On the Security of Generalized Selective Decryption

Abstract

Generalized Selective Decryption (GSD) is an easy to define game based on a symmetric encryption scheme Enc. It was introduced by Panjwani [TCC'07] to capture the difficulty of proving adaptive security of certain protocols. In the GSD game there are n keys $k_1, \ldots, k_n$, which the adversary may adaptively corrupt (i.e., 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 the encryption scheme is surprisingly hard. One can prove security using "complexity leveraging", but this reduction loses a factor exponential in n, which makes the proof basically useless.

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 an improved reduction that lost only a factor exponential in ℓ (not n). To date, this is the only non-trivial result known for GSD or related problems.

In this paper we give almost polynomial reductions for large classes of graphs. Most importantly, we show that the security of the GSD game restricted to trees (which is an important special case abstracting some real-world protocols like the Logical Key Hierarchy protocol) can be proven losing only a quasi-polynomial factor $n^{3\log n+5}$. Our proof borrows ideas from the "nested hybrids" technique recently introduced by Fuchsbauer at 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 often 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].

Panjwani [Pan07] argues that the two common approaches to achieving adaptive security, namely requiring that all parties erase past data [BH92], 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, while capturing 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 gets $c = \mathsf{Enc}_{k_i}(k_j)$; corruption query: on input i, she gets 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.[1] Finally, A outputs

[1]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]).

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).[2]

It is convenient to think of the GSD game as dynamically building a graph, which we call key graph. Initially, we have 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 j where there's a path from i to j in the key graph by simply decrypting the keys along the path. To make the game non-trivial, we therefore only allow challenge queries 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. The reason is simply that we cannot hope to prove security when allowing cycles assuming only a standard security notion (in our case IND-CPA) of the underlying encryption scheme. This would require circular (or key-dependent-message) security [BRS02], 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.[3]

Selective security of GSD. In order to prove security of the GSD game, one must show how to turn an adversary A who breaks the GSD game with some advantage $\epsilon = |\Pr[b = b'] - 1/2|$ into an adversary B who breaks the security of Enc with 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 security game an adversary B is given access to an encryption oracle $\mathsf{Enc}_k(.)$. 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 finally must output a guess b'. The advantage of B is $|\Pr[b = b'] - 1/2|$.

It's not at all clear how to construct an adversary B that breaks IND-CPA from an A who 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 who 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. In particular, if we have an A with advantage ϵ against GSD, 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 Figure 1.

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 relevant choices to be made by the adaptive adversary and committing to those (as required by the selective security game). If during the security game the adaptive choices by the adversary disagree with our initial guess, we simply abort. The problem with this approach is that

[2]A related problem to GSD is 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). Also this problem is notoriously hard, and no reduction to IND-CPA security of the underlying scheme is known.

[3]Concretely, this would require that for any k, k', m one cannot distinguish $\mathsf{Enc}_k(m)$ from $\mathsf{Enc}_{k'}(m)$; see Footnote 11.

