Be Adaptive, Avoid Overcommitting

Zahra Jafargholi* Chethan Kamath† Karen Klein‡ Ilan Komargodski§
Krzysztof Pietrzak† Daniel Wichs ¶

June 2, 2017

Abstract

For many cryptographic primitives, it is relatively easy to achieve *selective security* (where the adversary commits a-priori to some of the choices to be made later in the attack) but appears difficult to achieve the more natural notion of *adaptive security* (where the adversary can make all choices on the go as the attack progresses). A series of several recent works shows how to cleverly achieve adaptive security in several such scenarios including *generalized selective decryption* (Panjwani, TCC '07 and Fuchsbauer et al., CRYPTO '15), *constrained PRFs* (Fuchsbauer et al., ASIACRYPT '14), and *Yao garbled circuits* (Jafargholi and Wichs, TCC '16b). Although the above works expressed vague intuition that they share a common technique, the connection was never made precise. In this work we present a new framework that connects all of these works and allows us to present them in a unified and simplified fashion. Moreover, we use the framework to derive a new result for adaptively secure *secret sharing over access structures defined via monotone circuits*. We envision that further applications will follow in the future.

Underlying our framework is the following simple idea. It is well known that selective security, where the adversary commits to n-bits of information about his future choices, automatically implies adaptive security at the cost of amplifying the adversary's advantage by a factor of up to 2^n. However, in some cases the proof of selective security proceeds via a sequence of hybrids, where each pair of adjacent hybrids locally only requires some smaller partial information consisting of $m \ll n$ bits. The partial information needed might be completely different between different pairs of hybrids, and if we look across all the hybrids we might rely on the entire n-bit commitment. Nevertheless, the above is sufficient to prove adaptive security, at the cost of amplifying the adversary's advantage by a factor of only $2^m \ll 2^n$.

In all of our examples using the above framework, the different hybrids are captured by some sort of a *graph pebbling game* and the amount of information that the adversary needs to commit to in each pair of hybrids is bounded by the maximum number of pebbles in play at any point in time. Therefore, coming up with better strategies for proving adaptive security translates to various pebbling strategies for different types of graphs.

*Aarhus University, Denmark. Email: z.jafargholi@gmail.com.

†IST Austria, Am Campus 1, 3400 Klosterneuburg, Austria. Email: {ckamath,pietrzak}@ist.ac.at. Supported by the European Research Council, ERC consolidator grant (682815 - TOCNeT).

‡IST Austria, Am Campus 1, 3400 Klosterneuburg, Austria. Email: karen.klein@ist.ac.at.

§Department of Computer Science and Applied Mathematics, Weizmann Institute of Science Israel, Rehovot 76100, Israel. Email: ilan.komargodski@weizmann.ac.il. Supported in part by grants from the Israel Science Foundation and by a Levzion Fellowship.

¶Northeastern University. Department of Computer Science. wichs@ccs.neu.edu. Research supported by NSF grants CNS-1314722, CNS-1413964.

1 Introduction

Many security definitions come in two flavors: a stronger "adaptive" flavor, where the adversary can arbitrarily make various choices during the course of the attack, and a weaker "selective" flavor where the adversary must commit to some or all of his choices a-priori. For example, in the context of *identity-based encryption*, selective security requires the adversary to decide on the identity of the attacked party at the very beginning of the game whereas adaptive security allows the attacker to first see the master public key and some secret keys before making this choice. Often, it appears to be much easier to achieve selective security than it is to achieve adaptive security.

A series of recent works achieves adaptive security in several such scenarios where we previously only knew how to achieve selective security: *generalized selective decryption (GSD)* [Pan07, FJP15], *constrained PRFs* [FKPR14], and *garbled circuits* [JW16]. Although some of these works suggest a vague intuition that there is a general technique at play, there was no attempt to make this precise and to crystallize what the technique is or how these results are connected. In this work we present a new framework that connects all of these works and allows us to present them in a unified and simplified fashion. Moreover, we use the framework to derive a new result for adaptively secure secret sharing over access structures defined via monotone circuits.

At a high level, our framework carefully combines two basic tools commonly used throughout cryptography: *random guessing* (of the adaptive choices to be made by the adversary)[1] and *the hybrid argument*. Firstly, "random guessing" gives us a generic way to qualitatively upgrade selective security to adaptive security at a quantitative cost in the amount of security. In particular, assume we can prove the security of a selective game where the adversary commits to n-bits of information about his future choices. Then, we can also prove adaptive security by guessing this commitment and taking a factor of 2^n loss in the security advantage. However, this quantitative loss is often too high and hence we usually wish to avoid it or at least lower it. Secondly, the hybrid argument allows us to prove the indistinguishability of two games G_L and G_R by defining a sequence of hybrid games $\mathsf{G}_\mathsf{L} \equiv \mathsf{H}_0, \mathsf{H}_1, \ldots, \mathsf{H}_\ell \equiv \mathsf{G}_\mathsf{R}$ and showing that each pair of neighboring hybrids H_i and H_{i+1} are indistinguishable.

Our Framework. Our framework starts with two adaptive games G_L and G_R that we wish to show indistinguishable but we don't initially have any direct way of doing so. Let H_L and H_R be selective versions of the two games respectively, where the adversary initially has to commit to some information $w \in \{0,1\}^n$ about his future choices. Furthermore, assume there is some sequence of selective hybrids $\mathsf{H}_\mathsf{L} = \mathsf{H}_0, \mathsf{H}_1, \ldots, \mathsf{H}_\ell \equiv \mathsf{H}_\mathsf{R}$ such that we can show that H_i and H_{i+1} are indistinguishable. A naïve combination of the hybrid argument and random guessing shows that G_L and G_R are indistinguishable at a factor of $2^n \cdot \ell$ loss in security, but we want to do better.

Recall that the hybrids H_i are selective and require the adversary to commit to w. However, it might be the case that for each i we can prove that H_i and H_{i+1} would be indistinguishable even if the adversary didn't have to commit to all of w but only some partial-information $h_i(w) \in \{0,1\}^m$ for $m \ll n$ (formalizing this condition precisely requires great care and is the major source of subtlety in our framework). Notice that the partial information that we need to know about w may

[1] In many previous works – including [FJP15, FKPR14, JW16], and by the authors of this paper – this random guessing was referred to as "complexity leveraging", but this seems to be an abuse of the term. Instead, complexity leveraging [CGGM00] refers to the use of two different schemes, S_1, S_2, where the two schemes are chosen with different values of the security parameter, k_1 and k_2, where $k_1 < k_2$, and such that an adversary against S_2 (or perhaps even the honest user of S_2) can break the security of S_1.

be completely different for different pairs of hybrids, and if we look across all hybrids then we may need to know all of w. Nevertheless, we prove that this suffices to show that the adaptive games G_{L} and G_{R} are indistinguishable with only a $2^m \cdot \ell \ll 2^n \cdot \ell$ loss of security.

Applications of Our Framework. We show how to understand all of the prior works mentioned above as applications of our framework. In many cases, this vastly simplifies prior works. We also use the framework to derive a new result, proving the adaptive security of Yao's secret sharing scheme for access structures defined via monotone circuits.

In all of the examples, we get a series of selective hybrids $\mathsf{H}_1, \ldots, \mathsf{H}_\ell$ that correspond to *pebbling configurations* in some graph pebbling game. The amount of information needed to show that neighboring hybrids H_i and H_{i+1} are indistinguishable only depends on the configuration of the pebbles in the i'th step of the game. Therefore, using our framework, we translate the problem of coming up with adaptive security proofs to the problem of coming up with pebbling strategies that only require a succinct representation of each pebbling configuration.

We now proceed to give a high level overview of each of our results applying our general framework to specific problems, and refer to the main body for technical details.

1.1 Adaptive Secret Sharing for Monotone Circuits

Secret sharing schemes, introduced by Blakley [Bla79] and Shamir [Sha79], are methods that enable a dealer, that has a secret piece of information, to distribute this secret among n parties such that a "qualified" subset of parties has enough information to reconstruct the secret while any "unqualified" subset of parties learns nothing about the secret. The monotone collection of "qualified" subsets is known as an *access structure*. Any access structure admits a secret sharing scheme but the share size could be exponential in n [ISN87]. We are interested in efficient schemes in which the share size is polynomial (in n and possibly in a security parameter).

Many of the classical schemes for secret sharing are *perfectly* (information theoretically) secure. The largest class of access structures that admit such a (perfect and efficient) scheme was obtained by Karchmer and Wigderson [KW93] for the class of all functions that can be computed by monotone span programs. This result generalized a previous work of Benaloh and Leichter [BL90] (which, in turn, improved a result of Ito, Saito and Nishizeki [ISN87]) that showed the same result but for a smaller class of access structures: those functions that can be computed by monotone Boolean formulas. Under cryptographic hardness assumptions, efficient schemes for more general access structures are known (but security is only for bounded adversaries). In particular, in an unpublished work (mentioned in [Bei11], see also Vinod et al. [VNS+03]), Yao showed how to realize schemes for access structures that are described by monotone *circuits*. This construction could be used for access structures which are known to be computed by monotone circuits but are not known to be computed by monotone span programs, e.g., directed connectivity [KW88, RPRC16].[2] Komargodski, Naor, and Yogev [KNY17] showed how to realize the class of access structures described by monotone functions in NP[3] under the assumption that witness encryption for NP [GGSW13]

[2] In the access structure for directed connectivity, the parties correspond to an edge in the complete *directed* graph and the "qualified" subsets are those edges that connect two distinguished nodes s and t.

[3] For access structures in NP, a qualified set of parties needs to know an NP witness that they are qualified.

and one-way functions exist.[4][5]

Selective vs. adaptive security. All of the schemes described above guarantee security against static adversaries, where the adversary chooses a subset of parties it controls before it sees any of the shares. A more natural security guarantee would be to require that even an adversary that chooses its set of parties in an *adaptive* manner (i.e., based on the shares it has seen so far) is unable to learn the secret (or any partial information about it).

It is known that the schemes that satisfy perfect security (including the works [ISN87, BL90, KW93] mentioned above) actually satisfy this stronger notion of adaptive security. However, the situation for the schemes that are based on cryptographic assumptions (including Yao's scheme and the scheme of [KNY17]) is much less clear. Using random guessing (see Lemma 1) it can be shown that these schemes are adaptively secure, but this reduction loses an exponential (in the number of parties) factor in the security of the scheme. Additionally, as noted in [KNY17], their scheme can be shown to be adaptively secure if the witness encryption scheme is *extractable*.[6] The latter is a somewhat controversial assumption that we prefer to avoid.

Our results. We analyze the adaptive security of Yao's scheme under our framework and show that in some cases the security loss is much smaller than 2^n. Roughly, we show that if the access structure can be described by a monotone circuit of depth d and s gates (with unbounded fan-in and fan-out) the security loss is proportional to $s^{O(d)}$. Thus, for shallow circuits our analysis shows that an exponential loss is avoidable.

To exemplify the usefulness of the result, consider, for instance, the directed st-connectivity access structure mentioned in Footnote 2. It is known that it can be computed by a monotone circuit of size $O(n^3 \log n)$ and depth $O(\log^2 n)$, but its monotone formula and span-program complexity is $2^{\Omega(\log^2 n)}$ [KW88, RPRC16]. Thus, no perfectly secure scheme is known, and our proof shows that Yao's scheme for this access structure is secure based on the assumption that quasi-polynomially-secure one-way functions exist.

Yao's scheme. In this scheme, an access structure is described by a monotone circuit. The sharing procedure first labels the output wire of the circuit with the shared secret and then proceeds to assign labels to all wires of the circuit; in the end the label on each input wire is included in the share of the corresponding party. The procedure for assigning labels is recursive and in each step it labels the input wires of a gate g assuming its output wires are already labeled (recall that we assume unbounded fan-in and fan-out so there are many input and output wires). To do so, we first sample a fresh encryption key s for a symmetric-key encryption scheme. If the gate is an AND gate, then we label each input wire with a random string conditioned on their XOR being s, and if the gate is an OR gate, then we label each input wire with s. In either case, we encrypt the labels of the output wires under s and include these ciphertexts associated with the gate g as part of ever party's share. The reconstruction of the scheme works by reversing the above procedure from the

[4] Witness encryption for a language $L \in \mathsf{NP}$ allows to encrypt a message relative to a statement $x \in L$ such that anyone holding a witness to the statement can decrypt the message, but if $x \notin L$, then the message is computationally hidden.

[5] One can relax the additional assumption of one-way functions to an average-case hardness assumption in NP [KMN+14].

[6] This is a knowledge assumption that says that if an adversary can decrypt a witness encryption ciphertext, then it must *know* a witness which can be extracted from it.

leaves to the root. This scheme is indeed efficient for access structures that have polynomial-size monotone circuits.

Security proof. Our goal is to show that as long as an adversary controls an unqualified set, he cannot learn anything about the secret. We start by outlining the selective security proof (following the argument of [VNS+03]), where the adversary first commits to the "corrupted" set. The proof is via a series of hybrids in which we slowly replace the ciphertexts associated with various gates g with bogus ciphertexts. Once we do this for the output gate, the shares become independent of the secret which proves security. The gates for which we can replace the ciphertexts with bogus ones are the gates for which the adversary cannot compute the corresponding encryption key. Since the adversary controls an unqualified set, a sequence which eventually results with replacing the encryption of the root gate must exist. Since in every hybrid we "handle" one gate and never consider it again, the number of hybrids is at most the number of gates in the circuit.

The problem with lifting this proof to the adaptive case is that it seems inherent to know the corrupted set of parties in order to know for which gates g to switch the ciphertexts from real to bogus (and in what order). However, in the adaptive game this set is not known during the sharing procedure. A naïve use of random guessing would result in an exponential security loss 2^n, where n is the number of parties.

To overcome this we associate each intermediate hybrid H_i with a *pebbling configuration* in which each gate in the circuit is either pebbled (ciphertexts are bogus) or unpebbled (ciphertexts are real). The pebbling rules are:

1. Can place or remove a pebble on any AND gate for which (at least) one input wire is either *not* corrupted or comes out of a gate with a pebble on it.

2. Can place or remove a pebble on any OR gate for which all of the incoming wires are either non-corrupted input wires or come out of gates all of which have pebbles on them.

The initial hybrid corresponds to the case in which all gates are unpebbled and the final hybrid corresponds to the case in which all gates are unpebbled except the root gate which has a pebble. Now, any pebbling strategy that takes us from the initial configuration to the final one, corresponds to a sequence of selective hybrids H_i. Furthermore, to prove indistinguishability of neighboring hybrids $\mathsf{H}_i, \mathsf{H}_{i+1}$ we don't need the adversary to commit to the entire set of corrupted parties ahead of time but it suffices if the adversary only commits to the pebble configuration in steps i and $i+1$. Therefore, if the pebbling strategy has the property that each configuration requires few bits to describe, then we would be able to use our framework. We show that for every corrupted set and any monotone circuit of depth d and s gates, there exists such a pebbling strategy, where the number of moves is roughly $2^{O(d)}$ and each configuration has a very succinct representation: roughly $d \cdot \log s$ bits. Plugging this into our framework, we get a proof of adaptive security with security loss proportional to $s^{O(d)}$. We refer to Section 4 for the precise details.

1.2 Generalized Selective Decryption

Generalized Selective Decryption (GSD), introduced by Panjwani [Pan07], is a game that captures the difficulty of proving adaptive security of certain protocols, most notably the Logical Key Hierarchy (LKH) multicast encryption protocol. On a high level, it deals with scenario where we have many secret keys k_i and various ciphertexts encrypting one key under another (but no cycles).

We will discuss this problem in depth in Section A, here giving a high level overview on how our framework applies to this problem.

Let $(\mathsf{Enc}, \mathsf{Dec})$ be a CPA-secure symmetric encryption scheme with (probabilistic) $\mathsf{Enc}\colon \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ and $\mathsf{Dec}\colon \mathcal{K} \times \mathcal{C} \to \mathcal{M}$. We assume $\mathcal{K} \subseteq \mathcal{M}$, i.e., we can encrypt keys. In the game, the challenger — either G_L or G_R — picks $n+1$ random keys $k_0, \ldots, k_n \in \mathcal{K}$, and the adversary A is then allowed to make three types of queries:[7]

- Encryption query: on input $(\texttt{encrypt}, i, j)$ receives $\mathsf{Enc}(k_i, k_j)$.
- Corruption queries: on input $(\texttt{corrupt}, i)$ receives k_i.
- Challenge query, only one is allowed: on input $(\texttt{challenge}, i)$ receives k_i in the real game G_L, and a random value in the random game G_R.

We think of this game as generating a directed graph, with vertex set $\mathcal{V} = \{0, \ldots, n\}$, where every $(\texttt{encrypt}, i, j)$ query adds a directed edge (i, j), and we say a vertex v_i is corrupted if a query $(\texttt{corrupt}, i)$ was made, or v_i can be reached from a corrupted vertex. The goal of the adversary is to distinguish the games G_L or G_R, with the restriction that the constructed graph has no cycles, and the challenge vertex is a sink. To prove security, i.e., reduce the indistinguishability of G_L or G_R to the security of Enc, we can consider a selectivized version of this game where A must commit to the graph as described above (which uses $< n^2$ bits). The security of this selectivized game can then be reduced to the security of Enc by a series of $< n^2$ hybrids, where a distinguisher for any two consecutive hybrids can be used to break the security of Enc with the same advantage. Using random guessing followed by a hybrid argument we conclude that if Enc is δ-secure, the GSD game is $\delta \cdot n^2 \cdot 2^{n^2}$-secure. Thus, we lose an exponential in n^2 factor in the reduction.

Fortunately, if we look at the actual protocols that GSD is supposed to capture, it turns out that the graphs that A can generate are not totally arbitrary. Two interesting cases are given by GSD restricted to graphs of bounded depth, and to trees. For these cases better reductions exist. Panjwani [Pan07] shows that if the adversary is restricted to play the game such that the resulting graph is of depth at most d, a reduction losing a factor $(2n)^d$ exists. Moreover, Fuchsbauer et al. [FJP15] give a reduction losing a factor $n^{3 \log n}$ when the underlying graph is a tree. In Section A we prove these results in our framework. Our proofs are much simpler than the original ones, especially than the proof of [Pan07] which is very long and technical. This is thanks to our modular approach, where our general framework takes care of delicate probabilistic arguments, and basically just leaves us with the task of designing pebbling strategies, where each pebbling configuration has a succinct description, for various graphs, which is a clean combinatorial problem. The generic connection between adaptive security proofs of the GSD problem and graph pebbling is entirely new to this work.

GSD on a Path. Let us sketch the proof idea for the [FJP15] result, but for an even more restricted case where the graph is a path visiting every node exactly once. In other words there is a permutation σ over $\{0, \ldots, n\}$ and the adversary's queries are of the form $(\texttt{encrypt}, \sigma(i-1), \sigma(i))$ and $(\texttt{challenge}, \sigma(n))$. We first consider the selective game where A must commit to this permutation σ ahead of time. Let $\mathsf{H}_\mathsf{L}, \mathsf{H}_\mathsf{R}$ be the selectivized versions of G_L, G_R respectively.

[7] In the actual game the adversary can also make standard CPA encryption queries $\mathsf{Enc}(k_i, m)$ for chosen m, i. As this doesn't meaningfully change the security proof we ignore this here.

