblivious it has to commit to some vertex cover $\mathcal{S} \subset [N]$ in the beginning of the game. Since the Builder queries edges uniformly at random, $\mathcal{S}$ is a uniformly random subset of $[N]$. Thus, by Lemma 1, the probability that $\tilde{\mathcal{P}}$ is good is at most

$$\frac{N^{\log\log N}}{N^{\log(N)/8}} = \frac{1}{N^{\log(N)/8-\log\log(N)}}.$$

This proves the theorem. □

4.2 Combinatorial Upper Bounds for Binary Trees

In the case we restrict the Builder-Pebbler Game to binary trees, we show that any oblivious Pebbler playing the Builder-Pebbler Game against a random Builder with a definition of cut that is again related to pebbling lower bounds for trees (see §4.2.1) has advantage at most quasi-polynomial in N (§4.1.2). As a warm up, we analyse in §4.2.2 the advantage of an oblivious Pebbler when the size of the vertex cover is bounded (i.e, $o(N)$ to be precise). We then extend this to arbitrary oblivious strategies for Pebbler (§4.2.3). The main idea is to borrow ideas from pebbling lower bounds for binary trees as described in the introduction (and recalled below).

4.2.1 Pebbling Characteristics of Binary Trees.

It is known that the number of pebbles that are needed to pebble a perfect binary tree B_n of depth n, and therefore of size $N = 2^{n+1} - 1$, is at least n, and the argument is as follows (refer to [51] for example). Consider a pebbling sequence for perfect binary tree: at the beginning of the sequence *none* of the 2^n paths from the root to the leaves carry a pebble, whereas at the end of the sequence (at which point the root carries a pebble) *all* the paths from the root to the leaves carry a pebble. Furthermore, only

new pebbles on leaves decrease the number of paths that carry a pebble, because a pebble can only be placed on an inner node, if both children are already pebbled. Hence, all paths through this inner node already carry a pebble. So any pebbling move can only decrease the number of paths that carry a pebble by 1. Therefore there has to exist two consecutive configurations in the pebbling sequence such that in the first configuration there exists a path that does not carry a pebble but in the next configuration every path carries a pebble. At this point at least all the vertices on the copath — i.e., the siblings of each vertex on the path – need to be pebbled, and in particular there exists a pebbling configuration where there are at least $\log N$ pebbles. Such pairs of configurations serve as the cut for the winning condition. A formal definition follows.

Definition 12 (Good pebbling configurations, cuts and cut function for binary trees)**.** Let $\boldsymbol{\mathcal{P}}$ be the set of pebbling configurations on B_n such that B_n contains at least one path from a leaf to the root that does not carry a pebble, i.e. all edges on this path are unpebbled. As the cut-set on (the configuration graph of) B_n we choose $\mathcal{X} = \{(\mathcal{P}_i, \mathcal{P}_j) \mid \mathcal{P}_i \in \boldsymbol{\mathcal{P}} \wedge \mathcal{P}_j \notin \boldsymbol{\mathcal{P}}\}$. Note that any $\mathcal{P}_i$ with $(\mathcal{P}_i, \mathcal{P}_j) \in \mathcal{X}$ for some $\mathcal{P}_i$ must have exactly one path from some leaf to the root not carry a pebble while every other path must carry a pebble. The cut function $X_{\mathcal{B}}$ is defined as in Definition 8 as the frontier of this cut.

4.2.2 Warm-up: Upper Bound for Bounded Vertex Cover.

Consider an oblivious Pebbler that selects at most s (random) vertices on the binary tree as the vertex cover. (Note that since the Pebbler is oblivious and the Builder picks a uniformly random permutation of the graph, we can view any oblivious Pebbler as selecting the vertices in the cover at random.) For the ease of analysis, we will consider a slightly different Pebbler which – instead of selecting s vertices at random – will include each vertex in the cover with probability $\alpha := s/N$. We first show in Lemma 2 that this cannot decrease the success probability too much, so any super-polynomial lower bound we obtain in this way holds in general.

Lemma 2. *Let P_s be a Pebbler that selects s vertices at random and let $\mathsf{P}_{\alpha(s)}$ be a Pebbler that behaves exactly like P_s but for every one of the N vertices chooses to include it in the vertex cover i.i.d. with probability $\alpha(s) = s/N$. Then for any event E over the output of P_s we have $\mathsf{Pr}[\mathsf{P}_s \to E] = O(\sqrt{N})\mathsf{Pr}[\mathsf{P}_{\alpha(s)} \to E]$.*

Proof. Let L be the event that $\mathsf{P}_{\alpha(s)}$ selects exactly s vertices. Then we have $\mathsf{Pr}[\mathsf{P}_s \to E] = \mathsf{Pr}[\mathsf{P}_{\alpha(s)} \to E \mid L]$. On the other hand, we have

$$\mathsf{Pr}[\mathsf{P}_{\alpha(s)} \to E] = \mathsf{Pr}[\mathsf{P}_{\alpha(s)} \to E \mid L]\mathsf{Pr}[L] + \mathsf{Pr}[\mathsf{P}_{\alpha(s)} \to E \mid \bar{L}]\mathsf{Pr}[\bar{L}]$$

where $\bar{L}$ is the complementary event to L. This implies

$$\mathsf{Pr}[\mathsf{P}_s \to E] \leq \mathsf{Pr}[\mathsf{P}_{\alpha(s)} \to E]/\mathsf{Pr}[L].$$

It remains to bound $\mathsf{Pr}[L]$ from below:

$$\mathsf{Pr}[L] = \binom{N}{s}\left(\frac{s}{N}\right)^s\left(\frac{N-s}{N}\right)^{N-s} = \frac{N!}{N^N}\frac{s^s}{s!}\frac{(N-s)^{N-s}}{(N-s)!} \geq \sqrt{\frac{N}{2\pi e s(N-s)}}.$$

□

Theorem 7 (Combinatorial Upper Bound for Binary Trees: Bounded VC)**.** *Let $\mathcal{B}_n$ be the class of perfect binary trees of depth n and size $N = 2^{n+1}-1$. Then any oblivious Pebbler P in the $(N, \mathcal{B}_n, \mathcal{B}_n)$-Builder-Pebbler Game with perfect binary trees with cut $X_{\mathcal{B}}$ defined as in Definition 12 has an advantage of at most*

$$\pi \leq \begin{cases} 1/N^{\log N} & \text{for } s = o(N) \\ 1/N^{\omega(1)} & \text{for } s = N^{\epsilon} \text{ with } \epsilon < 1 \text{ constant.} \end{cases}$$

against a random Builder B.

Proof. Note first that since B queries a random graph $B_n \in \mathcal{B}_n$, one can view P_s as choosing the s vertices in the cover uniformly at random. By Lemma 2 we can instead bound the advantage of $\mathsf{P}_{\alpha(s)}$, which chooses for each vertex i.i.d. if it will be included in the cover with probability $\alpha = s/N$.

Fix a path p from a leaf to the root in B_n. Define $\boldsymbol{\mathcal{P}}_p \subset \boldsymbol{\mathcal{P}}$ to be the set of configurations in which p does not carry a pebble but every other path does. In the following we say that a subtree is *covered* by $\mathcal{S}$, if there exists a configuration $\mathcal{P}$ in which all paths from the leaves to the root of this subtree carry a pebble, such that $\mathcal{S}$ is a vertex cover of $\mathcal{P}$ and $\mathcal{S} \subset \mathcal{P}$. Let $P(d)$ denote the probability that a perfect binary tree of depth d is covered when vertices are included in the cover independently using coin toss of bias $\alpha = s/N$. We argue via induction that $P(d) \leq 2\alpha$. For the base case, note that $P(1) = \alpha + (1-\alpha)\alpha^2 \leq 2\alpha$. Suppose that the hypothesis is true for binary tree of depth $d-1$. It is not hard to see that

$$P(d) = \alpha + (1-\alpha)P(d-1)^2.$$

It follows that $P(d) \leq \alpha + (1-\alpha)4\alpha^2$, and it suffices to show that

$$(1-\alpha)4\alpha^2 \leq \alpha \Leftrightarrow (1-\alpha)\alpha \leq 1/4.$$

This is indeed true since $(1-\alpha)\alpha$ is a quadratic polynomial which is maximized at $\alpha = 1/2$.

In order for a configuration to be in $\boldsymbol{\mathcal{P}}_p$, all subtrees that are rooted in the copath of p must be covered by the selected vertex cover or the parent in the path must be in the vertex cover. The probability of this is $\leq \alpha + 2\alpha = 3\alpha$. Finally, since the vertices involved in each subtree are disjoint, we get that the probability of a vertex cover that is minimal for some configuration in $\boldsymbol{\mathcal{P}}_p$ is less than

$$\prod_{i=1}^{n-1} 3\alpha = (3\alpha)^{n-1}$$

where n, if you recall, is the depth of B_n.

By applying the union bound, we have at the probability that there exists *some* unpebbled path is at most $N/2 \cdot (3\alpha)^{n+1}$. It follows that $\lambda \geq 2/(N \cdot (3\alpha)^{n+1})$, which is quasi-polynomial when $s = N^{\epsilon}$ for a constant $\epsilon < 1$, and super-polynomial when $s = o(N)$. □

4.2.3 Upper Bound for Unbounded Vertex Cover.

Unfortunately, the above Builder strategy does not work when the Pebbler is allowed an unbounded number of vertices in the cover: in particular, in case the bias $\alpha = 1/2$ — in which case it places around $N/2$ pebbles — it gets into the cut with high probability. Thus, we need to somehow limit the number of pebbles that the Pebbler places, and this is accomplished by adding a second binary tree in the game. In the new strategy, the Builder randomly queries *two* binary trees and then proceeds to challenge one of these trees picked uniformly at random; Recall that if any edge in the other binary tree is pebbled, the Pebbler immediately loses. In case the Pebbler places too many pebbles, it is likely that it gets caught in this process. We show in the analysis that this intuition is in fact correct and consequently we obtain a tighter upper bound.