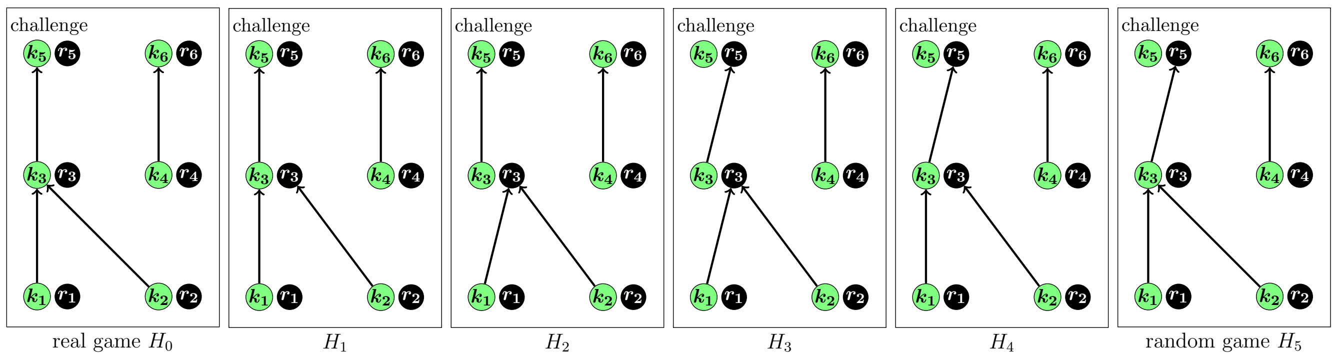


Figure 1: Illustration of the selective security game. The green nodes correspond to keys, the dark nodes are random values. The adversary commits to encryption queries $(1, 3), (2, 3), (3, 5)$ and challenge 5 (there's also an encryption query $(4, 6)$, but it's not in the connected component containing the challenge, and thus not relevant for defining the hybrids. The adversary can also corrupt keys 4 and 6 as they are outside of the component containing the challenge). The hybrid H_0 corresponds to the real game, the 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 this 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'll call k_5 and r_5). Now B can simulate the game H_2 for A, but when A makes the encryption query $(3, 5)$ we answer with z. If $z = k_5$, then we simulate the game H_2, but if $z = r_5$ we simulate the 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.

assuming the adaptive adversary has advantage ϵ, the selective adversary we get via complexity leveraging only has advantage ϵ/P where $1/P$ is the probability that our guess is correct, and typically is 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 security 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 the 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 and related problems.

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 we don't need the entire key graph to 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 (note that ℓ, d and s are at most n, and the above bound was derived setting $\ell = d = s = n$)

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

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 are the leaves of this tree, and keys are associated with all the nodes from leaves to the root. 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 $n^{\log(n)+\log\log(n)+4}$ bound 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) 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 vertices disconnects u from v (note that a tree has cut-width $w = 1$). For graphs with cut-width w we get

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

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 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 already sufficient to get asymptotically overwhelming security (again assuming Enc is exponentially hard).

Nested hybrids. In a classical paper [GGM86] Goldreich, Goldwasser and Mical gave a construction of a pseudorandom function (PRF) from a length-doubling pseudorandom generator (PRG). More recently, three papers independently [BW13, KPTZ13, BGI14] observed that this construction can also be used as a so called *constrained* PRF, where one can, for every string x, output a constrained key k_x that allows to evaluate the PRF on all inputs with prefix x. Informally, the security required here is that an adversary who 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,

[4]Let us stress that the graph we get 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.

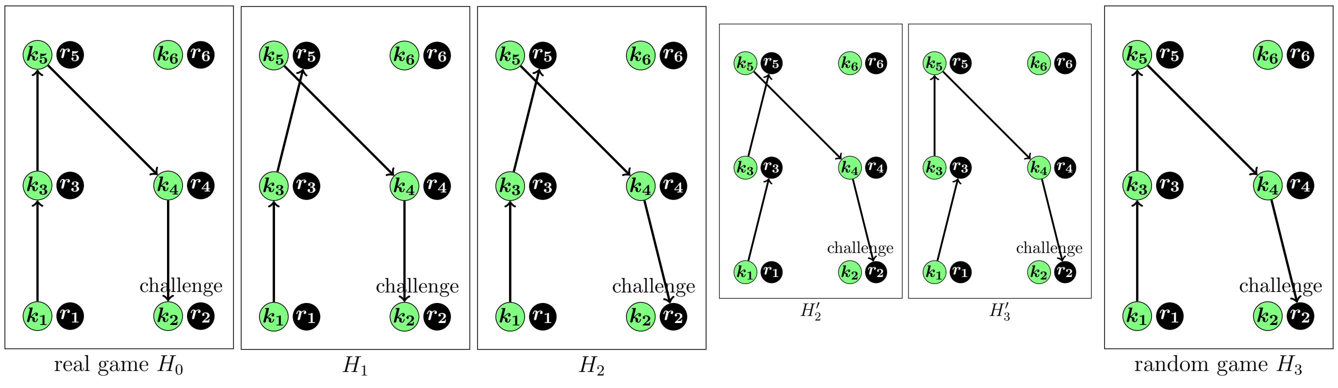


Figure 2: Illustration of our adaptive security proof for paths.

where the adversary must commit to the input on which it wants to be challenged before making any queries (this proof is a hybrid argument losing a factor $2m$ in the distinguishing advantage, where m is the input length of the PRF). 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 $(3q)^{\log m}$ factor, 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. As just outlined, 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 as 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, where one just guesses 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, only reduced to the same problem, but now 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 a similar approach as 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 $2^{\Theta(n)}$. Here we'll 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 Figure 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 selective" game where the adversary must commit initially to this middle node, at the

[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 $\mathsf{Enc}_k(k') = k \oplus k'$), as 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.

