The Cost of Adaptivity in Security Games on Graphs

Chethan Kamath[1]*, Karen Klein[2]**, Krzysztof Pietrzak[3]***, and Michael Walter[4]

[1] Tel Aviv University, ckamath@protonmail.com
[2] ETH Zurich, karen.h.klein@protonmail.com
[3] IST Austria, pietrzak@ist.ac.at
[4] Zama, Paris michael.walter@zama.ai

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, prefix-constrained PRFs and generalized selective decryption, a security game that captures the security of certain group messaging and broadcast encryption schemes. Those primitives have in common that their security game involves an underlying graph that can be adaptively built by the adversary.

Some of our lower bounds only apply to a restricted class of black-box reductions which we term "oblivious" (the existing upper bounds are of this restricted type), some apply to the broader but still restricted class of non-rewinding reductions, while our lower bound for proxy re-encryption applies to all black-box reductions. The fact that some of our lower bounds seem to crucially rely on obliviousness or at least a non-rewinding reduction hints to the exciting possibility that the existing upper bounds can be improved by using more sophisticated reductions.

Our main conceptual contribution is a two-player multi-stage game called the Builder-Pebbler Game. We can translate bounds on the winning probabilities for various instantiations of this game into cryptographic lower bounds for the above-mentioned primitives using oracle separation techniques.

* Supported by Azrieli International Postdoctoral Fellowship. Most of the work was done while the author was at Northeastern University and Charles University, funded by the IARPA grant IARPA/2019-19-020700009 and project PRIMUS/17/SCI/9, respectively.

** Supported in part by ERC CoG grant 724307. Most of the work was done while the author was at IST Austria funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (682815 - TOCNeT).

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

1 Introduction

Consider the following game played between a challenger C and an adversary A using a symmetric-key encryption (SKE) scheme $(\mathsf{Enc}, \mathsf{Dec})$. The challenger first samples, independently and uniformly at random, N keys $\mathtt{k}_1, \ldots, \mathtt{k}_N$. 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 $\mathtt{k}_j$ under the key $\mathtt{k}_i$ to obtain $\mathsf{Enc}(\mathtt{k}_i, \mathtt{k}_j)$, or
2. Corrupt a user U_i to obtain the key $\mathtt{k}_i$.

At the end of the game, A challenges C on a user U_{i^*} and is given either the real key $\mathtt{k}_{i^*}$ 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 requests the encryption of $\mathtt{k}_j$ under the key $\mathtt{k}_i$. 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 $\mathtt{k}_{i^*}$.

The above game is called *generalised selective decryption* (GSD) and it captures the security of protocols for multicast encryption [43] and continuous group key agreements (CGKA) [2, 1]. We will use GSD in this introduction as the running example to convey our ideas. The main question regarding GSD 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 SKE.[5] For this we need to prove a computational soundness (i.e., security) theorem of the form: if the SKE 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., $\epsilon' = \epsilon \cdot \mathbf{poly}(N)$. Otherwise, this requires to either set the security parameter of the underlying SKE very large if one wants to maintain provable security guarantees, which will lead to inefficiency. Or the provided security is only heuristic, leaving the possibility of an attack against GSD which does not break the underlying SKE.

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 $O(N^2)$ hybrids, each of which can be shown indistinguishable under the security of the

[5] In case the key-graph contains cycles, one must additionally assume that the SKE is key-dependent message (KDM) secure [7]. 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 where the acyclicity is enforced by the protocol rules.

SKE (see, e.g., [31]). 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 [43] and remains notoriously hard. In particular, non-trivial results are only known in settings, where the adversary is restricted to query (subgraphs of) 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 [31, 38].

1.1 Our Results

The Piecewise-Guessing Framework has been successfully used to give improved security guarantees against adaptive attacks for various applications [38, 20, 1, 2, 35], but there still are significant gaps to knows attacks. In this paper we approach this question from the other "lower bounds" direction, and for several applications show that this will not be possible, at least not when using existing techniques. In particular, (in the full version [33] of this paper) we show that there do not exist efficient non-rewinding black-box reductions – henceforth called "straight-line" reductions for brevity – that prove security of

- certain forms of restricted GSD (including its public key variant) based on the IND-CPA security of the underlying SKE (see Section 6),
- popular protocols for CGKA based on the IND-CPA security of the underlying public-key encryption (PKE) (see full version [33, Section 7]),
- the GGM construction for prefix-constrained PRFs based on the pseudorandomness of the underlying PRG (see Section 7)
- proxy re-encryption[6] (PRE) schemes [8] based on the IND-CPA security of the PKE and N-weak key privacy (see full version [33, Section 9])

with only polynomial loss in advantage. For PRE we can even rule out general (i.e., rewinding) black-box reductions. For the theorem statements of the latter three results, we refer to the corresponding sections, but we will discuss GSD in a little more detail, so we provide an informal statement here.

Theorem 1 (Informally Stated, Corollary 1). *Any straight-line reduction proving security of unrestricted adaptive* GSD *based on the IND-CPA security of the underlying SKE scheme loses at least a factor that is* super-polynomial *($N^{\Omega(\log N)}$) in the number of users N.*

For the proof we rely heavily on the adversary's freedom to query arbitrary directed acyclic graphs (DAG). (Actually, the graphs have some structure and so certain conditions may be imposed on it but these restrictions are very weak.)

[6] 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.

[7] Recall that a tree does not necessarily have to be rooted, so this includes any DAG such that the corresponding undirected graph does not contain any cycles.

Application	Underlying Graph	Lower Bound	Reduction	Upper Bound
GSD	Path P_N	$N^{\Omega(\log(N))}$	Oblivious	$N^{O(\log(N))}$ [19]
	Rooted Binary In-Tree B_n	$N^{\Omega(\log(N))}$	Oblivious	$N^{O(\log(N))}$ [43]
	Tree[7]	$N^{\Omega(\log(N))}$	Straight-line	$N^{O(\log(N))}$ [19]
	Arbitrary DAG	$2^{\Omega(\sqrt{N})}$	Oblivious	$N^{O(N/\log(N))}$ [31]
PRE	Path P_N	$N^{\Omega(\log(N))}$	Oblivious	$N^{O(\log(N))}$ [20]
	Binary Tree B_n	$N^{\Omega(\log(N))}$	Oblivious	$N^{O(\log(N))}$ [20]
	Arbitrary DAG	$2^{\Omega(N)}$	Arbitrary	$N^{O(N/\log(N))}$ [20]
GGM CPRF	Tree	$n^{\Omega(\log(n))}$	Straight-line	$n^{O(\log(n))}$ [21]
TreeKEM	Regular Tree	$M^{\Omega(\log(\log(M)))}$	Straight-line	$Q^{O(\log(M))}$ [1]

Table 1: Summary of lower bounds on the loss in security established in our work. $N = 2^n$ denotes the size of the graph. Therefore, in the case of GGM constrained PRF, n denotes the length of the input string. For TreeKEM, M denotes the number of users and Q refers to the number of queries allowed to the adversary.

In many applications however, the adversary is much more restricted in terms of the graphs it can query, e.g. in protocols for multicast encryption like logical key hierarchies (LKH) [51, 52, 13], and hence our bound does not apply. However, for a certain sub-class of straight-line reductions, which we term "oblivious" (see discussion below), we obtain results for such applications. These results show that the upper bounds for GSD given in [31], which are oblivious, are essentially tight and can only be improved by exploiting new non-oblivious techniques (and similarly for the bounds for PRE given in [20]), as stated informally below.

Theorem 2 (Informally Stated, Corollaries 2 to 4 in Full Version [33]). *Any oblivious reduction proving security of adaptive GSD restricted to* paths or binary trees *based on the IND-CPA security of the underlying SKE scheme loses a factor that is* super-polynomial *($N^{\Omega(\log N)}$) in the number of users N; for unrestricted GSD the loss is* sub-exponential *($2^{\Omega(\sqrt{N})}$).*

Our results for PRE have a similar flavor, but are even stronger, since in this case the reduction is naturally more restricted. A summary of the results can be found in Table 1.