Theorem 8 (Combinatorial Upper Bound for Binary Trees: Unbounded VC)**.** *Let $\mathcal{D}_n$ denote the class of DAGs where D_n consists of two binary trees $B_{n-1,0}$ and $B_{n-1,1}$ of depth $n-1$, and let $N = 2(2^n - 1)$. Then any oblivious Pebbler P_s which commits to a vertex cover of bounded size s in the $(N, \mathcal{D}_n, \mathcal{B}_{n-1})-$Builder-Pebbler Game with the cut function $X_{\mathcal{B}}$ defined as in Definition 12 has an advantage of at most*

$$\pi \leq 1/N^{\log N - \log\log N}$$

against a random Builder B*.*

Proof. The random Builder B plays the Builder-Pebbler Game on $\mathcal{D}$ as follows: it picks $D_n \in \mathcal{D}$ at random and then challenges one of the binary trees ($\mathcal{B}_{n-1,b}$) at random. Again, as in Theorem 7, by Lemma 2 we can bound the advantage of a Pebbler $\mathsf{P}_{\alpha(s)}$, which chooses for each vertex i.i.d. if it will be included in the cover with probability $\alpha = s/N$. Clearly, such a Pebbler has probability $(1-\alpha)^{N/2}$ of not selecting any vertex in $B_{n-1,1-b}$ (note that this is a requirement, since by definition of oblivious Pebblers any node in the vertex cover must be adjacent to at least one pebbled edge and there must not be any pebbled edges in the non-challenge part of the graph). By combining this with the bound obtained in Theorem 7, the probability of $\mathsf{P}_{\alpha(s)}$ selecting a vertex cover that is minimal for a configuration that is in $\mathcal{X}$ and is entirely unpebbled in $B_{n-1,1-b}$ is less than

$$\frac{N}{2} 3^{n-1} (1-\alpha)^{N/2} \alpha^{n-1}.$$

As a function of α, this expression is maximized for $\alpha = 2n/(N+2n)$ and yields the bound

$$N^{-\log N + \log\log(N/2) + o(1)}.$$

□

4.3 Combinatorial Upper Bound for Unrestricted Games

In the following we prove an almost exponential upper bound on the advantage of oblivious Pebblers in the Builder-Pebbler Game on complete graphs. Obviously, this implies a subexponential upper bound for oblivious Pebblers whenever the Builder is not restricted at all and, in particular, can query a complete graph.

4.3.1 Pebbling Characteristics of Complete Graphs.

The best known pebbling strategy $\boldsymbol{\mathcal{P}} = (\mathcal{P}_{start}, \ldots, \mathcal{P}_{\text{target}})$ for a complete graph K_N of size N has vertex cover $\mathsf{VC}(\boldsymbol{\mathcal{P}}) = N/2 + 1$, which implies an exponential upper bound. Note, this is not trivial since the complete graph has VC-complexity $N-1$. The strategy works as follows: First, greedily pebble all edges connected to the first half $[N/2]$ of the nodes in topological order; this can trivially be done at VC-complexity $N/2$. Next, unpebble all edges *within* the first half starting from those incident on node $N/2$ up to those on node 2; this still has VC-complexity $N/2$ since only edges were removed. At this point, all edges from $[N/2]$ to $[N/2+1, N]$ are pebbled, but there are no pebbles within either part of the graph; this configuration can be covered by the set $[N/2]$, but also by $[N/2+1, N]$ which will be a minimal cover for all subsequent configurations. Now, pebble all edges within the second half starting with those outgoing from node $N/2+1$ up to node $N-1$, which can be done since all ingoing edges from the first half are already pebbled; all these configurations can be covered by the set $[N/2+1, N]$. Finally, unpebble all edges not incident on N by following the sequence in reverse order, keeping N in each minimal vertex cover. This gives a valid pebbling strategy with VC-complexity $N/2+1$.

Unfortunately, our lower bound doesn't match this upper bound, but clearly gives a nontrivial result as stated in the lemma below.

Lemma 3 (Lower Bound on VC-complexity of Complete Graphs)**.** *Let* $\boldsymbol{\mathcal{P}}_N = (\mathcal{P}_{\text{start}}, \ldots, \mathcal{P}_{\text{target}})$ *be a valid (edge-) pebbling sequence of the complete graph* K_N *of size* N*. Then*

$$\mathsf{VC}(\boldsymbol{\mathcal{P}}_N) \geq \sqrt{N} - 1.$$

Proof. We argue via induction on N. For $N = 1$, the claim is trivially true. Now, assume it holds for all $N' < N$. Let $\boldsymbol{\mathcal{P}}$ be a minimal (w.r.t. VC-complexity) pebbling sequence. W.l.o.g., we can assume that $\boldsymbol{\mathcal{P}}$ is *reduced* and, in particular, edges incident on N are never unpebbled again. Let $\mathcal{P}^*$ be the first configuration in $\boldsymbol{\mathcal{P}}$ where an edge (i^*, N) incident on N is pebbled and S^* be a minimal vertex cover of $\mathcal{P}^*$. If

$\mathsf{VC}(\mathcal{P}^*) \geq \sqrt{N} - 1$ the claim trivially follows from $\mathsf{VC}(\boldsymbol{\mathcal{P}}_N) \geq \mathsf{VC}(\mathcal{P}^*)$. Thus, in the following we consider the case $|S^*| = \mathsf{VC}(\mathcal{P}^*) < \sqrt{N} - 1$.
When we remove the set S^* as well as the two nodes i^* and N (where at least one of them is contained in S^*) from the graph K_N, we end up with a complete (sub)graph K^* which is entirely unpebbled and will be pebbled during the configurations $\mathcal{P}^*, \ldots, \mathcal{P}_{\text{target}}$. It holds

$$N - 2 \;\geq\; |V(K^*)| \;\geq\; |K_N| - |S^*| - 1 > N - (\sqrt{N} - 1) - 1 = N - \sqrt{N}.$$

By induction hypothesis, any valid pebbling sequence on K^* has VC-complexity at least $\sqrt{|V(K^*)|} - 1 \geq \sqrt{N - \sqrt{N}} - 1$; this in particular also holds for the pebbling sequence on K^* induced by $\boldsymbol{\mathcal{P}}$. Since the edge (i^*, N) remains pebbled throughout $\mathcal{P}^*, \ldots, \mathcal{P}_{\text{target}}$ and is node-disjoint with K^*, it follows

$$\mathsf{VC}(\boldsymbol{\mathcal{P}}_N) \geq \mathsf{VC}(\boldsymbol{\mathcal{P}}) \geq (\sqrt{N - \sqrt{N}} - 1) + 1 = \sqrt{N - \sqrt{N}}.$$

The claim now follows since $\sqrt{N - \sqrt{N}} \geq \sqrt{N} - 1$ for all $N \geq 1$. □

This also yields the following definition of good configuration for the complete graph.

Definition 13 (Good pebbling configurations, cuts and cut function for complete graphs)**.** We call a pebbling configuration $\mathcal{P}$ for the complete graph K_N of size N *good* if the VC-complexity of $\mathcal{P}$ is $\sqrt{N} - 2$ and there exists a valid pebbling sequence $\boldsymbol{\mathcal{P}} = (\mathcal{P}_{\text{start}}, \ldots, \mathcal{P})$ such that the VC-complexity of the sequence $\mathsf{VC}(\boldsymbol{\mathcal{P}}) \leq \sqrt{N} - 2$. As the cut-set on (the configuration graph of) K_N we choose the $\mathcal{X}$ to be defined as the set of pairs $(\mathcal{P}_i, \mathcal{P}_j)$ such that $\mathcal{P}_i$ is good, $\mathcal{P}_j$ is not good, and $\mathcal{P}_j$ differs from $\mathcal{P}_i$ in one valid pebbling step. The cut function $X_{\mathcal{K}}$ is defined as in Definition 8 as the frontier of this cut.

4.3.2 The Upper Bound.

Lemma 3 implies the following upper bound on the advantage π for oblivious Pebblers against a random Builder on the Builder-Pebbler Game played on a complete challenge graph.

Theorem 9 (Combinatorial Upper Bound for Complete Graphs)**.** *Let $\mathcal{K}_{a,b}$ denote the class of graphs consisting of two complete graphs K_a and K_b. For any parameter $N \in \mathbb{N}$, any oblivious Pebbler in the $(N, \mathcal{K}_{n,N-n}, \mathcal{K}_{N-n})$-Builder-Pebbler Game, where $n = N/e^3$, and the cut function $X_{\mathcal{K}}$ defined in Definition 13 has advantage at most*

$$\pi \leq e^{-2(\sqrt{N/(e^3)}-1)}.$$

against a random Builder B.

Proof. We use a random Builder: that is, the graph structure B queries consists of two complete directed graphs of sizes n and $N-n$, respectively, where we will define n later in this proof. Since, by assumption, the reduction committed to a non-trivial vertex cover $S \subset [N]$ in the beginning of the game and B chose a permutation of $[N]$ independently and uniformly at random, the probability that S lies completely in the first part of the graph is at most

$$\frac{\binom{n}{\sqrt{n}-1}}{\binom{N}{\sqrt{n}-1}} \leq \left(\frac{n}{\sqrt{n}-1}\right)^{\sqrt{n}-1} \left(\frac{(\sqrt{n}-1)\cdot e}{N}\right)^{\sqrt{n}-1} = \left(\frac{ne}{N}\right)^{\sqrt{n}-1}$$

By computing the derivative of the latter function one finds that it takes its minimum close to $n = N/e^3$, hence B will use this value for n. Thus, $\pi \leq e^{-2(\sqrt{N/(e^3)}-1)}$. This proves the claim. □

4.4 Cryptographic Lower Bound: GSD

The generalized selective decryption game (GSD) was informally introduced in §1. In this section, we formally define GSD and interpret the lower bounds from §§4.1 through 4.3 for GSD.

4.4.1 Definition and Security Assumption.

We use the definitions from [35].

Let $(\mathsf{Enc}, \mathsf{Dec})$ be a symmetric encryption scheme with $\mathsf{Enc}\colon \mathcal{K}\times\mathcal{M} \to \mathcal{C}$, $\mathsf{Dec}\colon \mathcal{K}\times\mathcal{C} \to \mathcal{M}$ and we assume $\mathcal{K} \subseteq \mathcal{M}$ (i.e., we can encrypt keys). We assume that $(\mathsf{Enc}, \mathsf{Dec})$ is correct, i.e.,

$$\forall k \in \mathcal{K}, m \in \mathcal{M} \ : \ \mathsf{Pr}[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1$$

and that it is ε-indistinguishable under chosen-plaintext attack (IND-CPA) – see Definition 14.