To prove selective security, we can define a sequence of hybrid games $\mathsf{H}_\mathsf{L} = \mathsf{H}_0, \ldots, \mathsf{H}_\ell = \mathsf{H}_\mathsf{R}$. Each hybrid is defined by a path, $0 \to 1 \to \ldots \to n$, with a subset of the edges holding a black pebble. In the hybrid games, a pebble on $(i, i+1)$ means that instead of answering the query $(\texttt{encrypt}, \sigma(i), \sigma(i+1))$ with the "real" answer $\mathsf{Enc}(k_{\sigma(i)}, k_{\sigma(i+1)})$, we answer it with a "fake" answer $\mathsf{Enc}(k_{\sigma(i)}, r)$ for a random r. The goal is to move from a hybrid with no pebbles (this corresponds to H_L) to one with a single black pebble on the "sink" edge $(n-1, n)$ (this corresponds to H_R). We can prove that neighboring hybrids are indistinguishable via a reduction from CPA security as long as the pebbling configurations are only modified via the following legal moves:

1. We can put/remove a pebble on the source edge $(0, 1)$ at any time.
2. We can put/remove a pebble on an edge $(i, i+1)$ if the preceding edge $(i-1, i)$ has a pebble.

This is because adding/removing a pebble $(i, i+1)$ means changing what we encrypt under key $k_{\sigma(i)}$ and therefore we need to make sure that either the edge is a source edge or there is already a pebble on the preceding edge to ensure that the key $k_{\sigma(i)}$ is never being encrypted under some other key.

The simplest "basic pebbling strategy" consists of $2n$ moves where we add pebbles on the path $0 \to 1 \to \ldots \to n$, one by one starting on the left and then remove one by one starting on the right, keeping only the pebble on the sink edge $(n-1, n)$. This is illustrated in Figure 1.(a) for $n = 8$. The strategy uses n pebbles. However, there are other pebbling strategies that allow us to trade off more moves for fewer pebbles. For example there is a "recursive strategy" (recursively pebble the middle vertex, then recursively pebble the right-most vertex, then recursively remove the pebble from the middle vertex) that uses at most $\log n + 1$ pebbles (instead of n), but requires $3^{\log n} + 1$ moves (instead of just $2n$). This is illustrated in Figure 1.(b).

As we described, each pebbling strategy with ℓ moves gives us a sequence of hybrids $\mathsf{H}_\mathsf{L} = \mathsf{H}_0, \ldots, \mathsf{H}_\ell = \mathsf{H}_\mathsf{R}$ that allows us to prove selective security. Furthermore, we can prove relatively easily that neighboring hybrids $\mathsf{H}_j, \mathsf{H}_{j+1}$ are indistinguishable even if the adversary doesn't commit to the entire permutation σ but only to the value $\sigma(i)$ of vertices i where either H_j or H_{j+1} has a pebble on the edge $(i-1, i)$. Using our framework, we therefore get a proof of adaptive security where the security loss is $\ell \cdot n^p$ where p is the maximum number of pebbles used and ℓ is the number of pebbling moves. In particular, if we use the recursive pebbling strategy described above we only suffer a quasipolynomial security loss $3^{\log n} \cdot n^{\log n + 1}$, as compared with $2n \cdot (n+1)!$ for naïve random guessing where the adversary commits to the entire permutation σ.

GSD on Low Depth and Other Families of Graphs. The proof outline for GSD on paths is just a very special case of our general result for GSD for various classes of graphs, which we discuss in §A. If we consider a class of graphs which can be pebbled using ℓ pebbling configurations, each containing at most q pebbles, we get a reduction showing that GSD for this class is $\delta \cdot \ell \cdot 2^q$ secure, assuming the underlying Enc scheme is δ-secure.

Unfortunately, this approach will not gain us much for graphs with high in-degree: we can only put a pebble on an edge (i, j) if all the edges $(*, i)$ going into node i are pebbled. So if we consider graphs which can have large in-degree d, any pebbling strategy must at some point have pebbled all the parents of i, and thus we'll lose at least a factor 2^d in the reduction. But remember that to apply our Theorem 2, we just need to be able to "compress" the information required to simulate the hybrids. So even if the hybrids correspond to configurations with many pebbles, that is fine as

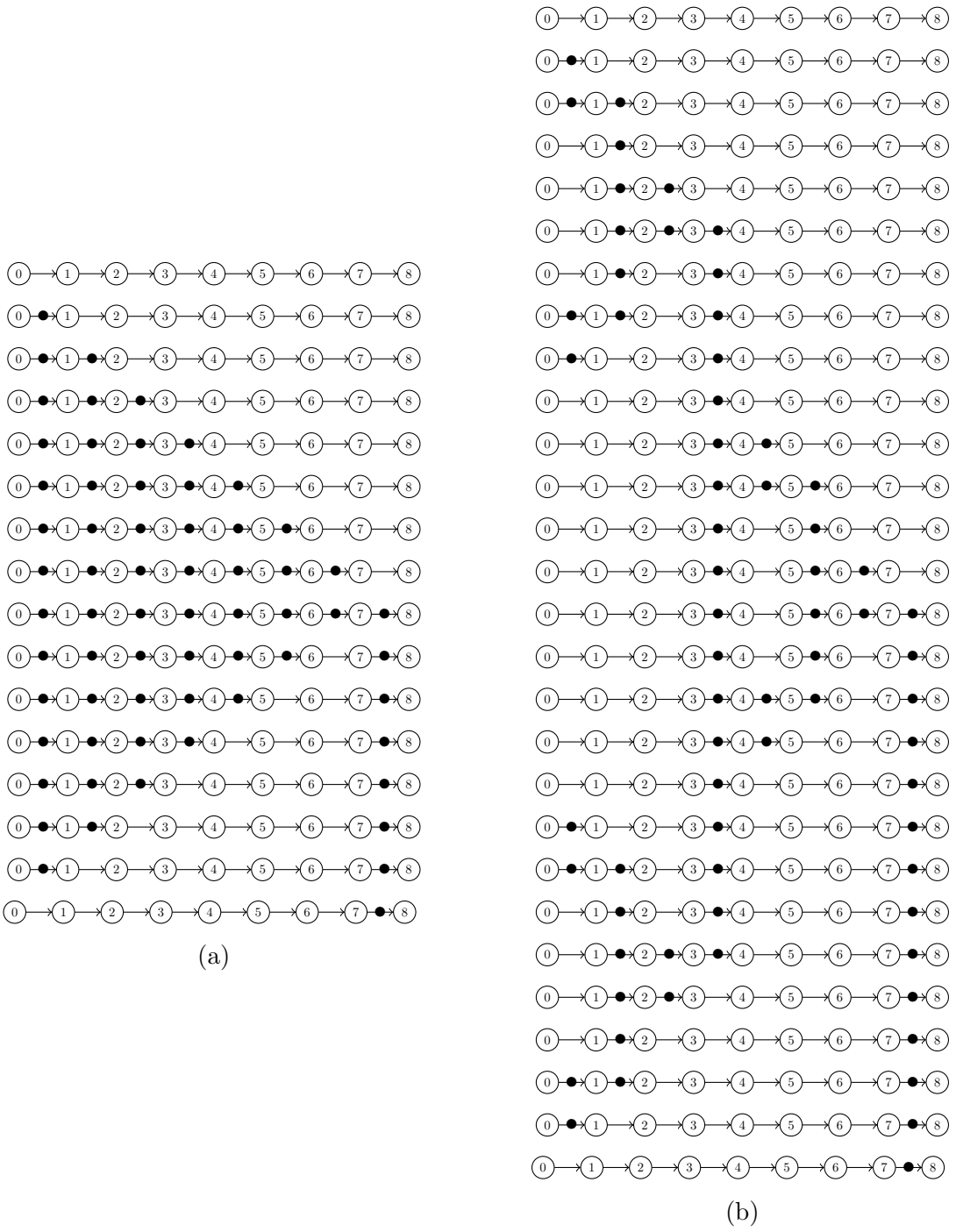

Figure 1: "Classical" hybrid argument vs. improved hybrid argument. In both diagrams, the edges that carry a pebble are faked. (a). Illustration of the classical hybrids $\mathsf{H}_0, \ldots, \mathsf{H}_{15}$ for GSD on a path graph with $n = 8$ edges: the number of hybrids is $2n = 16$, and the number of fake edges is at most n. (b.) A sequence of hybrids $\tilde{\mathsf{H}}_0, \ldots, \tilde{\mathsf{H}}_{27}$ that use fewer fake edges: even though the number of hybrids is $3^{\log n} + 1 = 28$, the number of fake edges is at most $\log n + 1 = 4$. The argument on the right is identical to the one using nested hybrids in [FJP15], which implicitly uses the edge-pebbling generated by Algorithm 4

long as we can generate a short hint which will allow to emulate it (we use the same idea in the proof of adaptive security of the secret sharing scheme for monotone circuits with large fan-in).

Consider the selective GSD game, where the adversary commits to all of its queries, we can think of this as a DAG, where each edge comes with an index indicating in which query this node was added. Assume the adversary is restricted to choose DAGs of depth l (but no bound on the in-degree). One can show that there exists a pebbling sequence (of length $(2n)^l$), such that in any pebbling configuration, all pebbles lie on a path from a sink to a root (which is of length at most l), and on edges going into this path. Moreover, we can ensure that in any configuration the following holds: if for a node j on this path, there is a pebble on edge (i, j) with index t, then all edges of the form $(*, j)$ with index $< t$ must also have a pebble.

To describe such a configuration, we will output the $\leq l$ nodes on the path, specify for every edge on this path if it is pebbled, and for any node j on the path, the number of edges going into j that have a pebble (note that there are at most $2^l n^{2l}$ choices for this hint). The hint is sufficient to emulate a hybrid, as for any query $(\texttt{encrypt}, i, j)$ the adversary makes, we will know if the corresponding edge has a pebble or not. This is clear if the edge (i, j) is on the path, as we know this path in full. But also for the other edges that can hold a pebble, where j is on the path but i is not. The reason is that we just have to count which query of the form $(*, j)$ this is, as we got a number c telling us that the first c such edges will have a pebble.

Applying Theorem 2, we recover Panjwani's result [Pan07] showing that if the GSD game restricted to graphs of depth l only loses a factor $n^{O(l)}$ in the reduction.

1.3 Yao's Garbled Circuits

Garbled circuits, introduced by Yao in (oral presentations of) [Yao82, Yao86], can be used to garble a circuit C and an input x in a way that reveals $C(x)$ but hides everything else. More precisely, a garbling scheme has three procedures; one to garble the circuit C and produce a garbled circuit $\widetilde{C}$, one to garble the input x and produce a garbled input $\widetilde{x}$, and one that evaluates the garbled circuit $\widetilde{C}$ on the garbled input $\widetilde{x}$ to get $C(x)$. Furthermore, to prove security, there must be a simulator that only gets the output of the computation $C(x)$ and can simulate the garbled circuit $\widetilde{C}$ and input $\widetilde{x}$, such that no PPT adversary can distinguish them from the real garbling.

Adaptive vs. Selective Security. In the adaptive setting, the adversary A first chooses the circuit C and gets back the garbled circuit $\widetilde{C}$, then chooses the input x, and gets back garbled input $\widetilde{x}$. The adversary's goal is to decide whether he was interacting with the real garbling scheme or the simulator. In the selective setting, the adversary has to choose the circuit C as well as the input x at the very beginning and only then gets back $\widetilde{C}, \widetilde{x}$.

Prior Work. The work of Bellare, Hoang and Rogaway [BHR12] raised the question of whether Yao's construction or indeed any construction of garbled circuits achieves adaptive security. The work of Hemenway et al. [HJO$^+$16] gave the first construction of non-trivial adaptively secure garbled circuits based on one-way functions, by modifying Yao's construction with an added layer of encryption having some special properties. Most recently, the work of Jafargholi and Wichs [JW16] gives the first analysis of adaptive security for Yao's unmodified garbled circuit construction which significantly improves on the parameters of trivial random guessing. See [JW16] for a more comprehensive introduction and broader background on garbled circuits and adaptive security.

Here, we present the work of [JW16] as a special case of our general framework. Indeed, the work of [JW16] already implicitly follows our general framework fairly closely and therefore we only give a high level overview of how it fits into it.

Selective Hybrids. We start by outlining the selective security proof for Yao's garbled circuits, following the presentation of [HJO+16, JW16] which is in turn based on the proof of Lindell and Pinkas [LP09]. Essentially the proof proceeds via series of hybrids which modify one garbled gate at a time from the Real distribution to a Simulated one. However, this cannot be done directly in one step and instead requires going through an intermediate distribution called InputDep (we explain the name later). There are important restrictions on the order in which these steps can be taken:

1. We can switch a gate from Real to InputDep (and vice versa) if it is at the input level or if its predecessor gates are already InputDep.
2. We can switch a gate from InputDep to Simulated (and vice versa) if it is at the output level or if its successor gates are already Simulated.

The simplest strategy to switch all gates from Real to Simulated is to start with the input level and go up one level at a time switching all gates to InputDep. Then start with the output level and go down one level at a time switching all gates to Simulated. This corresponds to the basic proof of selective security of Yao garbled circuits.

However, the above is not the only possibility. In particular, any strategy for switching all gates from Real to Simulated following rules (1) and (2) corresponds to a sequence of hybrid games for proving selective security. We can identify the above with a *pebbling game* where one can place pebbles on the gates of the circuit. The Real distribution corresponds to not having a pebble and there are two types of pebbles corresponding to the InputDep and Simulated distributions. The goal is to start with no pebbles and finish by placing a Simulated pebble on every gate in the circuit while only performing legal moves according to rules (1) and (2) above. Every pebbling strategy gives rise to a sequence of hybrid games $\mathsf{H}_0, \mathsf{H}_1, \ldots, \mathsf{H}_\ell$ for proving selective security, where the number of hybrids ℓ corresponds to the number of moves and each hybrid H_i is defined by the configuration of pebbles after i moves.

From Selective to Adaptive. The problem with translating selective security proofs into the adaptive setting lies with the InputDep distribution of a gate. This distribution depends on the input x (hence the name) and, in the adaptive setting, the input x that the adversary will choose is not yet known at the time when the garbled circuit is created. To be more precise, the InputDep distribution of a gate i only depends on the 1-bit value going over the output wire of that gate during the computation $C(x)$. Moreover, if we take any two fixed hybrid games $\mathsf{H}_i, \mathsf{H}_{i+1}$ corresponding to two neighboring pebble configurations (ones which differ by a single move) we can prove indistinguishability even if the adversary does not commit to the entire n-bit input x ahead of time but only commits to the bits going over the output wires of all gates i that are in InputDep mode in either configuration. This means that as long as the pebbling strategy only uses m pebbles of the InputDep type at any point in time, each pair of hybrids $\mathsf{H}_i, \mathsf{H}_{i+1}$ can proved indistinguishable in a partially selective setting where the adversary only commits to m bits of information about his input ahead of time, rather than committing to the entire n bit input x. Using our framework,

this shows that whenever there is a pebbling strategy for the circuit C that requires ℓ moves and uses at most m pebbles of the $\mathsf{InputDep}$ type, we can translate the selective hybrids into a proof of adaptive security where the security loss is $\ell \cdot 2^m$.

It turns out that for any graph of depth d there is a pebbling strategy that uses $O(d)$ pebbles and $\ell = 2^{O(d)}$ moves, meaning that we can prove adaptive security with a $2^{O(d)}$ security loss. This leads to a proof of adaptive security for NC^1 circuits where the reduction has only polynomial security loss, but more generally we can often get a much smaller security loss than the trivial 2^n bound achieved by naïve random guessing.[8]

1.4 Constrained Pseudorandom Functions

Goldreich et al. [GGM84] introduced the notion of a pseudorandom function (PRF). A PRF is an efficiently computable keyed function $\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, where $\mathsf{F}(k, \cdot)$, instantiated with a random key $k \leftarrow \mathcal{K}$, cannot be distinguished from a function randomly chosen from the set of all functions $\mathcal{X} \to \mathcal{Y}$ with non-negligible probability. More recently, the notion of constrained pseudorandom functions (CPRF) was introduced as an extension of PRFs, by Boneh and Waters [BW13], Boyle et al. [BGI14] and Kiayias et al. [KPTZ13], independently. Informally, a constrained PRF allows the holder of a master key to derive keys which are constrained to a set, in the sense that such a key can be used to evaluate the PRF on that set, while the outputs on inputs outside of this set remain indistinguishable from random.

Goldreich et al., in addition to formally defining PRFs, gave a construction of a PRF from any length doubling pseudorandom generator (PRG). Their construction is depicted in Figure 2. All three of the aforementioned results [BW13, BGI14, KPTZ13] show that this GGM construction already gives a so-called "prefix-constrained" PRF, which is a CPRF where for any $x \in \{0,1\}^*$, one can give out keys which allow to evaluate the PRF on all inputs whose prefix is x. This is a simple but already very interesting class of CPRFs as it can be used to construct a punctured PRF, which in turn is a major tool in constructing various sophisticated primitives based on indistinguishability obfuscation (see, for example, [BW13, SW14, HSW14]).

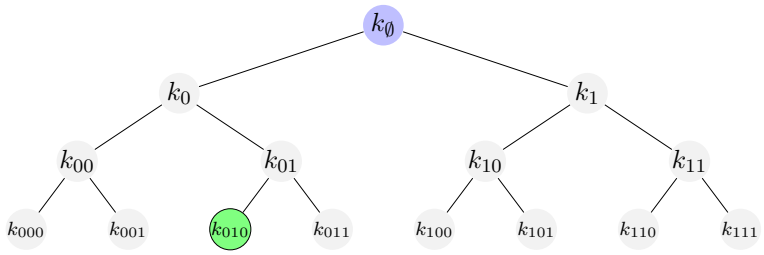

Figure 2: Illustration of the GGM PRF. Every left child $k_{x\|0}$ of a node k_x is defined as the first half of $\mathsf{PRG}(k_x)$, the right child $k_{x\|1}$ as the second half. The circled node corresponds to $\mathsf{GGM}(k_\emptyset, 010)$.

[8] The presentation in [JW16] follows the above outline fairly closely and the reader can easily match it with our general framework. The one conceptual difference is that we think of all the hybrids H_i as existing in the selective setting where the adversary commits to the entire input but then we analyze indistinguishability of neighboring hybrids in a partially selective setting. The work of [JW16] thought of the hybrids H_i as already being partially selective, which made it difficult to compare neighboring hybrids, since the adversary was expected to commit to different information in each one. We view our new framework as being conceptually simpler.

Prior work. To show that the GGM construction is a prefix-constrained PRF one must show how to transform an adversary that breaks GGM as a prefix-constrained PRF into a distinguisher for the underlying PRG. The proofs in [BW13, BGI14, KPTZ13] only show selective security, where the adversary must initially commit to the output he wants to be challenged on in the security game. There is a loss in tightness by a factor of $2n$. This can then be turned into a proof against adaptive adversaries via random guessing, losing an additional exponential factor 2^n in the input length n.