The common thread to the applications we consider is that their security game can be abstracted out by a two-player multi-stage game which we call the "Builder-Pebbler Game". We are unable to establish lower bounds for other applications of the Piecewise-Guessing Framework (e.g., computational secret sharing or garbling circuits) 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., the circuit to be garbled or the access structure) in these applications is fixed ahead of the time and the adaptivity comes from other sources (e.g., choice of garbling input or targeted user). Therefore we would require other combinatorial abstractions to establish lower bounds for them. In fact, building on the high level ideas introduced in this work, [34] showed lower bounds for adaptive security of Yao's garbling (see Section 1.2.2 for a comparison). We defer the discussion on the

Builder-Pebbler Game to the next section (Section 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 reductions. Oblivious reductions are a certain class of black-box reductions and our definition is motivated by the reductions in [31]. 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 inconsistencies 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*.[8] 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. A formal definition of an oblivious GSD reduction is given in the full version [33, Definition 21]; the corresponding definition for PREs is given in the full version [33, Definition 34].

Why oblivious reductions? We note that oblivious reductions are a quite natural notion, since they can easily be defined uniformly for all adversaries. Not surprisingly, they encompass some of the key reductions in the literature. Beside the reductions proposed and analysed in [31] (and its follow-up works), partitioning-based reductions, which have been successfully employed in a plethora of works [15], also roughly behave in an oblivious manner.[9] Moreover, oblivious reductions encompass the currently-known techniques for establishing upper bounds for primitives with dynamic graph-based security games, like GSD, PRE, CPRFs

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

[9] On every signature query issued by the adversary, the reduction in [15] 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.

etc.. Therefore, our results imply that in order to obtain better upper bounds on the loss function Λ even in the more restricted settings, one needs to deviate significantly from the current proof techniques (i.e., non-oblivious or rewinding reductions for GSD and restricted PRE). Accordingly, our results on oblivious reductions should not be viewed as separations, but rather as a guide towards new avenues to finding better reductions by ruling out a large class of reductions – such possibilities are discussed in the full version [33, Section 10].

1.2 Related Work

1.2.1 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 [12]. 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 [41].

The notion of generalised selective decryption (GSD) was introduced by Panjwani [43] to study adaptive corruption in restricted settings. His motivation was to better understand the problem of selective decommitment [16] (which is also known as selective opening in some works [5]) and the closely-related problem of selective decryption. The problem was further studied by Fuchsbauer et al. [19] 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. [4], Hemenway et al. [29] and Jafargholi and Wichs [32]. The latter two works are especially relevant since they established a relationship between adaptive security and graph pebbling. It is also worth noting that the study of adaptive security of garbled RAM was carried out in [23, 22].

The above two series of works culminated in the Piecewise-Guessing Framework of Jafargholi et al. [31] who managed to abstract out the ideas therein and give even more fine-grained reductions. In addition to capturing the results from [32, 19, 21], 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 [38], proxy re-encryption schemes [20], continuous group key-agreement [1, 2] and non-interactive zero-knowledge [35].

1.2.2 Limitations of Reductions The study of limitations of reductions (see Footnote 13) was initiated in the seminal work of Impagliazzo and Rudich [30]. They used *oracle separations* to rule out fully black-box reduction of key agreement to symmetric-key primitives. This approach turned out quite useful and has been further exploited to rule out fully black-box reduction of a variety of cryptographic primitives from one another (e.g., [48, 50]). A fine-grained study of the notion of reductions and separations was later carried out by Reingold et al. [47].

In addition to ruling out reductions, the more fine-grained question of efficiency of reduction of one primitive to another has also been studied [25, 24, 37]. 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 [40], who showed that the security of adaptively-secure hierarchical identity-based encryption must degrade exponentially in the depth, and Fuchsbauer et al. [21], who showed that certain types of constrained PRFs must incur an exponential loss (in the size of the input) in adaptive security. Note that this class of constrained PRFs does not include the prefix-constrained PRF construction we consider in this work. Both aforementioned works employ the more recent meta-reduction technique [9, 26, 45], which is of different flavour from oracle separations.

Comparison with [34]. Building on the high level ideas in this paper, [34] showed lower bounds on the adaptive security of Yao's garbling scheme. As pointed out in the introduction, the graph (i.e., the circuit) in Yao's garbling scheme is fixed ahead of time and the adaptivity comes from the choice of (garbling) input. (The difficulty of the reduction comes from having to guess the bits running over a subset of wires during evaluation of the circuit.) Therefore they had to rely on a different combinatorial abstraction from Builder-Pebbler Game (viz., a black-gray pebble game on the circuit) to establish their lower bound. However, since the security game for Yao's garbling consists of just two rounds, [34] did not encounter some of the difficulties (to do with the multiple rounds of interaction) we do and therefore were able to rule out arbitrary black-box reductions. While both [34] and this work model choices made by a reduction by putting pebbles on a graph structure, the analogy basically ends there. None of the main ideas from [34] seem applicable in this setting and vice versa.

1.2.3 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 [17, 18, 3]. The notion of node pebbling first appeared (albeit implicitly) in [46], whereas the notion of *reversible* node pebbling was introduced by Bennett to study reversible computation [6]. The notion of edge pebbling used in this work is defined in [31]. 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 [39]. As for the lower bound on the node pebbling complexity for binary trees, a proof can be found in [49]. We refer the reader to the textbook by Savage [49] or the excellent survey by Nordström [42] for more details on pebbling.

2 Technical Overview

figuration graph, cuts in pebbling configuration graphs, counting arguments On a high level, our approach can be divided into two steps. In the first step (Section 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 (Section 2.3), translated to lower bounds on the loss in security of concrete cryptographic protocols using oracle separation techniques to yield the results stated in Section 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.

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 SKE scheme loses a significant (super-polynomial) factor in the advantage. Since we are aiming to rule out black-box reductions, we have the luxury of constructing inefficient adversaries and SKE schemes. 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 6, any reduction attempting to take advantage of our adversaries must send its 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. Its 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-pebbling* 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 reduction's 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 $2^{|\mathcal{E}|}$

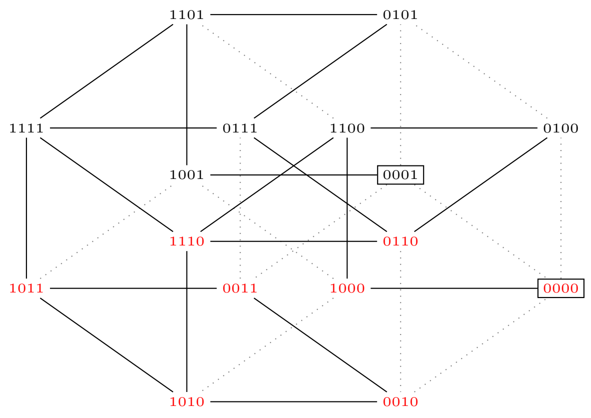

Fig. 1: Configuration graph for paths of length four, $C_4 = ([5], \{(1,2),(2,3),(3,4),(4,5)\})$. It is a subgraph of the Boolean hypercube of dimension four (the missing edges are dotted). The labels of the vertices encode the pebbling status of the corresponding edge and therefore represents a pebbling configuration: e.g., the vertex labelled 0000 is completely unpebbled (configuration $\mathcal{P} = \emptyset$) whereas the vertex labelled 1000 has a pebble only on the first edge $(1,2)$ (configuration $\mathcal{P} = \{(1,2)\}$). 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. The special vertices for $\mathcal{P}^{C_4}$ are $\mathcal{P}_{\text{start}} = 0000$ and $\mathcal{P}_{\text{target}} = 0001$ (both boxed). A cut for this configuration graph consists of the set of (red) vertices that lie on the 'bottom' half of the graph: $\{0000, 0010, 1010, 1011, 0011, 1000, 0110, 1110\}$. The set of edges from the top half to the bottom half form cut set: $\{(1111, 1110), (1111, 1011), (1100, 1110), (1100, 1000), (0111, 0110), (0100, 0110)\}$.