Definition 14 (IND-CPA)**.** The game is played between a challenger (either G_0 or G_1) and an adversary on the symmetric encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$. The challenger chooses the challenge key $k \leftarrow \mathcal{K}$. The adversary can make two types of queries:

- Encryption queries $(\texttt{encrypt}, m)$, $m \in \mathcal{M}$: the challenger returns $\mathsf{Enc}(k, m)$.
- One challenge query $(\texttt{challenge}, m_0, m_1)$, $m_0, m_1 \in \mathcal{M}$: the challenger when simulating G_b returns the challenge ciphertext $\mathsf{Enc}(k, m_b)$.

An encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ is said to be ε-indistinguishable under chosen-plaintext attack, if G_0 and G_1 are ε-indistinguishable.

The GSD game is defined as follows:

Definition 15 (Adaptive GSD [45, 35])**.** The game is played between a challenger G (which is either G_L or G_R) and an adversary A using $(\mathsf{Enc}, \mathsf{Dec})$. G picks n keys $\{k_1, \ldots, k_n\} \leftarrow \mathcal{K}$ uniformly at random, and initialises a graph $G_\kappa := (\{v_1, \ldots, v_n\}, \emptyset)$; it also initialises a set $\mathcal{C} = \emptyset$. A can make three types of queries:

- Encryption queries, $(\texttt{encrypt}, v_i, v_j)$: G returns $\mathsf{Enc}(k_i, k_j)$, and adds (v_i, v_j) to $\mathcal{E}$.
- Corruption queries, $(\texttt{corrupt}, v_i)$: G returns k_i,and adds v_i to C.
- One challenge query $(\texttt{challenge}, v_i)$: Here the answer differs between G_L and G_R: G_L answers with k_i (real key), whereas G_R answers with $r \leftarrow \mathcal{K}$ (fake key) sampled uniformly at random — for the task to be non-trivial, v_i must be a sink and must *not* be reachable from any vertex in C.

Definition 16. An encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ is called ε-adaptive GSD-secure if G_L and G_R are ε-indistinguishable.

4.4.2 Lower Bounds for GSD.

In many applications one considers games where the adversary's queries are restricted to certain graph structures, e.g., paths, trees, or low-depth graphs. While interesting upper bounds are known for these specific settings, for oblivious black-box reductions R reducing adaptive GSD security to IND-CPA security (short, GSD reductions), our results now allow us to prove lower bounds on λ for paths and binary trees.

Definition 17 (Oblivious GSD Reduction)**.** A GSD reduction R is *oblivious* if

- it is non-rewinding, and
- at the beginning of the game
  - assigns secret keys to all nodes, and
  - commits to a non-trivial vertex cover of all inconsistent edges.

The conditions on R with respect to non-rewinding and vertex cover naturally follow from Definition 10, while the requirement to assign keys to nodes at the beginning is not as obvious. It is due to the fact that Pebblers in Builder-Pebbler Game commits to whether an edge is pebbled or not as soon as they respond to the query. Without the above requirement, this is not true for reductions in the GSD game, since they could potentially respond to a query and decide later if that edge is consistent or inconsistent by choosing the key for the target accordingly (as long as this node does not have an outgoing edge). However, we remark that this requirement should not be seen as a very limiting restriction, but we introduce it for ease of exposition, since there are several "work arounds" to this issue. 1) One could use an

adversary that "fingerprints" the keys by querying the encryption of some message under each key before starting the rest of the query phase. This would entail adding the corresponding oracle to the GSD game, which seems reasonable in many (but not all) applications, since the keys are often not created for their own sake, but to encrypt messages. Additionally, some protocols are based on a public key version of GSD rather than the secret key version we consider here (see e.g. [1]). In such cases the public keys may be known to the adversary and fix the corresponding secret keys. (Our proof below does not cover such public key variants, but we do not see an obstacle in adapting it to such a setting.) 2) There is a generic fix where the adversary abuses the `encrypt` oracle to achieve this fingerprinting by introducing a new node and querying the edges from every other node to this new node. This introduces only a slight loss in the number N of nodes, but would make the proof more complicated: recall that the challenge node must be a sink, so this approach cannot be applied to it. We can still apply this approach to all other nodes, thereby giving away the challenge node right at the start of the game. But this can only increase the reduction's advantage by a factor N, since it could also simply guess the challenge node. Since we are only interested in super-polynomial losses in this work, this would not affect the results. But for the sake of clarity we refrain from applying this workaround and simply keep this mild condition on the GSD reductions.

We now give a general lemma that allows to turn lower bounds for the Builder-Pebbler Game game into lower bounds for the GSD game.

Lemma 4. *Let $\mathcal{G}, \mathcal{G}^*$ be a families of graphs and X a cut function. Let B be an oblivious Builder in the $(N, \mathcal{G}, \mathcal{G}^*)$-Builder-Pebbler Game with winning condition X. Then there exists*

1. *an ideal SKE scheme $\Pi = (\mathsf{Enc}, \mathsf{Dec})$*
2. *a GSD adversary A in PSPACE*

such that for any oblivious black-box PPT reduction R that reduces GSD to the IND-CPA security of the underlying SKE scheme there exists an oblivious Pebbler P such that the advantage of R in reducing GSD to the IND-CPA security of Π is at most the advantage of P against B (up to an additive term $\mathsf{poly}(N)/2^{\Omega(N)}$).

Proof. We first construct $\Pi = (\mathsf{Enc}, \mathsf{Dec})$: We will pick Enc to be a random expanding function (which is injective with overwhelming probability). More precisely, assuming (for simplicity) the key k, the message m and the randomness r are all λ-bit long, $\mathsf{Enc}(k, m; r)$ maps to a random ciphertext of length, say, 6λ with $\lambda = \Theta(N)$. Dec is simulated accordingly to be always consistent with Enc.

We now define a map ϕ from GSD adversaries and reductions to Builder-Pebbler Game Builders and Pebblers:

- The number N of nodes in the Builder-Pebbler Game corresponds to the number N of keys in the GSD game.

- An encryption query $(\texttt{encrypt}, v_i, v_j)$ maps to an edge query (i, j) in the Builder-Pebbler Game.
- A response to a query $(\texttt{encrypt}, v_i, v_j)$ is mapped to "no pebble" if it consists of a valid encryption of k_j under the key k_i, and to "pebble" otherwise. (Note that this is always well-defined for oblivious GSD reductions, because these need to commit to an assignment of keys at the beginning.)
- A corruption query $(\texttt{corrupt}, v_i)$ is ignored in the Builder-Pebbler Game.
- The challenge query $(\texttt{challenge}, v_t)$ is mapped to the challenge node t.

Let $\mathsf{A} \in \mathsf{PSPACE}$ be the following preimage of B under ϕ: A performs the same encryption queries as B and selects its GSD challenge node as the challenge node chosen by B. It then corrupts all nodes not in the challenge graph G^t. If there is an inconsistency (i.e. a pebble) in $G \setminus G^t$, A aborts and outputs 0. Finally, it uses its computational power to decrypt all the received ciphertexts and determines the resulting pebbling configuration $\mathcal{P}$ on G^t. If $\mathcal{P}$ is in the cut defined by the frontier $X(G^t)$, A outputs 0, otherwise it outputs 1. Clearly, A wins the GSD game against Π with probability 1. We will now show that the advantage of R against A when used to reduce GSD (with Π) to the IND-CPA security of Π is at most the advantage of $\mathsf{P} = \phi(\mathsf{R})$ against B (up to a negligible additive term).

Note that since Enc is a random function, the GSD game is entirely independent of the challenge bit b until the tuple (k, m_b, r) such that $c^* = \mathsf{Enc}(k, m_b; r)$ (where c^* is the challenge ciphertext) is queried to Enc. Since R is PPT, the probability of R doing this is at most $\mathsf{poly}(N)/2^{\Omega(N)}$. Accordingly, to gain a larger advantage, R must send c^* to A as response to some edge query. Since $\mathsf{B} = \phi(\mathsf{A})$ is oblivious, the behaviour of A does not depend on c^* (and thus not on b) during the entire query phase. This means that the statistical distance of A induced by $b = 0$ and $b = 1$ is

$$\sum_{(\mathcal{P}_i, \mathcal{P}_j) \in \mathcal{P}^{G^t}} p_{i,j} |\mathsf{Pr}\left[\mathsf{A}(\mathcal{P}_i) \to 1\right] - \mathsf{Pr}\left[\mathsf{A}(\mathcal{P}_j) \to 1\right]|$$

where $p_{i,j}$ is the probability that the query phase results in the configuration $\mathcal{P}_i$ or $\mathcal{P}_j$ depending on c^*. More formally, for an edge $(\mathcal{P}_i, \mathcal{P}_j)$ in the configuration graph $\mathcal{P}^{G^t}$, let $\mathcal{P}^c_{ij}$ be the "configuration" that is equal to $\mathcal{P}_i$ if c^* represents a consistent encryption edge (i.e. is not a pebble) and equal to $\mathcal{P}_j$ if c is inconsistent (i.e. a pebble). Then we define $p_{i,j}$ as the probability of the query phase resulting in $\mathcal{P}^c_{ij}$. Clearly, we have $|\mathsf{Pr}\left[\mathsf{A}(\mathcal{P}_1) \to 1\right] - \mathsf{Pr}\left[\mathsf{A}(\mathcal{P}_2) \to 1\right]| = 0$ for any edge $(\mathcal{P}_1, \mathcal{P}_2)$ where $\mathcal{P}_1 \notin X(G^t)$ and 1 otherwise. The statistical distance of A induced by b is thus bounded by the probability of the querying phase ending up in a configuration in $X(G^t)$ (if c^* is considered not a pebble for this argument). This is exactly the advantage of Pebbler $\mathsf{P} = \phi(\mathsf{R})$ in the Builder-Pebbler Game against B. By data processing inequality, this also means that the advantage of R is bounded from above by the same quantity. □

The following lower bounds on GSD now easily follow from Lemma 4 and the theorems in the previous section (Theorem 6, 8, 9, resp.).