Fuchsbauer et al. [FKPR14] showed that it is possible to achieve adaptive security by losing only factor of $(3q)^{\log n}$, where q denotes the number of queries made by the adversary — if q is polynomial, the loss is not exponential as before, but just quasi-polynomial. The bound relies on the so-called "nested hybrids" technique. Informally, the idea is to iterate random guessing and hybrid arguments several times. The random guessing is done in a way where one only has to guess some tiny amount of information, which although insufficient to get a full reduction using the hybrid argument, nevertheless reduces the complexity of the task significantly. Every such iteration "cuts" the domain in half, so after logarithmically many iterations the reduction is done. If the number of iterations is small, and the amount of information guessed in each iteration tiny, this can still lead to a reduction with much smaller loss than "single shot" random guessing.

Our results. We cast the result in [FKPR14] in our framework, giving an arguably simpler and more intuitive proof. To this aim, we first describe the GGM construction and sketch its security proof.

Given a $\mathsf{PRG}\colon \{0,1\}^m \to \{0,1\}^{2m}$, the PRF $\mathsf{GGM}\colon \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^m$ is defined recursively as

$$\mathsf{GGM}(k,x) = k_x \text{ where } k_\emptyset = k \text{ and } k_{x\|0}\|k_{x\|1} = \mathsf{PRG}(k_x).$$

The construction is also a prefix-constrained PRF: given a key k_x for any $x \in \{0,1\}^*$, one can evaluate $\mathsf{GGM}(k,x')$ for all x' whose prefix is x.

The security of the GGM as a PRF is given in [GGM84]. In particular, they show that if an adversary exists who distinguishes $\mathsf{GGM}(k,\cdot)$ (real experiment) from a uniformly random function (random experiment) with advantage ϵ making q (adaptive) queries, then an adversary of roughly the same complexity exists who distinguishes $\mathsf{PRG}(U_m)$ from U_{2m} with advantage ϵ/nq. Thus if we assume that PRG is δ-secure, then GGM is δnq-secure against any q-query adversary of the same complexity. This is one of the earliest applications of the hybrid argument.

The security definition for CPRFs is quite different from that of standard PRFs: the adversary will get to query the CPRF $\mathsf{F}(k,\cdot)$ in both, the real and random experiment (and can ask for constrained keys, not just regular outputs), and only at the very end the adversary will choose a challenge query x^*, which is then answered with either the correct CPRF output $\mathsf{F}(k,x^*)$ (in the real experiment) or a random value (in the random experiment). In the selective version of these security experiments, the adversary has to choose the challenge x^* before making any queries. In particular, for the case of prefix-constrained PRFs, the experiment is as follows. The challenger samples $k \in \{0,1\}^n$ uniformly at random. The adversary $\mathcal{A}$ first commits to some $x^* \in \{0,1\}^n$. Then it can make *constrain* queries $x \in \{0,1\}^*$ for any x which is not a prefix of x^*, and receives the constrained key k_x in return. Finally, $\mathcal{A}$ gets either $\mathsf{GGM}(k,x^*)$ (in the real game) or a random value, and must guess which is the case.

Selective hybrids. A naïve sequence of selective hybrids, which is of length $2n$, relies just on the knowledge of x^*. For $n = 8$ the corresponding 16 hybrid games are illustrated in Figure 1.a. Each path $\mathrm{C}(n)$ corresponds to a hybrid, and it "encodes" how the value of the function F is computed on the challenge input x^* (and this determines how the function is computed on the rest of the inputs too). An edge that does not carry a pebble is computed, normally, as defined in GGM — i.e., if the ith edge is not pebbled then $k_{x^*[1,i-1]\|0}\|k_{x^*[1,i-1]\|1}$ is set to $\mathsf{PRG}(k_{x[1,i-1]})$, where for $x \in \{0,1\}^n$, $x[1,i]$ denotes its i bit prefix. On the other hand, for an edge with a pebble, we replace the PRG output with a random value — i.e., $k_{x^*[1,i-1]\|0}\|k_{x^*[1,i-1]\|1}$ is set to a uniformly random string in $\{0,1\}^{2m}$. It's not hard to see that any distinguisher for two consecutive hybrids can be directly used to break the PRG with the same advantage by embedding the PRG-challenge – which is either U_{2m} or $\mathsf{PRG}(U_m)$ – at the right place. Using standard random guessing we can get adaptive security losing an additional factor 2^n in the distinguishing advantage by initially guessing $x^* \in \{0,1\}^n$.

From selective to adaptive. Before we explain the improved reduction, we take a step back and consider an even more selective game where A must commit, in addition to the challenge query $x_q = x^*$, also to the constrain queries $\{x_1, \ldots, x_{q-1}\}$. We can use the knowledge of $x_1, \ldots, x_{q-1}$ to get a better sequence of hybrids: this requires two tricks. First, as in GSD on a path, instead of using the pebbling strategy in Figure 1.a, we switch to the recursive pebbling sequence in Figure 1.b. Second, we need a more concise "indexing" for the pebbles: unlike in the proof for GSD, here we can't simply give the positions of the (up to $\log n+1$) pebbles as hint to simulate the hybrids, as the graph has exponential size, thus even the position of a single pebble would require as many bits to encode as the challenge x^*. Instead, we assume there's an upper bound q on the number of queries made by the adversary. For a pebble on the ith edge, we just give the index of the first constrain query whose i bit prefix coincides with x^*, i.e., the minimum j such that $x_j[1,i] = x^*[1,i]$. This information is sufficient to tell when exactly during the experiment we have to compute a value that corresponds to a pebbled edge.

As there are $3^{\log n}$ hybrids, and each hint comes from a set of size $q^{\log n}$ (i.e., a value $\leq q$ for every pebble), our Theorem 2 implies that GGM is a $\delta(3q)^{\log n}$ secure prefix-constrained PRF if PRG is δ secure.

2 Notation

Throughout, we use λ to denote the security parameter. We use capital letters like X to denote variables, small letters like x to denote concrete values, calligraphic letters like $\mathcal{X}$ to denote sets and sans-serif letters like X to denote algorithms. Our algorithms can all be modelled as (potentially interactive, probabilistic, polynomial time) Turing machines. With $\mathsf{X} \equiv \mathsf{Y}$ we denote that X has exactly the same input/output distribution as Y, and $X \sim Y$ denotes that X and Y have the same distributions. $U_{\mathcal{X}}$ denotes the uniform distribution over $\mathcal{X}$. In particular, U_n denotes the uniform distribution over $\{0,1\}^n$. For a set $\mathcal{X}$, $s_{\mathcal{X}}$ denotes the complexity of sampling uniformly at random from $\mathcal{X}$. For $a, b \in \mathbb{N}$, $a \geq b$, by $[a,b]$ we denote the set $\{a, a+1, \ldots, b\}$. For $x \in \{0,1\}^n$ we'll denote with $x[1,i]$ its i bit prefix.

3 The Framework

We consider a game described via a challenger G which interacts with an adversary A. At the end of the interaction, G outputs a decision bit b and we let $\langle \mathsf{A}, \mathsf{G} \rangle$ denote the random variable corresponding to that bit.

Definition 1. *We say that two games defined via challengers G_0 and G_1 are (s, ε)-indistinguishable if for any adversary A of size at most s:*

$$|\Pr[\langle \mathsf{A}, \mathsf{G}_0 \rangle = 1] - \Pr[\langle \mathsf{A}, \mathsf{G}_1 \rangle = 1]| \leq \varepsilon.$$

We say that two games are perfectly indistinguishable and write $\mathsf{G}_0 \equiv \mathsf{G}_1$ if they are $(\infty, 0)$-indistinguishable.

Selectivized Games. We define two operations that convert adaptive or partially selective games into further selective games.

Definition 2 (Selectivized Game)**.** *Given an (adaptive) game G and some function $g\colon \{0,1\}^* \to \mathcal{W}$ we define the selectivized game $\mathsf{H} = \mathsf{SEL}_{\mathcal{W}}[\mathsf{G}, g]$ which works as follows. The adversary A first sends a commitment $w \in \mathcal{W}$ to H. Then H runs the challenger G against A, at the end of which G outputs a bit b'. Let $\mathsf{transcript}$ denote all communication exchanged between G and A. If $g(\mathsf{transcript}) = w$ then H outputs the bit b' and else it outputs 0. See Figure 3.(a).*

Note that the selectivized game gets a commitment w from the adversary but essentially ignores it during the rest of the game. Only, at the very end of the game, it checks that the commitment matches what actually happened during the game.

Definition 3 (Further Selectivized Game)**.** *Assume $\hat{\mathsf{H}}$ is a (partially selective) game which expects to receive some commitment $u \in \mathcal{U}$ from the adversary in the first round. Given functions $g\colon \{0,1\}^* \to \mathcal{W}$ and $h\colon \mathcal{W} \to \mathcal{U}$ we define the further selectivized game $\mathsf{H} = \mathsf{SEL}_{\mathcal{U} \to \mathcal{W}}[\hat{\mathsf{H}}, g, h]$ as follows. The adversary A first sends a commitment $w \in \mathcal{W}$ to H and H begins running $\hat{\mathsf{H}}$ and passes it $u = h(w)$. It then continues running the game between $\hat{\mathsf{H}}$ and A at the end of which $\hat{\mathsf{H}}$ outputs a bit b'. Let $\mathsf{transcript}$ denote all communication exchanged between $\hat{\mathsf{H}}$ and A. If $g(\mathsf{transcript}) = w$ then H outputs the bit b' and else it outputs 0. See Figure 3.(b).*

Note that if $\hat{\mathsf{H}}$ is a (partially selective) game where the adversary sends some commitment u, then in the further selectivized game the adversary might have to commit to more information w. The further selectivized game essentially ignores w and only relies on the partial information $u = h(w)$ during the course of the game but only at the very end is still checks that the full commitment w matches what actually happened during the game.

Random Guessing. We first present the basic reduction using random guessing.

Lemma 1. *Assume we have two games defined via challengers G_0 and G_1 respectively. Let $g\colon \{0,1\}^* \to \mathcal{W}$ be an arbitrary function and define the selectivized games $\mathsf{H}_b = \mathsf{SEL}_{\mathcal{W}}[\mathsf{G}_b, g]$ for $b \in \{0,1\}$. If H_0, H_1 are (s, ε)-indistinguishable then G_0, G_1 are $(s - s_{\mathcal{W}}, \varepsilon \cdot |\mathcal{W}|)$-indistinguishable, where $s_{\mathcal{W}}$ denotes the complexity of sampling uniformly at random from $\mathcal{W}$.*

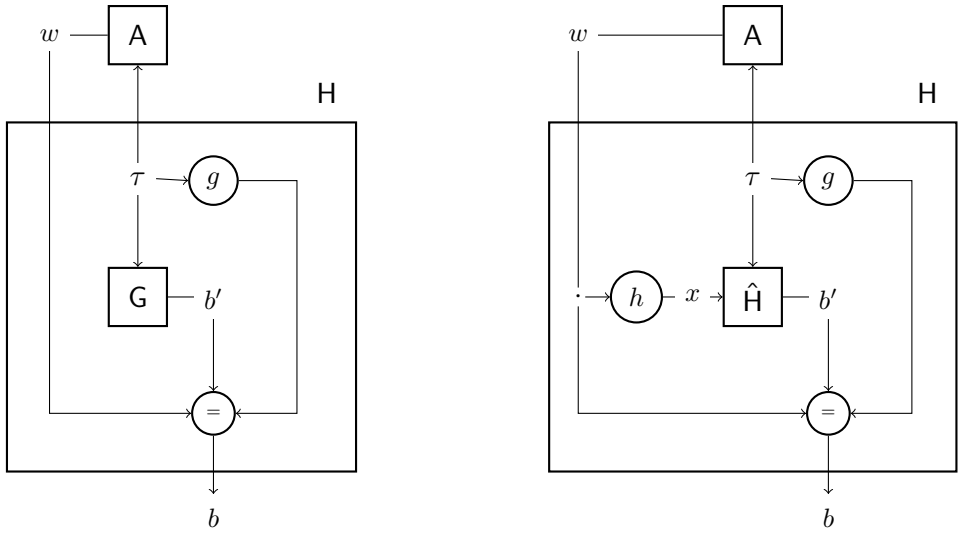

Figure 3: *Selectivizing.* (a): $\mathsf{SEL}_{\mathcal{W}}[\mathsf{G}, g]$, and (b): $\mathsf{SEL}_{\mathcal{U}\to\mathcal{W}}[\hat{\mathsf{H}}, g, h]$. The symbol τ is short for $\mathsf{transcript}$, the nodes with g and h compute the respective functions, whereas the node with $=$ outputs a bit b as prescribed in the consistency check.

Proof. We prove the contrapositive. Assume that there is an adversary A of size $s' = s - s_{\mathcal{W}}$ such that

$$|\Pr[\langle \mathsf{A}, \mathsf{G}_0\rangle = 1] - \Pr[\langle \mathsf{A}, \mathsf{G}_1\rangle = 1]| > \varepsilon \cdot |\mathcal{W}|.$$

Let A^* be the adversary that first chooses a uniformly random $w \leftarrow \mathcal{W}$ and then runs A. Then for $b \in \{0,1\}$:

$$\Pr[\langle \mathsf{A}^*, \mathsf{H}_b\rangle = 1] = \Pr[\langle \mathsf{A}, \mathsf{G}_b\rangle = 1]/|\mathcal{W}|$$

and therefore

$$|\Pr[\langle \mathsf{A}^*, \mathsf{H}_0\rangle = 1] - \Pr[\langle \mathsf{A}^*, \mathsf{H}_1\rangle = 1]| > \varepsilon.$$

Moreover, since A^* is of size $s' + s_{\mathcal{W}} = s$ this shows that H_0 and H_1 are not (s,ε)-indistinguishable. □

3.0.1 Partially Selective Hybrids.

Consider the following setup. We have two adaptive games G_L and G_R. For some function $g\colon \{0,1\}^* \to \mathcal{W}$ we define the selectivized games $\mathsf{H}_\mathsf{L} = \mathsf{SEL}_{\mathcal{W}}[\mathsf{G}_\mathsf{L}, g]$, $\mathsf{H}_\mathsf{R} = \mathsf{SEL}_{\mathcal{W}}[\mathsf{G}_\mathsf{R}, g]$ where the adversary commits to some information $w \in \mathcal{W}$. Moreover, to show the indisitinguishability of $\mathsf{H}_\mathsf{L}, \mathsf{H}_\mathsf{R}$ we have a sequence of ℓ (selective) hybrid games $\mathsf{H}_\mathsf{L} = \mathsf{H}_0, \mathsf{H}_1, \ldots, \mathsf{H}_\ell = \mathsf{H}_\mathsf{R}$.

If we only assume that neighboring hybrids $\mathsf{H}_i, \mathsf{H}_{i+1}$ are indistinguishable then by combining the hybrid argument and random guessing we know that G_L and G_R are indistinguishable at a security loss of $\ell \cdot |\mathcal{W}|$.

Theorem 1. *Assume that for each $i \in \{0,\ldots,\ell-1\}$, the games $\mathsf{H}_i, \mathsf{H}_{i+1}$ are (s,ε)-indistinguishable. Then G_L and G_R are $(s - s_{\mathcal{W}}, \varepsilon \cdot \ell \cdot |\mathcal{W}|)$-indistinguishable, where $s_{\mathcal{W}}$ denotes the complexity of sampling uniformly at random from $\mathcal{W}$.*

Proof. Follows from Lemma 1 and the hybrid argument. □

Our goal is to avoid the loss of $|\mathcal{W}|$ in the above theorem. To achieve this, we will assume a stronger condition: not only are neighboring hybrids $\mathsf{H}_i, \mathsf{H}_{i+1}$ indistinguishable, but they are selectivized versions of less selective games $\hat{\mathsf{H}}_{i,0}, \hat{\mathsf{H}}_{i,1}$ which are already indistinguishable. In particular, we assume that for each pair of neighboring hybrids $\mathsf{H}_i, \mathsf{H}_{i+1}$ there exist some less selective hybrids $\hat{\mathsf{H}}_{i,0}, \hat{\mathsf{H}}_{i,1}$ where the adversary only commits to much less information $h_i(w) \in \mathcal{U}$ instead of $w \in \mathcal{W}$. In more detail, for each i there is some function $h_i \colon \mathcal{W} \to \mathcal{U}$ that lets us interpret H_{i+b} as a selectivized version of $\hat{\mathsf{H}}_{i,b}$ via $\mathsf{H}_{i+b} \equiv \mathsf{SEL}_{\mathcal{U}\to\mathcal{W}}[\hat{\mathsf{H}}_{i,b}, g, h_i]$. In that case, the next theorem shows that we only get a security loss proportional to $|\mathcal{U}|$ rather than $|\mathcal{W}|$. Note that different pairs of "less selective hybrids" $\hat{\mathsf{H}}_{i,0}, \hat{\mathsf{H}}_{i,1}$ rely on completely different partial information $h_i(w)$ about the adversary's choices. Moreover, the "less selective" hybrid that we associate with each H_i can be different when we compare $\mathsf{H}_{i-1}, \mathsf{H}_i$ (in which case it is $\hat{\mathsf{H}}_{i-1,1}$) and when we compare H_i and H_{i+1} (in which case it is $\hat{\mathsf{H}}_{i,0}$).

Theorem 2 (main)**.** *Let G_L and G_R be two adaptive games. For some function $g \colon \{0,1\}^* \to \mathcal{W}$ we define the selectivized games $\mathsf{H}_\mathsf{L} = \mathsf{SEL}_\mathcal{W}[\mathsf{G}_\mathsf{L}, g]$, $\mathsf{H}_\mathsf{R} = \mathsf{SEL}_\mathcal{W}[\mathsf{G}_\mathsf{R}, g]$. Let $\mathsf{H}_\mathsf{L} = \mathsf{H}_0, \mathsf{H}_1, \ldots, \mathsf{H}_\ell = \mathsf{H}_\mathsf{R}$ be some sequence of hybrid games.*

Assume that for each $i \in \{0, \ldots, \ell - 1\}$, there exists a function $h_i \colon \mathcal{W} \to \mathcal{U}$ and games $\hat{\mathsf{H}}_{i,0}, \hat{\mathsf{H}}_{i,1}$ such that:

$$\mathsf{H}_i \equiv \mathsf{SEL}_{\mathcal{U}\to\mathcal{W}}[\hat{\mathsf{H}}_{i,0}, g, h_i] \quad , \quad \mathsf{H}_{i+1} \equiv \mathsf{SEL}_{\mathcal{U}\to\mathcal{W}}[\hat{\mathsf{H}}_{i,1}, g, h_i]. \tag{1}$$

Furthermore, assume that $\hat{\mathsf{H}}_{i,0}, \hat{\mathsf{H}}_{i,1}$ are (s, ε)-indistinguishable. Then G_L and G_R are $(s - s_\mathcal{U}, \varepsilon \cdot \ell \cdot |\mathcal{U}|)$-indistinguishable, where $s_\mathcal{U}$ denotes the complexity of sampling uniformly at random from $\mathcal{U}$.

