A Quasipolynomial Reduction for Generalized Selective Decryption on Trees

Georg Fuchsbauer[1,⋆], Zahra Jafargholi[2], and Krzysztof Pietrzak[1,⋆]

[1] Institute of Science and Technology Austria
{gfuchsbauer|pietrzak}@ist.ac.at
[2] Northeastern University, USA
z.jafargholi@gmail.com

Abstract. Generalized Selective Decryption (GSD), introduced by Panjwani [TCC'07], is a game for a symmetric encryption scheme Enc that captures the difficulty of proving adaptive security of certain protocols, most notably the Logical Key Hierarchy (LKH) multicast encryption protocol. In the GSD game there are n keys $k_1, \ldots, k_n$, which the adversary may adaptively corrupt (learn); moreover, it can ask for encryptions $\mathsf{Enc}_{k_i}(k_j)$ of keys under other keys. The adversary's task is to distinguish keys (which it cannot trivially compute) from random. Proving the hardness of GSD assuming only IND-CPA security of Enc is surprisingly hard. Using "complexity leveraging" loses a factor exponential in n, which makes the proof practically meaningless.

We can think of the GSD game as building a graph on n vertices, where we add an edge $i \to j$ when the adversary asks for an encryption of k_j under k_i. If restricted to graphs of depth ℓ, Panjwani gave a reduction that loses only a factor exponential in ℓ (not n). To date, this is the only non-trivial result known for GSD.

In this paper we give almost-polynomial reductions for large classes of graphs. Most importantly, we prove the security of the GSD game restricted to trees losing only a quasi-polynomial factor $n^{3\log n+5}$. Trees are an important special case capturing real-world protocols like the LKH protocol. Our new bound improves upon Panjwani's on some LKH variants proposed in the literature where the underlying tree is not balanced. Our proof builds on ideas from the "nested hybrids" technique recently introduced by Fuchsbauer et al. [Asiacrypt'14] for proving the adaptive security of constrained PRFs.

1 Introduction

Proving security of protocols where an adversary can make queries and/or corrupt players *adaptively* is a notoriously hard problem. Selective security, where the adversary must commit to its queries before the protocol starts, often allows for an easy proof, but in general does not imply (the practically relevant) adaptive security notion [CFGN96].

⋆ Supported by the European Research Council, ERC Starting Grant (259668-PSPC).

Panjwani [Pan07] argues that the two common approaches to achieving adaptive security, namely requiring that all parties erase past data [BH93], or using *non-committing* encryption [CFGN96] are not satisfactory. He introduces the *generalized selective decryption* (GSD) problem and uses it as an abstraction of security requirements of multicast encryption protocols [WGL00,MP06]. GSD is defined by a very simple game that captures the difficulty of proving adaptive security of some interesting protocols.

The generalized selective decryption (GSD) game. In the GSD game we consider a symmetric encryption scheme Enc and a parameter $n \in \mathbb{N}$. Initially, we sample n random keys $k_1, \ldots, k_n$ and a bit $b \in \{0,1\}$. During the game the adversary A can make two types of queries. Encryption query: on input (i,j) she receives $c = \mathsf{Enc}_{k_i}(k_j)$; corruption query: on input i, she receives k_i. At some point, A chooses some i to be challenged on. If $b = 0$, she gets the key k_i; if $b = 1$, she gets a uniformly random r_i.[3] Finally, A outputs a guess bit b'. The goal is prove that for any efficient A, $|\Pr[b = b'] - 1/2|$ is negligible (or, equivalently, k_i is pseudorandom) assuming only that Enc is a secure encryption scheme. We only allow one challenge query, but this notion is equivalent to allowing any number of challenge queries by a standard hybrid argument (losing a factor that is only the number of challenge queries).

It is convenient to think of the GSD game as dynamically building a graph, which we call key graph. We start with a graph with n vertices labeled $1, \ldots, n$, where we associate vertex i with key k_i. On an encryption query $\mathsf{Enc}_{k_i}(k_j)$ we add a directed edge $i \to j$. On a corruption query i we label the vertex i as corrupted. Note that if i is corrupted then A also learns all keys k_j for which there is a path from i to j in the key graph by simply decrypting the keys along that path. To make the game non-trivial, challenge queries are thus only allowed for keys that are not reachable from any corrupted key. Another restriction we must make is to disallow encryption cycles, i.e., loops in the graph. Otherwise we cannot hope to prove security assuming only standard security (in our case IND-CPA) of the underlying encryption scheme, as this would require circular (or key-dependent-message) security [BRS03], which is stronger than IND-CPA [ABBC10]. Finally, we require that the challenge query is a leaf in the graph; this restriction too is necessary unless we make additional assumptions on the underlying encryption scheme (cf. Footnote 11).

Selective security of GSD. In order to prove security of the GSD game, one must turn an adversary A that breaks the GSD game with some advantage $\epsilon = |\Pr[b = b'] - 1/2|$ into an adversary B that breaks the security of Enc with

[3] Below, we will consider a (seemingly) different experiment and output k_i in both cases ($b = 0$ and $b = 1$), but if $b = 1$, then on any query (j,i), we will encrypt $\mathsf{Enc}_{k_j}(r_i)$ and not $\mathsf{Enc}_{k_j}(k_i)$. This is just a semantic change assuming the following: during the experiment we always answer encryption queries of the form (a,b) with $\mathsf{Enc}_{k_a}(k_b)$ (note that we don't know if we're encrypting the challenge at this point), and once the adversary chooses a challenge i, if $b = 1$, we simply switch the values of r_i and k_i (this trick is already used in [Pan07]).

some advantage $\epsilon' = \epsilon'(\epsilon)$. The security notion we consider is the standard notion of indistinguishability under chosen plaintext attacks (IND-CPA). Recall that in the IND-CPA game an adversary B is given access to an encryption oracle $\mathsf{Enc}_k(\cdot)$. At some point B chooses a pair of messages (m_0, m_1), then gets a challenge ciphertext $c = \mathsf{Enc}_k(m_b)$ for a random bit b, and must output a guess b'. The advantage of B is $|\Pr[b = b'] - 1/2|$.

It is not at all clear how to construct an adversary B that breaks IND-CPA from an A that breaks GSD. This problem becomes much easier if we assume that A breaks the *selective* security of GSD, where A must choose all its encryption, corruption and challenge queries before the experiment starts.

In fact, it is sufficient to know the topology of the connected component in the key graph that contains the challenge node. Let α denote the number of edges in this component. One can now define a sequence of 2α hybrid games $H_0, \ldots, H_{2\alpha-1}$, where the first game is the real game (i.e., the GSD game with $b = 0$ where the adversary gets the key), the last hybrid is the random game $(b = 1)$, and moreover, from any adversary that distinguishes H_i from H_{i+1} with some advantage ϵ', we get an adversary against the IND-CPA security of Enc with the same advantage. Thus, given an A breaking GSD with advantage ϵ, we can break the IND-CPA security with advantage $\epsilon' \geq \epsilon/(2\alpha - 1) \geq \epsilon/n^2$ (as an n vertex graph has $\leq n^2$ edges). We illustrate this reduction in Fig. 1.

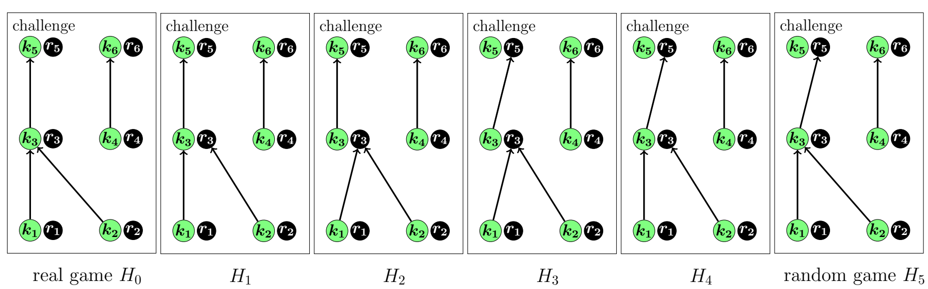


Fig. 1: Hybrids for the selective security proof. Green nodes correspond to keys, dark nodes are random values. The adversary A commits to encryption queries $(1, 3)$, $(2, 3)$, $(3, 5)$ and challenge 5 (Encryption query $(4, 6)$ is outside the connected component containing the challenge and thus not relevant for the hybrids. A could also corrupt keys 4 and 6, which are also outside.) Hybrid H_0 is the real game, hybrid H_5 is the random game, where instead of an encryption of the challenge key $\mathsf{Enc}_{k_3}(k_5)$, the adversary gets an encryption of the random value $\mathsf{Enc}_{k_3}(r_5)$. If an adversary A can distinguish any two consecutive hybrids H_i and H_{i+1} with some advantage δ, we can use A to construct B which breaks the IND-CPA security of Enc with the same advantage δ: E.g., assume B is given an IND-CPA challenge $C = \mathsf{Enc}_k(z)$ where z is one of two messages (which we call k_5 and r_5). Now B can simulate game H_2 for A, but when A makes the encryption query $(3, 5)$, B answers with C. If $z = k_5$ then B simulates game H_2; but if $z = r_5$, it simulates game H_3. Note that B can simulate the games because k_3, which in the simulation is B's challenger's key, is not used anywhere else. Thus, B has the same advantage in the IND-CPA game as A has in distinguishing H_3 from H_4.