Corollary 1 (Lower bound for GSD restricted to paths)**.** *Let N be the number of users in the GSD game. Then any oblivious black-box reduction from adaptive GSD restricted to paths to IND-CPA security of the underlying encryption scheme loses at least a factor*

$$\lambda \geq N^{\log(N)/8 - \log\log(N)}.$$

Corollary 2 (Lower bound for GSD on binary trees)**.** *Any oblivious black-box reduction from adaptive GSD on N users restricted to binary trees to IND-CPA security of the underlying encryption scheme loses at least a factor*

$$\lambda \geq N^{\log N - \log\log N}.$$

For adversaries which are allowed to query any acyclic graph structure on N vertices, and in particular a complete graph, Theorem 9 gives the following result.

Corollary 3 (Lower bound for GSD on arbitrary acyclic graphs)**.** *Any oblivious black-box reduction from adaptive security of GSD restricted to acyclic graphs on N users to IND-CPA security of the underlying encryption scheme loses at least a factor*

$$\lambda \geq 2^{2(\sqrt{N/(e^3)}-1)}.$$

5 Lower Bounds for Node Pebbling

Here, we consider reductions that are *restricted to node-pebbling*, i.e., R is only allowed to either pebble *all* ingoing edges to a node, or none (see Definition 18). For example, to prove adaptive security of proxy re-encryption schemes, Fuchsbauer et al. [21] suggest to reduce to two basic security assumptions: First, the necessary condition of IND-CPA security of the underlying encryption scheme, which implies indistinguishability of games where no reencryptions of the challenge ciphertext are issued, and second, δ-weak key privacy, which is used to prove indistinguishability of subsequent hybrid games that differ in how rekey queries are answered. The notion of δ-weak key privacy clearly translates to node pebbling and a relation to the more general edge pebbling is not clear.

Also other applications considered by Jafargholi et al. [35], such as Secret Sharing and Yao's Garbled Circuit, as well as the recent application to ABE by Kowalczyk and Wee [39] use node-pebbling reductions. However, in all three of these applications of the framework from [35], the graph structure is known to the reduction in the beginning of the game, which allows for some compression of the representation of the pebbling configurations. We will not discuss these applications in the current work.

We can still use the Builder-Pebbler Game to capture applications of node-pebbling reductions by placing the corresponding restriction on the Pebbler.

Definition 18 (Node-pebbling Pebbler)**.** A Pebbler is called node-pebbling, if for all nodes v it either puts a pebble on all edges incident on v or on none.

Since node-pebbling reductions are a subclass of edge-pebbling reductions (and node pebbling strategies are a subclass of all pebbling strategies), all previous results carry over. For certain graphs of high indegree, however, we will prove much stronger lower bounds. Not only quantitatively, but also qualitatively as the bound holds for general black-box reductions. In particular we don't require the reduction to be oblivious.

5.1 Combinatorial Upper Bound for Complete Graphs

For node-pebbling Pebblers in an unrestricted Builder-Pebbler Game, we prove an exponential upper bound on the Pebbler'a advantage. To define a suitable cut, we exploit the crucial difference between edge and node pebbling in terms of VC-complexity regarding graphs of high indegree: Let u be an intermediate node which has high indegree in the challenge graph. In the edge pebbling game, to be able to pebble an edge (u, v), we need to have all edges incident on u pebbled; there might be up to N edges involved but, however, one can cover all these edges with the single node u. On the other hand, in the node pebbling game, to pebble node u, all the parent *nodes* need to be pebbled and, in general, the only way for the reduction to get into this configuration is to guess all parents correctly. This is formalised in the following definition and theorem.

Definition 19. For a node $v \in \mathcal{V}$, let the *reachability graph* $\mathcal{S}_v \subset G$ be the subgraph induced by the nodes in $\mathcal{V}$ that can be reached from v (but not v itself). Furthermore, define the *level 2 predecessor graph* $\mathcal{P}_v^2 \subset G$ as the subgraph induced by all the nodes in $\mathcal{V}$ from which v can be reached through a path of length at most 2 (but again not v itself). Finally, for a graph G define $D(G) = \max\{|E_d| \mid E_d \subset E(G) \wedge |\{u \mid (u, v) \vee (v, u) \in E_d\}| = 2|E_d|\}$ to be the maximum number of pairwise disjoint edges in G.

Theorem 10. *For any graph family $\mathcal{G}$ containing all graphs isomorphic to some connected DAG $G = (\mathcal{V}, \mathcal{E})$, there exists a cut function X and a Builder B such that any (not necessarily oblivious) node-pebbling Pebbler P has advantage at most*

$$\pi \leq \left(\max_{v \in \mathcal{V}} \binom{D(\mathcal{S}_v) + D(\mathcal{P}_v^2)}{D(\mathcal{P}_v^2)} \right)^{-1}$$

in the $(N, \mathcal{G}, \mathcal{G})$-Builder-Pebbler Game.

We remark that for any $v \in \mathcal{V}$, $D(\mathcal{P}_v^2)$ must be smaller than or equal the in-degree of v. So, Theorem 10 only yields interesting results for graphs with large degree (but not all of them). Furthermore, if $\mathcal{S}_v$ and $\mathcal{P}_v^2$ contain long paths, then they have many disjoint edges.

Proof. Let v be such that it maximizes the quantity in the theorem. We define a cut S on $\mathcal{P}^G$ as containing all configurations where v and $\mathcal{S}_v$ are entirely unpebbled. The cut function X is now defined as the frontier of that cut (after applying the isomorphism). Note that for any configuration in $X(G)$ all edges in $\mathcal{P}_v^2$ are pebbled, while all edges in $\mathcal{S}_v$ are unpebbled. The Builder B picks a random graph G' in $\mathcal{G}$ and first queries for the disjoint edges in $\mathcal{P}_v^2$ and $\mathcal{S}_v$ in a random order. Note that the Pebbler has no information about which edge is in $\mathcal{P}_v^2$ and which is in $\mathcal{S}_v$. Accordingly, the probability of the challenge graph being in $X(G')$ at the end of the query phase is at most

$$\binom{D(\mathcal{S}_v) + D(\mathcal{P}_v^2)}{D(\mathcal{P}_v^2)}^{-1}. \tag{1}$$

Note that the above argument still works if we let B send all the queries of the first phase (i.e., randomly permuted $\mathcal{P}_v^2$ and $\mathcal{S}_v$ edges) at once: as they are disjoint, getting them all at once is of no help to the Pebbler for guessing whether an edge belongs to $\mathcal{P}_v^2$ or $\mathcal{S}_v$. As for a single query there's no distinction between an oblivious or non-oblivious Pebbler (as there's no second query that could depend on the answer to the first), this upper bound applies to non-oblivious Pebblers. □

Corollary 4. *For any graph family $\mathcal{G}$ containing all graphs isomorphic to the complete DAG $G = (\mathcal{V}, \mathcal{E})$, there exists a cut function X and a Builder B such that any (not necessarily oblivious) node-pebbling Pebbler P has advantage at most*

$$\pi \lesssim 2^{-N/2}$$

in the $(N, \mathcal{G}, \mathcal{G})$-Builder-Pebbler Game.

Proof. Invoke Theorem 10 with $v = N/2$. Note that $\mathcal{P}_v^2$ is the entire subgraph induced by $[N/2 - 1]$, and similarily $\mathcal{S}_v$ is the entire subgraph induced by $\{N/2 + 1, \ldots, N\}$. Both $\mathcal{P}_v^2$ and $\mathcal{S}_v$ have about $N/4$ disjoint edges (simply pick every second edge along the longest path), so by Theorem 10 any node-pebbling Pebbler has advantage at most

$$\pi \leq \binom{N/2}{N/4}^{-1} \approx 2^{-N/2}.$$

□

5.2 Cryptographic Lower Bound: Proxy Re-encryption

As an application of our results on node-pebbling Pebblers, we consider the recent work by Fuchsbauer et al. [21] on adaptively secure proxy re-encryption (PRE). In particular, they identify two natural security properties – indistinguishability of ciphertexts and δ-weak key privacy – which allow them to prove adaptive CPA-security via a black-box reduction. The current work now allows us to prove lower bounds on the security loss involved by any black-box reduction that reduces the CPA security of a PRE scheme to these two basic security properties.

5.2.1 Definitions and Security Assumptions.

A PRE scheme is a public-key encryption scheme that allows the holder of a key $\mathtt{pk}$ to derive a re-encryption key (short, rekey) $\mathtt{rk}$ for any other key $\mathtt{pk}'$ [11]. This rekey lets anyone transform ciphertexts under $\mathtt{pk}$ into ciphertexts under $\mathtt{pk}'$ without having to know the underlying message. We say that a PRE is *unidirectional* if $\mathtt{rk}$ does not allow transformations from $\mathtt{pk}'$ to $\mathtt{pk}$ [5]. Moreover if ciphertext c' for $\mathtt{pk}'$ that was derived from a ciphertext c for $\mathtt{pk}$, can be further transformed to another ciphertext c'' corresponding to public key $\mathtt{pk}''$ using a rekey $\mathtt{rk}'$, the PRE is said to allow two "hops". A PRE that allows multiple hops, i.e. a multi-hop PRE, can be defined analogously. A more formal definition of multi-hop, unidirectional PRE, to which we apply our lower bounds, is given below – we use the definitions and security assumptions from [21].

Definition 20 (Multi-hop, unidirectional PRE [21])**.** A multi-hop, unidirectional PRE scheme for a message space $\mathcal{M}$ consists of the six-tuple of algorithms $(\mathsf{S}, \mathsf{K}, \mathsf{RK}, \mathsf{E}, \mathsf{D}, \mathsf{RE})$, which are explained below.

$\mathsf{S}(1^\kappa, 1^\nu) \to \mathtt{pp}$**:** On input the security parameter κ and the maximum level ν (both in unary) supported by the scheme, **setup** outputs the public parameters $\mathtt{pp}$. We assume that $\mathtt{pp}$ is implicit in other function calls.

$\mathsf{K}(\mathtt{pp}) \to (\mathtt{pk}, \mathtt{sk})$**: Key generation** returns a public key $\mathtt{pk}$ and the corresponding secret key $\mathtt{sk}$.

$\mathsf{RK}((\mathtt{pk}_i, \mathtt{sk}_i), \mathtt{pk}_j) \to \mathtt{rk}_{i,j}$**:** On input a source key pair $(\mathtt{pk}_i, \mathtt{sk}_i)$ and a target public key $\mathtt{pk}_j$, **re-key generation** generates a unidirectional re-encryption key (rekey, for short) $\mathtt{rk}_{i,j}$.