Proof. Assume that A is an adaptive distinguisher for G_L and G_R of size s' such that

$$|\Pr[\langle \mathsf{A}, \mathsf{G}_\mathsf{L} \rangle = 1] - \Pr[\langle \mathsf{A}, \mathsf{G}_\mathsf{R} \rangle = 1]| > \varepsilon'.$$

Let A^* be a fully selective distinguisher that guesses $w \leftarrow \mathcal{W}$ uniformly at random in the first round and then runs A. By the same argument as in Lemma 1 and Theorem 1 we know that there exists some $i \in [0, \ell)$ such that:

$$|\Pr[\langle \mathsf{A}^*, \mathsf{H}_i \rangle = 1] - \Pr[\langle \mathsf{A}^*, \mathsf{H}_{i+1} \rangle = 1]| \geq \varepsilon'/(\ell \cdot |\mathcal{W}|) \tag{2}$$

Let A' be a partially selective distinguisher that guesses $u \leftarrow \mathcal{U}$ uniformly at random in the first round and then runs A. We want to relate the probabilities $\Pr[\langle \mathsf{A}^*, \mathsf{H}_{i+b} \rangle = 1]$ and $\Pr[\langle \mathsf{A}', \hat{\mathsf{H}}_{i,b} \rangle = 1]$.

Recall that the game $\langle \mathsf{A}^*, \mathsf{H}_{i+b} \rangle$ consists of A^* selecting a uniformly random value $w \leftarrow \mathcal{W}$ (which we denote by the random variable W) and then we run A against $\hat{\mathsf{H}}_{i,b}(u)$ (denoting the challenger $\hat{\mathsf{H}}_{i,b}$ that gets a commitment u in first round) which results in some $\mathsf{transcript}$ and an output bit b^*; if $g(\mathsf{transcript}) = w$ the final output is b^* else 0.

Similarly, the game $\langle \mathsf{A}', \hat{\mathsf{H}}_{i,b} \rangle$ consists of A' selecting a uniformly random value $u \leftarrow \mathcal{U}$ (which we denote by the random variable U) and then we run A against $\hat{\mathsf{H}}_{i,b}(u)$. Therefore:

$$
\begin{aligned}
&\Pr[\langle \mathsf{A}^*, \mathsf{H}_{i+b}\rangle = 1] \\
&\quad = \sum_{u\in\mathcal{U}} \Pr[\underbrace{h_i(W) = u}_{\text{I}}] \cdot \Pr[\underbrace{\langle \mathsf{A}, \hat{\mathsf{H}}_{i,b}(u)\rangle = 1}_{\text{II}}] \cdot \Pr[W = g(\mathsf{transcript})|\text{I}, \text{II}] \\
&\quad = \sum_{u\in\mathcal{U}} \frac{|h_i^{-1}(u)|}{|\mathcal{W}|} \cdot \Pr[\langle \mathsf{A}, \hat{\mathsf{H}}_{i,b}(u)\rangle = 1] \cdot \frac{1}{|h_i^{-1}(u)|} \\
&\quad = \frac{1}{|\mathcal{W}|} \cdot \sum_{u\in\mathcal{U}} \Pr[\langle \mathsf{A}, \hat{\mathsf{H}}_{i,b}(u)\rangle = 1] \\
&\quad = \frac{|\mathcal{U}|}{|\mathcal{W}|} \cdot \sum_{u\in\mathcal{U}} \Pr[\langle \mathsf{A}, \hat{\mathsf{H}}_{i,b}(u)\rangle = 1] \cdot \Pr[U = u] \\
&\quad = \frac{|\mathcal{U}|}{|\mathcal{W}|} \cdot \Pr[\langle \mathsf{A}', \hat{\mathsf{H}}_{i,b}\rangle = 1]
\end{aligned}
$$

Combining the above with equation 2 we get:

$$
|\Pr[\langle \mathsf{A}', \hat{\mathsf{H}}_{i,0}\rangle = 1] - \Pr[\langle \mathsf{A}', \hat{\mathsf{H}}_{i,1}\rangle = 1]| \geq \varepsilon'/(\ell \cdot |\mathcal{U}|)
$$

Since by assumption $\hat{\mathsf{H}}_{i,0}, \hat{\mathsf{H}}_{i,1}$ are (s, ε)-indistinguishable and A' is of size $s' + s_{\mathcal{U}}$ this shows that when $s' = s - s_{\mathcal{U}}$ then $\varepsilon' \leq \varepsilon \cdot \ell \cdot |\mathcal{U}|$ which proves the theorem.

□

3.1 Example: GSD on a Path

As an example, we consider the problem of generalised selective decryption (GSD) on a path graph with n edges, where n is a power of two.

Let $(\mathsf{Enc}, \mathsf{Dec})$ be a symmetric encryption scheme with (probabilistic) $\mathsf{Enc}\colon \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ and $\mathsf{Dec}\colon \mathcal{K} \times \mathcal{C} \to \mathcal{M}$. We assume $\mathcal{K} \subseteq \mathcal{M}$ so that we can encrypt keys, and that the encryption scheme is (s, δ)-indistinguishable under chosen-plaintext attack.[9] In the game, the challenger — either G_L or G_R — picks $n+1$ random keys $k_0, \ldots, k_n \in \mathcal{K}$, and the adversary A is then allowed to make two types of queries:

- Encryption queries, $(\texttt{encrypt}, v_i, v_j)$: it receives back $\mathsf{Enc}(k_i, k_j)$.
- Challenge query, $(\texttt{challenge}, v_{i^*})$: here the answer differs between G_L and G_R, with G_L answering with k_{i^*} (real key) and G_R answering with $r \leftarrow \mathcal{K}$ (random, "fake" key).

A cannot ask arbitrary queries: it is restricted to encryption queries that form a path graph with the challenge query as the sink. That is, a valid attacker A is allowed exactly n encryption queries $(\texttt{encrypt}, v_{i_t}, v_{j_t})$, for $t = 1, \ldots, n$, and a single $(\texttt{challenge}, v_{i^*})$ query such that the directed graph $\mathsf{G}_\kappa = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{v_0, \ldots, v_n\}$ and $\mathcal{E} = \{(v_{i_1}, v_{j_1}), \ldots, (v_{i_n}, v_{j_n})\}$ forms a path with sink v_{i^*}.

[9] To be precise, we only need the encryption scheme to be secure in a weaker model where encryptions of two random messages $m_0, m_1 \in \mathcal{K}$ under a random key $k \in \mathcal{K}$ are (s, δ)-indistinguishable, with the adversary having access to ciphertexts on random messages from $\mathcal{K}$.

Fully selective hybrids. Let's look at a naïve sequence of intermediate hybrids $\mathsf{H}_0, \ldots, \mathsf{H}_{2n-1}$. The fully selective challenger H_I receives as commitment the exact permutation σ that A will query — i.e, $v_{\sigma(i)}$ is the ith vertex on the path. Therefore, $\mathcal{W} = S_{n+1}$ (the symmetry group over $0, \ldots, n$) and g is the function that outputs the observed permutation from $\mathsf{transcript}$. Next, H_I samples $2(n+1)$ keys $k_0, \ldots, k_n, r_0, \ldots, r_n$, and when A makes a query $(\mathtt{encrypt}, v_{\sigma(i)}, v_{\sigma(i+1)})$, it returns

for $0 \le I \le n$:

$$\begin{array}{lll} \mathsf{Enc}(k_{\sigma(i)}, r_{\sigma(i+1)}) & \textbf{if } (0 \le i \le I) & \text{(Fake edge)} \\ \mathsf{Enc}(k_{\sigma(i)}, k_{\sigma(i+1)}) & \textbf{otherwise}. & \text{(Real edge)} \end{array}$$

for $n < I \le 2n-1$:

$$\begin{array}{lll} \mathsf{Enc}(k_{\sigma(i)}, r_{\sigma(i+1)}) & \textbf{if } (0 \le i \le 2n-1-I) \vee (i = n-1) & \text{(Fake edge)} \\ \mathsf{Enc}(k_{\sigma(i)}, k_{\sigma(i+1)}) & \textbf{otherwise}. & \text{(Real edge)} \end{array} \tag{3}$$

Thus, in the sequence $\mathsf{H}_0, \ldots, \mathsf{H}_{2n-1}$, edges are "faked" sequentially down the path, and then "restored", except for the last edge, in the reverse order up the path— see Figure 1.a. By definition, $\mathsf{H}_0 = \mathsf{G}_\mathsf{L}$ and $\mathsf{H}_{2n-1} = \mathsf{G}_\mathsf{R}$. Moreover, H_I and H_{I+1} can be shown (s, δ)-indistinguishable: when A queries for $(\mathtt{encrypt}, v_{\sigma(I)}, v_{\sigma(I+1)})$, the reduction R_I returns the challenge ciphertext

$$\begin{array}{lll} \mathsf{C}(\cdot, k_{\sigma(I+1)}, r_{\sigma(I+1)}) & \textbf{if } (I \le n) & \text{(Real to fake)} \\ \mathsf{C}(\cdot, r_{\sigma(I+1)}, k_{\sigma(I+1)}) & \textbf{otherwise}. & \text{(Fake to real)} \end{array} \tag{4}$$

For the rest of the queries, R_I works as prescribed in eq.3.[10] It is easy to see that R_I simulates H_I when the ciphertext corresponds to the first message, and H_{I+1} otherwise. By Theorem 1, $(s - n \cdot s_{\mathsf{Enc}}, \delta(2n+1)(n+1)!)$-indistinguishability of G_L and G_R follows, where s_{Enc} is the complexity of the Enc algorithm and the $(n+1)!$ factor is the size of the set $\mathcal{W} = S_{n+1}$.

Partially selective hybrids. In order to simulate according to the strategy just described, it *suffices* for the hybrid (as well as the reduction) to guess the edges that are faked — however, this number can be at most n (e.g., in the middle hybrids) and, therefore, the simulator guesses the whole path anyway. Intuitively, this is where the overall looseness of the bound stems from. Now, consider the alternative sequence of hybrids $\tilde{\mathsf{H}}_0, \ldots, \tilde{\mathsf{H}}_{27}$ given in Figure 1.b: the edges in this sequence are faked and restored, *one* at a time, in a recursive manner to ensure that at most four edges end up fake per hybrid. In particular, the new hybrid $\tilde{\mathsf{H}}_I$, fakes all the edges that belong to a set $\mathcal{P}_I \subseteq \mathcal{E}$. That is, when A makes a query $(\mathtt{encrypt}, v_i, v_j)$ — instead of following eq.3, — $\tilde{\mathsf{H}}_I$ returns

$$\begin{array}{lll} \mathsf{Enc}(k_i, r_j) & \textbf{if } ((v_i, v_j) \in \mathcal{P}_I) & \text{(Fake edge)} \\ \mathsf{Enc}(k_i, k_j) & \textbf{otherwise}. & \text{(Real edge)} \end{array} \tag{5}$$

This strategy can be extended to arbitrary n, and there exists such a sequence of sets $\mathcal{P}_0, \ldots, \mathcal{P}_{3^{\log n}}$ where the sets are of size at most $\log n + 1$.[11]

Next, we show that the above simulation strategy satisfies the requirements for applying Theorem 2. Firstly, as shown in Algorithm 1, the strategy is partially selective — i.e., $\tilde{\mathsf{H}}_{I+b} = \mathsf{SEL}_{\mathcal{U} \to \mathcal{W}}[\hat{\mathsf{H}}_{I,b}, g, h_I]$, where, for $I \in [0, \ell = 3^{\log n}]$, the function $h_I : S_{n+1} \to \mathcal{E}^{\log n + 1}$ computes $\mathcal{P}_I$. Secondly, as the simulation in $\hat{\mathsf{H}}_{I,0}$ and $\hat{\mathsf{H}}_{I,1}$ differ by exactly one edge — which is real

[10] Even though R_I does not know the key $k_{\sigma(I)}$, the query $(\mathtt{encrypt}, v_{\sigma(I-1)}, v_{\sigma(I)})$ does not cause a problem as its response is $\mathsf{Enc}(k_{\sigma(I)}, r_{\sigma(I-1)})$.

[11] Later, in §A.2.3, we see that the sequence $\mathcal{P}_0, \ldots, \mathcal{P}_{3^{\log n}}$ corresponds to an "edge-pebbling" of the path graph.

$\tilde{\mathsf{H}}^{\mathsf{A}}_{I+b}$
1: Obtain $\sigma \in S_{n+1}$ from A
2: Compute $\mathcal{P} := \mathcal{P}_0, \ldots, \mathcal{P}_\ell$
3: Run $\hat{\mathsf{H}}_{I,b}((\mathcal{P}_I, \mathcal{P}_{I+1}))$
4: **if** $g(\mathsf{transcript}) = \sigma$ **then**
5: **return** $\hat{\mathsf{H}}_{I,b}$'s output
6: **else return** 0
7: **end if**

$\hat{\mathsf{H}}^{\mathsf{A}}_{I,b}((\mathcal{P}_I, \mathcal{P}_{I+1}))$
1: Choose $2n$ keys $r_1, \ldots, r_n, k_1, \ldots, k_n \leftarrow \mathcal{K}$
2: Whenever A queries $(\texttt{encrypt}, v_i, v_j)$:
3: **if** $(v_i, v_j) \in \mathcal{P}_{I+b}$ **then return** $\mathsf{Enc}(k_i, r_j)$
4: **else return** $\mathsf{Enc}(k_i, k_j)$
5: **end if**
6: **return** A's output

Algorithm 1: $\tilde{\mathsf{H}}_{I+b} = \mathsf{SEL}_{\mathcal{U}\to\mathcal{W}}[\hat{\mathsf{H}}_{I,b}, g, h_I]$

in one and fake in the other — they can be shown to be (s, δ)-indistinguishable. To be precise, if $(v_{i^*}, v_{j^*}) := \mathcal{P}_I \triangle \mathcal{P}_{I+1}$, where $\triangle$ denotes the symmetric difference, when A queries for $(\texttt{encrypt}, v_{i^*}, v_{j^*})$, the reduction $\tilde{\mathsf{R}}_I$ returns

$$\begin{array}{lll} \mathsf{C}(\cdot, k_{j^*}, r_{j^*}) & \textbf{if } (\mathcal{P}_I \subset \mathcal{P}_{I+1}) & \text{(Real to fake)} \\ \mathsf{C}(\cdot, r_{j^*}, k_{j^*}) & \textbf{otherwise}. & \text{(Fake to real)} \end{array} \tag{6}$$

with the rest of the queries answered as in eq.5.

Although, the number of hybrids is greater than in the previous sequence, the number of fake edges in any hybrid is at most $\log n + 1$. Thus, the reduction can work with less information than earlier. By Theorem 2, $(s - n \cdot s_{\mathsf{Enc}} - s_{\mathcal{P}}, \delta \cdot 3^{\log n} \cdot n^{2(\log n+1)})$-indistinguishability of G_{L} and G_{R} follows, where $s_{\mathcal{P}}$ is the size of the algorithm that generates the set $\mathcal{P} = \{\mathcal{P}_0, \ldots, \mathcal{P}_\ell\}$, and the $n^{2(\log n+1)}$ factor results from the fact that the compressed set $\mathcal{U} = \mathcal{E}^{\log n+1}$. Thus, the bound is improved considerably from exponential to quasi-polynomial. A more formal treatment is given in §A.

4 Adaptive Secret Sharing for Monotone Circuits

Throughout history there have been many formulations of secret sharing schemes, each providing a different notion of correctness or security. We focus here on the computational setting and adapt the definitions of [KNY17] for our purposes. Bellare and Rogaway [RB07] survey many different definitions, so we refer there for more information.

A computational secret sharing scheme involves a dealer who has a secret, a set of n parties, and a collection M of "qualified" subsets of parties called the access structure.

Definition 4 (Access structure)**.** *An* access structure *M on parties $[n]$ is a monotone set of subsets of $[n]$. That is, $M \subseteq 2^{[n]}$ and for all $X \in M$ and $X \subseteq X'$ it holds that $X' \in M$.*

We sometimes think of M as a characteristic function $M\colon 2^{[n]} \to \{0, 1\}$ that outputs 1 on input X if and only if X is in the access structure. Here, we mostly consider access structures that can be described by a *monotone Boolean circuit*. These are directed acyclic graphs (DAGs) in which leaves are labeled by input variables and every internal node is labeled by an OR or AND operation. We assume that the circuit has fan-in k_{in} and fan-out (at most) k_{out}. The computation is done in the natural way from the leaves to the root which corresponds to the output of the computation. A circuit in which every gate has fan-out $k_{\mathsf{out}} = 1$ is called a *formula*.

A secret sharing scheme for M is a method by which the dealer efficiently distributes shares to the parties such that (1) any subset in M can efficiently reconstruct the secret from its shares, and (2) any subset not in M cannot efficiently reveal any partial information on the secret. We denote by Π_i the share of party i and by Π_X the joint shares of parties $X \subseteq [n]$.

Definition 5 (Secret sharing)**.** *Let $M\colon 2^{[n]} \to \{0,1\}$ be an access structure. A secret sharing scheme for M consists and secret space $\mathcal{S}$ of efficient sharing and reconstruction procedures S and R, respectively, that satisfy the following requirements:*

1. *$\mathsf{S}(1^\lambda, n, S)$ gets as input the unary representation of a security parameter, the number of parties and a secret $S \in \mathcal{S}$, and generates a share for each party.*
2. *$\mathsf{R}(1^\lambda, \Pi_X)$ gets as input the unary representation of a security parameter, the shares of a subset of parties X, and outputs a string S'.*
3. Completeness: *For a qualified set $X \in M$ the reconstruction procedure R outputs the shared secret:*

$$\Pr\left[\mathsf{R}(1^\lambda, \Pi_X) = S\right] = 1,$$

   where the probability is over the randomness of the sharing procedure $\Pi_1, \ldots, \Pi_n \leftarrow \mathsf{S}(1^\lambda, n, S)$.
4. Adaptive security: *For every adversary A of size s it holds that*

$$|\Pr[\langle \mathsf{A}, \mathsf{G}_0\rangle = 1] - \Pr[\langle \mathsf{A}, \mathsf{G}_1\rangle = 1]| \leq \epsilon,$$

   where the challenger G_b is defined as follows:

   (a) *The adversary A specifies a secret $S \in \mathcal{S}$.*
      i. *If $b = 0$: the challenger generate shares $\Pi_1, \ldots, \Pi_n \leftarrow \mathsf{S}(1^\lambda, n, S)$.*
      ii. *If $b = 1$: the challenger samples a random $S' \in \mathcal{S}$ and generate shares $\Pi_1, \ldots, \Pi_n \leftarrow \mathsf{S}(1^\lambda, n, S')$.*

   (b) *The adversary* adaptively *specifies an index $i \in [n]$ and if the set of parties he requested so far is unqualified, he gets back Π_i, the share of the i-th party.*

   (c) *Finally, the adversary outputs a bit b', which is the output of the experiment.*

The selective security variant is obtained by changing item 4b in the definition of the challenger G_b so that the adversary first sends a commitment to the set of shares X he wants to see ahead of time before seeing any share. We denote this challenger by $\mathsf{H}_b = \mathsf{SEL}_{2^{[n]}}[\mathsf{G}_b, X]$.

4.1 The scheme of Yao

Here we describe the scheme of Yao (mentioned in [Bei11], see also Vinod et al. [VNS+03]). The access structure M is given by a monotone Boolean circuit that is composed of AND and OR gates with fan-in k_{in} and fan-out (at most) k_{out}. Each leaf in the circuit is associated with an input variable $x_1, \ldots, x_n$ (there may be multiple inputs corresponding to the same input variable). During the sharing process, each wire in the circuit is assigned a label and the shares of party $i \in [n]$ corresponds to the labels of the wires corresponding to the input variable x_i. The sharing is

done from the output wire to the leaves. The reconstruction is done in reverse: using the shares of the parties (that correspond to labels of the input wires), we recover the label of the output wire which will correspond to the secret.