Adaptive security of GSD. In the selective security proof for GSD we crucially relied on the fact that we knew the topology of the underlying key graph. Proving adaptive security, where the adversary decides what queries to ask adaptively during the experiment, is much more difficult. A generic trick to prove adaptive security is "complexity leveraging", where one simply turns an adaptive adversary into a selective one by initially guessing the adaptive adversary's choices and committing to those (as required by the selective security game). If during the security game the adaptive choices by the adversary disagree with the guessed ones, we simply abort. The problem with this approach is that assuming the adaptive adversary has advantage ϵ, the constructed selective adversary only has advantage ϵ/P where $1/P$ is the probability of that our guess is correct, which is typically exponentially small. Concretely, in the GSD game we need to guess the nodes in the connected component containing the challenge, and as the number of such choices is exponential in the number of keys n, this probability is $2^{-\Theta(n)}$.

No proofs for the adaptive security of GSD with a subexponential (in n) security loss are known in general. But remember that the GSD problem abstracts problems we encounter in proving adaptive security of many real-world applications where the underlying key graph is typically not completely arbitrary, but often has some special structure. Motivated by this, Panjwani [Pan07] investigated better reductions assuming some special structure of the key graph. He gives a proof where the security degradation is only exponential in the *depth* of the key graph, as opposed to its size. Concretely, he proves that if the encryption scheme is ϵ-IND-CPA secure then the adaptive GSD game with n keys where the adversary is restricted to key graphs of depth ℓ is ϵ'-secure where

$$\epsilon' = \epsilon \cdot O(n \cdot (2n)^{\ell}) \ .$$

Until today, Panjawain's bound is the only non-trivial improvement over the $2^{\Theta(n)}$ loss for GSD.

Our result. The main result of this paper is Theorem 2, which states that GSD restricted to trees can be proven secure with only a quasi-polynomial loss

$$\epsilon' = \epsilon \cdot n^{3\log(n)+5} \ .$$

Our bound is actually even stronger as the entire key graph need not be a tree; it is sufficient that the subgraph containing only the nodes from which the challenge node can be reached is a tree (when ignoring edge directions).

The bound above is derived from a more fine-grained bound: assuming that the longest path in the key graph is of length ℓ, the in-degree of every node is at most d and the challenge node can be reached from at most s sources (i.e., nodes with in-degree 0) we get

$$\epsilon' = \epsilon \cdot dn((2d+1)n)^{\lceil \log s \rceil} (3n)^{\lceil \log \ell \rceil} \ .$$

Note that ℓ, d and s are at most n and the previous bound was derived from this by setting $\ell = d = s = n$. Panjwani [Pan07] uses his bound to give a quasi-polynomial reduction of the Logical Key Hierarchy (LKH) protocol [WGL00].

Panjwani first fixes a flaw in LKH, and calls the new protocol rLKH with "r" for repaired. rLKH is basically the GSD game restricted to a binary tree.[4]

The users correspond to the leaves of this tree, and their keys consists of all the nodes from the root to their leaf. Thus, if the tree is almost full and balanced, then it has only depth $\ell \approx \log n$ and Panjwani's bound loses only a quasi-polynomial factor $n^{\log(n)+2}$ (if $\ell = \log n$). As here $d = 2, \ell = \log n, s = n$, our bound gives a slightly worse bound $n^{\log(n)+\log\log(n)+4}$ for this particular problem, but this is only the case if a large fraction of the keys are actually used, and the adversary gets to see almost all of them. If ℓ is significantly larger than $\log n$ (e.g., because only few of the keys are active, or the tree is constructed in an unbalanced way like e.g. proposed in [SS00]), our bounds decrease only marginally, as opposed to exponentially fast in ℓ in [Pan07].

Graphs with small cut-width. The reason our result is restricted to trees is that in the process of generating the hybrids, we have to guess nodes such that removing this node splits the tree in a "nice" way (this has to be done $\log n$ times, losing a factor n in the distinguishing advantage every time).

One can generalize this technique (but we do not work out the details in this paper) to graphs with small "cut-width", where we say that a graph has cut-width w if for any two vertices u, v that are not connected by an edge, there exists a set of at most w vertices such that removing those disconnects u from v (a tree has cut-width $w = 1$). For graphs with cut-width w we get

$$\epsilon' = \epsilon \cdot n^{(2w+1)\log(n)+4} \quad ,$$

which is subexponential in n, and thus beats the existing exponential bound whenever $w = o(n/\log^2(n))$. Whether there exists a subexponential reduction which works for any graph is an intriguing open problem.

Shorter keys from better reduction. An exponential security loss (as via complexity leveraging) means that, even when assuming exponential hardness of Enc (which is a typical assumption for symmetric encryption schemes like AES), one needs to use keys for Enc whose length is at least linear in n to get any security guarantee for the hardness of GSD at all. Whereas our bound for trees means that a key of length *polylog*(n) is sufficient to get asymptotically overwhelming security (again assuming Enc is exponentially hard).

Nested hybrids. In a classical paper [GGM86] Goldreich, Goldwasser and Micali constructed a pseudorandom function (PRF) from a pseudorandom generator (PRG). More recently, three papers independently [BW13,KPTZ13,BGI14] observed that this construction is also a so-called *constrained* PRF, where for every string x one can compute a constrained key k_x that allows evaluation of

[4] Let us stress that the graph obtained when just adding an edge for every encryption query in rLKH is not a tree after a rekeying operation. But for every node v, the subgraph we get when only keeping the nodes from which v can be reached is a tree, and as explained above, this is sufficient.

the PRF on all inputs with prefix x. Informally, the security requirement is that an adversary that can ask for constrained keys cannot distinguish the output of the PRF on some challenge input from random.

All three papers [BW13,KPTZ13,BGI14] only prove *selective* security of this constrained PRF, where before any queries the adversary must commit to the input on which it wants to be challenged. This proof is a hybrid argument losing a factor $2m$ in the distinguishing advantage, where m is the PRF input length. One can then get adaptive security losing a huge exponential factor 2^m via complexity leveraging. Subsequently, Fuchsbauer et al. [FKPR14] gave a reduction that only loses a quasi-polynomial factor $(3q)^{\log m}$, where q denotes the number of queries made by the adversary. Our proofs borrows ideas from their work.

Very informally, the idea behind their proof is the following. In the standard proof for adaptive security using leveraging one first guesses the challenge query (losing a huge factor 2^m), which basically turns the adaptive attacker into a selective one, followed by a simple hybrid argument (losing a small factor $2m$) to prove selective security. The proof from [FKPR14] also first makes a guessing step, but a much simpler one, namely which of the q queries made by the adversary is the first to coincide with the challenge query on the first $m/2$ bits. This is followed by a hybrid argument losing a factor 3, so both steps together lose a factor $3q$. At this point the reduction is not finished yet, but intuitively the problem was reduced to itself but on inputs of only half the size $m/2$. These two steps can be iterated $\log m$ times (losing a total factor of $(3q)^{\log m}$) to get a reduction to the security of the underlying PRG.

Proof outline for paths. Our proof for GSD uses an approach similar to the one just explained, iterating fairly simple guessing steps with hybrid arguments, but the analogy ends here, as the actual steps are very different.