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. 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 [43, 31]. 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.

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.

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: Section 2.2), one of the players. Then we will use oracle separation techniques to translate these upper bounds into cryptographic lower bounds (Step II: Section 2.3).

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 Section 4). 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 pebbling configuration $\mathcal{P}$ that is in a designated subset of all configurations. This winning set 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 Builder is restricted to query graphs G that are subgraphs of some family of graphs $\mathcal{G}$. 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 winning configuration and the Builder preventing this from happening.[10]

[10] This is reminiscent of Maker-Breaker games [28], 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

Defining winning configurations via cuts of the configuration graph. Although the Builder-Pebbler Game is meaningful for any notion of winning configuration, we are interested in a particular definition that is essential in establishing our cryptographic lower bounds: we will set the winning configurations as the ones that belong to bottleneck configurations in the configuration graph of G. The goal is to prove that it will be difficult for Pebblers to get into such configurations. In some cases we can do so directly, but in others the Pebbler may be able to achieve this by "flooding" the graph with many pebbles. Our solution is to "artificially" restrict the Pebbler to placing very few pebbles by requiring it to leave the part of the query graph that is not in the challenge graph entirely unpebbled, i.e., if at the end of the game there is a pebble on an edge that is not rooted in the challenge graph, the Pebbler loses. Note that this does not trivialize our task of finding a suitable Builder, because for our application to cryptographic lower bounds to work, the Builder's querying strategy (including the challenge) needs to be independent of the pebbles placed by the Pebbler. (We call such Builders also *oblivious*, see below.) Of course, care must be taken that this behaviour cannot be exploited by the reduction. Intuitively, the reason this works is that in all our applications, if the reduction were to embed the challenge outside of the challenge graph, our adversaries will almost always interpret it to be a pebble, no matter if the challenge was real or random.

Combinatorial Upper Bounds in the Builder-Pebbler Game. We bound the advantage of Pebblers from above against Builders with varying degree of freedom, i.e., Builders that are restricted to querying certain classes of graphs. The upper bounds in Theorems 3 to 5 are (almost) tight since a random Pebbler yields (almost) matching lower bounds.

Theorem 3 (Informally Stated, Theorems 6 and 8 in Full Version [33]). *Any oblivious*[11] *Pebbler in the Builder-Pebbler Game restricted to* paths or binary trees *has advantage at most inverse* quasi-polynomial *($N^{-\Omega(\log N)}$) in N, the size of the graph.*

Theorem 4 (Informally Stated, Theorem 9 in Full Version [33]). *Any oblivious Pebbler in the unrestricted Builder-Pebbler Game has advantage at most inverse* sub-exponential *($2^{-\Omega(\sqrt{N})}$) in N, the size of the graph.*

Theorem 5 (Informally Stated, Theorem 6). *Any Pebbler in the Builder-Pebbler Game restricted to* trees *has advantage at most inverse* quasi-polynomial *($N^{-\Omega(\log N)}$) in N, the size of the graph.*

in Maker-Breaker games one usually considers optimal (deterministic) strategies, whereas we consider randomised strategies for the 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.

[11] The notion of obliviousness for Pebblers is naturally derived from the one for reductions, see discussion above and Definition 9.

Remark 1 (On Builder Obliviousness). It is worth mentioning that all our Builder strategies are also *oblivious*, where oblivious is defined different for Builders than for Pebblers: it means that the query strategy is independent of the Pebbler's responses (see Section 4.1).[12] 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.

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

For translating upper bounds established in Step I into loss in security of concrete cryptographic protocols, we adapt ideas from oracle separations.

Ruling out tight black-box reductions. Oracle separations are used to rule out the reduction[13] of a primitive Q (e.g., PKE) to another primitive P (e.g., SKE). Our case is slightly different since it involves a primitive P (e.g., SKE) that is used in a graph-based "multi-instance" setting Q^P (e.g., GSD with SKE). In this setting, we are interested in the more fine-grained question of bounding Λ, the loss in security incurred by any efficient black-box reduction R that breaks P when given black-box access to an adversary that breaks Q^P (i.e., from P to Q^P). This means we must show that for *every* R, there exists

- an instance P (not necessarily efficiently-implementable) of P and
- an adversary A_Q (not necessarily efficient) that breaks Q^{P}

such that the loss in security incurred by R in breaking P is at least Λ.[14] To this end, we establish a tight *coupling* between the security game for Q^P and the Builder-Pebbler Game (e.g., Lemma 2). If Q^P involves a graph family $\mathcal{G}$ then the Builder-Pebbler Game will be played on $\mathcal{G}$ (or sometimes another family

[12] 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* [44].

[13] The usage of the word 'reduction' here and in Section 1.2.2 is in a constructive sense [47]: a primitive Q is reduced to another primitive P if (i) there is an efficient *construction* C that takes an implementation P of P and gives an implementation Q of Q and (ii) there is an efficient *security reduction* R which takes an adversary A_Q that breaks Q and constructs an adversary A_P that breaks P. For example, the most common type of reduction used in cryptography is a *fully* black-box reduction where both R and C are black-box in that they only have black-box access to P and A_Q, respectively. In the rest of the paper, 'reduction' is used to refer to a security reduction as in (ii).

[14] This is obtained by simply negating the definition of a black-box reduction: *there exists an efficient reduction* R *such that for every (not necessarily efficient) implementation* P *of* P *and for every (not necessarily efficient) adversary* A_Q *that breaks* Q^{P} *the loss in security is at most* Λ.

related to $\mathcal{G}$) and the winning condition is determined by the cut for $\mathcal{G}$. The coupling is established using a Builder strategy B and a related adversary A_Q such that

– every reduction R can be translated to a Pebbler strategy P against B on $\mathcal{G}$, and
– if R has a security loss of at most Λ then B's advantage against P is at least $1/\Lambda$ (up to negligible additive factors).

If $\mathcal{G}$ is a class for which we derived an upper bound of π for Pebbler strategies (in Step I) then any reduction R such that $1/\Lambda > \pi$ cannot exist. Put differently, 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. In the remainder of the section, we explain how the coupling works in a bit more detail using GSD on binary trees as the running example. To keep the exposition simple, we will brush a lot of issues (e.g., dealing with 'flooding' reductions) under the rug and refer the readers to Section 6 for a more formal treatment.

Example: GSD on Binary Trees. Let's consider the case where P is SKE and Q^P is the GSD game played on $\mathcal{G} = \mathcal{B}_n$, the class of binary trees of depth n. Intuitively, the GSD adversary A_Q "simulates" the oblivious Builder B used to derive Theorem 3. That is, it

1. chooses a binary tree $B_n \in \mathcal{B}_n$ uniformly at random,
2. queries, in a random order, each edge $(u, v) \in E(B_n)$ to obtain the corresponding ciphertext $\mathsf{Enc}(\mathtt{k}_u, \mathtt{k}_v)$ from the reduction R and
3. challenges the sole sink T at the end of the game.

For it to be a valid adversary, A_Q must distinguish the extreme games, i.e., the real game where all the ciphertexts are real and the random game where the ciphertexts incoming to T are both random. To this end, it looks at the ciphertexts it obtained and extracts a pebbling configuration $\mathcal{P}$ from it (as described in Section 2.1). Note that the extreme hybrids corresponds to $\mathcal{P}_{\text{start}} = \emptyset$ (real) and $\mathcal{P}_{\text{target}}$ such that both the edges incoming to T carry a pebble (random). A_Q distinguishes these by concentrating all its advantage in the cut in the configuration graph of B_n defined in Section 2.1: i.e., it outputs 0 if $\mathcal{P}$ is on one side of the cut and 1 otherwise. To help A_Q faithfully distinguish real ciphertexts from random ones so that it can infer the exact pebbling configuration $\mathcal{P}$, we fix P to be an ideal implementation $(\mathsf{Enc}, \mathsf{Dec})$ of SKE:

– Enc is a random expanding function that implements encryption and
– Dec is the decryption function defined to be "consistent" with Enc.[15]

[15] Since most of our ideal functionalities are implemented using random oracles, it is possible using standard tricks [30] to switch the order of the quantifiers and establish the stronger statement that there exists a *single* oracle P and adversary A_Q which work *for all* reductions.