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 a factor of n). We have illustrated this with a small example featuring a path of length 4 in Figure 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 $\epsilon/3n$. If we look at two consecutive games H_i and H_{i+1}, then 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 we had for 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 Figure 2, we can guess the node which halves the path leading to 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 $\mathsf{Enc}_{k_1}(r_3)$ of the random r_3 instead of $\mathsf{Enc}_{k_1}(k_3)$.

If we start with a path of length $\ell \leq n$, after $\log \ell \leq \log n$ iterations of this process we have proven the existence of two consecutive games (let's call them G_0 and G_1) which differ only in a single edge, and in the underlying key graph, this edge forms a path $j \to i$ of length 1 (i.e., the vertex j has in-degree 0). That is, both games are identical, except that in one game we must answer the encryption query (j, i) with $\mathsf{Enc}_{k_j}(k_i)$, 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 has committed to i).

Given a distinguisher A for G_0 and G_1, we can now construct a new attacker B that breaks the IND-CPA security of the underlying encryption scheme with the same advantage. For this, we let B choose the two messages m_0, m_1 it wants to be challenged on in the IND-CPA security game at random.[7] The IND-CPA game samples a random bit b, and returns the challenge $C = \mathsf{Enc}_k(m_b)$ to B, who 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 guessing bit b'. Identifying (k, m_0, m_1) with (k_j, k_i, r_i) we see that depending on whether $b = 0$ or $b = 1$, we simulate either the game G_0 or G_1. Thus, whatever advantage $|\Pr[b = b'] - 1/2|$ the adversary A has in distinguishing G_0 from G_1, B will break the IND-CPA security of Enc with the same advantage.

[6]Technically, we never actually construct this "slightly selective" adversary, but (exactly 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 can also output a constant value, the point is that the advantage of the adversary, conditioned on our guess being wrong, is zero; whereas the advantage, conditioned on the guess being correct, is the same as the advantage of the adaptive adversary). Instead of this experiment, it is easier to follow our proof outline by thinking of this experiment as having the adversary actually commit to its choices initially, but the reduction paying a factor (on the distinguishing advantage of the adversary who is allowed to make this choice adaptively) that corresponds to the size of the sample space of this guess.

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

[8]Note that here we need the fact that node j has in-degree 0. This allows us to indentify the key k_j with the key k used by the IND-CPA experiment, as we never have to encrypt k_j.

Proof outline for trees. We will now outline our reduction of the adaptive security of the GSD game to the IND-CPA security of Enc for a more general case: The adversary is restricted in that the key graph resulting from his 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-degree ≤ 1. Let us first consider the case where the connected component containing the challenge is a tree, and moreover all vertices in it have in-degree 0 or 1. It turns out that this case is not really more difficult to prove than the case of paths. The proof outlined for paths goes through with only minor changes for such trees. In particular, note that given such a tree, there is 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 path, and do a reduction as the one outlined above. The only interesting 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 don't know k_j (we identified k_j with the key used by the IND-CPA challenger). But recall that in the IND-CPA security game we have access to an encryption oracle $\mathsf{Enc}_{k_j}(.)$, and can thus simply ask the oracle for the answer $\mathsf{Enc}_{k_j}(k_x)$ to such encryption queries.