We first outline the proof for the adaptive security of the GSD game for a special case where the adversary is restricted in the sense that the connected component in the key graph containing the challenge must be a path. Even for this very special case, currently the best reduction [Pan07] loses an *exponential* factor $2^{\Theta(n)}$. We will now outline a reduction losing only a *quasi-polynomial* $n^{\log n}$ factor.[5] Recall that the standard way to prove adaptive security is to first guess the entire connected component containing the challenge, and then prove selective security as illustrated in Fig. 1.

Our approach is not to guess the entire path, but in a first step only the node in the middle of the path (as we make a uniform guess, it will be correct with probability $1/n$). This reduces the adaptive security game to a "slightly

[5] Let us mention that it is trivial to prove security of GSD restricted to paths if we additionally assume that for random keys k, k' the ciphertext $\mathsf{Enc}_k(k')$ is uniform given k' (this is e.g. the case for one-time pad encryption $\mathsf{Enc}_k(k') = k \oplus k'$): then the real and random challenge have the same distribution (they're uniform) and thus even a computationally unbounded adversary has zero advantage. (This is because in the path case, every key is used only once to encrypt.) The proof we outline here does not require this special property of Enc, and this will be crucial to later generalize it to more interesting graphs.

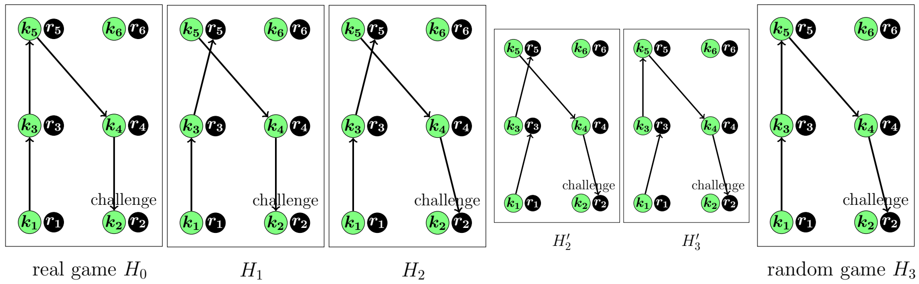


Fig. 2: Illustration of our adaptive security proof for paths.

selective" game where the adversary must commit initially to this middle node, at the price of losing a factor n in the distinguishing advantage.[6]

Let H_0 and H_3 denote these "slightly selective" real and random GSD games (we also assume that the adversary initially commits to the challenge query, which costs another factor of n). We illustrate this with a small example featuring a path of length 4 in Fig. 2. The correct guess for the middle node for the particular run of the experiment illustrated in the figure is $i = 5$. As now we know the middle vertex is $i = 5$, we can define new games H_1 and H_2 which are derived from H_0 and H_3, respectively, by replacing the ciphertext $\mathsf{Enc}_{k_j}(k_i)$ with an encryption $\mathsf{Enc}_{k_j}(r_i)$ of a random value (in the figure this is illustrated by replacing the edge $k_j \to k_i$ with $k_j \to r_i$).

So, what have we gained? If our adaptive adversary has advantage ϵ in distinguishing the real and random games then she has advantage at least ϵ/n to distinguish the "slightly selective" real and random games H_0 and H_3, and thus for some $i \in \{0,1,2\}$ she can distinguish the games H_i and H_{i+1} with advantage $\epsilon/3n$. Looking at two consecutive games H_i and H_{i+1}, we see that they only differ in one edge (e.g., in H_2 we answer the query $(3,5)$ with $\mathsf{Enc}_{k_3}(r_5)$, in H_3 with $\mathsf{Enc}_{k_3}(k_5)$), and moreover this edge will be at the end of a path that now has only length 2, that is, half the length of the path in our original real and random games.

We can now continue this process, constructing new games where the path length is halved, paying a factor $3n$ in distinguishing advantage. For example, as illustrated in Fig. 2, we can guess the node that halves the path leading to

[6] We never actually construct this "slightly selective" adversary, but (as in complexity leveraging) we simply commit to a random guess, then run the adaptive adversary, and if its queries are not consistent with our guess, we abort outputting a random value. (We could also output a constant value; the point is that the advantage of the adversary, conditioned on our guess being wrong, is zero; whereas, conditioned on the guess being correct, it is the same as the advantage of the adaptive adversary). However, instead of this experiment it is easier to follow our proof outline by thinking of the adversary actually committing to its choices initially, but the reduction paying a factor (in the distinguishing advantage of the adversary that is allowed to make this choice adaptively) that corresponds to the size of the sample space of this guess.

the differing query in games H_2 and H_3 (for the illustrated path this would be $i = 3$), then define new games where we assume the adversary commits to this node (paying a factor n), and then define two new games H_2' and H_3', which are derived from games H_2 and H_3 (which now are augmented by our new guess), respectively, by answering the query (j, i) that asks for an encryption of this node (in the figure $(j, i) = (1, 3)$) with an encryption $\mathsf{Enc}_{k_1}(r_3)$ instead of $\mathsf{Enc}_{k_1}(k_3)$.

If we start with a path of length $\ell \leq n$ then after $\log \ell \leq \log n$ iterations of this process we proved the existence of two consecutive games (call them G_0 and G_1) that differ only in a single edge $j \to i$ and the vertex j has in-degree 0. That is, both games are identical, except that in one game the encryption query (j, i) is answered with $\mathsf{Enc}_{k_j}(k_i)$ and in the other with $\mathsf{Enc}_{k_j}(r_i)$. Moreover, the key k_j is not used anywhere else in the experiment and we know exactly when this query is made during the experiment (as the adversary committed to i).

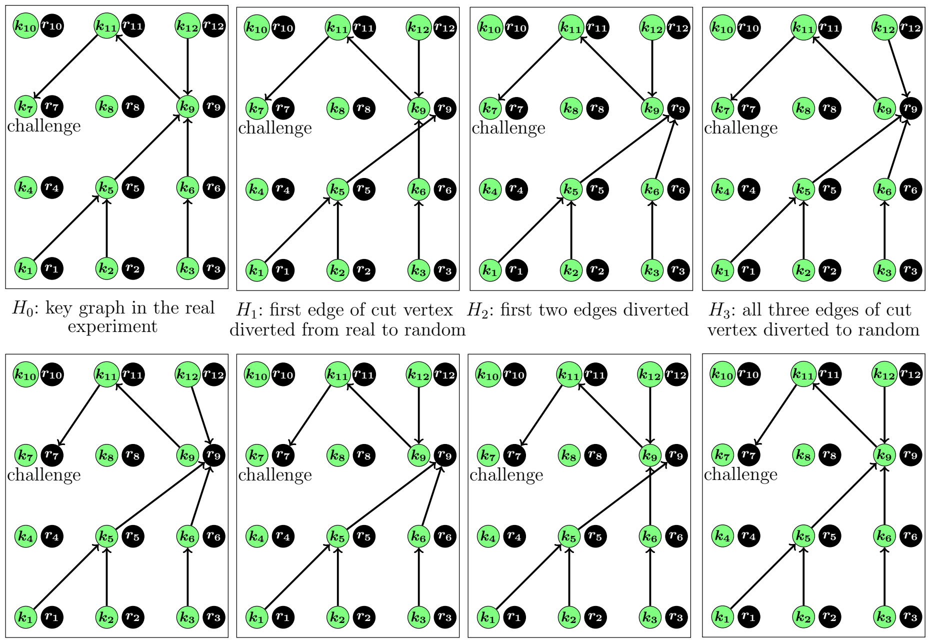
Given a distinguisher A for G_0 and G_1, we can now construct an attacker B that breaks the IND-CPA security of the underlying encryption scheme with the same advantage: in the IND-CPA game B chooses two random messages m_0, m_1 and asks to be challenged on them.[7] The game samples a random bit b and returns the challenge $C = \mathsf{Enc}_k(m_b)$ to B, which must then output a guess b' for b. At this point, B invokes A and simulates the game G_0 for it, choosing all keys at random, except that it uses C to answer the encryption query (j, i).[8] Finally, B forwards A's guess b'. Identifying (k, m_0, m_1) with (k_j, k_i, r_i), we see that depending on whether $b = 0$ or $b = 1$, B simulates either G_0 or G_1. Thus, whatever advantage A has in distinguishing G_0 from G_1, B will break the IND-CPA security of Enc with the same advantage.