Since Enc is injective with overwhelming probability, given a ciphertext A_Q can brute force Enc to determine (exactly) whether or not the ciphertext corresponding to an edge is real. By carrying this out for all the edges, it can extract a *unique* pebbling configuration corresponding to R's simulation. Since A_Q concentrates its advantage in the cut, for R to have any chance of winning, its own challenge c^* must be 'embedded at the cut' so that – depending on whether or not c^* is real – $\mathcal{P}$ switches from one side of the cut to the other. Since this is the *only* way R can exploit A_Q, we may infer that a reduction with loss in security at most Λ ends up in the cut with probability at least $1/\Lambda$. However, thanks to the fidelity of the extraction, this also means that the natural Pebbler strategy P that underlies R, which simply places a pebble whenever R fakes, wins against B in the Builder-Pebbler Game on $\mathcal{B}_n$ with an advantage at least $\pi = 1/\Lambda$ (formally, Lemma 1). If particular, if Λ is significantly less than quasi-polynomial in $N = 2^n$, it would imply the existence of a Pebbler that is successful with a probability greater than inverse quasi-polynomial, a contradiction to Theorem 3. Since Theorem 3 only holds for oblivious Pebblers, the bound on Λ only holds for oblivious GSD reductions.

3 Preliminaries

We use the notation $[N] = \{1, \ldots, N\}$ and $[N]_0 = \{0\} \cup [N]$. For a string $x = x_0, \ldots, x_{n-1} \in \{0,1\}^n$, for $0 \leq a \leq b < n$, we use $x[a, b]$ to denote the substring $x_a, \ldots, x_b$.

3.1 Graph Theory

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 set of sinks $\mathcal{T}$. For a subset $\mathcal{S} \subseteq [N]$ of nodes, let $\mathrm{in}(\mathcal{S})$ denote the set of ingoing edges and $\mathrm{parents}(\mathcal{S})$ 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 $E(G)$ (resp., $V(G)$) the edges $\mathcal{E}$ (resp., vertices $\mathcal{V}$) of G. By B_n, we denote a binary tree of depth n – the binary tree is *perfect* if it has all $2^{n+1} - 1$ vertices. We assume the standard indexing of the vertices in B_n by associating them with binary strings in $\{0,1\}^{\leq n}$ determined by their position in the tree: i.e., the root has index ε and the left (resp., right) child of a vertex with index i is $i\|0$ (resp., $i\|1$).

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 $\mathcal{P} \subseteq \mathcal{E}$ 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 $\mathcal{S} \subseteq \mathcal{V}(\mathcal{P})$. We denote the size of a minimal vertex cover of $\mathcal{P}$ by*

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

3.2 Graph Pebbling

A *pebbling configuration* on the graph $\mathcal{G}$ is a set $\mathcal{P} \subseteq \mathcal{E}$ 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 $\boldsymbol{\mathcal{P}} = (\mathcal{P}_0, \ldots, \mathcal{P}_\ell)$, 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 $\mathcal{P}_{\text{target}} = \text{in}(T) = \{(i, T) \in \mathcal{E}\}$ denote the configuration where only all the edges incident on some sink $T \in \mathcal{T}$ 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 pebbling game from [31].

Definition 3 (Edge-Pebbling). *An edge pebbling of a DAG $G = (\mathcal{V}, \mathcal{E})$ with* unique sink *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* $\mathcal{P}_i = \mathcal{P}_{i-1} \setminus \{(u, v)\}$,
- $\text{in}(u) \subseteq \mathcal{P}_{i-1}$.

Definition 4 (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 Definition 3, the configuration graph is undirected.

4 The 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 5 (N- and $(N, \mathcal{G})$-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 $(N, \mathcal{G})$-Builder-Pebbler Game, the Builder is restricted to building graphs (isomorphic to subgraphs of) $G \in \mathcal{G}$ for a class of graphs $\mathcal{G}$.

Definition 6 (Winning Condition and Advantage for $(N, \mathcal{G})$-Builder-Pebbler Game). *Consider an $(N, \mathcal{G})$-Builder-Pebbler Game and let $G =$*

$(\mathcal{V}, \mathcal{E})$, $G^* = (\mathcal{V}^*, \mathcal{E}^*)$ and $\mathcal{P}$ be as in Definition 5. 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 $(N, \mathcal{G})$-Builder-Pebbler Game under winning condition X if the following two conditions are satisfied:

1. *only edges rooted in $\mathcal{V}^*$ are pebbled, i.e. $\mathcal{P} \subseteq \{(u, v) \in \mathcal{E} \mid u \in \mathcal{V}^*\}$*
2. *the pebbling induced on G^* satisfies the winning condition, i.e., $\mathcal{P}|_{G^*} \in X(G^*)$.*

Otherwise, the Builder is declared the winner. In case the strategies are randomised, we call the probability (over the randomness of the strategies) 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 2. The corresponding definitions for the N-Builder-Pebbler Game can be obtained by simply ignoring the restriction to $\mathcal{G}$.

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 7 (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 $G \in \mathcal{G}$. For a cut function X defined on $\mathcal{G}$ and $G \notin \mathcal{G}$, we set $X(G) = \emptyset$.*

4.1 Player Strategies

Builder strategies. As motivated in Remark 1, we will be dealing in this paper mostly with a class of Builders who play *independently* of the Pebbler's strategy.[16]

Definition 8 (Oblivious Builders). *We say that a Builder's strategy in the $(N, \mathcal{G})$-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 serves two main purposes.

1. Firstly, it ensures that the Builder-Pebbler Game is not trivial for the cut functions we are interested in: otherwise, it is easy to come up with Builder strategies in which any Pebbler has advantage 0.
2. Moreover, non-oblivious Builder strategies are less interesting in our setting since they could potentially allow reductions to exploit the query behaviour of the adversary built on top of a non-oblivious Builder to gain advantage in the security game.

[16] The exact definition of the strategy will depend on the graph and the application.

Pebbler strategies. Ideally, we would like to establish lower bounds that hold against all Pebblers. Since this is not always possible, we consider Builder-Pebbler Games where the Pebbler strategy is restricted to *oblivious* strategies.

Definition 9 (Oblivious Pebbler). *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 $\mathcal{P}$.*

Note that the notion of obliviousness differs from that in Definition 8. Definition 9 is motivated by oblivious reductions used in [31] (see Section 1.1) and the goal is to capture *prior knowledge* that a Pebbler may have about the graph structure that a Builder builds during the query phase. This is captured in Definition 9 by requiring the Pebbler to commit to a non-trivial vertex cover of the pebbling configuration. This allows compressing of pebbling configurations based on the graph structure: e.g., 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 $[1, N]$. On the other hand, using a minimal vertex cover seems too strong, since we do not actually require it to prove our bounds.

Remark 3. Note that restricting the Builder strategy does not weaken our results: we are constructing lower bounds for reductions and an oblivious Builder gives rise to oblivious adversaries. In contrast restricting to oblivious does weaken the result. However, looking ahead, these restrictions allow us to prove much stronger bounds compared to an unrestricted Pebbler.

5 Combinatorial Upper Bounds

In this section we show upper bounds for Pebblers in the Builder-Pebbler Game by constructing Builders (potentially in a restricted Builder-Pebbler Game) and then showing that no Pebbler can have a good advantage against such a Builder. In the following, we show a bound that holds for arbitrary Pebblers. In the full version of this paper, we also provide bounds for oblivious Pebblers [33, Section 5.1] and so-called *node Pebblers* [33, Section 5.2], i.e. Pebblers that may only pebble all or none of the edges incident on any node.