$\mathsf{E}(\mathtt{pk}, (m, \ell)) \to (c, \ell)$**: Encryption** takes as input the public key $\mathtt{pk}$, a message m and a level $\ell \in [\nu]$, and outputs a level-ℓ ciphertext (c, ℓ).

$\mathsf{D}(\mathtt{sk}, (c, \ell)) \to m$: On input a ciphertext (c, ℓ) and the secret key $\mathtt{sk}$, **decryption** outputs a message m, or the symbol $\perp$ (if the ciphertext is invalid).

$\mathsf{RE}(\mathtt{rk}_{i,j}, \mathtt{pk}_i, \mathtt{pk}_j, (c_i, \ell)) \to (c_j, \ell + 1)$**: Reencryption** takes a re-key $\mathtt{rk}_{i,j}$, a source public key $\mathtt{pk}_i$, a target public key $\mathtt{pk}_j$ and a level-ℓ ciphertext c_i under $\mathtt{pk}_i$ and transforms it to a level-$(\ell + 1)$ ciphertext c_j under $\mathtt{pk}_j$. Only ciphertexts belonging to levels $\ell \in [\nu - 1]$ can be re-encrypted.

Definition 21 (Correctness [4, 21])**.** A proxy re-encryption scheme (as in Definition 20) is correct w.r.t. the message space $\mathcal{M}$ if the following two properties hold:

1. *Correctness of encryption:* $\forall \kappa, \nu \in \mathbb{N}\ \forall\, \mathtt{pp} \in [\mathsf{S}(1^\kappa, 1^\nu)]\ \forall\, (\mathtt{pk}, \mathtt{sk}) \in [\mathsf{K}(\mathtt{pp})]$ $\forall\, (m, \ell) \in \mathcal{M} \times [\nu]$:

$$\Pr\left[\mathsf{D}\big(\mathtt{sk}, \mathsf{E}(\mathtt{pk}, (m, \ell))\big) \neq m\right] = \mathsf{negl}(\kappa, \nu),$$

where the probability is over the random coins of E.

2. *Correctness of re-encryption:* $\forall\, \kappa, \nu \in \mathbb{N}\ \forall\, \mathtt{pp} \in [\mathsf{S}(1^\kappa, 1^\nu)]\ \forall\, (\mathtt{pk}_i, \mathtt{sk}_i)$, $(\mathtt{pk}_j, \mathtt{sk}_j) \in [\mathsf{K}(\mathtt{pp})]\ \forall\, \mathtt{rk}_{i,j} \in [\mathsf{RK}((\mathtt{pk}_i, \mathtt{sk}_i), \mathtt{pk}_j)]\ \forall\, (m, \ell) \in \mathcal{M} \times [\nu - 1]$:

$$\mathsf{Pr}\left[\mathsf{D}\big(\mathtt{sk}_j, \mathsf{RE}(\mathtt{rk}_{i,j}, \mathtt{pk}_i, \mathtt{pk}_j, (c_i, \ell))\big) \neq m\right] = \mathsf{negl}(\kappa, \nu),$$

where (c_i, ℓ) is a level-ℓ ciphertext of m under $\mathtt{pk}_i$ resulting either from direct encryption or a reencryption of level-$\ell - 1$ ciphertext, and the probability is over the random coins of E and RE.

We consider the CPA-security from [21] as defined in Game 1. In the security game an adversary first receives the public keys of all users and then can adaptively do the following queries: It can corrupt a party and receive its secret key, it can query for rekeys between two users or for a re-encryption of a ciphertext encrypted under the public key of one user to an encryption of the same plaintext under the public key of another user, and, only once, it can issue a challenge query where it chooses a challenge user, two messages m_0, m_1 as well as a level and receives an encryption of m_b to the chosen levelunder the challenge user's public key. The adversary's goal is to guess the bit b.
Consider the graph structure on the set of users which is defined throughout the game as follows (see Figure 3): Whenever the adversary queries a rekey or a reencryption of some ciphertext from user i to user j, this is represented as an edge from j to i (note, for CPA-security we do not distinguish between rekey and reencryption queries).[10]
To avoid trivial wins, we need to restrict the adversary so that it can not simply reencrypt the challenge ciphertext to a corrupted party and then use the known secret key to decrypt. Thus, for CPA-security[11], the adversary is not allowed to query any paths of rekey or reencryption queries from the challenge user to a corrupted user. Considering the query graph in Figure 3, this corresponds to the requirement that the challenge node is not reachable from any corrupt node.

Definition 22 (PRE-CPA-security [21])**.** A PRE scheme is ϵ-adaptively secure against chosen-plaintext attack if there is no adversary which can distinguish CPA^0 from CPA^1 with advantage larger than ϵ, where CPA^b is defined in Game 1.

[10] In [21], edges were defined in a more natural way, opposite to here, which led to an *inverse* pebbling game where a node can be pebbled/unpebbled if all its children are pebbled. For the ease of presentation, we chose to define the query graph so that it fits our general framework and the usual reversible pebbling game. Analogously to the GSD game, corrupting a node allows the adversary to decrypt ciphertexts encrypted under the public key of any node which is reachable from it in the graph.

[11] In [21], the authors also consider the stronger and less restrictive notion of security under *honest re-encryption attack* (HRA) which was introduced in [15] and distinguishes between (iterated) re-encryptions of the challenge ciphertext and unrelated ciphertexts. They prove HRA-security for PRE schemes which satisfy one more basic property called *source-hiding*. Our lower bounds also hold for black-box reductions reducing the HRA-security of the scheme to these three basic properties.

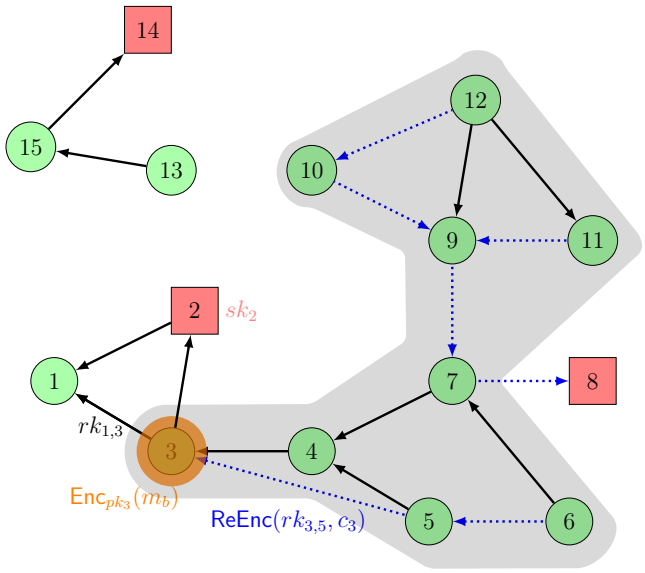

Figure 3: Recoding and challenge graph as generated in game CPA^b. The round green nodes represent the honest users, the square red ones the corrupted users. Solid black arrows from node i to node j represent that a rekey $rk_{j,i}$ from j to i was issued, and, similarly, dotted blue arrows represent re-encryption queries. For the definition of the recoding graph we do not distinguish between these two types of edges. The challenge node is marked in orange, here node 3, and the challenge graph (shaded grey) is the subgraph induced on all the ancestors of the challenge node.

Fuchsbauer et al. [21] reduce the CPA security of a PRE scheme to the following two basic security properties which are naturally satisfied by the popular constructions they analysed. The first basic security property is indistinguishability of ciphertexts, as defined for public-key encryption in [30], but on all levels:

Definition 23 (Indistinguishability)**.** A proxy re-encryption scheme PRE has ϵ-indistinguishable ciphertexts if no adversary can distinguish IND^0 from IND^1 with advantage larger than ϵ, with IND as in Game 2.

The second security property is δ-weak key privacy, which says that a set of δ re-encryption keys $\mathtt{rk}_{0,i}$ from a given source key $(\mathtt{pk}_0, \mathtt{sk}_0)$ to δ given target public keys $\mathtt{pk}_i$, where $i \in [\delta]$, is indistinguishable from a set of δ rekeys which were generated from a freshly sampled source key pair $(\mathtt{pk}'_0, \mathtt{sk}'_0)$. The security game for weak key-privacy as considered in [21] is given in Game 3 where the simulator RK^* is defined as

$$\mathsf{RK}^*(\mathtt{pp}, \mathtt{pk}_1) := \mathsf{RK}((\mathtt{pk}'_0, \mathtt{sk}'_0), \mathtt{pk}_1) : (\mathtt{pk}'_0, \mathtt{sk}'_0) \leftarrow \mathsf{K}(\mathtt{pp}).$$

Definition 24 (Weak key-privacy [21])**.** Let $\delta \in \mathbb{N}$. A proxy re-encryption scheme PRE is (ϵ, δ)-weakly key-private if no adversary can distinguish KP^0 from KP^1 with advantage larger than ϵ, where KP is defined in Game 3.