Proof outline for trees. We will now outline our reduction of the adaptive security of GSD to the IND-CPA security of Enc for a more general case. Namely, the adversary is only restricted in that the key graph resulting from its queries is such that the connected component containing the challenge is a tree. (Recall that we already disallowed cycles in the key graph as this would require circular security. Being a tree means that we also have no cycles in the key graph when ignoring edge directions). Note that paths as discussed in the previous section are very special trees. The GSD problem on trees is particularly interesting, as it captures some multicast encryption protocols like the Logical Key Hierarchy (LKH) protocol [WGL00]. We refer the reader to [Pan07] for details.

TREES WITH IN-DEGREES ≤ 1. Let us first consider the case where the connected component containing the challenge is a tree, and moreover all its vertices have in-degree 0 or 1. It turns out that the proof outlined for paths goes through with only minor changes for such trees. Note that such a tree has exactly one vertex with in-degree 0, which we call the *root*, and there is a unique path from the root to the challenge node. We can basically ignore all the edges not on this

[7] Note that B makes no encryption queries at all (which are allowed by the IND-CPA experiment).

[8] Note that since node j has in-degree 0, we can identify k_j with the key k used by the IND-CPA experiment, as we never have to encrypt k_j.



H_0: key graph in the real experiment

H_1: first edge of cut vertex diverted from real to random

H_2: first two edges diverted

H_3: all three edges of cut vertex diverted to random

H_4: divert challenge from real k_7 to random r_7

H_5: divert first edge of cut vertex back to real

H_6: divert back second edge

H_7: key graph in the random experiment

Fig. 3: Illustration of our adaptive security proof for general trees.

path and do a reduction as the one outlined above. The only difference is that now, when simulating the game G_b (where b is 0 or 1 depending on the whether the challenge C with which we answer the encryption query (j, i) is $\mathsf{Enc}_{k_j}(k_i)$ or $\mathsf{Enc}_{k_j}(r_i)$), the adversary can also ask for encryption queries (j, x) for any x. This might seem like a problem as we do not know k_j (we identified k_j with the key used by the IND-CPA challenger). But recall that in the IND-CPA game there is an encryption oracle $\mathsf{Enc}_{k_j}(\cdot)$, which we can query for the answer $\mathsf{Enc}_{k_j}(k_x)$ to such encryption queries.

General Trees. For general trees, where nodes can have in-degree greater than 1, we need to work more. The proof for paths does not directly generalize, as now nodes (in particular, the challenge) can be reached from more than one node with in-degree 0. We call these the *sources* of this node; for example in the tree H_0 in Fig. 3, the (challenge) node k_7 has 4 sources k_1, k_2, k_3 and k_{12}.

On a high level, our proof strategy will be to start with a tree where the challenge node c has s sources (more precisely, we have two games that differ in one edge that points to k_i in one game, and to r_i in the other, like games H_0 and H_7 in Fig. 3). We then guess a node v that "splits" the tree in a nice way, by which we mean the following: Assume v has in-degree d and we divert every edge going into v to a freshly generated node; let's call them $v_1, \ldots, v_d$. Then this splits the tree into a forest consisting of $d+1$ trees (the component containing the challenge and one component for every v_i). The node v "well-divides" the tree if after the split the node c and all of $v_1, \ldots, v_d$ have at most $\lceil s/2 \rceil$ sources.

As an example, consider again the tree H_0 in Fig. 3, where the challenge node k_7 has 4 sources. The node k_9 would be a good guess, as it well-divides the tree: consider the forest after splitting at this node as described above (creating new nodes v_1, v_2, v_3 and diverting the edges going into k_9 to them, i.e., replacing $k_5 \to k_9$ by $k_5 \to v_1$, $k_6 \to k_9$ by $k_6 \to v_2$, and $k_{12} \to k_9$ by $k_{12} \to v_3$). Then we obtain 4 trees, where now $c = k_7$ has only one source (k_9) and the new nodes v_1, v_2, v_3 have $2, 1$ and 1 sources, respectively.

Once we have guessed a well-dividing node v (or equivalently, the adversary has committed to such a node), we define $2d$ hybrid games (where d is the degree of the well-dividing node) between the two initial games, which we call H_0 and H_{2d+1}, as follows. H_1 is derived from H_0 by diverting the first encryption query that asks for an encryption of v (i.e., that is of the form (j, v) for some j) from real to random; that is, we answer with $\mathsf{Enc}_{k_j}(r_v)$ instead of $\mathsf{Enc}_{k_j}(k_v)$. For $i \leq d$, H_i is derived from H_0 by diverting the first i encryption queries. H_{d+1} is derived from H_d by diverting the encryption query that asks for an encryption of the challenge c from real to random. The final $d-1$ hybrids games are used to switch the encryption of v back from random to real, one edge at a time. This process is illustrated in the games H_0 to H_7 in Fig. 3.

Because v was well-dividing (and we show in the full version that such a node always exists), we can prove the following property for any two consecutive games H_i and H_{i+1}: they differ in exactly one edge, which for some j, v in one game is $k_j \to k_v$ and $k_j \to r_v$ in the other, and moreover, k_j has at most $\lceil s/2 \rceil$ sources.

If an adversary can distinguish H_0 and H_{2d+1} with advantage ϵ then it must distinguish two hybrids H_i and H_{i+1} with advantage $\epsilon/((2d+1)n)$ (where n accounts for guessing the well-dividing node). But any such two hybrids now only have at most $\lceil s/2 \rceil$ sources. If we repeat this guessing/hybrid steps $\log s$ times, we end up with two games G_0 and G_1 which differ in one edge that has only one source. At this point we can then use our reduction for trees with only one source outlined above.

Analyzing the Security Loss. To halve the number of sources, we guess a well-dividing vertex (which costs a factor n in the reduction), and then must add up to $2d$ intermediate hybrids (where d is the maximum in-degree of any node), costing another factor $2d+1$. Assuming that the number of sources is bounded by s, we have to iterate the process at most $\log s$ times. Finally, we lose another factor d (but only once) because our final node can have more than one ingoing edge. Overall, assuming the adversary breaks the GSD game with advantage ϵ on trees with at most s sources and in-degree at most d, our reduction yields an attacker against the IND-CPA security of Enc with advantage

$$\epsilon / \, dn((2d+1)n)^{\lceil \log s \rceil} (3n)^{\lceil \log \ell \rceil} \; .$$

For general trees, since $s, d \leq n$, we have $\epsilon / \, n^{3 \log n + 5}$.

2 Preliminaries

For $a \in \mathbb{N}$, we let $[a] = \{1, 2, \ldots, a\}$ and $[a]_0 = [a] \cup \{0\}$. We say adversary (or distinguisher) D is t-bounded if D runs in time t.

Definition 1. ***(Indistinguishability)*** *Two distributions X and Y are (ϵ, t)-indistinguishable, denoted $Y \sim_{(\epsilon,t)} X$ or $\Delta_t(Y, X) \leq \epsilon$, if no t-bounded distinguisher D can distinguish them with advantage greater than ϵ, i.e.,*

$$\Delta_t(Y, X) \leq \epsilon \iff \forall\, \mathsf{D}_t : \big| \Pr[\mathsf{D}_t(X) = 1] - \Pr[\mathsf{D}_t(Y) = 1] \big| \leq \epsilon \ .$$

Symmetric encryption. A pair of algorithms $(\mathsf{Enc}, \mathsf{Dec})$ with input $k \in \{0,1\}^\lambda$, where λ is the security parameter, and a message m (or a ciphertext) from $\{0,1\}^*$ is a symmetric-key encryption scheme if for all k, m we have $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$. Consider the game $\mathbf{Exp}^{\text{IND-CPA-}b}_{\mathsf{Enc},\,\mathsf{D}}$ between a challenger C and a distinguisher D: C chooses a uniformly random key $k \in \{0,1\}^\lambda$ and a bit $b \in \{0,1\}$; D can make encryption queries for messages m and receives $\mathsf{Enc}_k(m)$; finally, D outputs a pair (m_0, m_1), is given $\mathsf{Enc}_k(m_b)$ and outputs a bit $b' \in \{0,1\}$, which is also the output of $\mathbf{Exp}^{\text{IND-CPA-}b}_{\mathsf{Enc},\,\mathsf{D}}$.[9]