The scheme (S, R) uses a symmetric-key encryption scheme $\mathsf{SKE} = (\mathsf{Enc}, \mathsf{Dec})$ in which keys are uniformly random strings in $\{0,1\}^\lambda$ and is ϵ-secure: any polynomial-time adversary cannot distinguish the encryption of $m_1 \in \{0,1\}^\lambda$ from an encryption of $m_2 \in \{0,1\}^\lambda$ with probability larger than ϵ. The sharing procedure S is described in Figure 4.

The sharing procedure S:
Input: A secret $S \in \{0,1\}^\lambda$.

1. Initialize $\Pi(S,i) = \emptyset$ for every $i \in [n]$.
2. Label the output wire ow with the secret $\ell_{ow} = S$.
3. Repeat the following until all input wires of the circuit are labeled.
   (a) Let g be a gate with k_{in} input wires and (at most) k_{out} output wires. Let $w'_1, \ldots, w'_{k_{\mathsf{out}}}$ be the output wires of g having labels $\ell_{w'_1}, \ldots, \ell_{w'_{k_{\mathsf{out}}}}$ and $w_1, \ldots, w_{k_{\mathsf{in}}}$ be the input wires. Associate with g a fresh encryption key $s_g \leftarrow \{0,1\}^\lambda$.
   (b) If $g = \text{AND}$, assign the label of $w_1, \ldots, w_{k_{\mathsf{in}}}$ to be random conditioned on $\ell_{w_1} \oplus \cdots \oplus \ell_{w_{k_{\mathsf{in}}}} = s_g$.
   (c) If $g = \text{OR}$, assign the label of $w_1, \ldots, w_{k_{\mathsf{in}}}$ to be s_g.
   (d) For every $i \in [n]$, add to the share of the ith party an encryption of the labels of the w'_i's under s_g. That is,
   $$\Pi(S,i) = \Pi(S,i) \cup \{(g, \mathsf{Enc}_{s_g}(\ell_{w'_1}), \ldots, \mathsf{Enc}_{s_g}(\ell_{w'_{k_{\mathsf{out}}}}))\}.$$
4. For every input wire w associated with the input variable x_i, add to the share of the ith party the tuple that consists of the name of the wire and its label:
   $$\Pi(S,i) = \Pi(S,i) \cup \{(w, \ell_w)\}.$$
5. Output $\Pi(S,1), \ldots, \Pi(S,n)$.

Figure 4: Yao's secret sharing scheme (S, R) for an access structure M described by a monotone Boolean circuit.

The reconstruction procedure R of the scheme is essentially applying the reverse operations from the leaves of the circuit to the root. Given the labels of the input wires of an AND gate g, we recover the key associated with g by applying a XOR operation on the labels of the input wires, and then recover the labels of the output wires by decrypting the corresponding ciphertexts. Given the labels of the input wires of an OR gate g, we recover the key associated with g by setting it to be the label of any input wire, and then recover the labels of the output wires by decrypting the corresponding ciphertexts. The label of the output wire of the root gate is the recovered secret.

The scheme is efficient in the sense that the share size of each party is bounded by $k_{\mathsf{out}} \cdot \lambda \cdot s$, where s is the number of gates in the circuit. So, if the circuit is of polynomial-size (in n), then the share size is also polynomial (in n and in the security parameter).

Correctness of the scheme follows by an induction on the depth of the circuit and we omit

further details here. Vinod et al. [VNS+03] proved that this scheme[12] is selectively secure by a sequence of roughly s hybrid arguments, where s is the number of gates in the circuit representation of M. By the basic random guessing lemma (Lemma 1), this scheme is also adaptively secure but the security loss is exponential in the number of players the adversary requests to see. The later can be exponential in $O(n)$ so for our scheme to be adaptively secure, we need the encryption scheme to be exponentially secure.

Theorem 3 ([VNS+03])**.** *Assume that* SKE *is a* ϵ*-secure symmetric-key encryption scheme. Then, for any polynomial-time adversary* A *and any access structure on* n *parties described by a monotone circuit with* s *gates, it holds that*

$$|\Pr[\langle \mathsf{A}, \mathsf{H}_0 \rangle = 1] - \Pr[\langle \mathsf{A}, \mathsf{H}_1 \rangle = 1]| \leq k_{\mathsf{out}} \cdot s \cdot \epsilon,$$

and (using Lemma 1),

$$|\Pr[\langle \mathsf{A}, \mathsf{G}_0 \rangle = 1] - \Pr[\langle \mathsf{A}, \mathsf{G}_1 \rangle = 1]| \leq 2^n \cdot k_{\mathsf{out}} \cdot s \cdot \epsilon,$$

In the following subsection we prove that the scheme is adaptively secure and the security loss is roughly $2^{d \cdot \log s}$, where d and s are the depth and number of gates, respectively, in the circuit representing the access structure.

Theorem 4. *Assume that* SKE *is* ϵ*-secure. Then, for any polynomial-time adversary* A *and any access structure on* n *parties described by a monotone circuit of depth* d *and* s *gates with fan-in* k_{in} *and fan-out* k_{out}*, it holds that*

$$|\Pr[\langle \mathsf{A}, \mathsf{G}_0 \rangle = 1] - \Pr[\langle \mathsf{A}, \mathsf{G}_1 \rangle = 1]| \leq 2^{d \cdot (\log s + \log k_{\mathsf{in}} + 2)} \cdot (2k_{\mathsf{in}})^{2d} \cdot k_{\mathsf{out}} \cdot \epsilon.$$

4.2 Hybrids and pebbling configurations

To prove Theorem 4 we rely on the framework introduced in Theorem 2 that we briefly recall here. Our goal is to prove that an adversary cannot distinguish the challengers $\mathsf{G}_{\mathsf{L}} = \mathsf{G}_0$ and $\mathsf{G}_{\mathsf{R}} = \mathsf{G}_1$, corresponding to the adaptive game. We define the selective version of the games $\mathsf{H}_{\mathsf{L}} = \mathsf{SEL}_{2^{[n]}}[\mathsf{G}_{\mathsf{L}}, X]$ and $\mathsf{H}_{\mathsf{R}} = \mathsf{SEL}_{2^{[n]}}[\mathsf{G}_{\mathsf{R}}, X]$, where the adversary has to commit to the whole set of shares it wished to see ahead of time. We construct a sequence of ℓ selective hybrid games $\mathsf{H}_{\mathsf{L}} = \mathsf{H}_0, \mathsf{H}_1, \ldots, \mathsf{H}_{\ell-1}, \mathsf{H}_\ell = \mathsf{H}_{\mathsf{R}}$. For each H_i we define two selective games $\hat{\mathsf{H}}_{i,0}$ and $\hat{\mathsf{H}}_{i,1}$ and show that for every $i \in \{0, \ldots, \ell - 1\}$, there exists a mapping h_i such that the games H_{i+b} and $\hat{\mathsf{H}}_{i,b}$ (for $b \in \{0, 1\}$) are equivalent up to the encoding of the inputs to the games (given by h_i). Then, we can apply Theorem 2 and obtain our result.

The fully-selective hybrids. The sequence of fully selective hybrids $\mathsf{H}_{\mathsf{L}} = \mathsf{H}_0, \mathsf{H}_1, \ldots, \mathsf{H}_{\ell-1}, \mathsf{H}_\ell = \mathsf{H}_{\mathsf{R}}$ is defined such that each experiment is associated with a pebbling configuration. In a pebbling configuration, each gate is either pebbled or unpebbled. A configuration is specified by a compressed string that fully specifies the names of the gates which have a pebble on them (and the rest of the gates implicitly do not). We will define the possible pebbling configurations later but for now let us denote by Q the number of possible pebbling configurations.

[12] To be more precise, the scheme that Vinod et al. presented and analyzed is slightly different. Specifically, they considered AND and OR gates with fan-out 1 and showed how to separately handle FAN-OUT gates (gates that have fan-in 1 and fan-out 2). Their analysis can be modified to handle our scheme.

We define for every $j \in [Q]$, a hybrid experiment H_j in which the adversary first commits to the set X of parties it wishes to see their shares, and then the challenger executes a *new sharing procedure* S^j that depends on the j-th pebbling configuration. Roughly, this sharing procedure acts exactly as the original sharing procedure S, but whenever it encounters a gate with a pebble it generates bogus ciphertexts rather than the real ones. This sharing procedure is described in Figure 5.

The sharing procedure S^j:
Input: A secret $S \in \{0,1\}^\lambda$.

1. Initialize $\Pi_i = \emptyset$ for every $i \in [n]$.
2. Label the output wire ow with the secret $\ell_{ow} = S$.
3. Repeat the following until all input wires of the circuit are labeled.
   (a) Let g be a gate with k_{in} input wires and (at most) k_{out} output wires. Let $w'_1, \ldots, w'_{k_{\mathsf{out}}}$ be the output wires of g having labels $\ell_{w'_1}, \ldots, \ell_{w'_{k_{\mathsf{out}}}}$ and $w_1, \ldots, w_{k_{\mathsf{in}}}$ be the input wires. Associate with g a fresh encryption key $s_g \leftarrow \{0,1\}^\lambda$.
   (b) If $g = \text{AND}$, assign the label of $w_1, \ldots, w_{k_{\mathsf{in}}}$ to be random conditioned on $\ell_{w_1} \oplus \cdots \oplus \ell_{w_{k_{\mathsf{in}}}} = s_g$.
   (c) If $g = \text{OR}$, assign the label of $w_1, \ldots, w_{k_{\mathsf{in}}}$ to be s_g.
   (d) If g has no pebble on it: For every $i \in [n]$, add to the share of the ith party an encryption of the labels of the w'_i's under s_g. That is,
   $$\Pi_i = \Pi_i \cup \left(g, \mathsf{Enc}_{s_g}(\ell_{w'_1}), \ldots, \mathsf{Enc}_{s_g}(\ell_{w'_{k_{\mathsf{out}}}})\right).$$
   (e) If g has a pebble on it: Sample fresh random strings $r_1, \ldots, r_{k_{\mathsf{out}}}$ and for every $i \in [n]$, add to the share of the ith party an encryption of r_i and under s_g. That is,
   $$\Pi_i = \Pi_i \cup \{(g, \mathsf{Enc}_{s_g}(r_1), \ldots, \mathsf{Enc}_{s_g}(r_{k_{\mathsf{out}}}))\}.$$
4. For every input wire w associated with the input variable x_i, add to the share of the ith party the tuple that consists of the name of the wire and its label:
   $$\Pi_i = \Pi_i \cup \{(w, \ell_w)\}.$$
5. Output $\Pi_1, \ldots, \Pi_n$.

Figure 5: The sharing procedure S^j for an access structure M, described by a monotone Boolean circuit, and the j-th pebbling configuration which encodes the color of the pebble on each gates.

Observe that the hybrid that corresponds to the configuration in which all gates are unpebbled is identical to the experiment H_L and the configuration in which there is a pebble only on the root gate corresponds to the experiment H_R.

Pebbling rules and strategies. The rules of the pebbling game depend on the subset of parties whose shares the adversary sees. The rules are:

1. Can place or remove a pebble on any AND gate for which (at least) one input wire is either *not* in X or comes out of a gate with a pebble on it.

2. Can place or remove a pebble on any OR gate for which all of the incoming wires are either input wires *not* in X or come out of gates all of which have pebbles on them.

Our goal is to find a sequence of pebbling rules so that starting with the initial configuration (in which there are no pebbles at all) will end up with a pebbling configuration in which only the root has a pebble. Jumping ahead, we would like for the sequence of pebbling rules to have the property that each configuration is as short to describe as possible (i.e., minimize Q). One way to achieve this is to have at any configuration along the way, as few pebbles as possible. An even more succinct representation can be obtained if we allow many pebbles but have a way to succinctly represent their location. This is what we achieve in the following lemma.

Lemma 2. *For every subset of parties X and any monotone circuit of depth d, fan-in k_{in}, and s gates, there exists a sequence of $(2k_{\mathsf{in}})^{2d}$ pebbling rules such that every pebbling configuration can be uniquely described by at most $d \cdot (\log s + \log k_{\mathsf{in}} + 1)$ bits.*

Proof. A pebbling configuration is described by a list of pairs (gate name, counter), where the counter is a number between 1 and k_{in}, and another bit b to specify whether the root gate has a pebble or not. The counter will represent the number of predecessors, ordered from left to right, that have a pebble on them. Any encoding uniquely defines a pebbling configuration (but notice that the converse is not true).

Denote by $T_X(d)$ the number of pebbling rules needed (i.e., the length of the sequence) and by $P_X(d)$ the maximum size of the description of the pebbling configuration during the sequence. The sequence of pebbling rules is defined via a recursive procedure in the depth d. We first pebble each of the k_{in} predecessors of the root *from left to right* and add a pair (root gate, counter) to the configuration. After we finish pebbling each predecessor we increase the counter by 1 to keep track of how many predecessors have been pebbled. To pebble all predecessors we used $k_{\mathsf{in}} \cdot T_X(d-1)$ pebbling rules and the maximal size of a configuration is at most $P_X(d-1) + (\log s + \log k_{\mathsf{in}} + 1)$. The $\log s$ term comes from specifying the name of the root gate, the $\log k_{\mathsf{in}}$ term come from the number of predecessors of the root gate that have a pebble on them, and the single bit is to signal whether the root gate is pebbled or not.

After this recursive pebbling each of the predecessors have a pebble only at their root gate and the root (of the depth d circuit) has no pebble. Now, we need to remove the pebble from the root of every predecessor of the root gate and put a pebble on the root gate. For the latter we apply one pebbling rule and put a pebble on the root gate. To remove the pebbles from the predecessors of the root gate we reverse the recursive pebbling procedure (by "unpebbling" from right to left and updating the counter appropriately), resulting in the application of additional $k_{\mathsf{in}} \cdot T_X(d-1)$ pebbling rules. When we finish unpebbling, since the root has no predecessors with pebbles, we remove from the description of the configuration the pair corresponding to the root gate. Thus, we get that the maximum size of a pebbling configuration at any point in time is is

$$P_X(d) \leq P_X(d-1) + (\log s + \log k_{\mathsf{in}} + 1) \Rightarrow P_X(d) \leq d \cdot (\log s + \log k_{\mathsf{in}} + 1).$$

The total number of pebbling rules we apply is

$$T_X(d) \leq 2k_{\mathsf{in}} \cdot T_X(d-1) + 1 \Rightarrow T_X(d) \leq (2k_{\mathsf{in}})^{2d}.$$

This completes the proof of the lemma. □

Recall that we denote by Q the number of possible pebbling configurations. Using the pebbling strategy from Lemma 2, we get that

$$Q \leq 2^{d\cdot(\log s+\log k_{\mathsf{in}}+1)}.$$

The partially-selective hybrids. We define the partially selective hybrids $\hat{\mathsf{H}}_{j,0}$ and $\hat{\mathsf{H}}_{j,1}$ for every H_j and $j \in [Q]$. In both hybrid games $\hat{\mathsf{H}}_{j,0}$ and $\hat{\mathsf{H}}_{j,1}$, the adversary first commits to the *j-th pebbling configuration* and *the next pebbling rule to apply*. Denote by $j' \in [Q]$ the index of the pebbling configuration resulting from applying the next configuration rule to the j-th pebbling configuration. In $\hat{\mathsf{H}}_{j,0}$ the challenger samples the shares from S^j and in $\hat{\mathsf{H}}_{j,1}$ the challenger samples the shares from $\mathsf{S}^{j'}$ (but other than this the games do not change).

Denote by $\mathcal{U}$ the space of messages that the adversary has to commit in the partially selective hybrids $\hat{\mathsf{H}}_{j,b}$. This space includes all tuples of pebbling configurations and an additional valid pebbling rule. First, recall that there are Q possible pebbling configurations. Seocnd, observe that a pebbling rule can be described by a gate name: a pebbling rule is just flipping the color of the pebble on that gate. For a circuit with s gates this requires additional $\log s$ bits. Thus, $\mathcal{U} = \{(i, g) \mid i \in [Q], g \in [s]\}$ and this means that the size of $\mathcal{U}$ is bounded by

$$|\mathcal{U}| \leq Q \cdot s \leq 2^{d\cdot(\log s+\log k_{\mathsf{in}}+1)} \cdot s.$$

By semantic security of the symmetric-key encryption scheme and the fact that we replace k_{out} ciphertexts with bogus ones, we have that the games $\hat{\mathsf{H}}_{j,0}$ and $\hat{\mathsf{H}}_{j,1}$ are indistinguishable. The proof is by planting the challenge ciphertext as the ciphertext in the gate where the "next pebbling rule" is applied. In $\hat{\mathsf{H}}_{j,0}$ it is a "real" ciphertext while in $\hat{\mathsf{H}}_{j,1}$ it is a bogus one.

Lemma 3. *Assume that* SKE *is ϵ-secure. Then, for any polynomial-time adversary* A *and any access structure on n parties described by a monotone circuit it holds that*

$$|\Pr[\langle \mathsf{A}, \hat{\mathsf{H}}_{j,0}\rangle = 1] - \Pr[\langle \mathsf{A}, \hat{\mathsf{H}}_{j,1}\rangle = 1]| \leq k_{\mathsf{out}} \cdot \epsilon.$$

Applying Theorem 2 with the fact that $\ell \leq (2k_{\mathsf{in}})^{2d}$ and $|\mathcal{U}| \leq 2^{d\cdot(\log s+\log k_{\mathsf{in}}+1)} \cdot s$, we get that if SKE is ϵ-secure, then for any polynomial-time adversary A and any access structure on n parties described by a monotone circuit of depth d and s gates of fan-in k_{in} and fan-out k_{out}, it holds that

$$\begin{aligned}|\Pr[\langle \mathsf{A}, \mathsf{G}_0\rangle = 1] - \Pr[\langle \mathsf{A}, \mathsf{G}_1\rangle = 1]| &\leq 2^{d\cdot(\log s+\log k_{\mathsf{in}}+1)} \cdot s \cdot (2k_{\mathsf{in}})^{2d} \cdot k_{\mathsf{out}} \cdot \epsilon \\ &\leq 2^{d\cdot(\log s+\log k_{\mathsf{in}}+2)} \cdot (2k_{\mathsf{in}})^{2d} \cdot k_{\mathsf{out}} \cdot \epsilon.\end{aligned}$$

5 Open Problems

In this work we presented a framework for proving adaptive security of various schemes including secret sharing over access structures defined via monotone circuits, generalized selective decryption, constrained PRFs, and Yao's garbled circuits. The most natural future direction is to find more applications where our framework can be used to prove adaptive security with better security loss than using the standard random guessing. Also, improving our results in terms of security loss is an open problem.

In all of our applications of the framework, the security loss of a scheme is captured by the existence of some pebbling strategy. Does there exist a connection in the opposite direction between the security loss of a scheme and possible pebbling strategies? That is, is it possible to use lower bounds for pebbling strategies to show that various security losses are necessary?

Acknowledgments

The fourth author thanks his advisor Moni Naor for asking whether Yao's secret sharing scheme is adaptively secure and for his support.

References

[Bei11] Amos Beimel. Secret-sharing schemes: A survey. In *3rd International Workshop, IWCC*, pages 11–46, 2011.

[Ben89] Charles H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776, 1989.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 134–153. Springer, Heidelberg, December 2012.