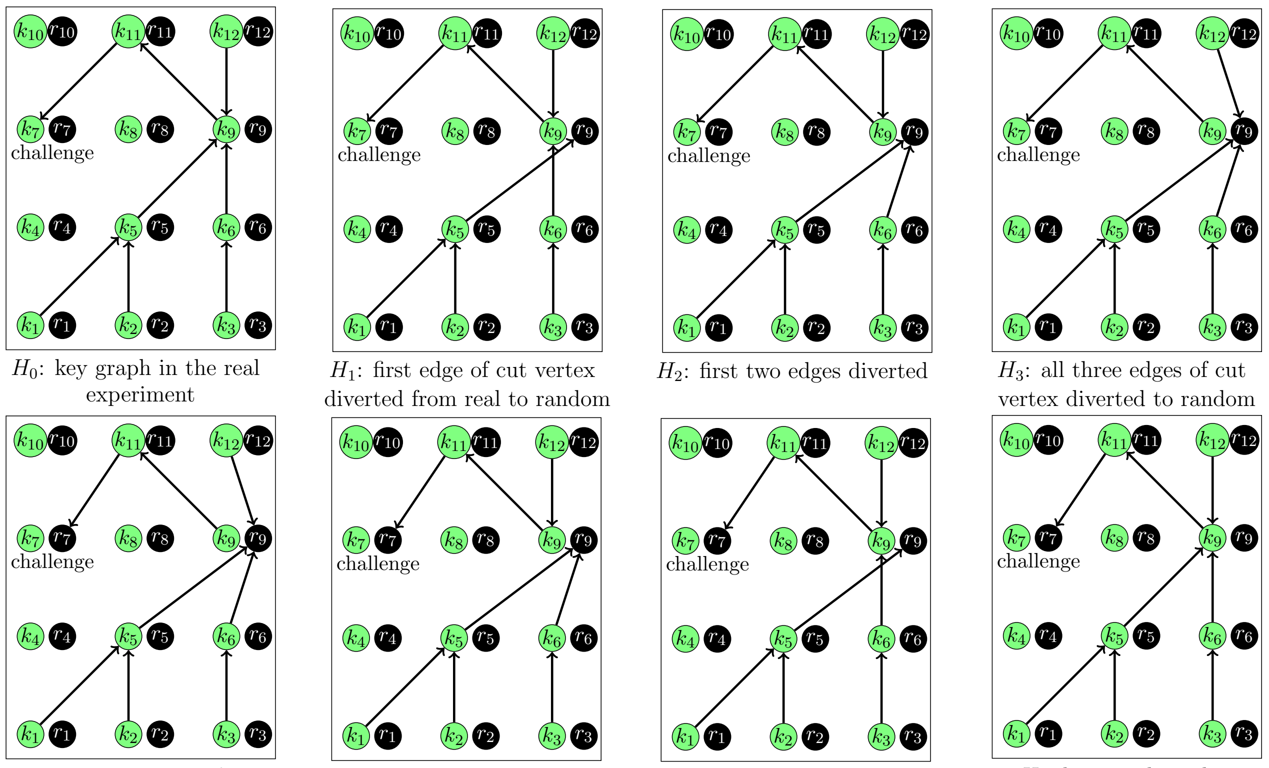
General Trees. For general trees, where nodes can have in-degree greater than 1, we need to work a bit more. We cannot directly generalize the proof for paths as now nodes (in particular, the challenge) can be reached from more than 1 node with in-degree 0. We'll call these the sources of this node, for example in the tree H_0 in Figure 3, the (challenge) node k_7 has four 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 Figure 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, let's consider again the tree H_0 in Figure 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 $k_5 \to v_1$, $k_3 \to k_9$ by $k_3 \to v_2$, 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 will 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 existing 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., it's of the form (j,v) for some j) from real to random, i.e., 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 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 Figure 3.

Because v was well-dividing (and we show in Lemma 2 that such a node always exists), it's not hard



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

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

to prove the following property for any two consecutive games H_i and H_{i+1}. Those games 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 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 use our reduction for trees with only one source outlined above.

Analyzing the Security Loss. Every time we halve the number of sources, we have to guess a well-dividing vertex (that costs us 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 us another factor $2d$. 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 $\epsilon / d(2d)^{\log s} n^{\log s} (3n)^{\log \ell}$. 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 $\Delta_t(Y,X) \le \epsilon$, if no t-bounded distinguisher D can distinguish them with advantage greater than ϵ, i.e.,

$$\Delta_t(Y,X) \le \epsilon \iff \forall \mathsf{D}_t : \left| \Pr\left[\mathsf{D}_t(X) = 1\right] - \Pr\left[\mathsf{D}_t(Y) = 1\right] \right| \le \epsilon \ . \qquad \diamondsuit$$

Symmetric encryption. A pair of algorithms $(\mathsf{Enc}, \mathsf{Dec})$ which take an 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

$$\left| \Pr\left[\mathbf{Exp}^{\text{IND-CPA-1}}_{\mathsf{Enc},\mathsf{D}} = 1\right] - \Pr\left[\mathbf{Exp}^{\text{IND-CPA-0}}_{\mathsf{Enc},\mathsf{D}} = 1\right] \right| \le \epsilon \ . \qquad \diamondsuit$$

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. The game starts with B sampling 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 a ciphertext $c \leftarrow \mathsf{Enc}_{k_i}(k_j)$.
- $\mathsf{corrupt}$: 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 $b = 0$ then the answer is k_i; if $b = 1$ then the answer is a random value $r_i \in \{0,1\}^\lambda$.