Definition 2. *Let $t \in \mathbb{N}^+$ and $0 < \epsilon < 1$. An encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ is (t, ϵ)-IND-CPA secure if for any t-bounded distinguisher D, we have*

$$\big| \Pr\big[\mathbf{Exp}^{\text{IND-CPA-}1}_{\mathsf{Enc},\,\mathsf{D}} = 1\big] - \Pr\big[\mathbf{Exp}^{\text{IND-CPA-}0}_{\mathsf{Enc},\,\mathsf{D}} = 1\big] \big| \leq \epsilon \ .$$

3 The GSD Game

In this section we describe the generalized selective decryption game as defined in [Pan07] and give our main theorem. Consider the following game, $\mathbf{Exp}^{\text{GSD-}(n,\,b)}_{\mathsf{Enc},\,\mathsf{A}}$ called the generalized selective decryption (GSD) game, parameterized by an encryption scheme Enc,[10] an integer n and a bit b. It is played by the adversary A and the challenger B. First B samples n keys $k_1, k_2, \ldots, k_n$ uniformly at random from $\{0,1\}^\lambda$. A can make three types of queries during the game:

- $\mathsf{encrypt}$: A query of the form $\mathsf{encrypt}(i, j)$ is answered with $c \leftarrow \mathsf{Enc}_{k_i}(k_j)$.
- $\mathsf{corrupt}$: A query of the form $\mathsf{corrupt}(i)$ is answered with k_i.
- $\mathsf{challenge}$: The response to $\mathsf{challenge}(i)$ depends on the bit b: if $b = 0$, the answer is k_i; if $b = 1$, the answer is a random value $r_i \in \{0,1\}^\lambda$.

[9] For this notion to be satisfied, Enc must be probabilistic. In this paper one may also consider deterministic encryption, in which case the security definition must explicitly require that the challenge messages are fresh in the sense that D has not asked for encryptions of them already.

[10] We will never actually use the decryption algorithm Dec in the game, and thus will not mention it explicitly.

A can make multiple queries of each type, adaptively and in any order. It can also make several challenge queries at any point in the in the game. Allowing multiple challenge queries models the fact that the respective keys are jointly pseudorandom (as opposed to individual keys being pseudorandom by themselves). Allowing to interleave challenges with other queries models that they remain pseudorandom even after corrupting more keys or seeing further ciphertexts.

We can think of the n keys that B creates as n vertices, labeled $1, 2, \ldots, n$, in a graph. In the beginning of the game there are no edges, but every time A queries $\mathsf{encrypt}(i, j)$, we add the edge $i \to j$ to the graph. When A queries $\mathsf{corrupt}(i)$ for some $i \in [n]$, we mark i as a corrupt vertex; when A queries $\mathsf{challenge}(i)$, we mark it as a challenge vertex. For an adversary A we call this graph the *key graph*, denoted $G(\mathsf{A})$ and we write $V^{\mathrm{corr}}(\mathsf{A})$ and $V^{\mathrm{chal}}(\mathsf{A})$ for the sets of corrupt and challenge nodes, respectively. (Note that $G(\mathsf{A})$ is a random variable depending on the randomness used by A and its challenger.)

Legitimate adversaries. Consider an adversary that corrupts a node i in $G(\mathsf{A})$ and queries $\mathsf{challenge}(j)$ for some j which is reachable from i. Then A can successively decrypt the keys on the path from i to j, in particular k_j, and thus deduce the bit b. We only consider non-trivial breaks and require that no challenge node is reachable from a corrupt node in $G(\mathsf{A})$.

Two more restrictions must be imposed on $G(\mathsf{A})$ if we only want to assume that Enc satisfies IND-CPA. First, we do not allow key cycles, that is, queries yielding

$$\mathsf{Enc}_{k_{i_1}}(k_{i_2}), \mathsf{Enc}_{k_{i_2}}(k_{i_3}), \ldots, \mathsf{Enc}_{k_{i_{s-1}}}(k_{i_s}), \mathsf{Enc}_{k_s}(k_{i_1}) \enspace ,$$

as this would require the scheme to satisfy key-dependent-message (a.k.a. circular) security [BRS03,CL01].

Second, IND-CPA security does not imply that keys under which one has seen encryptions of random messages remain pseudorandom.[11] Pseudorandomness of keys (assuming only IND-CPA security of the underlying scheme) can thus only hold if their corresponding node does not have any outgoing edges. We thus require that all challenge nodes in the key graph are sinks (i.e., their out-degree is 0). The requirements (as formalized also in [Pan07]) are summarized in the following.

Definition 3. *An adversary* A *is* legitimate *if in any execution of* A *in the GSD game the values of* $G(\mathsf{A})$, $V^{corr}(\mathsf{A})$ *and* $V^{chal}(\mathsf{A})$ *are such that:*

- *For all* $i \in V^{corr}(\mathsf{A})$ *and* $j \in V^{chal}(\mathsf{A})$*:* j *is unreachable from* i *in* $G(\mathsf{A})$.
- $G(\mathsf{A})$ *is a directed acyclic graph (DAG) and every node in* $V^{chal}(\mathsf{A})$ *is a sink.*

[11] Consider any IND-CPA-secure scheme ($\mathsf{Enc}, \mathsf{Dec}$) and define a new scheme as follows: keys are doubled in length and encryption under $k = k_1||k_2$ is defined as $\mathsf{Enc}_k(m) = \mathsf{Enc}_{k_1}(m)||k_2$. This scheme is still IND-CPA, but given a ciphertext $C = \mathsf{Enc}_k(m)$ one can easily distinguish k from a random value even if m is random and unknown.

*Let $n \in \mathbb{N}^+$ and $\mathcal{G}$ be a class of DAGs with n vertices. We say that a legitimate adversary A is a $\mathcal{G}$-*adversary *if in any execution the key graph belongs to $\mathcal{G}$, i.e., $G(\mathsf{A}) \in \mathcal{G}$.*

Definition 4. *Let $t \in \mathbb{N}^+$, $0 < \epsilon < 1$. An encryption scheme Enc is called $(n, t, \epsilon, \mathcal{G})$-GSD secure if for every $\mathcal{G}$-adversary A running in time t, we have*

$$\left| \Pr\left[\mathbf{Exp}^{\text{GSD-}(n,1)}_{\mathsf{Enc},\mathsf{A}} = 1\right] - \Pr\left[\mathbf{Exp}^{\text{GSD-}(n,0)}_{\mathsf{Enc},\mathsf{A}} = 1\right]\right| \leq \epsilon \ .$$

Assuming one challenge query is enough. Although the definition of GSD allows the adversary to make any number of corruption queries, Panjwani [Pan07] observes that by a standard hybrid argument one can turn any adversary with advantage ϵ (which makes at most $q \leq n$ challenge queries) into an adversary that makes only one challenge query, but still has advantage at least ϵ/q. From now on we therefore only consider adversaries that make exactly one challenge query (keeping in mind that we have to pay an extra factor n in the final distinguishing advantage for statements about general adversaries).

4 Single Source

In this section we will analyze the GSD game for key graphs in which the challenge node is only reachable from one source node. That is, for some $q \leq n$ there is a path $p_1 \to p_2 \to \ldots \to p_q$ where p_1 has in-degree 0, all nodes p_i, $2 \leq i \leq q$ have in-degree 1 (but arbitrary out-degree) and the (single) challenge query is $\mathsf{challenge}(p_q)$ (recall that the challenge has out-degree 0). Let $\mathcal{G}_1$ be the set of all such graphs, and $\mathcal{G}_1^\ell \subseteq \mathcal{G}_1$ be the subset where this path has length at most ℓ.

Theorem 1 (GSD on trees with one path to challenge). *Let $t \in \mathbb{N}$, $0 < \epsilon < 1$ and $\mathcal{G}_1$ be the class of key graphs just defined. If an encryption scheme is (t, ϵ)-IND-CPA secure then it is also $(n, t', \epsilon', \mathcal{G}_1)$-GSD secure for*

$$\epsilon' = \epsilon \cdot n\,(3n)^{\lceil \log n \rceil} \qquad \text{and} \qquad t' = t - Q_{\mathsf{Adv}} T_{\mathsf{Enc}} - \tilde{O}(Q_{\mathsf{Adv}}) \ ,$$

where T_{Enc} denotes the time required to encrypt a key, and Q_{Adv} denotes an upper bound on the number of queries made by the adversary.[12] *More generally, if we replace $\mathcal{G}_1$ with $\mathcal{G}_1^\ell$, we get*

$$\epsilon' = \epsilon \cdot n\,(3n)^{\lceil \log \ell \rceil} \qquad \text{and} \qquad t' = t - Q_{\mathsf{Adv}} T_{\mathsf{Enc}} - \tilde{O}(Q_{\mathsf{Adv}}) \ .$$

[12] If Enc is deterministic then w.l.o.g. we can assume $Q_{\mathsf{Adv}} \leq n^2$ as there are at most $n(n-1)/2$ possible encryption queries (plus $\leq n$ corruption and challenge queries). If Enc is probabilistic then A is allowed any number of encryption queries.