[BL90] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 27–35. Springer, Heidelberg, August 1990.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48:313–317, 1979.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *32nd ACM STOC*, pages 235–244. ACM Press, May 2000.

[FJP15] Georg Fuchsbauer, Zahra Jafargholi, and Krzysztof Pietrzak. A quasipolynomial reduction for generalized selective decryption on trees. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, volume 9215 of *Lecture Notes in Computer Science*, pages 601–620. Springer Berlin Heidelberg, 2015.

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained prfs. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, volume 8874 of *Lecture Notes in Computer Science*, pages 82–101. Springer Berlin Heidelberg, 2014.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 276–288. Springer, Heidelberg, August 1984.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

[HJO+16] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 149–178. Springer, Heidelberg, August 2016.

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 201–220. Springer, Heidelberg, May 2014.

[ISN87] M. Ito, A. Saito, and Takao Nishizeki. Secret sharing schemes realizing general access structure. In *Proc. IEEE Global Telecommunication Conf. (Globecom'87)*, pages 99–102, 1987.

[JW16] Zahra Jafargholi and Daniel Wichs. Adaptive security of yao's garbled circuits. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 433–458, 2016.

[KMN+14] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. In *55th FOCS*, pages 374–383. IEEE Computer Society Press, October 2014.

[KNY17] Ilan Komargodski, Moni Naor, and Eylon Yogev. Secret-sharing for NP. *J. Cryptology*, 30(2):444–469, 2017.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 669–684. ACM Press, November 2013.

[Krá01] Richard Královic. Time and space complexity of reversible pebbling. In *SOFSEM 2001: Theory and Practice of Informatics, 28th Conference on Current Trends in Theory and Practice of Informatics Piestany, Slovak Republic, November 24 - December 1, 2001, Proceedings*, volume 2234 of *Lecture Notes in Computer Science*, pages 292–303. Springer, 2001.

[KW88] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *20th ACM STOC*, pages 539–550. ACM Press, May 1988.

[KW93] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of Structures in Complexity Theory*, pages 102–111, 1993.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

[Pan07] Saurabh Panjwani. Tackling adaptive corruptions in multicast encryption protocols. In SalilP. Vadhan, editor, *Theory of Cryptography*, volume 4392 of *Lecture Notes in Computer Science*, pages 21–40. Springer Berlin Heidelberg, 2007.

[RB07] Phillip Rogaway and Mihir Bellare. Robust computational secret sharing and a unified account of classical secret-sharing goals. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 07*, pages 172–184. ACM Press, October 2007.

[RPRC16] Robert Robere, Toniann Pitassi, Benjamin Rossman, and Stephen A. Cook. Exponential lower bounds for monotone span programs. In *57th FOCS*, pages 406–415. IEEE Computer Society Press, 2016.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

[VNS+03] V. Vinod, Arvind Narayanan, K. Srinathan, C. Pandu Rangan, and Kwangjo Kim. On the power of computational secret sharing. In Thomas Johansson and Subhamoy Maitra, editors, *INDOCRYPT 2003*, volume 2904 of *LNCS*, pages 162–176. Springer, Heidelberg, December 2003.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

A Generalised Selective Decryption

The generalised selective decryption (GSD) game was introduced in [Pan07] in order to capture the hardness of proving adaptive security of cryptographic protocols. [Pan07] then showed how GSD can be used to show adaptive security of multicast encryption protocol and logical key hierarchy.

Notation. For $n = n(\lambda)$, $\mathrm{G} = (\mathcal{V}, \mathcal{E})$ denotes a directed acyclic graph (DAG) with $\mathcal{V} = \{v_1, \ldots, v_n\}$ and $\mathcal{E} \subseteq \mathcal{V}^2$. The set of all graphs of n vertices is denoted $\mathcal{G}(n)$. The indegree (resp., outdegree) of a vertex is defined as the number of edges coming in to (resp., going out of) that vertex. The indegree (resp., outdegree) of the graph is the maximum indegree (resp., outdegree) over all the vertices. A vertex with indegree (resp., outdegree) zero is called a source (resp., sink). Let $\mathcal{S}$ (resp., $\mathcal{T}$) denote the set of source (resp., sink) vertices of a graph. We assume that the set of edges $\mathcal{E}$ is (totally) ordered: we use $(u, v) < (u', v')$ to denote that (u, v) precedes (u', v') in the set. For $v \in \mathcal{V}$, we define $\mathsf{parents}(v) := \{u : (u, v) \in \mathcal{E}\}$ and $\mathsf{in}(v) := \{(u, v) : u \in \mathsf{parents}(v)\}$. We assume that $\mathsf{parents}(\cdot)$ preserves the order on $\mathcal{E}$ — i.e., if $(u, v) < (u', v)$ then u precedes u' in $\mathsf{parents}(v)$. Finally, $\mathsf{parents}^{-1}(v)$ denotes the set $\mathsf{parents}(v)$ with elements in the reverse order.

A.1 Formal Definitions

We generalise the definition of GSD given in §3.1. Let (Enc,Dec) be a symmetric encryption scheme with $\mathsf{Enc}\colon \mathcal{K} \times \mathcal{M} \to \mathcal{C}$, $\mathsf{Dec}\colon \mathcal{K} \times \mathcal{C} \to \mathcal{M}$ and we assume $\mathcal{K} \subseteq \mathcal{M}$ (so we can encrypt keys). We assume that (Enc,Dec) is correct, i.e.,

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

and that it is (s, ε)-indistinguishable under chosen-plaintext attack (IND-CPA) — see Definition 6.

Definition 6 (IND-CPA)**.** *The game is played between a challenger (either* G_0 *or* G_1*) and an adversary on the symmetric encryption scheme* $(\mathsf{Enc}, \mathsf{Dec})$*. The challenger chooses the challenge key* $k \leftarrow \mathcal{K}$*. The adversary can make two types of queries:*

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

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

Definition 7 (Adaptive GSD [Pan07])**.** *The game is played between a challenger* G *(which is either* G_L *or* G_R*) and an adversary* A *using* $(\mathsf{Enc}, \mathsf{Dec})$*.* G *picks* n *keys* $k_1, \ldots, k_n \leftarrow \mathcal{K}$ *uniformly at random, and initialises a graph* $G_\kappa := (\{v_1, \ldots, v_n\}, \emptyset)$*; it also initialises a set* $\mathcal{C} = \emptyset$*.* A *can make three types of queries:*

- *Encryption queries,* $(\texttt{encrypt}, v_i, v_j)$*:* G *returns* $\mathsf{Enc}(k_i, k_j)$*, and adds* (v_i, v_j) *to* $\mathcal{E}$*.*
- *Corruption queries,* $(\texttt{corrupt}, v_i)$*:* G *returns* k_i*,and adds* v_i *to* $\mathcal{C}$*.*
- *One challenge query* $(\texttt{challenge}, v_i)$*: Here the answer differs between* G_L *and* G_R*:* G_L *answers with* k_i *(real key), whereas* G_R *answers with* $r \leftarrow \mathcal{K}$ *(fake key) sampled uniformly at random — for the task to be non-trivial,* v_i *must be a sink and it* must not *be reachable from any vertex in* $\mathcal{C}$*.*[13]

Note that the order "<"on $\mathcal{E}$ considered here is the "temporal" order induced by the game: if A queried $(\texttt{encrypt}, u, v)$ before $(\texttt{encrypt}, u', v')$ in the game, then (u, v) was added to the set $\mathcal{E}$ before (u', v'), and hence $(u, v) < (u', v')$.[14] The graph G_κ, along with the order $<$, is called the *key-graph*. In the fully selectivized version of Definition 7, A must commit to the key-graph beforehand — i.e., the selective challenger $\mathsf{H}_\mathsf{L} = \mathsf{SEL}_{\mathcal{G}_\kappa}[\mathsf{G}_\mathsf{L}, g]$ (resp., $\mathsf{H}_\mathsf{R} = \mathsf{SEL}_{\mathcal{G}_\kappa}[\mathsf{G}_\mathsf{R}, g]$), where $\mathcal{G}_\kappa = \mathcal{G}_\kappa(n)$ is the set of all key-graphs of n vertices (i.e., the set of all graphs $\mathcal{G}(n)$ along with the set of all possible edge orderings on them), and g is the function that extracts the key-graph from the transcript.

Definition 8. *An encryption scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is called* (s, ε)*-adaptive (resp., selective) GSD-secure if* G_L *and* G_R *(resp.,* H_L *and* H_R*) are* (s, ε)*-indistinguishable.*

[13] As noted in [Pan07], through a standard hybrid argument, ε-security in the above model implies $(\varepsilon \cdot m)$-security in a (stronger) model where $m = m(\lambda)$ challenges are allowed.

[14] An order can be maintained even when there are parallel queries (viz., the order within the parallel query)

Existing Results. Let s_{Enc} denote the size of the algorithm Enc. [Pan07] shows that if the key-graph is of depth $l = l(n)$, then an (s, ε)-indistinguishable encryption scheme is also $(s - (n \cdot s_{\mathcal{K}} + n^2 \cdot s_{\mathsf{Enc}}), O(\varepsilon \cdot 2n^{l+1}))$-adaptive GSD-secure. As a corollary, for perfect binary trees the loss in tightness is only $O(n^{\log n+2})$. In [FJP15], it is shown that if the key-graph is a tree, then the encryption scheme is $(s - (n \cdot s_{\mathcal{K}} + n^2 \cdot s_{\mathsf{Enc}}), O(\varepsilon \cdot n^{O(\log n)})$-adaptive GSD-secure — the result is established using the nested hybrids technique [FKPR14]. We recapture the result in [Pan07], as well as the result for paths in [FJP15] in our framework.

A.1.1 Overview.

We prove selective security first (c.f., §A.2) and then apply Theorem 2 to establish adaptive security (c.f., §A.2.2). The main idea behind proving selective security is to associate, as in §4, a hybrid experiment to a pebbling configuration of the underlying key-graph (c.f., Lemma 4).

A.2 Hybrids and Pebbling Configurations

Recall that our goal is to show that the indistinguishability of the encryption scheme implies indistinguishability of the fully selectivized games H_{L} and H_{R}. To this aim, we first construct a set of $\ell - 1$ intermediate (fully selectives) hybrids $\mathsf{H}_{\mathsf{L}} = \mathsf{H}_0, \mathsf{H}_1, \ldots, \mathsf{H}_\ell = \mathsf{H}_{\mathsf{R}}$ by associating each experiment with a pebbling configuration. Then we further selectivize these games by showing that there exists $\hat{\mathsf{H}}_{i,b}$, $b \in \{0, 1\}$, for each H_i, $i \in [0, \ell]$. For this, we rely on the pebbling sequence to have certain desirable properties.

A.2.1 Fully Selective Hybrids.

Let's focus on the first part. For ease of exposition, let's restrict the adversary to commit key-graphs having only one sink which, by definition, is also the challenge vertex in the GSD game. Now, consider the structure of key-graphs for H_{L} and H_{R} — let's call them, respectively, the left and the right key-graph. In the left key-graph all the edges are real, whereas in the right key-graph only the edges that are incident on the sink are fake — some examples are given in Figure 6.

Consider the simplest case of G_1: the left and right key-graphs correspond, exactly, to the left and right games in IND-CPA. To be precise, they correspond to the IND-CPA game where k_2 is the challenge key, and the adversary challenges on the messages (k_1, r): if the challenger responds with a ciphertext that corresponds to k_1, then we end up with the left key-graph for G_1, and otherwise we end up with the right key-graph. Thus, in the case of G_2, changing an edge from real to fake is indistinguishable to a GSD adversary. For G_2, the edges incident on v_1 have to be faked iteratively: first (v_4, v_1), then (v_2, v_1) and finally (v_3, v_1), and we end up in the right key-graph. Thus, we get a sequence of hybrids $\mathsf{H}_{\mathsf{L}} = \mathsf{H}_0, \ldots, \mathsf{H}_3 = \mathsf{H}_{\mathsf{R}}$, one corresponding to each act of faking. Each of these moves can be shown indistinguishable by reducing to the case of G_1.

Next consider the slightly more involved graph G_3: unlike in the cases of G_1 and G_2, some of the edges in G_3 *cannot* be faked straight away. For example, consider the left-key graph and the key-graph that has (v_2, v_1) faked. An attempt to show that these two are indistinguishable would fail: such a reduction, which must set k_2 as the challenge key, would not be able to answer, for example, to the query $(\texttt{encrypt}, v_4, v_2)$. However, if we first fake all the edges that are incident on v_2 (iteratively, as in the case of G_2), then the query $(\texttt{encrypt}, v_4, v_2)$ does not pose a problem (as it is responded with $\mathsf{Enc}(v_4, r)$ for some $r \leftarrow \mathcal{K}$). Thus, before faking an edge (u, v) we must ensure that all the incoming edges to u are faked. Such a sequence is given in Figure 7.

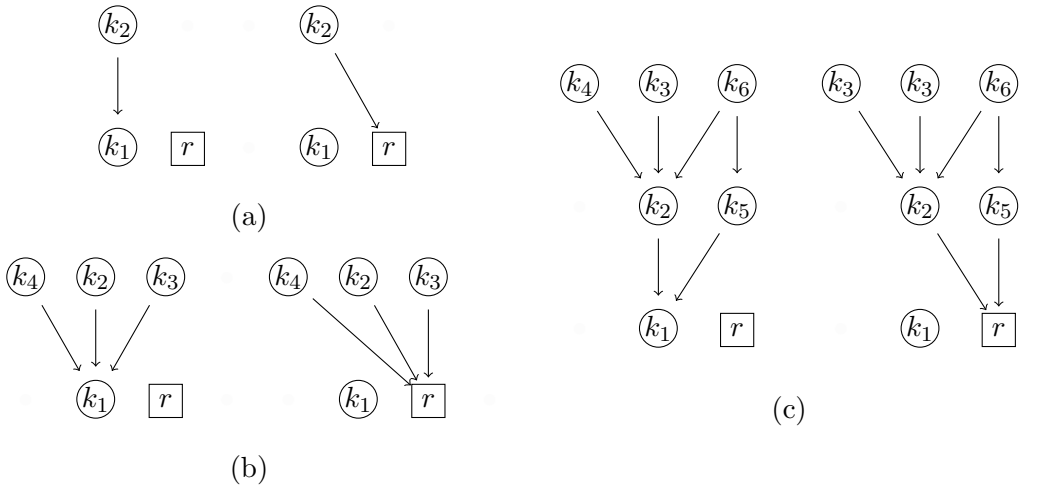

Figure 6: Examples of key-graphs for the extreme games $\mathsf{H_L}$ and $\mathsf{H_R}$. (a): G_1, (b): G_2 and (c): G_3. In the key-graphs given on the left (the left key-graph), all the edges are real, whereas in those given on the right (the right key-graph), only the edges that are incident on the challenge vertex (v_1) are fake.

To summarize, our goal is to start with the left key-graph and end up with the right key-graph by faking edges or restoring faked edges, *one* at a time keeping in mind that before faking or restoring an edge (u, v) all the edges coming in to u must be already fake. This can be abstracted out as an edge-pebbling game: faking (resp., restoring) an edge is equivalent to placing (resp., removing) a pebble on the edge, and the goal of the pebbling is to pebble all the incoming edges of the sink. A more formal definition follows.

Reversible edge-pebbling. The classical reversible black pebbling game on DAGs was introduced in [Ben89] to model reversible computation. A pebble in the reversible edge-pebbling game – unlike in the classical case – is placed on the edges. Thus, a pebbling configuration is a subset of the edges. *A vertex is deemed pebbled if all its incoming edges are pebbled* — i.e., $v \in \mathcal{V}$ is pebbled in a configuration $\mathcal{P}_i$ if $\mathsf{in}(v) \subseteq \mathcal{P}_i$.

Definition 9. *A **reversible edge-pebbling** of a DAG $G = (\mathcal{V}, \mathcal{E})$ is a sequence $\mathcal{P} := \mathcal{P}_0, \ldots, \mathcal{P}_\ell$ with each configuration $\mathcal{P}_i \subseteq \mathcal{E}$ and, in particular, $\mathcal{P}_0 = \emptyset$ and $\mathcal{P}_\ell = \cup_{v \in \mathcal{T}'} \mathsf{in}(v)$ for some $\emptyset \neq \mathcal{T}' \subseteq \mathcal{T}$. In a move, a pebble can be placed on or removed from an edge (u, v) iff the vertex u is pebbled — thus, edges going out from $\mathcal{S}$ can be pebbled or unpebbled in any move. Thus, $\mathcal{P}$ is a valid pebbling sequence iff*

$$\forall i \in [1, \ell], \exists! (u, v) \in \mathcal{P}_{i-1} \triangle \mathcal{P}_i, (\mathsf{in}(u) \subseteq \mathcal{P}_{i-1}).$$

Thus, for a target set $\mathcal{T}' \subseteq \mathcal{T}$, starting from a completely unpebbled graph, the aim is to achieve a pebbling configuration in which only the vertices $\mathcal{T}'$ are pebbles, and all other edges are unpebbled. The space complexity of an edge pebbling $\mathcal{P}$ is defined as $\mathsf{S}_{\mathcal{P}}(\mathrm{G}) := \max_{i \in [0,\ell]} |\mathcal{P}_i|$, and the space complexity of edge-pebbling a DAG is $\mathsf{S}(\mathrm{G}) := \min_{\mathcal{P}} \mathsf{S}_{\mathcal{P}}(\mathrm{G})$. Similarly, we for an edge-pebbling $\mathcal{P} = \mathcal{P}_0, \ldots, \mathcal{P}_\ell$ of a graph G, the number of moves (ℓ, i.e.,) is denoted by $\mathsf{T}_{\mathcal{P}}(\mathrm{G})$.

Definition 10 (Ordered edge-pebbling)**.** *An edge-pebbling $\mathcal{P} = \mathcal{P}_0, \ldots, \mathcal{P}_\ell$ of a graph $G = (\mathcal{V}, \mathcal{E})$ that has ordered $\mathcal{E}$ is said to be* ***ordered*** *if for each configuration $\mathcal{P}_i$ the following holds:*

$$(u, v) \in \mathcal{P}_i \implies \forall\ ((u', v) < (u, v)) : (u', v) \in \mathcal{P}_i.$$

In other words, if an edge (u, v) carries a pebble in $\mathcal{P}_i$ then all the edges that are incident on v and precede *(u, v) must also carry pebbles.*

For an ordered edge-pebbling, we are interested in a coarser measure of space complexity called the *lateral* space complexity, which takes into account the fact that an ordered pebbling is "compressible": in a configuration $\mathcal{P}_i$, if the pebbles that come in to a vertex v are $(u_1, v), \ldots, (u_p, v)$ (in that order), then it suffices just to store v and the number of incident pebbled edges, i.e., p. We call p the "index" of a vertex in a pebbling configuration. More formally, for a vertex $v \in \mathcal{V}$ and a pebbling configuration $\mathcal{P}_i$, let $\mathsf{index}(v, \mathcal{P}_i)$ denote the number of incoming edges to v that are pebbled — i.e.,

$$\mathsf{index}(v, \mathcal{P}_i) := \left|\{(u, v') \in \mathcal{P}_i : v' = v\}\right|.$$

Note that for a graph with bounded indegree d, the index is also bounded by d. Thus, an ordered pebbling configuration $\mathcal{P}_i$ is compressed as it suffices to store all the vertices that have non-zero index (along with the index), i.e., the set $\mathcal{Q}_i$ defined as

$$\mathcal{Q}_i := \{(v, \mathsf{index}(v, \mathcal{P}_i)) : v \in \mathcal{V}, \mathsf{index}(v, \mathcal{P}_i) > 0\}.$$

The maximum size of these sets is precisely defined as the lateral space-complexity S' of the edge pebbling — i.e., $\mathsf{S}'_{\mathcal{P}}(\mathrm{G}) := \max_{i \in [0,\ell]} |\mathcal{Q}_i|$. The lateral space complexity of edge-pebbling a DAG is $\mathsf{S}'(\mathrm{G}) := \min_{\mathcal{P}} \mathsf{S}'_{\mathcal{P}}(\mathrm{G})$.[15]

The edge-pebbling equivalent to the sequence of hybrids given in Figure 7 is given in Figure 8. However, it is *not* ordered. An alternative edge-pebbling, which is ordered, is given in Figure 9. Next, we formally show that an ordered edge-pebbling implies a sequence of fully selective hybrids.