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 the adversary to make multiple challenge queries means that the respective keys are jointly pseudorandom (as opposed to individual keys being pseudorandom by themselves). Allowing to interleave challenges with other queries models the requirement that they remain pseudorandom even after more 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 $i \in [n]$ then we mark i as a corrupt vertex; when A queries $\mathsf{challenge}(i)$ then 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 $V^{\text{corr}}(\mathsf{A})$ and $V^{\text{chal}}(\mathsf{A})$ 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.)

[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.

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 j to i, in particular k_j, and easily deduce the bit b. We only consider non-trivial breaks and require that no challenge node is reachable from a corrupt node in $G(\mathsf{A})$.

This is not the only restriction we must impose on $G(\mathsf{A})$ if all we want to assume is that the encryption scheme satisfies IND-CPA. First, we do not allow key cycles, that is, queries yielding

$$\mathsf{Enc}_{k_1}(k_2), \mathsf{Enc}_{k_2}(k_3), \ldots, \mathsf{Enc}_{k_{s-1}}(k_s), \mathsf{Enc}_{k_s}(k_1) \ ,$$

as this would require the scheme to satisfy key-dependent-message (a.k.a. circular) security [BRS02, CL01].

Second, IND-CPA security does not imply that keys under which one has seen encryptions 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 $j \in V^{chal}(\mathsf{A})$ we have that 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.

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

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 legitimate $\mathcal{G}$-adversary A running in time t, we have

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

It is enough to assume one challenge query. Although the definition of GSD allows an 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 (who makes at most $q \le n$ challenge queries) into a new adversary which makes only one challenge query, but still has advantage at least ϵ/q. From now on we therefore only consider adversaries who 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 begin with studying a simplified case. Assume that in the key graph constructed during the GSD game there is only one source node from which the challenge node is reachable. That is, there exists a path $p_1 \to p_2 \to \ldots \to p_q$ where p_1 has in-degree 0, p_q is the node for which A queries $\mathsf{challenge}$ and all nodes p_i, $i \in [q]$ have in-degree 1. Let $\mathcal{G}_1$ be the set of all such graphs, and $\mathcal{G}_1^\ell$ the subset of $\mathcal{G}_1$ where this path has length at most ℓ (note that $\ell \le n$).

[11]IND-CPA does not guarantee that encryptions under different keys are indistinguishable: Consider an IND-CPA-secure encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ and define a new scheme as follows: keys are doubled in length and an encryption of m under $k_1||k_2$ is defined as $\mathsf{Enc}_{k_1}(m)||k_2$. Then this scheme is still IND-CPA, encryptions can be linked to their keys.

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*

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

Here 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] *If we replace $\mathcal{G}_1$ with $\mathcal{G}_1^{\ell}$ we get*

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

GSD on single-source graphs. In GSD game $\mathbf{Exp}_{\mathsf{Enc}}^{\text{GSD-}(n,b)}$ on $\mathcal{G}_1$ between a challenger B and an adversary A B first samples n keys $k_1, k_2, \ldots, k_n$ at random from $\{0,1\}^{\lambda}$. Let us assume that B also samples fake keys $r_1, \ldots, r_n$. On all $\mathsf{encrypt}$ queries B returns real responses. If $b = 0$, the response to $\mathsf{challenge}(z)$ is k_z; if $b = 1$ then the response is r_z.

The connected component of the key graph containing z contains a path $p_1 \to p_2 \to \ldots \to p_q$ (with p_1 having in-degree 0, all other p_i having in-degree 1 and $p_q = z$ having out-degree 0.) Thus, during the game $\mathsf{encrypt}(p_{i-1}, p_i)$ was queried for all $i \in [q]$ but there were no other queries $\mathsf{encrypt}(x, p_i)$ for any $x \in [n]$ and p_i. 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

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

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 Defintion 2: D sets up all keys k_x, $x \in [n]$, samples a value r_{z^*} and runs A, answering all its 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 set the unknown key of its IND-CPA challenger C to be k_y.