5.1 Unrestricted Pebblers

In this section we prove a first combinatorial upper bound for unrestricted – i.e., non-oblivious – Pebblers in the Builder-Pebbler Game. While our upper bound on the advantage of unrestricted Pebblers is significantly weaker than the result for oblivious Pebblers in the full version of this paper [33, Section 5.1], it is still non-trivial.

Generalized Pebbling Characteristics of Paths. Let $k \in [N]$ be arbitrary. We prove that any pebbling sequence on a path of length N must contain a pebbling configuration such that $\lfloor \log(\lceil N/k \rceil) \rfloor + 1$ of the $\lceil N/k \rceil$ subpaths of length $\leq k$ contain at least one pebble respectively. For $k = 1$ this result is well-known, for a proof we refer to the full version [33, Section 5.1.1]. Assume, for contradiction, that there exists a $k > 1$ and a valid pebbling strategy $\boldsymbol{\mathcal{P}}$ for paths of length N such that the claim was false. Then this strategy implies a pebbling strategy $\boldsymbol{\mathcal{P}}'$ of complexity less than $\lfloor \log(\lceil N/k \rceil) \rfloor + 1$ for paths of length $\lceil N/k \rceil$ as follows: For each pebbling configuration $\mathcal{P}$ in $\boldsymbol{\mathcal{P}}$, define $\mathcal{P}'$ in $\boldsymbol{\mathcal{P}}'$ to contain a pebble on the ith edge if the ith subpath of $\mathcal{P}$ contains a pebble. Cancelling redundant steps in $\boldsymbol{\mathcal{P}}'$, i.e., configurations that equal the preceding configuration in the sequence, implies a valid pebbling sequence of complexity less than $\lfloor \log(\lceil N/k \rceil) \rfloor + 1$ for paths of length $\lceil N/k \rceil$ – a contradiction.

We will use the following definition of k-cuts for paths matching this generalized pebbling lower bound.

Definition 10 (k-good pebbling configurations, k-cuts and k-cut function for paths). *For $k \in \mathbb{N}$ we call a pebbling configuration $\mathcal{P}$ for a path $C = C_N$ on N nodes k-*good *if $\lfloor \log(\lceil N/k \rceil - 1) \rfloor$ of the $\lceil N/k \rceil - 1$ non-source subpaths of C of length $(\leq)k$ contain at least one pebble respectively*[17]*, and there exists a valid pebbling sequence $\boldsymbol{\mathcal{P}} = (\mathcal{P}_{\text{start}}, \ldots, \mathcal{P})$ such that in all configurations in $\boldsymbol{\mathcal{P}}$ at most $\lfloor \log(\lceil N/k \rceil - 1) \rfloor$ of the subpaths simultaneously carry a pebble. We define a k-*cut set *$\mathcal{X}$ in the configuration graph $\mathcal{P}^C$ as the set of all edges consisting of a k-good pebbling configuration and a configuration which can be obtained from this good configuration by* adding *one pebble (following the pebbling rules) in a previously unpebbled subpath. The k-cut function $X_{\mathcal{C},k}$ is defined as in Definition 7 as the frontier of this cut.*

The Upper Bound. The Builder strategy is to query a (polynomial-sized) subgraph of an exponential-sized tree of *outdegree* $\delta_{out} \geq 2$, so that in order to pebble any edge in the final challenge path the Pebbler has to guess one out of many source nodes at the same depth in the tree.

Theorem 6 (Combinatorial Upper Bound for Unrestricted Pebblers). *Let $\mathcal{G}$ be the family of directed trees on $N = 2^n$ nodes (with $n \in \mathbb{N}$). Then there exists a Builder strategy querying a challenge path $G^* \in \mathcal{C}_{\sqrt{N}}$, such that the advantage of* any *Pebbler against this Builder in the $(N, \mathcal{G})$-Builder-Pebbler Game with the winning condition $X_{\mathcal{C}_{\sqrt{N}},1}$ defined as in Definition 10 is at most*

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

Let $\mathcal{G}_2 \subset \mathcal{G}$ be the subset of graphs in $\mathcal{G}$ of bounded outdegree $\delta_{out} = 2$. Then there exists a Builder strategy querying a challenge path $G^* \in \mathcal{C}_{\sqrt{N}}$, such that the advantage of any *Pebbler against Builder in the $(N, \mathcal{G}_2)$-Builder-Pebbler Game*

[17] For technical reasons, we exclude the first subpath of length k in C.

with the winning condition $X_{\mathcal{C}_{\sqrt{N}},k}$ for $k = \log(N)/4$ defined as in Definition 10 is at most

$$\pi \leq 1/N^{\log(N)/8-\log(\log(N))/4}.$$

Proof. We define a Builder strategy B for graph family $\mathcal{G}_{\delta_{out}}$ of outdegree bounded by δ_{out} as follows: First, B chooses a source node in $[N]$ uniformly at random. It then proceeds in $D = N/\delta_{out}^{2k}$ rounds (where k is the 'overlap parameter' and will be specified later), increasing the current graph's depth by 1 in each round. In each round $R \leq 2k$ and each round $R \not\equiv 1 \mod k$, for all sinks at depth $R-1$ in the current graph B queries δ_{out} outgoing edges respectively. Note, after the first $2k$ rounds, B's queries form a δ_{out}-regular tree directed from root to leaves, with δ_{out}^{2k} sinks at depth $2k$. For all rounds such that $R > 2k$ and $R \equiv 1 \mod k$, the Builder B first chooses an integer $i \in [\delta_{out}^{k}]$ and then only queries edges outgoing from the ith batch of δ_{out}^{k} sinks at depth $R-1$. Finally, B chooses the target node uniformly at random from the δ_{out}^{2k} sinks at depth $D = N/\delta_{out}^{2k}$.

First note that B's queries involve less than $D \cdot \delta_{out}^{2k} = N$ nodes and the challenge graph forms a path of length D. To win the game, the Pebbler needs to place at least one pebble on $\lfloor \log(\lceil D/k \rceil - 1) \rfloor$ of the disjoint subpaths of length k in the challenge path respectively. But whenever it wants to place a pebble in a subpath starting from depth $i \cdot k$ with $i \geq 1$, the Pebbler has to at least guess which of the δ_{out}^{k} sources of edges at depth $i \cdot k$ will end up in the challenge graph. Since this choice is made uniformly at random by the Builder B only after all queries at depth $(i+1) \cdot k$ were made, the advantage of the Pebbler to correctly pebble an edge in the subpath sourced at depth $i \cdot k$ is at most $1/\delta_{out}^{k}$. Since this bound holds also conditioned on the event that previous guesses were done correctly, and to win the game, the Pebbler has to pebble $\lfloor \log(\lceil D/k \rceil - 1) \rfloor$ subpaths of the challenge path, we obtain

$$\pi \leq 1/\delta_{out}^{k \cdot \lfloor \log(\lceil D/k \rceil - 1) \rfloor}. \tag{1}$$

Now, for the graph family $\mathcal{G}$ of unbounded outdegree, we set $\delta_{out} = N^{1/4}$ and $k = 1$ to obtain $D = \sqrt{N}$ and hence $\pi \leq 1/N^{1/4 \log(\sqrt{N})} = 1/N^{\log(N)/8}$. For $\delta_{out} = 2$, on the other hand, we set $k = \log(N)/4$ to obtain $D = \sqrt{N}$ and $\pi \leq 1/N^{1/4(\log(\sqrt{N}) - \log(\log(N)/4))} = 1/N^{\log(N)/8 - \log\log(N)/4}$. ☐

6 Cryptographic Lower Bound I: Generalised Selective Decryption

The generalized selective decryption game (GSD) was informally introduced in Section 1; we refer to the full version [33, Section 6.1] for a formal definition. In the following we interpret the combinatorial upper bound from 5.1 for GSD. In the full version, we establish analogous lower bounds for *public-key* GSD, where PKE is used instead of SKE as the underlying primitive; these can be used as a basis for the lower bound on the continuous group key agreement protocol TreeKEM [33, Section 5.1].