Lemma 4. *For $G_\kappa \in \mathcal{G}_\kappa$, let $\mathcal{P}_0, \ldots, \mathcal{P}_\ell$ be an ordered edge-pebbling that is generated by a deterministic edge-pebbling algorithm P. Then the sequence of hybrids $\mathsf{H}_{\mathsf{P},I}$, $I \in [0, \ell]$ and with $\mathsf{H}_{\mathsf{P},I}$ defined in Algorithm 2, constitutes a valid sequence of fully selective hybrids. Furthermore, if s_{P} denotes the complexity of P, then an encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ that is (s, ε)-secure under IND-CPA, is $(s - O(s_{\mathsf{P}} + n^2 \cdot s_{\mathsf{Enc}}), \varepsilon \cdot \ell)$-selective GSD-secure.*

As a corollary to Lemma 4 and Theorem 1, an encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ that is (s, ε)-secure under IND-CPA, is $(s - O(s_{\mathsf{P}} + n^2 \cdot s_{\mathsf{Enc}} + s_{\mathcal{W}}), \varepsilon \cdot \ell \cdot exp(n))$-adaptive GSD-secure.

of lemma. Fix a graph G and consider the pebbling sequence $\mathcal{P} = \mathcal{P}_0, \ldots, \mathcal{P}_\ell \leftarrow \mathsf{P}(\mathrm{G})$. Each configuration $\mathcal{P}_I$ yields a challenger $\mathsf{H}_{I,\mathsf{P}}$ described in Algorithm 2. By the properties of the pebbling configurations $\mathcal{P}_0$ and $\mathcal{P}_\ell$, it is easy to see that $\mathsf{H}_{\mathsf{P},0} \equiv \mathsf{H}_{\mathsf{L}}$ and $\mathsf{H}_{\mathsf{P},\ell} \equiv \mathsf{H}_{\mathsf{R}}$. Moreover, $\mathsf{H}_{\mathsf{P},I}$ is $(s - O(s_{\mathsf{P}} + n^2 \cdot s_{\mathsf{Enc}}), \varepsilon)$-indistinguishable from $\mathsf{H}_{\mathsf{P},I+1}$, for $I \in [0, \ell - 1]$ — we do not give the details, but the main point is to plug in the challenge ciphertext on the (only) edge that is different in the two hybrids. Thus, $\mathsf{H}_{\mathsf{P},0}, \ldots, \mathsf{H}_{\mathsf{P},\ell}$ constitutes a valid sequence of fully selective hybrids. □ □

[15] Note that $\mathsf{S}'(\mathrm{G}) \leq \mathsf{S}(\mathrm{G})$, with equality holding, for example, for graphs with in-degree one — also, $\mathsf{S}(\mathrm{G}) \leq d \cdot \mathsf{S}'(\mathrm{G})$, where d is the in-degree of the DAG.

$\mathsf{H}^{\mathsf{A}}_{\mathsf{P},I}$

Obtain the key-graph $\mathrm{G}_\kappa \in \mathcal{G}_\kappa$ from A
Compute $\mathcal{P}_0, \ldots, \mathcal{P}_\ell \leftarrow \mathsf{P}(\mathrm{G}_\kappa)$ ▷ Generate the ordered pebbling
Initialise $c_1, \ldots, c_n := 0$ ▷ Counters for edges incident on each vertex
Compute $\mathcal{Q}_I$ ▷ Contains the set of "pebbled" vertices and their indices
Choose $2n$ keys $r_1, \ldots, r_n, k_1, \ldots, k_n \leftarrow \mathcal{K}$
Whenever A makes a query $(\texttt{encrypt}, v_i, v_j)$:
if $(v_j \in \mathcal{Q}_I)$ **then** ▷ Carries pebble?
    if $c_j \leq \text{index}_j$ **then return** $\mathsf{Enc}(k_i, r_j)$ ▷ Fake if within the index
    else return $\mathsf{Enc}(k_i, k_j)$ ▷ Real edge
    end if
    $c_j := c_j + 1$
else return $\mathsf{Enc}(k_i, k_j)$ ▷ Real edge
end if
Whenever A makes a query $(\texttt{corrupt}, v_i)$ or $(\texttt{challenge}, v_i)$ **return** k_i
return A's output

Algorithm 2: Template for generating fully selective hybrids.

A.2.2 Partially Selective Hybrids.

Here we show that the fully selective sequence of hybrids $\mathsf{H}_{\mathsf{P},0}, \ldots, \mathsf{H}_{\mathsf{P},\ell}$ constructed in the previous subsection is also partially selective, with each $\mathsf{H}_{\mathsf{P},I}$ of the form prescribed in eq.1. This is made possible by the fac that an ordered pebbling is, as we discussed, compressible. As a result, better bounds on adaptive security follows by Theorem 2.

Lemma 5. *For $d = d(n)$ and $l = l(n)$, let $\mathcal{G}_\kappa = \mathcal{G}_\kappa(n, d, l)$ denote the subset of key-graphs in $\mathcal{G}_\kappa(n)$ with in-degree d and depth l. For $G_\kappa \in \mathcal{G}_\kappa(n, d, l)$, let $\mathcal{P}_0, \ldots, \mathcal{P}_\ell$ be an ordered edge-pebbling that is generated by a deterministic edge-pebbling algorithm P. Let β denote its lateral space complexity. Then, for $I \in [0, \ell)$ and $b \in \{0, 1\}$,*

$$\mathsf{H}_{\mathsf{P},I+b} = \mathsf{SEL}_{\mathcal{U}\to\mathcal{G}_\kappa}[\hat{\mathsf{H}}_{I,b}, g, h_I],$$

where $\hat{\mathsf{H}}_{I,b}$ is defined in Algorithm 3, and $\mathcal{U}$ is the set $(\mathcal{V} \times [1, d])^{2\beta}$.

As a corollary to Lemma 5 and Theorem 2, we get our main result for the GSD game.

Theorem 5. *Let n, d, β, ℓ be as in Lemma 5. Let s_{P} denote the complexity of P, then an encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ that is (s, ε)-secure under IND-CPA, is $(s - O(s_{\mathsf{P}} + n^2 \cdot s_{\mathsf{Enc}}), \varepsilon \cdot \ell \cdot (n \cdot d)^{2\beta})$-adaptive GSD-secure, where $\mathcal{U} = (\mathcal{V} \times [1, d])^{2\beta}$.*

of lemma. The partial selectivising of $\mathsf{H}_{\mathsf{P},I+b} = \mathsf{SEL}_{\mathcal{U}\to\mathcal{G}_\kappa}[\hat{\mathsf{H}}_{I,b}, g, h_I]$ is shown in Algorithm 3. The proof for the indistinguishability of $\hat{\mathsf{H}}_{I,0}$ and $\hat{\mathsf{H}}_{I,1}$ follows the same line of argument as in the proof of Lemma 4 and is omitted. □

$\mathsf{H}^{\mathsf{A}}_{\mathsf{P},I}$

1: Obtain the key-graph $\mathrm{G}_\kappa \in \mathcal{G}_\kappa$ from A
2: Compute $\mathcal{P}_0, \ldots, \mathcal{P}_\ell \leftarrow \mathsf{P}(\mathrm{G}_\kappa)$
3: Compute $\mathcal{Q}_I$, $\mathcal{Q}_{I+1}$
4: Run $\hat{\mathsf{H}}_{I,b}(\mathcal{Q}_I, \mathcal{Q}_{I+1})$
5: **return** $\hat{\mathsf{H}}_{I,b}$'s output

$\hat{\mathsf{H}}^{\mathsf{A}}_{I,b}(\mathcal{Q}_I, \mathcal{Q}_{I+1})$

1: Initialise $c_1, \ldots, c_n := 0$
2: Choose $2n$ keys $r_1, \ldots, r_n, k_1, \ldots, k_n \leftarrow \mathcal{K}$
3: Whenever A makes a query $(\texttt{encrypt}, v_i, v_j)$:
4: **if** $(v_j \in \mathcal{Q}_{I+b})$ **then**
5: **if** $(c_j \leq \mathrm{index}_j \in \mathcal{Q}_{I+b})$ **then** ▷ The index of v_j in $\mathcal{Q}_I$ is compared
6: **return** $\mathsf{Enc}(k_i, r_j)$
7: **else return** $\mathsf{Enc}(k_i, k_j)$
8: **end if**
9: $c_j := c_j + 1$
10: **else return** $\mathsf{Enc}(k_i, k_j)$
11: **end if**
12: Whenever A makes a query $(\texttt{corrupt}, v_i)$ or $(\texttt{challenge}, v_i)$ **return** k_i
13: **return** A's output

Algorithm 3: Partially selectivized hybrids. $\mathsf{H}_{\mathsf{P},I+b} := \mathsf{SEL}_{\mathcal{U}\to\mathcal{G}_\kappa}[\hat{\mathsf{H}}_{I,b}, g, h_I]$, where $\mathcal{U}$ is the set $(\mathcal{V} \times [1,d])^{2\beta}$.

A.2.3 Corollaries.

As corollaries to Theorem 6, we capture the existing results on GSD: viz., the result on paths using the nested hybrid technique, given in [FJP15], which was briefly discussed in §3.1, and the bound for general DAGs in [Pan07].

Corollary 1 (GSD for path graphs is quasi-polynomial in length)**.** *Let* $\mathcal{C}(n) := \mathcal{G}_\kappa(n+1, 1, n+1)$ *denote the set of all chain key-graphs of* n *edges. An encryption scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *that is* (s, ε)*-secure under IND-CPA is* $(s - O(s_{\mathsf{P}_1} + n \cdot s_{\mathsf{Enc}}), \varepsilon \cdot 3^n \cdot n^{\log n + 1})$*-adaptive GSD-secure on* $\mathcal{C}(n)$*.*

Proof. Suppose that n is a power of two. An edge-pebbling algorithm for a chain graphs $\mathrm{C}(n) \in \mathcal{C}(n)$ is given in Algorithm 4 — the algorithm is to be invoked on $(\mathrm{G}, (0, n), (0, n))$ in order to pebble the sink. It is called the nested pebbling strategy as it is used implicitly in the argument for path graphs using nested hybrids in [FJP15]. The number of pebbles used by P_1 is captured by the recursion $\mathsf{S}_{\mathsf{P}_1}(\mathrm{C}(n)) = \mathsf{S}_{\mathsf{P}_1}(\mathrm{C}(n/2) + 1$, with $\mathsf{S}_{\mathsf{P}_1}(\mathrm{C}(1)) = 1$.[16] The number of moves, on the other hand, is captured by $T(n) = 3 \cdot T(n/2)$ with $T(1) = 1$. Therefore, $\mathsf{S}_{\mathsf{P}_1}(\mathcal{C}(n)) = \log n + 1$ and $\mathsf{T}_{\mathsf{P}_1}(\mathcal{C}(n)) = 3^{\log n}$. As the indegree of a path graph is one, all its pebbling sequence are ordered, and its lateral space complexity is the same as the number of pebbles — i.e., $\beta = \log n + 1$. The corollary now follows Theorem 6. A similar argument can be made for arbitrary n. □ □

```
P1((A, B), (a, b))
 1: if b = a + 1 then
 2:     if (a, a + 1) is pebbled then remove it        ▷ P_{i+1} := P_i \ {(a, a + 1)}
 3:     else place pebble on (a, a + 1)                ▷ P_{i+1} := P_i ∪ {(a, a + 1)}
 4:     end if                                         ▷ Increment counter i
 5: else
 6:     P1((A, B), (a, (a + b)/2))                     ▷ Recursively pebble left half
 7:     P1((A, B), ((a + b)/2, b))                     ▷ Recursively pebble right half
 8:     P1((A, B), (a, (a + b)/2)))                    ▷ Recursively unpebble left half
 9: end if
10: if (A, B) = (a, b) then HALT end if
```

Algorithm 4: Nested pebbling for path graphs with n edges, where n is a power of two. Note that the initial call must be $\mathsf{P}_1((0, n), (0, n))$.

Corollary 2 (GSD for DAGs is exponential in depth)**.** *For* $l = l(n)$*, let* $\mathcal{G}_\kappa(n, l)$ *denote the subset of graphs in* $\mathcal{G}_\kappa(n)$ *with depth* l*. An encryption scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *that is* (s, ε)*-secure under IND-CPA is* $(s - O(s_{\mathsf{P}_2} + n^2 \cdot s_{\mathsf{Enc}}), \varepsilon \cdot \ell \cdot n^{8l})$*-adaptive GSD-secure on* $\mathcal{G}_\kappa(n, d, l)$*.* [17]

Proof. A reversible edge-pebbling strategy that pebbles a single edge $((u^*, v^*))$ is given in Algorithm 5. In order to pebble a vertex v^*, the algorithm is to be called sequentially on all $u^* \in \mathsf{parents}(v^*)$. An example of this pebbling strategy is given in Figure 9. Let's first see why the pebbling sequence generated is ordered. For an edge (u, v), the order in which the edges incident on a vertex u are pebbled is determined by $\mathsf{parents}(u)$, which we assumed preserves the edge ordering.

[16] It is known that the strategy is optimal — i.e., $\mathsf{S}(\mathrm{C}(n)) = \mathsf{S}_{\mathsf{P}_1}(\mathrm{C}(n))$. [Krá01]

[17] Compare with [Pan07]

Moreover, the edges are restored in the opposite order. Together, these two steps ensure that in any configuration, if an edge (u, v) carries a pebble then all the edges that are incident on v and *precede* (u, v) must also carry pebbles. In addition, all the vertices that have edges with pebbles coming into the them lie along a path from a source to a sink — it follows that the lateral edge-pebbling complexity β is at most l. The number of moves in above strategy is captured by the recursion $T(l) \leq 2n \cdot T(l-1)$ with $T(1) \leq 2n$, and hence $\mathsf{T}_{\mathsf{P}_2}(\mathcal{G}_\kappa(n, l)) < (2n)^l$. Plugging in these values in Theorem 6 proves the corollary. □

$\mathsf{P}_2(\mathrm{G}, (u^*, v^*), (u, v))$

1: **for** $x \in \mathsf{parents}(u)$ **do** $\mathsf{P}_2(\mathrm{G}, (u^*, v^*), (x, u))$ ▷ Pebble parents recursively
2: **if** (u, v) is pebbled **then** remove pebble on (u, v) ▷ $\mathcal{P}_{i+1} := \mathcal{P}_i \setminus \{(u, v)\}$
3: **else** place pebble on (u, v) ▷ $\mathcal{P}_{i+1} := \mathcal{P}_i \cup \{(u, v)\}$
4: **end if** ▷ Increment counter i
5: **for** $x \in \mathsf{parents}^{-1}(u)$ **do** $\mathsf{P}_2(\mathrm{G}, (u^*, v^*), (x, u))$ ▷ Unpebble parents, in reverse
6: **if** $(u, v) = (u^*, v^*)$ **then** HALT **end if**

Algorithm 5: A recursive edge-pebbling algorithm. Note that to pebble an edge $(u, v) \in \mathcal{E}$, the initial call must be $\mathsf{P}_1(\mathrm{G}, (u, v), (u, v))$

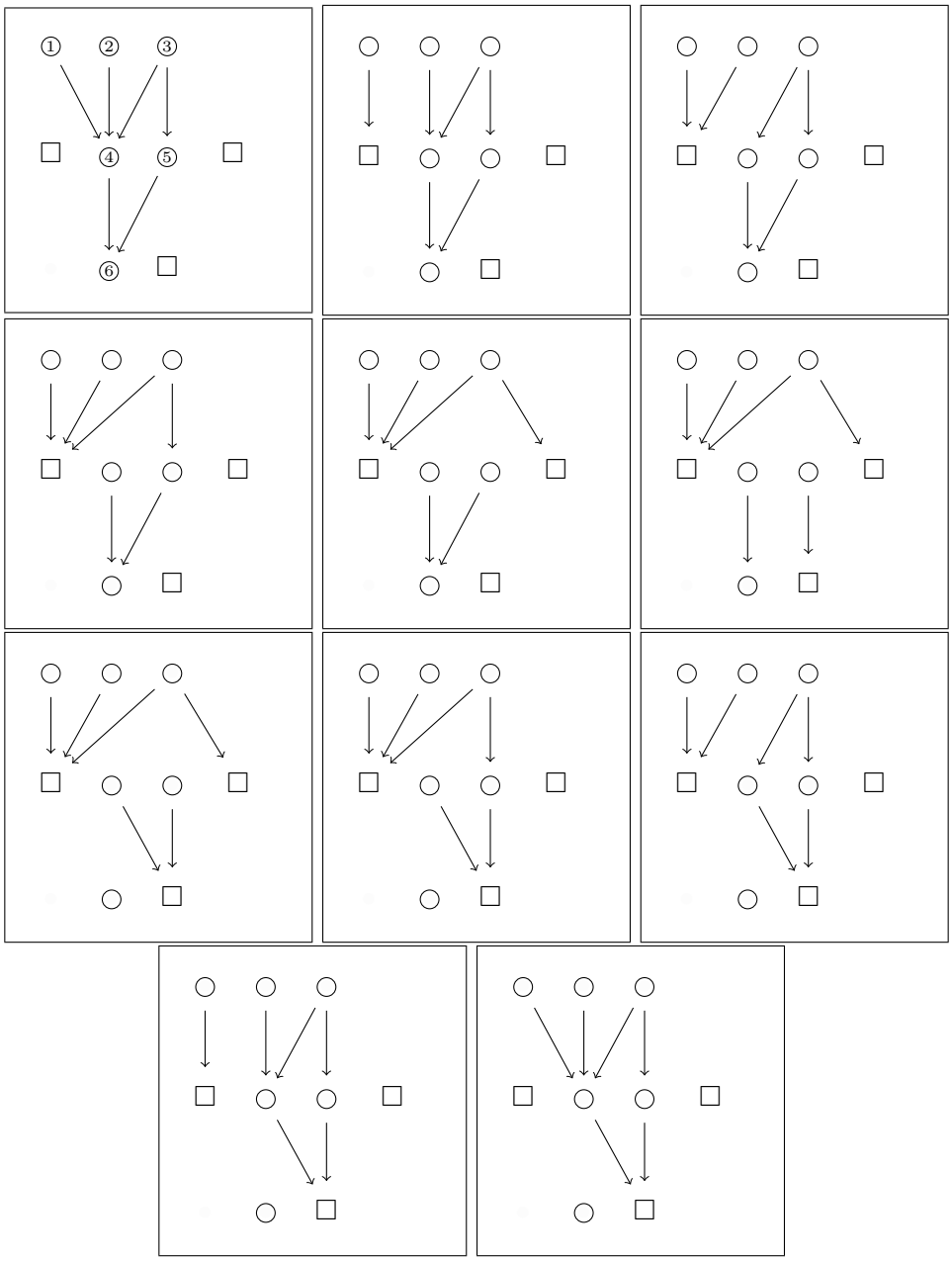

Figure 7: A sequence of hybrids for G_3. The square boxes indicate fake keys and, hence, the edges incident on these boxes are the faked edges.