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

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, for some y we need the two games to have the following properties:

- 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 Figure 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 of 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

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

Note that we 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 Figure 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, $\mathsf{Game}^{\{q/2,\,q\}}_{\{q/2\}}$, in which the query for the encryption of $k_{p_{q/2}}$ is responded to with a fake answer. In addition, we consider games $\mathsf{Game}^{\{q/2,\,q\}}_{\{q\}}$ 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 $\Pr[\mathsf{Game}^{\{q/2,\,q\}}_{\{q\}} = 1] = {}^1\!/_n \cdot \Pr[\mathsf{Game}^{\{q\}}_{\{q\}} = 1]$. Therefore $(t', \epsilon'/n)$-distinguishablity of $\mathsf{Game}^{\{q\}}_{\emptyset}$ and $\mathsf{Game}^{\{q\}}_{\{q\}}$ implies that $\mathsf{Game}^{\{q/2,\,q\}}_{\emptyset}$ and $\mathsf{Game}^{\{q/2,\,q\}}_{\{q\}}$ are $(t', \epsilon'/n^2)$-distinguishable, i.e. $\Delta_t\big(\mathsf{Game}^{\{q/2,\,q\}}_{\emptyset}, \mathsf{Game}^{\{q/2,\,q\}}_{\{q\}}\big) > \epsilon'/n^2$, and therefore by the triangle inequality

$$\begin{aligned}\Delta_t\big(\mathsf{Game}^{\{q/2,\,q\}}_{\emptyset}, \mathsf{Game}^{\{q/2,\,q\}}_{\{q/2\}}\big) + \Delta_t\big(\mathsf{Game}^{\{q/2,\,q\}}_{\{q/2\}}, \mathsf{Game}^{\{q/2,\,q\}}_{\{q/2,\,q\}}\big) + \Delta_t\big(\mathsf{Game}^{\{q/2,\,q\}}_{\{q/2,\,q\}}, \mathsf{Game}^{\{q/2,\,q\}}_{\{q\}}\big)& \\ \geq\ \Delta_t\big(\mathsf{Game}^{\{q/2,\,q\}}_{\emptyset}, \mathsf{Game}^{\{q/2,\,q\}}_{\{q\}}\big)\ >\ \frac{\epsilon'}{n^2}\ .& \qquad (2)\end{aligned}$$

By Equation (2), at least one of the pairs of games 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 $\mathsf{Game}^{\{q/2,\,q\}}_{\{q/2,\,q\}}$ and $\mathsf{Game}^{\{q/2,\,q\}}_{\{q\}}$ is the encryption of node $q/2$.

Recall that our goal is to construct 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 $\mathsf{Game}^{\{q\}}_{\emptyset}$ and $\mathsf{Game}^{\{q\}}_{\{q\}}$ differed in the last query in 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 games, $\mathsf{Game}^{\{q/2,\,q\}}_{\emptyset}, \mathsf{Game}^{\{q/2,\,n\}}_{\{q/2\}}$ and the last two games $\mathsf{Game}^{\{q/2,\,q\}}_{\{q/2,\,q\}}, \mathsf{Game}^{\{q/2,\,q\}}_{\{q\}}$ differ in a node that is only half way down the path; and the middle two games $\mathsf{Game}^{\{q/2,\,q\}}_{\{q/2\}}, \mathsf{Game}^{\{q/2,\,q\}}_{\{q/2,\,q\}}$ 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

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

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

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

by Equation (3) and the fact that the guess is correctly with probability ${}^1\!/_n$. We can now define two intermediate games

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

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 is $(t', \epsilon'/(3^2n^3))$-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

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 $i \in P$, we have $v_i = p_i$ then B returns b', that is, A's output. Otherwise, B returns 0.

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

1. We have thus found two games which 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 in order to give a proof of the theorem. In Figure 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 with $I \subseteq P$).

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*

- *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 Appendix A. 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 where $G \in \mathcal{G}_\tau$ if G contains one designated "challenge node" z and the graph T_z is a tree (when ignoring edge directions).