6.1 Lower Bounds for GSD

In many applications one considers games where the adversary's queries are restricted to certain graph structures, e.g., paths, "in-trees" (i.e. rooted trees directed from the leaves to the root), or low-depth graphs. These restrictions depend on the protocol under consideration and often allow to construct stronger reductions.

Interesting upper bounds are known for specific settings for (oblivious) black-box reductions R proving adaptive GSD security based on IND-CPA security (short, GSD reductions). Our results now allow us to prove lower bounds on Λ for GSD with various restrictions (which cover similar settings as known upper bounds). Note that our lower bounds are stronger and more widely applicable the more restrictions they can handle.

Definition 11 (Black-Box and Straight-Line GSD Reduction). R *is a* black-box *GSD reduction if for every SKE* $\mathsf{SKE} = (\mathsf{Enc}, \mathsf{Dec})$ *and every adversary* A *that wins the GSD game played on* SKE*,* R *breaks* SKE*. Moreover, if* A *is an* (ϵ, t) *GSD adversary and* R (ϵ', t')*-breaks* SKE *(where* ϵ' *and* t' *are functions of* ϵ *and* t*) then the loss in security is defined to be* $(t'\epsilon)/(t\epsilon')$*. A black-box GSD reduction* R *is* straight-line *if it, additionally, does not rewind* A.

The following definition mirrors the obliviousness of Pebblers in the context of the Builder-Pebbler Game (cf. Definition 9).

Definition 12 (Oblivious GSD Reduction). *A straight-line GSD reduction* R *(Definition 11) is* oblivious *if it commits to a non-trivial vertex cover of all inconsistent edges at the beginning of the game.*

In all our bounds we require the reduction to assign keys to nodes at the beginning of the game.

Definition 13 (Key-Committing GSD Reduction). *A black-box GSD reduction* R *is* key-commiting *if it commits to an assignment of keys to all nodes at the beginning of the game.*

This is due to the fact that Pebblers in the Builder-Pebbler Game commit to whether an edge is pebbled or not as soon as they respond to the query. Without this 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, 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. 2) In case the adversary is not too restricted (which is application dependent), 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.

Both of these approaches work, but would make the proof more complicated: recall that the challenge node must be a sink, so neither of the two fixes can be applied to it. We can still fix all other nodes (which is sufficient), 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. In the full version [33], we see that some protocols are based on a public key version of GSD rather than the secret key version we consider here. In such cases the public keys are known to the adversary and commit the reduction to the corresponding secret keys and thus no assumption or extra fix are required.

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

Lemma 1 (Coupling Lemma for GSD). *Let $\mathcal{G}$ be a family of DAGs and X a cut function. Let* B *be an oblivious Builder in the $(N, \mathcal{G})$-Builder-Pebbler Game with winning condition X. Then there exists*

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

such that for any key-committing straight-line *reduction* R *there exists a Pebbler* P *such that the advantage $1/\mathit{\Lambda}$ of* R *is at most the advantage π of* P *against* B *(up to an additive term $\mathbf{poly}(N)/2^{\Omega(N)}$). Moreover, if* R *is oblivious then so is* P*.*

Proof. We first construct $\mathit{\Pi} = (\mathsf{Enc}, \mathsf{Dec})$: 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, $\mathsf{Enc}(\mathtt{k}, m; r)$ 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 GameBuilders 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 $(\mathtt{encrypt}, v_i, v_j)$ maps to an edge query (i, j) in the Builder-Pebbler Game.
– A response to a query $(\mathtt{encrypt}, v_i, v_j)$ is mapped to "no pebble" if it consists of a valid encryption of $\mathtt{k}_j$ under the key $\mathtt{k}_i$, and to "pebble" otherwise. (Note that this is always well-defined for key-committing GSD reductions.)
– A corruption query $(\mathtt{corrupt}, v_i)$ is ignored in the Builder-Pebbler Game.

– The challenge query $(\texttt{challenge}, v_t)$ is mapped to the challenge node t.

Let $\mathsf{A} \in \mathbf{PSPACE}$ 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 in using the GSD-adversary A to break 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 $(\mathtt{k}, m_b, r)$ such that $c^* = \mathsf{Enc}(\mathtt{k}, m_b; r)$ (where c^* is the challenge ciphertext) is queried to Enc. Since R is PPT, the probability of R doing this is at most $\mathbf{poly}(N)/2^{\Omega(N)}$. 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

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

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.

For the final statement of the lemma, note that ϕ maps oblivious GSD reductions to oblivious Pebblers. □

The following lower bound on GSD now easily follows from Lemma 1 and Theorem 6; for stronger lower bounds for oblivious reductions we refer to the full version [33, Corollaries 2 to 4].

Corollary 1 (Lower bound for GSD on Trees, Straight-Line Reductions). *Let N be the number of users in the GSD game. Any* key-committing straight-line *reduction proving adaptive GSD-security restricted to* trees *based*

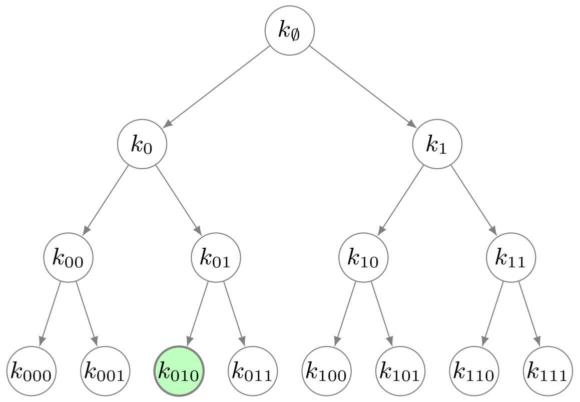

Fig. 2: Illustration of the GGM PRF. Every left child $k_{x\|0}$ of a node k_x is defined as the first half of $\mathsf{PRG}(k_x)$, the right child $k_{x\|1}$ as the second half. The thick node (shaded in green) corresponds to $\mathsf{F}_{GGM}(k_\emptyset, 010)$.

on the IND-CPA security of the underlying encryption scheme loses at least a factor

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

Even if the adversary is restricted to querying graphs with outdegree 2, the reduction loses at least a factor

$$\Lambda \geq N^{\log(N)/8-\log(\log(N))/4}.$$

7 Cryptographic Lower Bound II: Constrained PRF

In this section we use our combinatorial results for the Builder-Pebbler Game to prove that the constrained pseudorandom function (CPRF) [10, 11, 36] based on the GGM PRF [27] cannot be proven adaptively-secure based on the security of the underlying pseudorandom generator (PRG) using a *straight-line* reduction. Our lower bound almost matches the best-known upper bound by Fuchsbauer et al. [21].

7.1 Definition, Construction and Security Assumption

The following definitions are essentially taken from [31].

Definition 14 (GGM PRF). *Given a* $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$*, the PRF* $\mathsf{F}_{GGM} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda$ *is defined as*

$$\mathsf{F}_{GGM}(k,x) = k_x \textit{ where } k_\emptyset = k \textit{ and } \forall z \in \{0,1\}^* : k_{z\|0}\|k_{z\|1} = \mathsf{PRG}(k_z).$$

A graphical representation of the GGM construction is depicted in Figure 2.

Next, we give the definitions for CPRFs that are tailored to prefix-constrained PRFs.