GSD on single-source graphs. For $b \in \{0,1\}$, we consider the GSD game $\mathbf{Exp}_{\mathsf{Enc}}^{\text{GSD-}(n,b)}$ on $\mathcal{G}_1$ between B and an adversary A. Challenger B first samples n random keys $k_1, k_2, \ldots, k_n$ and we assume that already at this point B samples fake keys $r_1, \ldots, r_n$. On all $\mathsf{encrypt}(i,j)$ queries B returns real responses $\mathsf{Enc}_{k_i}(k_j)$. If $b=0$, the response to $\mathsf{challenge}(z)$ is k_z; if $b=1$, the response is r_z.

We require that the key graph is in $\mathcal{G}_1$, that is the connected component of the key graph which contains the challenge z has a path $p_1 \to p_2 \to \ldots \to p_q = z$ with p_1 having in-degree 0, all other p_i having in-degree 1 and $p_q = z$ having out-degree 0 (this means A made queries $\mathsf{encrypt}(p_{i-1}, p_i)$, but no queries $\mathsf{encrypt}(x, p_i)$ for $x \neq p_{i-1}$).

Eventually, A outputs a bit $b' \in \{0,1\}$, which is also the output of the game. If the encryption scheme Enc is not $(t', \epsilon', \mathcal{G}_1)$-GSD secure then there exists a $\mathcal{G}_1$-adversary A running in time t' such that

$$\left| \Pr\left[\mathbf{Exp}_{\mathsf{Enc},\,\mathsf{A}}^{\text{GSD-}(n,\,0)} = 1\right] - \Pr\left[\mathbf{Exp}_{\mathsf{Enc},\,\mathsf{A}}^{\text{GSD-}(n,\,1)} = 1\right] \right| > \epsilon' \ . \tag{1}$$

Our goal. Suppose we knew that our GSD adversary A wants to be challenged on a fixed node z^* and that it will make a query $\mathsf{encrypt}(y, z^*)$ for some y which it will not use in any other query. Then we could use A directly to construct a distinguisher D as in Definition 2: D sets up all keys k_x, $x \in [n]$, samples a value r_{z^*} and runs A, answering A's queries using its keys; except when $\mathsf{encrypt}(y, z^*)$ is queried for any $y \in [q]$, D queries its own challenger on (k_{z^*}, r_{z^*}) and forwards the answer to A. Moreover, $\mathsf{challenge}(z^*)$ is answered with k_{z^*}. If D's challenger C chose $b=0$, this perfectly simulates the real game for A. If $b=1$ then A gets an encryption of r_{z^*} and the challenge query is answered with k_{z^*}, although in the random GSD game A expects an encryption of k_{z^*} and $\mathsf{challenge}(z^*)$ to be answered with r_{z^*}. However, these two games are distributed identically, since both k_{z^*} and r_{z^*} are uniformly random values that do not occur anywhere else in the game. Thus D simulates the real game when $b=0$ and the random game when $b=1$. Note that D implicitly set k_y to the key that C chose, but that's fine, since we assumed that k_y is not used anywhere else in the game and thus not needed by D for the simulation.

Finally, suppose that, in addition to the challenge z^*, we knew y^* for which A will query $\mathsf{encrypt}(y^*, z^*)$. Then we could also allow A to issue queries of the form $\mathsf{encrypt}(y^*, x)$, for x other than z^*. D could easily simulate any such query by querying k_x to its encryption oracle.

Unfortunately, general GSD adversaries can decide adaptively on which node they want to be challenged, and worse, they can make queries $\mathsf{encrypt}(x,y)$, where y is a key that encrypts the challenge.

We will construct a series of hybrids where any two consecutive games **Game** and **Game′** are such that from a distinguisher A for them, we can construct an adversary D against the encryption scheme with the same advantage. For this, the two games should only differ in the response of one encryption query on the path to the challenge, say $\mathsf{encrypt}(y,z)$, which is responded to with a real ciphertext $\mathsf{Enc}_{k_y}(k_z)$ in **Game** and with a fake ciphertext $\mathsf{Enc}_{k_y}(r_z)$ in **Game′**.

Moreover, the key k_y must not be encrypted anywhere else in the game, as our distinguisher D will implicitly set k_y to be the key of its IND-CPA challenger C. Thus, in **Game** and **Game**$'$ all queries $\mathsf{encrypt}(x, y)$, for any x, are responded to with a fake ciphertext $\mathsf{Enc}_{k_x}(r_y)$. Summing up, we need the two games to have the following properties for some y:

- Property 1. **Game** and **Game**$'$ are identical except for the response to one query $\mathsf{encrypt}(y, z)$, which is replied to with a real ciphertext in **Game** and a fake one in **Game**$'$.
- Property 2. Queries $\mathsf{encrypt}(x, y)$ are replied to with a fake response in both games.

If we knew the entire key graph $G(\mathsf{A})$ before answering A's queries then we could define a series of $2q-1$ games as in Fig. 1 where we consecutively replace edges from the source to the challenge by fake nodes and then go back replacing fake edges with real ones starting with $p_{q-2} \to p_{q-1}$. Any two consecutive games in such a sequence would satisfy the two properties, so we could use them to break IND-CPA.

The problem is that in general the probability of guessing the connected component containing the challenge is exponentially small in n and consequently from a GSD adversary's advantage ϵ' we will obtain a distinguisher D with advantage $\epsilon = \epsilon'/O(n!)$. To avoid an exponential loss, we thus must avoid guessing the entire component at once.

The first step. Our first step is to define two new games $\mathsf{Game}_\emptyset^{\{q\}}$ and $\mathsf{Game}_{\{q\}}^{\{q\}}$, which are modifications of $\mathbf{Exp}^{\text{GSD-0}}$ and $\mathbf{Exp}^{\text{GSD-1}}$, respectively. Both new games have an extra step at the beginning of the game: B guesses which key is going to be the challenge key and at the end of the game only if its guess was correct, the output of the game is A's output and otherwise it is 0. Clearly B's guess is correct with probability $1/n$. Aside from this guessing step, $\mathsf{Game}_\emptyset^{\{q\}}$ is identical to $\mathbf{Exp}^{\text{GSD-0}}$; all responses are real. We therefore have $\Pr[\mathsf{Game}_\emptyset^{\{q\}} = 1] = 1/n \cdot \Pr[\mathbf{Exp}^{\text{GSD-0}} = 1]$.

Analogously, we define an auxiliary game, $\mathsf{Game}_1^{\{q\}}$, which is identical to $\mathbf{Exp}^{\text{GSD-1}}$, except for the guessing step. Again we have $\Pr[\mathsf{Game}_1^{\{q\}} = 1] = 1/n \cdot \Pr[\mathbf{Exp}^{\text{GSD-1}} = 1]$. We then define $\mathsf{Game}_{\{q\}}^{\{q\}}$ exactly as $\mathsf{Game}_1^{\{q\}}$, except for a syntactical change: Let z be the guessed value for the challenge node. Then any query $\mathsf{encrypt}(x, z)$ is replied to with $\mathsf{Enc}_{k_x}(r_z)$, that is, an encryption of the *fake* key r_z. (Note that this game can be simulated, since we "know" z when guessing correctly.) On the other hand, the query $\mathsf{challenge}(z)$ is answered with k_z (rather than r_z in $\mathbf{Exp}^{\text{GSD-1}}$). Since the difference between $\mathsf{Game}_1^{\{q\}}$ and $\mathsf{Game}_{\{q\}}^{\{q\}}$ is that we have replaced all occurrences of k_z by r_z and all occurrences of r_z by k_z, which are distributed identically (thus we've merely swapped the names of k_z and r_z), we have $\Pr[\mathsf{Game}_{\{q\}}^{\{q\}} = 1] = \Pr[\mathsf{Game}_1^{\{q\}} = 1] = 1/n \cdot \Pr[\mathbf{Exp}^{\text{GSD-1}} = 1]$.