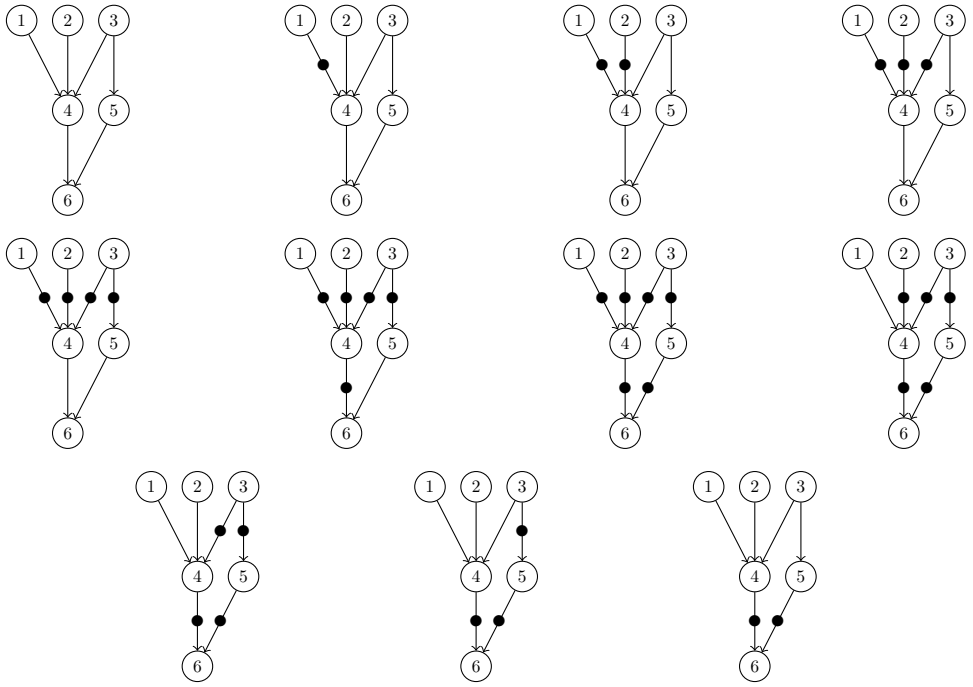

Figure 8: An example of an edge-pebbling sequence — it corresponds to a sequence of hybrids for the graph G_3. If an edge carries a pebble, then that edge is faked during the simulation. Thus, the first configuration is the real game, whereas the last configuration — with all the incoming edges to the challenge faked — is the random game.

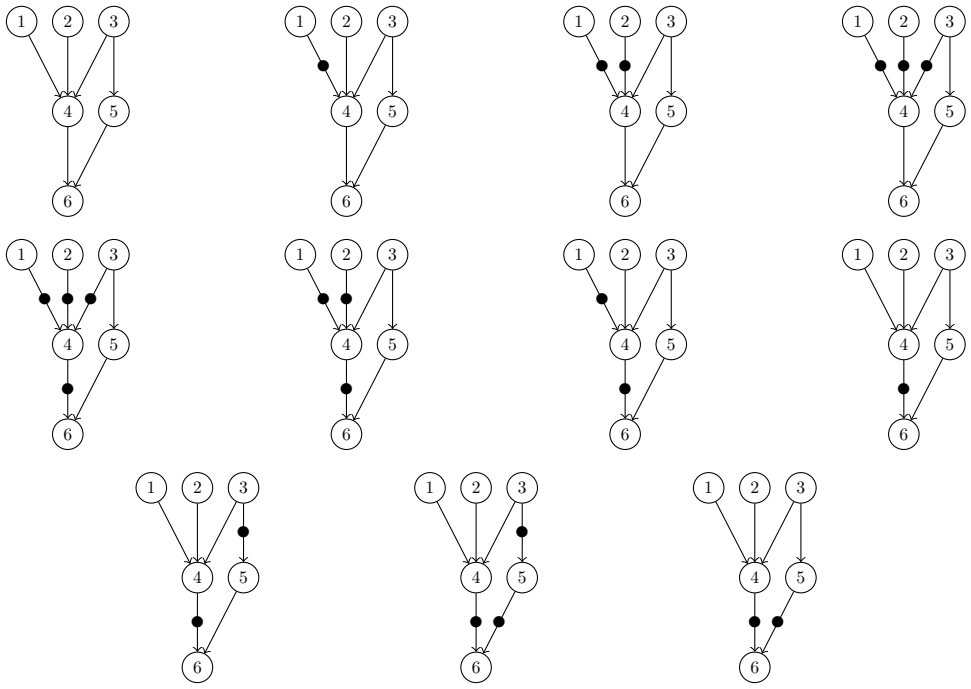

Figure 9: An example of edge-pebbling of G_3 that uses "fewer" pebbles than in Figure 8. Although the space complexity of the above DAG is four, its lateral space complexity is just two. Note that the edge-pebbling is ordered.

B Constrained Pseudorandom Functions

In this section, we formalise the high level ideas that we presented in §1.4

B.1 Formal Definitions

Definition 11 (GGM PRF)**.** *Given a* $\mathsf{PRG} : \{0,1\}^m \to \{0,1\}^{2m}$*, the PRF* $\mathsf{GGM} : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^m$ *is defined as*

$$\mathsf{GGM}(k,x) = k_x \text{ where } k_\emptyset = k \text{ and } k_{x\|0}\|k_{x\|1} = \mathsf{PRG}(k_x).$$

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

Definition 12. *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* $\mathsf{F.constrain} : \mathcal{K} \times \{0,1\}^n \to \mathcal{K}$ *and* $\mathsf{F.eval} : \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}$ *which for all* $k \in \mathcal{K}$*,* $x \in \{0,1\}^n$ *and* $k_x \leftarrow \mathsf{F.constrain}(k,x)$ *satisfy*

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

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

$$\mathsf{F}_{\mathsf{GGM}}.\mathsf{eval}(k_x, x') = \begin{cases} \mathsf{GGM}(k,x') & \textbf{if } x \text{ is a prefix of } x' \\ \bot & \textbf{otherwise}. \end{cases}$$

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

Definition 13. *The game is played between a challenger* G *(which is either* $\mathsf{G_L}$ *or* $\mathsf{G_R}$*) and an adversary* A *using* F*.* G *picks a random key* $k \leftarrow \mathcal{K}$*, and initialises a set* $\mathbf{x} = \emptyset$*.* 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* $\mathbf{x}$*.*
- *One challenge query* $(\texttt{challenge}, x^*)$*: Here the answer differs between* $\mathsf{G_L}$ *and* $\mathsf{G_R}$*:* $\mathsf{G_L}$ *answers with* $\mathsf{F.eval}(k,x^*)$ *(real output), whereas* $\mathsf{G_R}$ *answers with random* $r \leftarrow \mathcal{Y}$ *(fake, random output) — for the task to be non-trivial, no element in* $\mathbf{x}$ *must be a prefix of* x^**.* G *adds* x^* *to* $\mathbf{x}$*.*

In the fully selectivized version of Definition 13, A must commit to the to the whole set $\mathbf{x} := \{x_1, \ldots, x_{q-1}, x_q = x^*\}$. Therefore, the selective challenger is defined as $\mathsf{H_L} = \mathsf{SEL}_{\{0,1\}^{n\cdot q}}[\mathsf{G_L}, g]$ (resp., $\mathsf{H_R} = \mathsf{SEL}_{\{0,1\}^{n\cdot q}}[\mathsf{G_R}, g]$), where g is the function that extracts $\mathbf{x}$ from the transcript. Note that the amount of information that A commits to in the selectivised games is much more than the one defined in [FKPR14].

Definition 14. *A prefix-constrained PRF* F *is* (s,ε,q)*-adaptive-secure (resp., selective-secure) if* $\mathsf{G_L}$ *and* $\mathsf{G_R}$ *(resp.,* $\mathsf{H_L}$ *and* $\mathsf{H_R}$*) are* (s,ε)*-indistinguishable.*

B.2 Hybrids and Pebbling Configurations

B.2.1 Fully Selective Hybrids.

Let's recall briefly from §1.4 how we used the knowledge of $x_1, \ldots, x_{q-1}$ to get a better sequence of hybrids. First we switched to the recursive pebbling sequence in Figure 1.b. Second, we managed to shrink the index of the pebble from $[0, 2^n]$ to $[0, q]$ by assuming an upper bound q on the number of queries made by the adversary: we set the index of a pebble to the index of the first constrain query whose i bit prefix coincides with x^*. More formally, the index for a pebble on the ith edge is defined as

$$\mathsf{index}(\mathbf{x}, i) := \arg \min_{j \in [q-1]} \{x^*[i] = x_j[i]\}.$$

In particular, by definition of the security game, $\mathsf{index}(\mathbf{x}, n) = q$. The index of an edge-pebbling configuration is accordingly defined as

$$\mathsf{index}(\mathbf{x}, \mathcal{P}_I) := \{\text{index}_i : i \in \mathcal{P}_I, \text{index}_i = \mathsf{index}(\mathbf{x}, i)\}.$$

By using the edge-pebbling $\mathcal{P}_0, \ldots, \mathcal{P}_\ell$ generated by P_1 (Algorithm 4), where $\ell = 3^{\log n}$, we get a sequence of fully selective hybrids $\mathsf{H}(\mathcal{P}_0), \ldots, \mathsf{H}(\mathcal{P}_\ell)$, with H_I described in Algorithm 6, and the following lemma.

$\mathsf{H}^{\mathsf{A}}(\mathcal{P}_I)$

Obtain $\mathbf{x} \in \{0,1\}^{n \cdot q}$ from A
Compute **index** $:= \mathsf{index}(\mathbf{x}, \mathcal{P}_I)$
Sample key $k_\emptyset \leftarrow \mathcal{K}$, set $\forall x \in \{0,1\}^{\leq n} : k_x := \perp$ ▷ Initialise the keys
Initialise the counter $c = 1$
Whenever A makes a query $(\cdot, x)$:
Start
    if $(c = \text{index}_i \in \textbf{index})$ **then** set $k_{x[i]\|0} \| k_{x[i]\|1} \leftarrow U_{2m}$ ▷ Fake output
    Increment c by one
    return $\mathsf{K}(x)$ ▷ Compute the key using the sub-routine
End
return A's output

$\mathsf{K}(x)$:

if $k_x \neq \perp$ **then return** k_x ▷ Key already defined
Set $l = |x| - 1$, $k_{x[l]} = \mathsf{K}(x[l-1])$ ▷ Recursively compute the key
$k_{x[l-1]} \| k_{x[l-1]} := \mathsf{PRG}(k_{x[l]})$ ▷ Normal output
return k_x

Algorithm 6: Template for generating fully selective hybrids. The sub-routine $\mathsf{K}(x)$ computes the key k_x from the first key that has already been defined on the path from x to the root. We reuse this sub-routine in later part of the sections.

Lemma 6. *Let $\ell := 3^{\log n}$. The sequence of hybrids $\mathsf{H}(\mathcal{P}_0), \ldots, \mathsf{H}(\mathcal{P}_\ell)$ constitutes a valid sequence of fully selective hybrids. Furthermore, if the PRG is (s, ε)-indistinguishable then the constrained PRF $\mathsf{F}_{\mathsf{GGM}}$ is $(s - O(s_{\mathsf{P}_1} + qn \cdot s_{\mathsf{PRG}}), \varepsilon \cdot \ell, q)$-selective secure, where s_{P_1} (resp, s_{PRG}) denotes the complexity of P_1 (resp., PRG).*

Proof. By the properties of the pebbling configurations $\mathcal{P}_0$ and $\mathcal{P}_\ell$, it is easy to see that $\mathsf{H}(\mathcal{P}_0) \equiv \mathsf{H}_\mathsf{L}$ an $\mathsf{H}(\mathcal{P}_\ell) \equiv \mathsf{H}_\mathsf{R}$. In addition, the neighbouring hybrids $\mathsf{H}(\mathcal{P}_I)$ and $\mathsf{H}(\mathcal{P}_{I+1})$ are $(s - qn \cdot s_{\mathsf{PRG}}, \varepsilon, q)$-indistinguishable — see Algorithm 7 for a reduction. The lemma follows. □ □

$\mathsf{R}^{\mathsf{A}}(\mathcal{P}_I, \mathcal{P}_{I+1}, y)$ ▷ $y \in \{0,1\}^{2m}$ is the PRG challenge
1: Obtain $\mathbf{x} \in \{0,1\}^{n \cdot q}$ from A
2: **index** := $\mathsf{index}(\mathbf{x}, \mathcal{P}_I \cap \mathcal{P}_{I+1})$, $\mathrm{index}_{i^*} := \mathsf{index}(\mathbf{x}, \mathcal{P}_I \triangle \mathcal{P}_{I+1})$
3: Sample key $k_\emptyset \leftarrow \mathcal{K}$, set $\forall x \in \{0,1\}^{\leq n} : k_x := \perp$ ▷ Initialise the keys
4: Initialise the counter $c = 1$
5: Whenever A makes a query $(\cdot, x)$:
6: **Start**
7: **if** $(c = \mathrm{index}_{i^*})$ **then** set $k_{x[i^*]\|0} \| k_{x[i^*]\|1} := y$ ▷ Fake or real
8: **if** $(c = \mathrm{index}_i \in$ **index**$)$ **then** set $k_{x[i]\|0} \| k_{x[i]\|1} \leftarrow U_{2m}$ ▷ Fake output
9: Increment c by one
10: **return** $\mathsf{K}(x)$ ▷ Compute the key using the sub-routine
11: **End**
12: **return** A's output

Algorithm 7: The reduction algorithm that establishes the indistinguishability of $\mathsf{H}(\mathcal{P}_I)$ and $\mathsf{H}(\mathcal{P}_{I+1})$. (See Algorithm 6 for the description of K.)

Lemma 7. *The sequence of fully selective hybrids* $\mathsf{H}(\mathcal{P}_0), \ldots, \mathsf{H}(\mathcal{P}_{3^{\log n}})$ *are partially selective as*

$$\mathsf{H}(\mathcal{P}_{I+b}) = \mathsf{SEL}_{\{0,1\}^{2 \cdot \log n \cdot \log q} \to \{0,1\}^{n \cdot \log q}}[\hat{\mathsf{H}}_{I,b}, g, h_I], \tag{7}$$

where $\hat{\mathsf{H}}_{I,b}$ *is described in Algorithm 8, and* h_I *is the function that, on input the queries* $\mathbf{x}$*, computes the indices* $\mathbf{index}_I$ *and* $\mathbf{index}_{I+1}$ *of the pebbling configurations* $\mathcal{P}_I$ *and* $\mathcal{P}_{I+1}$*, respectively.*[18]

It follows from Theorem 2 and Lemma 7 that the prefix-constrained CPRF $\mathsf{F}_{\mathsf{GGM}}$ is adaptive-secure with a quasi-polynomial loss in tightness.

Theorem 6. *If the underlying PRG is* (s, ε)*-indistinguishable then* $\mathsf{F}_{\mathsf{GGM}}$ *is* $(s - O(s_{\mathsf{P}_1} + qn \cdot s_{\mathsf{PRG}}), \varepsilon \cdot 3^{\log n} \cdot n^{2 \cdot \log q})$*-adaptive-secure prefix-constrained PRF.*

of lemma. It remains to show that $\hat{\mathsf{H}}_0$ and $\hat{\mathsf{H}}_1$ are indistinguishable — the reduction is given in Algorithm 9. The lemma follows. □

[18] There are means to further compress h_I: it suffices that it returns the indices $\mathsf{index}(\mathbf{x}, \mathcal{P}_I \cap \mathcal{P}_{I+1})$ and $\mathsf{index}(\mathbf{x}, \mathcal{P}_I \triangle \mathcal{P}_{I+1})$, along with a bit b^* which indicates what the pebbling move is — i.e., if, in move $I+1$, the edge (i^*, i^*+1) was added to $\mathcal{P}_I$ then $b^* = 0$, otherwise $b^* = 1$. But we prefer the simpler h_I for the sake of simplicity of exposition.

$\hat{\mathsf{H}}_b^{\mathsf{A},h_I(\cdot)}(\mathcal{P}_I, \mathcal{P}_{I+1})$

Obtain $(\mathbf{index}_I, \mathbf{index}_{I+1})$ using $h_I(\cdot)$ from H
Sample key $k_\emptyset \leftarrow \mathcal{K}$, set $\forall x \in \{0,1\}^{\leq n} : k_x := \perp$ ▷ Initialise the keys
Initialise the counter $c = 1$
Whenever A makes a query $(\cdot, x)$:
Start
    if $(c = \text{index}_i \in \mathbf{index}_{I+b})$ **then** set $k_{x[i]\|0} \| k_{x[i]\|1} \leftarrow U_{2m}$ ▷ Faked
    Increment c by one
    return $\mathsf{K}(x)$ ▷ Compute the key using the sub-routine
End
return A's output

Algorithm 8: Partially selectivized hybrids $\hat{\mathsf{H}}_b$ for H_b. (See Algorithm 6 for the description of K.)

$\mathsf{R}'^{\mathsf{A},h_I(\cdot)}(\mathcal{P}_I, \mathcal{P}_{I+1}, y)$ ▷ $y \in \{0,1\}^{2m}$ is the PRG challenge

Obtain $(\mathbf{index}_I, \mathbf{index}_{I+1})$ using $h_I(\cdot)$ from A
Let $\text{index}_{i^*} := \mathbf{index}_I \triangle \mathbf{index}_{I+1}$, $\mathbf{index} := \mathbf{index}_I \cap \mathbf{index}_{I+1}$
Sample key $k_\emptyset \leftarrow \mathcal{K}$, set $\forall x \in \{0,1\}^{\leq n} : k_x := \perp$ ▷ Initialise the keys
Initialise the counter $c = 1$
Whenever A makes a query $(\cdot, x)$:
Start
    if $(c = \text{index}_{i^*})$ **then** set $k_{x[i^*]\|0} \| k_{x[i^*]\|1} := y$ ▷ Fake or real output
    if $(c = \text{index}_i \in \mathbf{index})$ **then** set $k_{x[i]\|0} \| k_{x[i]\|1} \leftarrow U_{2m}$ ▷ Fake output
    Increment c by one
    return $\mathsf{K}(x)$ ▷ Compute the key using the sub-routine
End
return A's output

Algorithm 9: The reduction algorithm that establishes the indistinguishability of the partial hybrids $\hat{\mathsf{H}}_0$ and $\hat{\mathsf{H}}_1$. (See Algorithm 6 for the description of K.)