Definition 15 (Prefix-constrained PRF). *For $n \in \mathbb{N}$, a function $\mathsf{F} : \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}$ is a* prefix-constrained PRF *if there are algorithms $\mathsf{F.Constrain} : \mathcal{K} \times \{0,1\}^{\leq n} \to \mathcal{K}_{pre}$ and $\mathsf{F.Eval} : \mathcal{K}_{pre} \times \{0,1\}^n \to \mathcal{Y}$ which for all $k \in \mathcal{K}$, $x \in \{0,1\}^{\leq n}$ and $k_x \leftarrow \mathsf{F.Constrain}(k,x)$ satisfy*

$$\mathsf{F.Eval}(k_x, x') = \begin{cases} \mathsf{F}(k, x') & \textit{if } x \textit{ is a prefix of } x' \\ \perp & \textit{otherwise.} \end{cases}$$

That is, $\mathsf{F.Constrain}(k,x)$ outputs a key k_x that allows evaluation of $\mathsf{F}(k,\cdot)$ on all inputs that have x as a prefix. We can derive a prefix-constrained PRF from the GGM construction by setting $\mathcal{K} = \{0,1\}^\lambda$, $\mathcal{Y} = \{0,1\}^\lambda$, and for a random $k \leftarrow \mathcal{K}$ and $x \in \{0,1\}^l$ with $l \leq n$ defining $\mathsf{F}_{GGM}.\mathsf{Constrain}(k,x) = (k_x^1, k_x^2) := (x, \mathsf{F}_{GGM}(k,x))$ and

$$\mathsf{F}_{GGM}.\mathsf{Eval}(k_x, x') := \begin{cases} \mathsf{F}_{GGM}(k_x^2, z) & \text{if } x' = x||z \text{ for some } z \in \{0,1\}^{n-l} \\ \perp & \text{otherwise.} \end{cases}$$

The security for prefix-constrained PRFs is argued using the following game.

Definition 16. *The game is played between a challenger G (which is either G_L or G_R) and an adversary A using F. The challenger G picks a random key $k \leftarrow \mathcal{K}$, and initialises a set $\mathcal{X} = \emptyset$. A can make at most $q = q(n)$ queries, which is either:*

- *Constrain queries, $(\texttt{constrain}, x)$: G returns $\mathsf{F.Constrain}(k,x)$, and adds x to $\mathcal{X}$.*
- *One challenge query $(\texttt{challenge}, x^*)$: Here the answer differs between G_L and G_R: G_L answers with $\mathsf{F.Eval}(k, x^*)$ (real output), whereas G_R answers with random $r \leftarrow \mathcal{Y}$ (fake, random output) – for the task to be non-trivial, no element in $\mathcal{X}$ must be a prefix of x^*. G adds x^* to $\mathcal{X}$.*

Definition 17. *A prefix-constrained PRF F is (s, ε, q)-adaptive-secure if G_L and G_R are (s, ε)-indistinguishable.*

7.2 Lower Bound for the GGM CPRF

To prove a lower bound for GGM, we use the combinatorial upper bound from Section 5.1 for non-oblivious Pebblers, restricted to the class of graphs with outdegree 2. The main challenge here is that – in contrast to our Builder from Section 5.1 – the constrain queries of an adversary in the security game for prefix-constrained PRFs correspond to paths in an exponentially large binary tree (see Figure 3). But it's not only that the adversary has to follow a certain query pattern, but more importantly for each query (which corresponds to a path of up to n edges) it only receives a single evaluation (and this evaluation allows A to efficiently compute any evaluations for the entire subtree below it). While A might be able to use its unrestricted computational power to distinguish whether the answer to its query lies in the image of the PRG (for an appropriately chosen PRG), it is impossible to extract a pebbling configuration on the entire path given just the single evaluation. This is why we follow a different approach and instead of choosing a PRG with sparse output range construct a PRG from two random permutations, which allows A to invert the function and compare whether two queries were computed from the same seed. Similar to the Builder strategy in Section 5.1, our adversary A makes bunches of queries forming complete binary subtrees, threaded along the challenge path. However, these queries are now paths of length n such that their *prefixes* cover the binary subtrees, respectively. Accordingly, we then map these bunches of queries to a pebbling strategy on the corresponding binary subtrees, instead of mapping single edges to a pebble or no pebble, as we did in previous applications. Fortunately, the combinatorial bound from Section 5.1 still holds for Builders revealing such bunches of queries at once.

Lemma 2 (Coupling Lemma for GGM CPRF). *Let $\mathcal{G}$ be the family of trees of depth D, size $N = \mathbf{poly}(D)$, indegree 1, outdegree 2 and a single source; i.e. $\mathcal{G}$ denotes the set of $\mathbf{poly}(D)$-sized subtrees of the binary tree of depth D which include the root, where edges are directed from the root to the leaves. Furthermore, let B and $X_{\mathcal{C}_D,k}$ for $k = \log(D)/2$ be the Builder and the cut from Theorem 6. Then there exists*

1. *an information-theoretically secure length-doubling PRG scheme* PRG
2. *a CPRF adversary* A *in* **PSPACE**

such that for any straight-line *reduction* R *that proves CPRF security of the GGM construction for input length $D+1$ based on the security of the underlying PRG scheme there exists a Pebbler* P *such that the advantage $1/\Lambda$ of* R *is at most the advantage of* P *against* B *(up to a negligible additive term $\mathbf{poly}(D)/2^{\Omega(D)}$).*

To prove this lemma, we will use the following construction of an information-theoretically secure PRG scheme.

Lemma 3. *Let $\pi_0, \pi_1 : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be two random permutations. Then $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ defined by $\mathsf{PRG}(x) := (\pi_0(x), \pi_1(x))$ is a $\mathbf{poly}(\lambda)/2^{\lambda/2}$-secure length-doubling PRG.*

Proof. Since random permutations are indistinguishable from random functions using only polynomially many queries, we may consider the PRG as a concatenation of two $\mathbf{poly}(\lambda)/2^{\lambda/2}$-secure PRFs by a hybrid argument. Again by hybrid argument, the concatenation of two secure PRFs yields a PRF from $\{0,1\}^\lambda$ to $\{0,1\}^{2\lambda}$. The lemma follows, since length extending PRFs are PRGs.

Having a construction of a PRG in place, we are now ready to prove Lemma 2.

Proof (Proof of Lemma 2). We pick the PRG from Lemma 3 for $\lambda = \Theta(D)$.

Analogously to the proof of Lemma 1 we define a map ϕ between the CPRF game and the Builder-Pebbler Game:

– For a constrain query by adversary A, $(\texttt{constrain}, x)$, we make a case distinction on the length l of x:

    - if $l = D + 1$, the Builder B extends the current tree in the natural way, ignoring k-sized blocks of trailing zeros in x and adding random nodes as needed. More formally, write $x = x^1||x^2||x^3 \in \{0,1\}^{l_1} \times \{0,1\}^{l_2} \times \{0\}^{l_3} \times \{0,1\}$ with $l_1, l_2, l_3 \geq 0$ and $k|l_3$, where x^1 is the longest prefix of x that has been queried so far. For each prefix x' of x with length between l_1+1 and $l_1 + l_2$, B chooses a uniformly random node (that is not associated to any prefix yet) and associates it to x'. Writing $x^2 = (x^2_1, x^2_2, \dots)$, it then queries the edges between the nodes associated with x^1 and $x^1||x^2_1$, between $x^1||x^2_1$ and $x^1||x^2_1||x^2_2$, etc.
    - if $l \leq D$, B ignores the query.
- For the challenge query $(\texttt{challenge}, x^*)$, proceed as for constrain queries to extend the tree. Choose the node associated to x^* as the challenge T.
- Pebbles are determined in the following way. Recall that the Builder from Theorem 6 always extends the tree in chunks of entire subtrees (and the queries comprising such a chunk can be sent at the same time). So we may restrict the definition of ϕ to preimages of such Builders. To determine which edges in such a subtree are pebbled, consider the responses y_i corresponding to the queries x_i in such a chunk. For each y_i invert π_0 repeatedly to obtain the seed associated to the i-th leaf in the subtree. Then for every node, bottom-up, if
    - the children are associated with seeds s_0, s_1, resp., check if $\pi_0^{-1}(s_0) = \pi_1^{-1}(s_1)$. If this is true, associate the node with this computed seed. Otherwise, consider both outgoing edges from this node as pebbled and set the seed of this node to $\perp$.
    - only the left (right) child is associated with a seed s, set the seed of this node to $\pi_0^{-1}(s)$ ($\pi_1^{-1}(s)$, resp.).
    - neither of the children is associated with a seed, set the seed of the current node to $\perp$.