Challenger $\mathsf{CPA}^b(1^\kappa, 1^\nu, n)$

1: Set $\mathcal{C}, \mathcal{E} = \emptyset$ ▷ Stores corrupt keys and issued re-keys and re-encryptions
2: $\mathtt{pp} \leftarrow \mathsf{PRE.S}(1^\kappa, 1^\nu)$, $(\mathtt{pk}_1, \mathtt{sk}_1), \ldots, (\mathtt{pk}_n, \mathtt{sk}_n) \leftarrow \mathsf{PRE.K}(\mathtt{pp})$ ▷ Generate keys
3: $\forall i, j \in [n], i \neq j : \mathtt{rk}_{i,j} \leftarrow \mathsf{PRE.RK}((\mathtt{pk}_i, \mathtt{sk}_i), \mathtt{pk}_j)$ ▷ Generate re-keys
4: $b' \leftarrow \mathsf{A}^{(\mathtt{corrupt},\cdot),(\mathtt{rekey},\cdot,\cdot),(\mathtt{reencrypt},\cdot,\cdot,\cdot),(\mathtt{challenge},\cdot,\cdot,\cdot)}(\mathtt{pp}, \mathtt{pk}_1, \ldots, \mathtt{pk}_n)$
5: **if** A made call $(\mathtt{challenge}, i^*, \cdot, \cdot)$ for some i^* **then** ▷ Check abort conditions
6: **if** $\exists\, i \in \mathcal{C} : i$ is connected to i^* in $([n], \mathcal{E})$ **then return** 0 **end if**
7: **end if**
8: **return** b'

Oracle $(\mathtt{corrupt}, i)$

1: Add i to $\mathcal{C}$
2: **return** $\mathtt{sk}_i$

Oracle $(\mathtt{rekey}, i, j)$

1: Add (j, i) to $\mathcal{E}$ ▷ Add to recoding graph
2: **return** $\mathtt{rk}_{i,j}$

Oracle $(\mathtt{reencrypt}, i, j, (c_i, \ell))$

1: Add (j, i) to $\mathcal{E}$ ▷ Add to recoding graph
2: **return** $(c_j, \ell + 1) \leftarrow \mathsf{PRE.RE}(\mathtt{rk}_{i,j}, \mathtt{pk}_i, \mathtt{pk}_j, (c_i, \ell))$

Oracle $(\mathtt{challenge}, i^*, (m_0^*, m_1^*), \ell^*)$ ▷ Single access

1: **return** $(c_{i^*}, \ell^*) \leftarrow \mathsf{PRE.E}(\mathtt{pk}_{i^*}, (m_b^*, \ell^*))$

Game 1: PRE-CPA[21]

Challenger $\mathsf{IND}^b(1^\kappa, 1^\nu)$

1: $\mathtt{pp} \leftarrow \mathsf{PRE.S}(1^\kappa, 1^\nu)$, $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{PRE.K}(\mathtt{pp})$
2: **return** $b' \leftarrow \mathsf{A}^{(\mathtt{challenge},\cdot,\cdot)}(\mathtt{pp}, \mathtt{pk})$

Oracle $(\mathtt{challenge}, (m_0^*, m_1^*), \ell^*)$

1: **return** $\mathsf{PRE.E}(\mathtt{pk}, (m_b^*, \ell^*))$

Game 2: Security game IND for ciphertext indistinguishability

Challenger $\mathsf{KP}^b(1^\kappa, 1^\nu)$

1: $\mathtt{pp} \leftarrow \mathsf{PRE.S}(1^\kappa, 1^\nu)$, $(\mathtt{pk}_0, \mathtt{sk}_0), \ldots, (\mathtt{pk}_\delta, \mathtt{sk}_\delta) \leftarrow \mathsf{K}(\mathtt{pp})$
2: $\forall j \in [\delta] : \mathtt{rk}_{0,j}^{(0)} \leftarrow \mathsf{RK}((\mathtt{pk}_0, \mathtt{sk}_0), \mathtt{pk}_j)$
3: $\mathtt{rk}_{0,j}^{(1)} \leftarrow \mathsf{RK}^*(\mathtt{pp}, \mathtt{pk}_j)$
4: **return** $b' \leftarrow \mathsf{A}(\mathtt{pp}, \mathtt{pk}_0, \ldots, \mathtt{pk}_\delta, \mathtt{rk}_{0,1}^{(b)}, \ldots, \mathtt{rk}_{0,\delta}^{(b)})$

Game 3: Security game KP for weak key-privacy [21]

5.2.2 Lower Bounds for PRE.

In applications of PRE schemes it often makes sense to only consider security against restricted classes of adversaries where the challenge graph can only have a specific form, such as a path (e.g., in the application of key rotation) or binary trees (e.g., in a hierarchy of low depth). While for these cases quasi-polynomial upper bounds on the security loss involved when reducing CPA-security of the PRE scheme to

IND-CPA security and δ-weak key privacy are known [21], our results allow us to prove quasi-polynomial lower bounds for all *oblivious* black-box reductions, which basically means, that only the development of new techniques can lead to significally better reductions and hence stronger security guarantees.

Definition 25 (Oblivious PRE Reduction)**.** A PRE reduction R is *oblivious* if

- it is non-rewinding, and
- at the beginning of the game
  - assigns key pairs to all nodes, and
  - commits to a non-trivial vertex cover of all inconsistent edges (rekey queries) at the beginning of the game.

Lemma 5. *Let $\mathcal{G}, \mathcal{G}^*$ be families of graphs and X a cut function. Let B be an oblivious Builder in the $(N, \mathcal{G}, \mathcal{G}^*)$-Builder-Pebbler Game with winning condition X. Then there exists*

1. *an ideal PRE scheme $\Pi = (\mathsf{S}, \mathsf{K}, \mathsf{RK}, \mathsf{E}, \mathsf{D}, \mathsf{RE})$*
2. *a PRE adversary A in PSPACE*

such that for any oblivious black-box PPT reduction R that reduces PRE to the IND-CPA security of the underlying PKE scheme and the δ-weak key privacy there exists an oblivious Pebbler P such that the advantage of R is at most the advantage of P against B (up to an additive term $\mathsf{poly}(N)/2^{\Omega(N)}$).

Proof. The proof is analogous to the one of Lemma 4, so we only point out the differences here. The ideal PRE scheme Π is defined as follows: we build on the ideal public-key encryption scheme from [28], from which ideal IND-CPA security follows. We now equip the PKE scheme with PRE capabilities by defining RK to respond with the output of a random function (with large enough co-domain, so that rekeys are sparsely distributed in the range of the function) under the query input. Upon re-encryption queries, the oracle 1) computes the secret and public keys that are consistent with the rekey and the ciphertext, 2.a) if the source key pair of the rekey coincides with the key pair associated with the ciphertext, it correctly decrypts the ciphertext and re-encrypts the message using the target public key of the rekey, 2.b) and otherwise (i.e., if the source key pair of the rekey and the key pair associated with the ciphertext do not match), it outputs a uniformly random string from the ciphertext space (i.e., co-domain of E). The scheme described is clearly correct, and one can show that it satisfies weak key privacy information-theoretically.

We now decribe the map ϕ that maps the parties in the PRE game to parties in a Builder-Pebbler Game:

- The number N of nodes in the Builder-Pebbler Game corresponds to the number N of keys in the PRE game.
- A rekey query $(\texttt{rekey}, i, j)$ maps to an edge query (j, i) (sic!) in the Builder-Pebbler Game.
- A response to a query $(\texttt{rekey}, i, j)$ is mapped to "no pebble" if it consists of a valid rekey from $\texttt{pk}_i$ to $\texttt{pk}_j$, and to "pebble" otherwise. (Note that this is always well-defined for oblivious PRE reductions, because these need to commit to an assignment of keys at the beginning.)
- Corruption and re-encryption queries are ignored in the Builder-Pebbler Game.
- The challenge query $(\texttt{challenge}, i^*)$ is mapped to the challenge node t.

Analogously to the adversary in Lemma 4, A is the preimage of B under ϕ: A performs the same rekey queries as B. When B selects a challenge node, A issues a challenge query on the same node with randomly chosen messages $m_0 \neq m_1$. A then corrupts all nodes that are not in the challenge graph. If there are any inconsistencies in the corrupted part, A aborts and outputs 0. Finally, A extracts the pebbling configuration $\mathcal{P}$ from the transcript and checks whether the challenge ciphertext is an encryption of m_0 or m_1 under the correct key. If the encrypted message is m_0 (resp. m_1) and the pebbling configuration is a valid node pebbling in the cut defined by $X(G^t)$, then A outputs 0 (resp. 1). Otherwise A outputs always 0. Clearly, this adversary has advantage 1 in the PRE-CPA game.

Since Π is information-theoretically IND-CPA secure, R can only gain any advantage in the IND-CPA game by sending A the challenge ciphertext as response to the challenge query. However, this means the challenge node is associated to the challenge public key. R does not know the corresponding secret key and thus, with overwhelming probability, will respond with a fake rekey when queried for the edge(s) incident on the challenge node. This means in the extracted configuration, the target node is pebbled, so the configuration is not in the cut. Accordingly, the output of A is independent of the IND-CPA challenge bit.

This means, R must reduce to δ-key privacy. The remaining proof is the same as for Lemma 4, with the δ-key privacy challenge taking the role of the IND-CPA challenge. □

Corollary 5 (Lower bound for PRE restricted to paths)**.** *Let N be the number of users. Then any oblivious black-box reduction from PRE-CPA restricted to paths to IND-CPA security and 1-weak key privacy loses at least a factor*

$$2 \cdot N^{\log(N)/8 - \log\log(N)}.$$

Corollary 6 (Lower bound for PRE restricted to binary trees)**.** *Any oblivious black-box reduction from PRE-CPA restricted to binary trees on N users to IND-CPA and*

2-*weak key privacy loses at least a factor*

$$N^{\log N - \log\log N}.$$

For adversaries that are allowed to query complete (directed acyclic) graphs, Corollary 4 implies an exponential lower bound on the security loss even for non-oblivious black-box reductions:

Corollary 7 (Lower Bound for PRE)**.** *Let N be the number of users. Any non-rewinding black-box reduction (possibly non-oblivious) from PRE-CPA restricted to acyclic graphs to IND-CPA security and N-weak key privacy loses at least a factor $2^{N/2}$.*

Handling Rewinding Reductions Theorem 10 does not hold (and thus neither Corollary 4) if we allow Pebbler P to rewind Builder B: P can invoke B once to learn which edges queried in the first phase belong to $\mathcal{P}_v^2$ and $\mathcal{S}_v$, respectively. Then rewind B, and in this 2nd execution the reduction can easily put pebbles so the graph ends up in the cut.

However, Corollary 7 can be extended to rewinding reductions in the following way. We can consider another adversary A^* who only at the end of the first phase decides which edges should belong to $\mathcal{P}_v^2$ and $\mathcal{S}_v$. A^* will derive the randomness for this assignment by using a random function (only known to A^*) on input the transcript of the first query phase. The reduction can get a fresh shot at guessing which edges belong to $\mathcal{P}_v^2$ and $\mathcal{S}_v$ by rewinding A^*, but the probability of any such guess being correct is upper bounded as in eq.(1) because every time the transcript changes, there's a completely new assignment, and thus the reduction cannot gradually learn anything about the edge assignments.