Together with Equation (1), we have thus

$$\begin{aligned} &\left|\Pr\left[\mathsf{Game}_{\emptyset}^{\{q\}} = 1\right] - \Pr\left[\mathsf{Game}_{\{q\}}^{\{q\}} = 1\right]\right| \\ &\qquad = {}^{1}\!/_{n} \cdot \left|\Pr\left[\mathbf{Exp}^{\text{GSD-0}} = 1\right] - \Pr\left[\mathbf{Exp}^{\text{GSD-1}} = 1\right]\right| > {}^{1}\!/_{n} \cdot \epsilon' \ . \end{aligned}$$

We continue to use the notational convention that for sets $I \subseteq P \subseteq [n]$, the game Game_I^P is derived from the real game by additionally guessing the nodes corresponding to P and answering encryptions of the nodes in I with fake keys. This is made formal in Fig. 4 below.

The second step. Assume q is a power of 2 and consider $\mathsf{Game}_{\emptyset}^{\{q/2,\, q\}}$, which is identical to $\mathsf{Game}_{\emptyset}^{\{q\}}$, except that in addition to the challenge node, B also guesses which node $x \in [n]$ is going to be the node in the middle of the path to the challenge, i.e. $p_{q/2} = x$. The output of $\mathsf{Game}_{\emptyset}^{\{q/2,\, q\}}$ is A's output if the guess was correct and 0 otherwise. Since B guesses correctly with probability ${}^{1}\!/_{n}$, we have

$$\Pr\left[\mathsf{Game}_{\emptyset}^{\{q/2,\, q\}} = 1\right] = {}^{1}\!/_{n} \cdot \Pr\left[\mathsf{Game}_{\emptyset}^{\{q/2\}} = 1\right] \ .$$

By guessing the middle node, we can assume the middle node is *known* and this will enable us to define a hybrid game, $\mathsf{Game}_{\{q/2\}}^{\{q/2,\, q\}}$, in which the query for the encryption of $k_{p_{q/2}}$ is responded to with a fake answer. In addition, we consider games $\mathsf{Game}_{\{q\}}^{\{q/2,\, q\}}$ and $\mathsf{Game}_{\{q/2,\, q\}}^{\{q/2,\, q\}}$ which are similarly defined by making the same changes to game $\mathsf{Game}_{\{q\}}^{\{q\}}$, i.e. guessing the middle node and replying to the encryption query of the guessed key with a fake and a real ciphertext respectively. Again, we have $\Pr[\mathsf{Game}_{\{q\}}^{\{q/2,\, q\}} = 1] = {}^{1}\!/_{n} \cdot \Pr[\mathsf{Game}_{\{q\}}^{\{q\}} = 1]$. Therefore $(t', \epsilon'/n)$-distinguishability of $\mathsf{Game}_{\emptyset}^{\{q\}}$ and $\mathsf{Game}_{\{q\}}^{\{q\}}$ implies that $\mathsf{Game}_{\emptyset}^{\{q/2,\, q\}}$ and $\mathsf{Game}_{\{q\}}^{\{q/2,\, q\}}$ are $(t', \epsilon'/n^2)$-distinguishable, i.e. $\mathit{\Delta}_t\big(\mathsf{Game}_{\emptyset}^{\{q/2,\, q\}}, \mathsf{Game}_{\{q\}}^{\{q/2,\, q\}}\big) > \epsilon'/n^2$, and therefore by the triangle inequality

$$\begin{aligned} &\mathit{\Delta}_t\Big(\mathsf{Game}_{\emptyset}^{\{q/2,\, q\}}, \mathsf{Game}_{\{q/2\}}^{\{q/2,\, q\}}\Big) + \mathit{\Delta}_t\Big(\mathsf{Game}_{\{q/2\}}^{\{q/2,\, q\}}, \mathsf{Game}_{\{q/2,\, q\}}^{\{q/2,\, q\}}\Big) \\ &\qquad + \mathit{\Delta}_t\Big(\mathsf{Game}_{\{q/2,\, q\}}^{\{q/2,\, q\}}, \mathsf{Game}_{\{q\}}^{\{q/2,\, q\}}\Big) \geq \ \mathit{\Delta}_t\Big(\mathsf{Game}_{\emptyset}^{\{q/2,\, q\}}, \mathsf{Game}_{\{q\}}^{\{q/2,\, q\}}\Big) \\ &\qquad\qquad > \ {}^{1}\!/_{n^2} \cdot \epsilon' \ . \end{aligned} \tag{2}$$

By Equation (2), at least one of the pairs of games on the left-hand side must be $(t', \epsilon'/3n^2)$-distinguishable. The two games of every pair differ in exactly one point, as determined by the subscript of each game. For instance, the difference between the last pair $\mathsf{Game}_{\{q/2,\, q\}}^{\{q/2,\, q\}}$ and $\mathsf{Game}_{\{q\}}^{\{q/2,\, q\}}$ is the encryption of node $q/2$.

Recall that our goal is to construct a pair of hybrids where the differing query $\mathsf{encrypt}(y, z)$ is such that all queries $\mathsf{encrypt}(x, y)$ are replied to with $\mathsf{Enc}_{k_x}(r_y)$, as formalized as Property 2. Games $\mathsf{Game}_{\emptyset}^{\{q\}}$ and $\mathsf{Game}_{\{q\}}^{\{q\}}$ differed in the last query on the path and the only key above it that is not encrypted anywhere is the start of the path. What we have achieved with our games above is to *halve* that distance: the first pair, $(\mathsf{Game}_{\emptyset}^{\{q/2,\, q\}}, \mathsf{Game}_{\{q/2\}}^{\{q/2,\, n\}})$, and the last pair, $(\mathsf{Game}_{\{q/2,\, q\}}^{\{q/2,\, q\}}, \mathsf{Game}_{\{q\}}^{\{q/2,\, q\}})$, differ in a node that is only half way down the path;

Game_I^P, with $I \subseteq P \subseteq [n]$ is defined as follows:

- For every $i \in P$, B chooses $v_i \leftarrow [n]$, which is B's guess for the node at position i in the final path.
- B chooses $2n$ keys $k_1, r_1, k_2, r_2 \ldots, k_n, r_n \leftarrow \{0,1\}^\lambda$ and runs A.
- Whenever A makes a query $\mathsf{encrypt}(x, y)$, B does the following: If $y = v_i$ for some $i \in I$ then reply with $\mathsf{Enc}_{k_x}(r_{v_i})$; otherwise reply with $\mathsf{Enc}_{k_x}(k_y)$.
- When A makes the query $\mathsf{challenge}(z)$, return k_z.
- Let $b' \in \{0,1\}$ be A's output. At then end of the game, consider the longest path $p_0 \to p_1 \to \cdots \to p_q$ in $G(\mathsf{A})$, with p_q being the argument of A's $\mathsf{challenge}$ query. If for all $i \in P$: $v_i = p_i$ then B returns b'; otherwise, B returns 0.

Fig. 4: Definition of Game_I^P for the single-source case.

and the middle pair, $(\mathsf{Game}_{\{q/2\}}^{\{q/2,\,q\}}, \mathsf{Game}_{\{q/2,\,q\}}^{\{q/2,\,q\}})$, differ in the last node, but half way up the path there is a key, namely $k_{q/2}$, which is not encrypted anywhere, as all queries $\mathsf{encrypt}(x, q/2)$ are answered with $\mathsf{Enc}_{k_x}(r_{q/2})$.