To give more fine-grained bounds we'll 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

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

Notation. As all our graphs will have vertex set $[n]$, with "graph" we also refer to the set of edges of a graph. Thus for two graphs (set of edges) T, T', by $\mathsf{T} \setminus \mathsf{T}'$ we mean the graph T (with vertex set $[n]$) after removing all edges of T' and $\mathsf{T} \cup \mathsf{T}'$ is the graph of all edges of T and T'.

For a node $x \in [n]$ we denote by T_x the graph (set of edges) of all the paths in $G(\mathsf{A})$ that reach x. Recall that a node with in-degree 0 is called a source and denote by $S(v)$ the number of sources in the tree T_v. Note that $S(v)$ is also the number of paths to a node v. For a tree T with at least 2 sources we say a node v *well-divides* T if the number of sources in each subtree obtained from removing all edges of the form $* \to v$, is less than or equal to half of the sources in T. If $(x, y) \in \mathsf{T}$, we call x a parent of y and y a child of x. Furthermore, we let $x[i]$ denote the i-th parent of node x, which means that in the GSD game resulting in $G(\mathsf{A})$ there was a query $\mathsf{encrypt}(x[i], x)$ and before that there were $i-1$ queries of the form $\mathsf{encrypt}(y, x)$ with $y \neq x[i]$.

Our approach. Generalizing the proof for single-source graphs from Section 4, we will define a sequence of hybrids such that there are two games **Game** and **Game$'$** which have the following properties:

- the two games differ in exactly one query, $\mathsf{encrypt}(y, z)$;
- the node y has only one source, i.e., $S(y) = 1$.

Finding **Game** and **Game$'$** will enable us to apply Theorem 1 to these two games and derive contradiction as we did before.

In the single-source game we guessed which node will be in the middle of the path to the challenge, as this allowed us to reduce the problem to smaller problems. Here we need a new metric since there are multiple paths. Whereas before we halved the *length* of the path to the challenge, we now first halve the *number* of such paths, by guessing a node v that well-divides the tree.

Let us look at the original games $\mathbf{Exp}^{\text{GSD-0}}$ and $\mathbf{Exp}^{\text{GSD-1}}$ and suppose they are δ-distinguishable. Unlike for the single-source case, these games differ not in one but in at most d edges: with z denoting the challenge, in $\mathbf{Exp}^{\text{GSD-0}}$ all edges $* \to z$ are real, whereas they are fake in $\mathbf{Exp}^{\text{GSD-1}}$ (and there are at most d of them). We start with defining two new games, where the only change is that we guess the challenge node. Other than that $\mathsf{Game}^{(0)}_{(0)}$ is defined as $\mathbf{Exp}^{\text{GSD-0}}$ and $\mathsf{Game}^{(0)}_{(d)}$ as $\mathbf{Exp}^{\text{GSD-1}}$ (except for the syntactical swap of k_z and r_z). Note that the subscript (d) of Game denotes that the "first d" (that is, all) queries of the form $\mathsf{encrypt}(*, z)$ are replied to with fake ciphertexts. Games $\mathsf{Game}^{(0)}_{(0)}$ and $\mathsf{Game}^{(0)}_{(d)}$ are thus (δ/n)-distinguishable.

We now define $d-1$ intermediate games $\mathsf{Game}^{(0)}_{(i)}$, $1 \leq i \leq d-1$, where in $\mathsf{Game}^{(0)}_{(i)}$ the first i responses to $\mathsf{encrypt}(*, z)$ are fake and the rest are real. For some $i \in [d]$, $\mathsf{Game}^{(0)}_{(i-1)}$ and $\mathsf{Game}^{(0)}_{(i)}$ are thus $(\delta/(dn))$-distinguishable. What we have gained is that now in the sequence of hybrids, two games only differ in the response of one query.