6 Discussion and Open Questions

In this work we have seen that the loss in adaptive security in GSD and PRE is inherent for many graph families, in the case of GSD when restricted to oblivious reductions. To the best of our knowledge, this is the first attempt so far to establish non-trivial *fine-grained* lower bounds for loss in adaptive security. Whilst the topic of adaptive vs. selective security is of theoretical interest, the question of exact security of GSD is of practical relevance since GSD has emerged as the basis of proving adaptive security of modern group messaging protocols (e.g. TreeKEM [1, 2]): see §6.5. Even though currently we only have two applications, we believe that our underlying approach (lower bounds for adaptive security from combinatorial games) will find further applications (see §§6.4 and 6.5).

Moreover, and more importantly, we have seen how the question about lower bounds on loss in adaptive security of certain applications can be posed as questions

on the upper bound on the probability of the success in the Builder-Pebbler Game. We believe the introduction of the game is an important contribution: it abstracts out the hardness of establishing cryptographic lower bounds and this problem can be posed to anyone (e.g., researchers in combinatorics or game theory) without having to bother them with the cryptography.

We conclude by explaining some of the open questions and avenues for further improving the results of our paper.

6.1 Non-Oblivious Pebbler/Reductions

Our lower bounds for GSD only apply to the rather restricted class of oblivious black-box reductions. It remains an exciting open question whether (i) this can be extended to rule out *all* black-box reductions; or, contrarily (ii) the partial information on the graph structure which the reduction learns throughout the game can indeed be exploited to find better reductions. In the following we show why our attempt at (i) failed; we explore one further tool (viz., rewinding) that could potentially be used to establish (ii) in the coming section.

Exploiting non-obliviousness. Recall from §4 that to prove lower bounds on loss λ in the GSD game for oblivious black-box reductions against adversaries restricted to paths (Corollary 1), we first established that every *oblivious* Pebbler P has only a negligible (inverse quasi-polynomial in the length of the path, to be precise) winning chance when playing against an oblivious Builder B (Theorem 6). We describe next a *non-oblivious* Pebbler strategy P^* that has better winning probability against this particular Builder B – i.e., Theorem 6 is *not* valid if one considers non-oblivious Pebbler strategies! It follows that we cannot rule out non-oblivious reductions for GSD using the techniques in Theorem 6.

Recall that the oblivious Builder B chooses a uniformly random path of length N and queries the edges of the path uniformly at random. If $\mathcal{P}$ denotes the (final) pebbling configuration at the end of the game, then B wins if $\mathcal{P}$ contains $\log(N)$ pebbles and is reachable with $\log(N)$ pebbles (i.e., there exists a valid pebbling sequence $\boldsymbol{\mathcal{P}} = (\mathcal{P}_0, \ldots, \mathcal{P})$ with $|\boldsymbol{\mathcal{P}}| \leq \log(N)$, where $\mathcal{P}_0$ is the empty configuration on the path). Now, consider the following non-oblivious Pebbler strategy P^* which simply "waits":

1. initially, P^* pebbles the first edge e_c which B queries – this will be a uniform edge on the path
2. for all further edges P^* checks whether the edge is connected to and precedes e_c; if so and there are less than $\log(N)$ pebbles on the graph, it puts a pebble, otherwise answers honestly

To see that P^* considerably outperforms all oblivious Pebbler strategies, consider the probability that $\mathcal{P}$ is a configuration with $\log(N)$ consecutive pebbles such that the

first pebble is in the first half of the path. This is a good pebbling configuration and will lead to the Pebbler winning. The probability that e_c is in position $\log(N)$ to $N/2 + \log(N)$ on the path is $\frac{1}{2}$. The probability that the $\log(N)$ preceding edges are queried in reverse order is $\frac{1}{(\log(N))!}$. This gives a (probably quite loose) lower bound on P^*'s success probability of

$$\mathsf{Pr}[\mathsf{P}^* \text{ wins}] \geq \frac{1}{2(\log(N))!} \geq \frac{1}{2(\log(N))^{\log(N)}} \geq \frac{1}{2N^{\log\log(N)}}.$$

A more careful analysis of the "waiting" strategy for the Pebbler might lead to it winning with even a non-negligible probability. However, it is not clear if or how such non-oblivious strategies could be turned into better Pebblers for *arbitrary* Builders. This leads us to the following open questions towards positive (better strategies) or negative (stronger bounds) results.

Open Question 1. Is there a Pebbler strategy P such that P has non-negligible probability of winning the Builder-Pebbler Game against *any* (even non-oblivious and optimal) Builder B?

We could also ask for a weaker result by considering *non-uniform* Pebbler strategies.

Open Question 2. Is there a Pebbler strategy $\mathsf{P}(\mathsf{B})$ for any arbitrary (even non-oblivious and optimal) Builder B such that $\mathsf{P}(\mathsf{B})$ has non-negligible probability of winning the Builder-Pebbler Game against B?

A weaker but still interesting result would be if there exist non-trivial[12] graph families such that the above questions could be answered positively.

On the negative side, we pose the following open questions on stronger bounds on the success probability of any Pebbler.

Open Question 3. Is there a Builder strategy B such that *every* (even non-oblivious and optimal) Pebbler P only has a negligible probability of winning the Builder-Pebbler Game?

While a positive result to Questions 1 or 2 would directly imply polynomial reductions for GSD (potentially restricted to certain graph structures), it is not clear how to translate a stronger upper bound for the Builder-Pebbler Game to a lower bound on the loss λ involved by arbitrary reductions. The reason for this is that the Builder-Pebbler Game does not capture the cryptographic application in its full generality and, in particular, does not capture rewinding reductions, which we will discuss in the following section: Opposed to the single bit a Pebbler chooses for each edge in the Builder-Pebbler Game, a reduction in the GSD game can choose arbitrary randomness to answer encryption queries. It is not clear at this point if the Builder-Pebbler Game can be adapted to also cover rewinding and our current results on edge-pebbling do not generailize to rewinding.

[12] Note, e.g. for constant-depth graphs it is trivial to construct Pebbler strategies with polynomial success probability.

6.2 Rewinding Reductions

Another large class of reductions that we do not include in this work are rewinding reductions. While our lower bound for *node*-pebbling black-box reductions against unrestricted adversaries allows rewinding, this is not the case for our *edge*-pebbling reductions. Let us again consider the oblivious adversary restricted to paths, which we constructed in the proof of Theorem 6. Since this adversary chooses a uniformly random path in the beginning of the game and then obliviously sticks to this graph structure, we can define a reduction which manages to get into any pebbling configuration it wishes: First, R runs the adversary once on an arbitrary pebbling strategy, e.g., it answers all queries real. Then it rewinds the adversary until the point after it chose the path. But now R knows the full path structure and can trivially embed the pebbles and its challenge such that it ends up in a configuration in the cut.

To fix this issue, we could consider an adversary who follows the same oblivious threshold strategy, but chooses the edges of the path uniformly at random while the game proceeds; i.e., it first chooses a uniform edge $e = (u, v) \leftarrow \mathcal{E} := [N]_0 \times [N]_0 \setminus \{(x, x) \mid x \in [N]_0\}$, then a uniform edge $e' \leftarrow \mathcal{E} \setminus \{(u', v') \mid u' = u \vee v' = v\}$, and so on. In particular, this adversary behaves randomly in each step, conditioned on ending up with a path structure on the set of nodes. However, also this oblivious adversary can be exploited by a (oblivious or non-oblivious) black-box reduction: Assume, R wants to end up with a specific pebbling configuration $\mathcal{P}$. When receiving A's first query, R guesses the position $(i, i+1)$ of this query on the path. If i is its challenge key it embeds the challenge ciphertext, if $(i, i+1) \in \mathcal{P}$ it places a pebble, otherwise it answers real. For the next query R rewinds the adversary until it receives a query which is connected to the first edge and, in particular, assuming its initial guess was correct, knows the position of this edge on the path. Thus, it answers this query according to the pebbling configuration it has in mind. R acts similarly for all following queries. If it realises that its initial guess was wrong, R stops and rewinds the adversary until the first query and starts another run of the game. Following this strategy, the reduction has to rewind on expectation $O(N^2)$ times for each of the expected $O(N)$ runs until its initial guess is correct. Thus, this reduction can use the considered adversary at an only polynomial slow-down.

Like for obliviousness, this example shows that assuming non-rewinding reductions is necessary for our proofs to go through, but it remains an open problem if rewinding can be exploited to get better reductions. We finally want to mention, that the Builder-Pebbler Game is not the right abstraction here, since it doesn't capture additional sources of randomness the reduction might choose (e.g. encryption randomness in the case of GSD).

6.3 Better Reductions for Graph Families

We showed in Section 3 that the advantage of a Pebbler in the (Restricted) Builder-Pebbler Game is intimately related to cuts in the configuration graph of the challenge

graph. Our lower bounds exploited that certain configuration graphs have low weight cuts, such that they are hard to exploit for a reduction. Assume, we could show that for certain graphs there is no such cut in the configuration graph. Could this be exploited to obtain better Pebbler strategies in the Builder-Pebbler Game restricted to such graphs? This has the potential of resulting in better reductions for such graphs in certain applications.

6.4 Node Pebbling on Known Graphs

It is not clear at this point if our techniques can be applied to other applications of the framework [35] where pebbling techniques have shown useful to prove adaptive security, but the security games involved are of a different flavor than our Builder-Pebbler Game. One such application is the construction of a prefix-constrained pseudorandom function (pcPRF). Fuchsbauer et al. in [22] show that the GGM PRF [29] is an *adaptively* secure pcPRF involving only a quasipolynomial security loss. Unlike in the Builder-Pebbler Game the graph structure - a binary tree of exponential size - involved here is known in the beginning of the game, but the cost of guessing the challenge leaf involves an exponential loss in security. Fuchsbauer et al. circumvent this issue and reduce the security of the GGM cpPRF to a pebbling game on the path from the challenge leaf to the root of the tree. It's an interesting open problem if – following the strategy presented in this work – lower bounds for pebbling games on paths can be used to prove optimality of this reduction.
Other interesting applications are Yao's Secret Sharing, Yao's Garbled Circuit, and ABE. For all three of these very different applications the best reductions to prove adaptive security fall in the framework of Jafargholi et al. While they are quite different in nature they all have in common that – unlike in the applications that were considered in this paper – the graph structure is known in the very beginning of the game, and good upper bounds on the security loss are achieved by compressing the description of the involved pebbling configurations.