The remaining steps. For any of the three pairs that is $(t', \epsilon'/3n^2)$-distinguishable (and by Equation (2) there must exist one), we can repeat the same process on the half of the path which ends with the query that is different in the two games. For example, assume this holds for the last pair, that is

$$\Delta_t\Big(\mathsf{Game}_{\{q/2,\,q\}}^{\{q/2,\,q\}}, \mathsf{Game}_{\{q\}}^{\{q/2,\,q\}}\Big) \;>\; \frac{\epsilon'}{3n^2} \;. \tag{3}$$

We repeat the process of guessing the middle node between the differing node and the random node above (in this case the root of the path), which is thus node $q/4$, and obtain a new pair which satisfies

$$\Delta_t\Big(\mathsf{Game}_{\{q/2,\,q\}}^{\{q/4,\,q/2,\,q\}}, \mathsf{Game}_{\{q\}}^{\{q/4,\,q/2,\,q\}}\Big) \;>\; \frac{\epsilon'}{3n^3} \;, \tag{4}$$

by Equation (3) and the fact that the guess is correctly with probability $1/n$. We can now define two intermediate games

$$\mathsf{Game}_{\{q/4,\,q/2,\,q\}}^{\{q/4,\,q/2,\,q\}} \qquad \text{and} \qquad \mathsf{Game}_{\{q/4,\,q\}}^{\{q/4,\,q/2,\,q\}} \tag{5}$$

where we replaced the encryption of $k_{p_{q/4}}$ by one of $r_{p_{q/4}}$. As in Equation (2), we can again define a sequence of games by putting the games in Equation (5) between the ones in Equation (4) and argue that by Equation (4), two consecutive hybrids must be $(t', \epsilon'/(3^2n^3))$-distinguishable. What we have gained is that any pair in this sequence differs by exactly one edge and the closest fake answer above is only a fourth of the path length away.

Repeating these two steps a maximum number of $\lceil \log q \rceil$ times, we arrive at two consecutive games, where the distance from the differing node to the closest "fake" node above is 1. We have thus found two games that satisfy Properties 1 and 2, meaning we can use a distinguisher A to construct an adversary D against the encryption scheme.

Since a path has at most n nodes, after at most $\log n$ steps we end up with two games that are $(t', \epsilon'/n(3n)^{\lceil \log n \rceil})$-distinguishable and which can be used to break the encryption scheme. If the adversary is restricted to paths of length ℓ (i.e., graphs in $\mathcal{G}_1^\ell$), this improves to $(t', \epsilon'/n(3n)^{\lceil \log \ell \rceil})$.

Proof of Theorem 1. We formalize our method to give a proof of the theorem. In Fig. 4 we describe game Game_I^P, which is defined by the nodes on the path that are guessed (represented by the set P) and the nodes where an encryption of a key is replaced with an encryption of a value r (represented by $I \subseteq P$).

Lemma 1. *Let $I \subseteq P \subseteq [n]$ and $z \in P \setminus I$. Also let y be the largest number in I such that $y < z$, and $y = 0$ if z is smaller than all elements in I. If Game_I^P and $\mathsf{Game}_{I\cup\{z\}}^P$ are (t, ϵ)-distinguishable then the following holds.*

- *If $z = y + 1$ then Enc is not $(t + Q_{\mathsf{Adv}} T_{\mathsf{Enc}} + \tilde{O}(Q_{\mathsf{Adv}}), \epsilon)$-IND-CPA-secure.*
- *If $z > y + 1$, define $z' = y + \lfloor (z - y)/2 \rfloor$, $P' = P \cup \{z'\}$ and*

$$I_1 = I \ , \qquad I_2 = I \cup \{z'\} \ , \qquad I_3 = I \cup \{z', z\} \ , \qquad I_4 = I \cup \{z\} \ .$$

Then for some $i \in \{1, 2, 3\}$, games $\mathsf{Game}_{I_i}^{P'}$ and $\mathsf{Game}_{I_{i+1}}^{P'}$ are $(t, \epsilon/3n)$-distinguishable.

The proof of this lemma can be found in the full version. Applying Lemma 1 repeatedly $\lceil \log n \rceil$ times (or $\lceil \log \ell \rceil$ if we know an upper bound on the path length ℓ), we obtain the proof of Theorem 1.

5 General Trees

For a node v in a directed graph G let T_v denote the subgraph of G we get when only keeping the edges on paths that lead to v. In this section we prove bounds for GSD if the underlying key graph is a tree. Concretely, let $\mathcal{G}_\tau$ be the class of key graphs that contain one designated "challenge node" z and where the graph T_z is a tree (when ignoring edge directions).

To give more fine-grained bounds we define a subset $\mathcal{G}_\tau^{s,d,\ell} \subseteq \mathcal{G}_\tau$ as follows. For $G \in \mathcal{G}_\tau$, let z be the challenge node and T_z as above. Then $G \in \mathcal{G}_\tau^{s,d,\ell}$ if the challenge node has at most s sources (i.e., there are at most s nodes u of in-degree 0 s.t. there is a directed path from u to z), every node in T_z has in-degree at most d and the longest path in T_z has length at most ℓ. Note that as $d < n, s < n$ and $\ell \leq n$ any $G \in \mathcal{G}_\tau$ with n nodes is trivially in $\mathcal{G}_\tau^{n-1,n-1,n}$.

Theorem 2 (Security of GSD on trees). *Let $n, t \in \mathbb{N}$, $0 < \epsilon < 1$ and $\mathcal{G}_\tau$ be the class of key graphs just defined. If an encryption scheme is (t, ϵ)-IND-CPA secure then it is also $(n, t', \epsilon', \mathcal{G}_\tau)$-GSD secure for*

$$\epsilon' = \epsilon \cdot n^2 (6n^3)^{\lceil \log n \rceil} \leq \epsilon \cdot n^{3\lceil \log n \rceil + 5} \qquad \textit{and} \qquad t' = t - Q_{\mathsf{Adv}} T_{\mathsf{Enc}} - \tilde{O}(Q_{\mathsf{Adv}})$$

(with $Q_{\mathsf{Adv}}, T_{\mathsf{Enc}}$ as in Theorem 1). If we replace $\mathcal{G}_\tau$ with $\mathcal{G}_\tau^{s,d,\ell}$ then

$$\epsilon' = \epsilon \cdot dn((2d+1)n)^{\lceil \log s \rceil} (3n)^{\lceil \log \ell \rceil} \quad \textit{and} \quad t' = t - Q_{\mathsf{Adv}} T_{\mathsf{Enc}} - \tilde{O}(Q_{\mathsf{Adv}}) \ .$$

For space reasons, the proof of this theorem is moved to the full version.

6 Conclusions and Open Problems

We showed a quasipolynomial reduction of the GSD game on trees to the security of the underlying symmetric encryption scheme. As already discussed in the introduction, it is an interesting open problem to extend our reduction to general (directed, acyclic) graphs or to understand why this is not possible. This is the second result using the "nested hybrids" technique (after its introduction in [FKPR14] to prove the security of constrained PRFs), and given that it found applications for two seemingly unrelated problems, we believe that there will be further applications in the future.

One candidate is the problem of proving security under selective opening attacks [DNRS99,FHKW10,BHY09], where one wants to prove security when correlated messages are encrypted under different keys. Here, the adversary may adaptively chose to corrupt some keys after seeing all ciphertexts, and one requires that the messages in the unopened ciphertexts are indistinguishable from random messages (sampled so they are consistent with the already opened ciphertexts). This problem is notoriously hard, and no reduction avoiding complexity leveraging to IND-CPA security of the underlying scheme is known.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments.

References

[ABBC10] Tolga Acar, Mira Belenkiy, Mihir Bellare, and David Cash. Cryptographic agility and its relation to circular encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 403–422. Springer, May 2010.

[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, March 2014.

[BH93] Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In Rainer A. Rueppel, editor, *EUROCRYPT'92*, volume 658 of *LNCS*, pages 307–323. Springer, May 1993.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, April 2009.

[BRS03] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, August 2003.

[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, December 2013.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, May 2001.

[DNRS99] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. In *40th FOCS*, pages 523–534. IEEE Computer Society Press, October 1999.

[FHKW10] Serge Fehr, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Encryption schemes secure against chosen-ciphertext selective opening attacks. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 381–402. Springer, May 2010.

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained PRFs. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 82–101. Springer, December 2014.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[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.

[MP06] Daniele Micciancio and Saurabh Panjwani. Corrupting one vs. corrupting many: The case of broadcast and multicast encryption. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 70–82. Springer, July 2006.

[Pan07] Saurabh Panjwani. Tackling adaptive corruptions in multicast encryption protocols. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 21–40. Springer, February 2007.

[SS00] Ali Aydin Selçuk and Deepinder P. Sidhu. Probabilistic methods in multicast key management. In *Information Security, Third International Workshop, ISW 2000, Wollongong, NSW, Australia, December 20-21, 2000, Proceedings*, pages 179–193, 2000.

[WGL00] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–30, February bbl 2000.