  For the root of the subtree, which already has a seed s (or $\perp$) associated to it, check if s is consistent with its children; if not, update to $\perp$ and pebble both outgoing edges.

Let A be the preimage under ϕ of B from Theorem 6 as follows: A first queries CPRF evaluations for $\{0,1\}^{2k}||0^{D-2k+1}$ in reverse order (i.e. starting from $1^{2k}||0^{D-2k+1}$)[18] – this is in analogy to the first $2k$ rounds of B (see Figure 3). Then it proceeds in $[D/k-2]$ rounds, where in round $j \in [D/k-2]$ it first samples $x^*_j \in \{0,1\}^k$ and then makes 2^{2k} queries $x^*_1||\dots||x^*_j||\{0,1\}^{2k}||0^{D-(j+2)k+1}$ in reverse order, starting with $x^*_1||\dots||x^*_j||1^{2k}||0^{D-(j+2)k+1}$. Next, A samples a challenge $x^* = (x^*_1, \dots, x^*_D, 1)$ in $x^*_1||\dots||x^*_{D/k-2}||\{0,1\}^{2k}||1$ uniformly at random. Furthermore, it makes constrain queries for all prefixes $(x^*_1, \dots, x^*_{j-1}, \bar{x}_j)$

[18] This is for technical reasons: We defined the mapping ϕ to ignore k-blocks of trailing zeros in order to associate queries $x||0^{D-2k+1}$ to (non-disjoint) paths of length $2k$. To this aim ϕ prolongs the longest already existing subpath associated to some prefix x' of x. If A now starts querying the string 0^{D+1}, this query would simply be ignored. On the other hand, if there was a preceding query $0^{2k-1}||1||0^{D-2k+1}$, then the query 0^{D+1} is mapped to an edge extending the path associated with the prefix 0^{2k-1}.

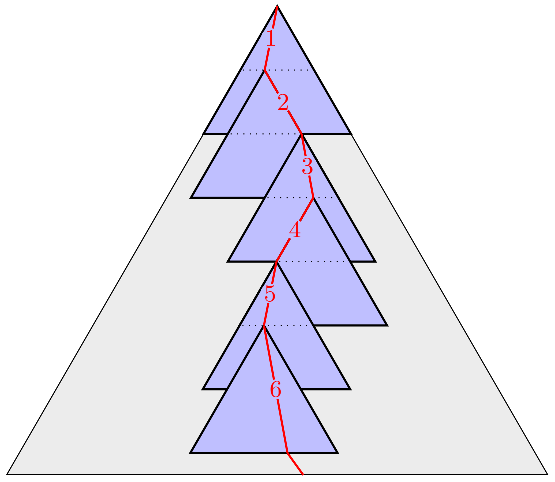

Fig. 3: A schematic diagram showing the adversarial query strategy for GGM CPRF in Lemma 2. The outer (gray) triangle represents the perfect binary tree of depth $D = 7k + 1$ representing the GGM PRF. The internal (blue) triangles represent perfect binary trees of depth $2k$ with the j-th triangle representing the 2^{2k} queries $x_1^*||\ldots||x_j^*||\{0,1\}^{2k}||0^{D-(j+2)k+1}$. The challenge x^* is highlighted (in red) with the label j indicating the string x_j^*.

for $j \in [D]$. If the answers to the prefixes are not consistent with the previous CPRF queries, then A aborts and outputs 0. Otherwise, A uses its unrestricted computational power to compute the mapping ϕ from the reduction's answers to its queries to a pebbling configuration on the subtree. Note that due to the previous check, there must not be any pebbles on edges rooted at nodes outside the challenge path. A now considers the pebbling configuration induced on the challenge path. If this pebbling configuration lies in the cut defined by $X_{\mathcal{C}_D,k}$, the adversary A outputs 0, otherwise 1.

Clearly, A wins the CPRF game with probability 1. Now, let R be an arbitrary straight-line reduction. First, note that the probability that R queries PRG on the challenge seed is negligibly small ($\mathbf{poly}(D)/2^{\Omega(D)}$). Assuming this does not happen, R can only gain a bigger advantage if it embeds its PRG challenge when interacting with A and manages to hit a pebbling configuration in the cut, i.e. such that depending on the challenge being real or random the pebbling configuration which A extracts lies either in the cut set or not. Note that choosing a value in the tree at random instead of applying PRG to the correct output is equivalent (w.r.t. A's behavior) to responding to the respective queries inconsistently and will thus yield a pebble with overwhelming probability. Furthermore, the consistency check after the constrain queries ensures that R may only place pebbles on edges rooted in the challenge graph and can only embed its challenge in the challenge graph. Similar to the proof in Lemma 1, one can see that R

maps (under ϕ) to a Pebbler in the Builder-Pebbler Game which has at least the same advantage of achieving such a configuration.

□

Using the above lemma, the following corollary now easily follows from Theorem 6.

Corollary 2 (Lower Bound for GGM). *Let n be the input length of the GGM CPRF scheme. Then any straight-line reduction proving cPRF security of the GGM construction based on the security of the underlying PRG scheme loses at least a factor $\Lambda \geq n^{(\log(n)-\log\log(n))/2}$.*

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. Cryptology ePrint Archive, Report 2019/1489, 2019. https://eprint.iacr.org/2019/1489.

[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, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 248–277. Springer, Heidelberg, Aug. 2020.

[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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, Dec. 2013.

[11] E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, Mar. 2014.

[12] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996.

[13] R. Canetti, J. A. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *IEEE INFOCOM'99*, pages 708–716, New York, NY, USA, Mar. 21–25, 1999.

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

[15] 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.

[16] C. Dwork, M. Naor, O. Reingold, and L. J. Stockmeyer. Magic functions. In *40th FOCS*, pages 523–534. IEEE Computer Society Press, Oct. 1999.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] D. Hefetz, M. Krivelevich, M. Stojakovic, and T. Szabó. *Positional Games.* Birkhäuser Basel, 2014.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] C. Kamath, K. Klein, K. Pietrzak, and M. Walter. The cost of adaptivity in security games on graphs. Cryptology ePrint Archive, Report 2021/059, 2021. https://eprint.iacr.org/2021/059.

[34] C. Kamath, K. Klein, K. Pietrzak, and D. Wichs. Limits on the adaptive security of yao's garbling. In T. Malkin and C. Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 486–515. Springer, 2021.

[35] 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.

[36] A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, Nov. 2013.

[37] 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.

[38] 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.

[39] R. Královič. *Time and Space Complexity of Reversible Pebbling*, pages 292–303. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[40] 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.

[41] 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.

[42] J. Nordström. *New Wine into Old Wineskins: A Survey of SomePebbling Classics with Supplemental Results.* 2015.

[43] 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.

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

[45] 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.

[46] 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.

[47] 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.

[48] S. Rudich. *Limits on the Provable Consequences of One-way Functions.* PhD thesis, EECS Department, University of California, Berkeley, Dec 1988.

[49] J. E. Savage. *Models of computation - exploring the power of computing.* Addison-Wesley, 1998.

[50] 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.

[51] D. M. Wallner, E. J. Harder, and R. C. Agee. Key management for multicast: Issues and architectures. Internet Draft, Sept. 1998. http://www.ietf.org/ID.html.

[52] C. K. Wong, M. G. Gouda, and S. S. Lam. Secure group communications using key graphs. *IEEE/ACM Trans. Netw.*, 8(1):16–30, 2000.