6.5 Application to Concrete Constructions

Another interesting open problem outside the scope of this work is the application of our results (especially for GSD) to concrete constructions. The GSD game was introduced by Panjwani in [45] to analyse adaptive security of multicast encryption protocols. While Panjwani used upper bounds on the security loss involved for GSD to gain a quasipolynomial security reduction for the Logical Key Hierarchy (LKH) protocol, extending the lower bounds for GSD developed in this work to LKH might allow to prove optimality of this reduction.
Our results for GSD could also find applications in the context of secure group messaging, in particular in the security analysis of a candidate continuous group key-agreement protocol called TreeKEM, which was proposed in [9]. To prove adaptive

security of (variants of) TreeKEM, [1] introduce a public-key variant of GSD. Our results easily extend to lower bounds for this variant of GSD, however it remains open if this can be used to prove a bound on the adaptive security of TreeKEM-like protocols: On one hand, to improve efficiency in these protocols not all keys are derived independently but a hierarchical key derivation is used, which incurs a second type of edges. On the other hand, the adversary's queries are very restricted due to the special functioning of update operations.

References

[1] J. Alwen, M. Capretto, M. Cueto, C. Kamath, K. Klein, I. Markov, G. Pascual-Perez, K. Pietrzak, M. Walter, and M. Yeo. Keep the dirt: Tainted treekem, adaptively and actively secure continuous group key agreement. 2019. https://eprint.iacr.org/2019/1489. (Cited on pages 2, 3, 12, 29, 40 and 45.)

[2] J. Alwen, S. Coretti, Y. Dodis, and Y. Tselekounis. Security analysis and improvements for the ietf mls standard for group messaging. In D. Micciancio and T. Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 248–277, Cham, 2020. Springer International Publishing. (Cited on pages 2, 3, 12 and 40.)

[3] J. Alwen and V. Serbinenko. High parallel complexity graphs and memory-hard functions. In R. A. Servedio and R. Rubinfeld, editors, *47th ACM STOC*, pages 595–603. ACM Press, June 2015. (Cited on page 13.)

[4] G. Ateniese, K. Benson, and S. Hohenberger. Key-private proxy re-encryption. In M. Fischlin, editor, *CT-RSA 2009*, volume 5473 of *LNCS*, pages 279–294. Springer, Heidelberg, Apr. 2009. (Cited on page 34.)

[5] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS 2005*. The Internet Society, Feb. 2005. (Cited on page 34.)

[6] M. Bellare, V. T. Hoang, and P. Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In X. Wang and K. Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 134–153. Springer, Heidelberg, Dec. 2012. (Cited on page 12.)

[7] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, Heidelberg, Apr. 2009. (Cited on page 12.)

[8] C. H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776, 1989. (Cited on pages 13, 14 and 18.)

[9] K. Bhargavan, R. Barnes, and E. Rescorla. TreeKEM: Asynchronous Decentralized Key Management for Large Dynamic Groups. May 2018. (Cited on page 44.)

[10] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In K. Nyberg and H. M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, Heidelberg, Aug. 2003. (Cited on page 2.)

[11] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In K. Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 127–144. Springer, Heidelberg, May / June 1998. (Cited on pages 3 and 34.)

[12] D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In K. Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 59–71. Springer, Heidelberg, May / June 1998. (Cited on page 13.)

[13] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multiparty computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996. (Cited on page 12.)

[14] F. Chung, P. Diaconis, and R. Graham. Combinatorics for the east model. *Advances in Applied Mathematics*, 27(1):192–206, 2001. (Cited on pages 13 and 18.)

[15] A. Cohen. What about bob? The inadequacy of CPA security for proxy reencryption. In D. Lin and K. Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 287–316. Springer, Heidelberg, Apr. 2019. (Cited on page 35.)

[16] J.-S. Coron. On the exact security of full domain hash. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Heidelberg, Aug. 2000. (Cited on page 5.)

[17] C. Dwork, M. Naor, O. Reingold, and L. J. Stockmeyer. Magic functions. In *40th FOCS*, pages 523–534. IEEE Computer Society Press, Oct. 1999. (Cited on page 12.)

[18] C. Dwork, M. Naor, and H. Wee. Pebbling and proofs of work. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 37–54. Springer, Heidelberg, Aug. 2005. (Cited on page 13.)

[19] S. Dziembowski, T. Kazana, and D. Wichs. One-time computable self-erasing functions. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 125–143. Springer, Heidelberg, Mar. 2011. (Cited on page 13.)

[20] G. Fuchsbauer, Z. Jafargholi, and K. Pietrzak. A quasipolynomial reduction for generalized selective decryption on trees. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 601–620. Springer, Heidelberg, Aug. 2015. (Cited on page 12.)

[21] G. Fuchsbauer, C. Kamath, K. Klein, and K. Pietrzak. Adaptively secure proxy re-encryption. In D. Lin and K. Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 317–346. Springer, Heidelberg, Apr. 2019. (Cited on pages 3, 12, 31, 33, 34, 35, 36, 37 and 38.)

[22] G. Fuchsbauer, M. Konstantinov, K. Pietrzak, and V. Rao. Adaptive security of constrained PRFs. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 82–101. Springer, Heidelberg, Dec. 2014. (Cited on pages 12, 13 and 44.)

[23] S. Garg, R. Ostrovsky, and A. Srinivasan. Adaptive garbled RAM from laconic oblivious transfer. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 515–544. Springer, Heidelberg, Aug. 2018. (Cited on page 12.)

[24] S. Garg and A. Srinivasan. Adaptively secure garbling with near optimal online complexity. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 535–565. Springer, Heidelberg, Apr. / May 2018. (Cited on page 12.)

[25] R. Gennaro, Y. Gertner, J. Katz, and L. Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Comput.*, 35(1):217–246, 2005. (Cited on page 13.)

[26] R. Gennaro and L. Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st FOCS*, pages 305–313. IEEE Computer Society Press, Nov. 2000. (Cited on page 13.)

[27] C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In L. Fortnow and S. P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. (Cited on page 13.)

[28] Y. Gertner, T. Malkin, and O. Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *42nd FOCS*, pages 126–135. IEEE Computer Society Press, Oct. 2001. (Cited on page 38.)

[29] O. Goldreich, S. Goldwasser, and S. Micali. On the cryptographic applications of random functions. In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 276–288. Springer, Heidelberg, Aug. 1984. (Cited on page 44.)

[30] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982. (Cited on page 36.)

[31] D. Hefetz, M. Krivelevich, M. Stojakovic, and T. Szabó. *Positional Games*. Birkhäuser Basel, 2014. (Cited on page 9.)

[32] B. Hemenway, Z. Jafargholi, R. Ostrovsky, A. Scafuro, and D. Wichs. Adaptively secure garbled circuits from one-way functions. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 149–178. Springer, Heidelberg, Aug. 2016. (Cited on page 12.)

[33] C.-Y. Hsiao and L. Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 92–105. Springer, Heidelberg, Aug. 2004. (Cited on page 10.)

[34] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989. (Cited on pages 10 and 12.)

[35] Z. Jafargholi, C. Kamath, K. Klein, I. Komargodski, K. Pietrzak, and D. Wichs. Be adaptive, avoid overcommitting. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 133–163. Springer, Heidelberg, Aug. 2017. (Cited on pages 3, 4, 6, 12, 13, 14, 27, 31 and 44.)

[36] Z. Jafargholi and D. Wichs. Adaptive security of Yao's garbled circuits. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 433–458. Springer, Heidelberg, Oct. / Nov. 2016. (Cited on page 12.)

[37] S. Katsumata, R. Nishimaki, S. Yamada, and T. Yamakawa. Compact NIZKs from standard assumptions on bilinear maps. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 379–409. Springer, Heidelberg, May 2020. (Cited on pages 3 and 12.)

[38] J. H. Kim, D. R. Simon, and P. Tetali. Limits on the efficiency of one-way permutation-based hash functions. In *40th FOCS*, pages 535–542. IEEE Computer Society Press, Oct. 1999. (Cited on page 13.)

[39] L. Kowalczyk and H. Wee. Compact adaptively secure ABE for NC^1 from k-lin. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476

of *LNCS*, pages 3–33. Springer, Heidelberg, May 2019. (Cited on pages 3, 12 and 31.)

[40] R. Královič. *Time and Space Complexity of Reversible Pebbling*, pages 292–303. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. (Cited on page 13.)

[41] A. B. Lewko and B. Waters. Why proving HIBE systems secure is difficult. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 58–76. Springer, Heidelberg, May 2014. (Cited on page 13.)

[42] M. Li, J. Tromp, and P. Vitányi. Reversible simulation of irreversible computation. *Physica D: Nonlinear Phenomena*, 120(1):168 – 176, 1998. Proceedings of the Fourth Workshop on Physics and Consumption. (Cited on page 19.)

[43] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, Heidelberg, Aug. 2002. (Cited on page 12.)

[44] J. Nordström. *New Wine into Old Wineskins: A Survey of SomePebbling Classics with Supplemental Results*. 2015. (Cited on page 13.)

[45] S. Panjwani. Tackling adaptive corruptions in multicast encryption protocols. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 21–40. Springer, Heidelberg, Feb. 2007. (Cited on pages 2, 3, 6, 12, 27 and 44.)

[46] C. H. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31(2):288 – 301, 1985. (Cited on page 10.)

[47] R. Pass. Unprovable security of perfect NIZK and non-interactive non-malleable commitments. In A. Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 334–354. Springer, Heidelberg, Mar. 2013. (Cited on page 13.)

[48] M. S. Paterson and C. E. Hewitt. Record of the project mac conference on concurrent systems and parallel computation. chapter Comparative Schematology, pages 119–127. ACM, New York, NY, USA, 1970. (Cited on page 13.)

[49] O. Reingold, L. Trevisan, and S. P. Vadhan. Notions of reducibility between cryptographic primitives. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 1–20. Springer, Heidelberg, Feb. 2004. (Cited on pages 4, 10 and 12.)

[50] S. Rudich. *Limits on the Provable Consequences of One-way Functions*. PhD thesis, EECS Department, University of California, Berkeley, Dec 1988. (Cited on page 12.)

[51] J. E. Savage. *Models of computation - exploring the power of computing*. Addison-Wesley, 1998. (Cited on pages 7, 13 and 21.)

[52] D. R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In K. Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 334–345. Springer, Heidelberg, May / June 1998. (Cited on page 12.)