Suppose that $\mathsf{Game}^{(0)}_{(j-1)}$ and $\mathsf{Game}^{(0)}_{(j)}$ are $(\delta/(dn))$-distinguishable and let z denote the challenge. The games differ in the edge $z[j] \to z$, which is real in $\mathsf{Game}^{(0)}_{(j-1)}$ and fake in $\mathsf{Game}^{(0)}_{(j)}$. The node $z[j]$ could itself still have s sources, so our next step is to reduce this. We thus define two new games where we guess a well-dividing node for $\mathsf{T}_{z[j]}$. We define $\mathsf{Game}^{(0,j)}_{(j-1)}$ as $\mathsf{Game}^{(0)}_{(j-1)}$, where in addition to the challenge, we guess the node v that well-divides $\mathsf{T}_{z[j]}$, that is the tree above the j-th edge that goes into z. We analogously define $\mathsf{Game}^{(0,j)}_{(j)}$ and have that $\mathsf{Game}^{(0,j)}_{(j-1)}$ and $\mathsf{Game}^{(0,j)}_{(j)}$ are $(\delta/(dn^2))$-distinguishable.

Having found a well-dividing node v, we can now define $2d$ intermediate hybrids, where we replace every edge $* \to v$, one by one, with a fake edge and call these hybrids $\mathsf{Game}^{(0,j)}_{(j-1,i)}$ for $i = 1, \ldots, d$. $\mathsf{Game}^{(0,j)}_{(j-1,i)}$ is thus defined as $\mathsf{Game}^{(0,j)}_{(j-1)}$, except that the first i answers to queries $\mathsf{encrypt}(*, v)$ are fake. In $\mathsf{Game}^{(0,j)}_{(j-1,d)}$, our splitting node v has thus only fake ingoing edges, which means that in the key graph we have removed $S(v)$ real sources from the challenge and added one source, namely v. The next hybrid is $\mathsf{Game}^{(0,j)}_{(j,d)}$ (where we changed the edge $z[j] \to z$ from real to random), and then a sequence of hybrids $\mathsf{Game}^{(0,j)}_{(j,i)}$ for $i = d, \ldots, 1$, which changes back the edges $* \to v$ from fake to real.

What have we gained by this? Since v well-divided the tree, each differing edge between two consecutive hybrids in the sequence

$$\mathsf{Game}^{(0,j)}_{(j-1)} = \mathsf{Game}^{(0,j)}_{(j-1,0)}, \ldots, \mathsf{Game}^{(0,j)}_{(j-1,d)}, \mathsf{Game}^{(0,j)}_{(j,d)}, \ldots, \mathsf{Game}^{(0,j)}_{(j,0)} = \mathsf{Game}^{(0,j)}_{(j)} \tag{6}$$

has at most $\lceil s/2 \rceil$ sources. Moreover, there must be two consecutive hybrids that are $\delta/((2d+1)dn^2)$-distinguishable .

Continuing with this pair and guessing a well-dividing node (paying a factor $1/n$), we embed $2d$ intermediate hybrids, of which a pair must be $(\delta/((2d+1)^2dn^3))$-distinguishable. The differing edge in this pair has now only $\lceil s/4 \rceil$ source nodes. We can now continue this recursion, e.g., guessing and then embedding between games $\mathsf{Game}^{(0,j_1,j_2)}_{(j_1,j_2-1)}$ and $\mathsf{Game}^{(0,j_1,j_2)}_{(j_1,j_2)}$ the sequence

$$\mathsf{Game}^{(0,j_1,j_2)}_{(j_1,j_2-1)} = \mathsf{Game}^{(0,j_1,j_2)}_{(j_1,j_2-1,0)}, \ldots, \mathsf{Game}^{(0,j_1,j_2)}_{(j_1,j_2-1,d)}, \mathsf{Game}^{(0,j_1,j_2)}_{(j_1,j_2,d)}, \ldots, \mathsf{Game}^{(0,j_1,j_2)}_{(j_1,j_2,0)} = \mathsf{Game}^{(0,j_1,j_2)}_{(j_1,j_2)} \ .$$

But what happens if in Equation (6), the middle games $\mathsf{Game}^{(0,j)}_{(j-1,d)}$ and $\mathsf{Game}^{(0,j)}_{(j,d)}$ are distinguishable? They differ in the j-th edge to the challenge, i.e., $z[j] \to z$. Since in both games all edges going into v (the first splitting node) are fake, we now need to find a node which well-divides the tree $\mathsf{T}_z \setminus \mathsf{T}_v$, as this is the tree containing all the real sources leading to z. Whereas $j \in [d]$ in the superscript denoted guessing a well-dividing node for the j-th tree above, we denote by '0' the fact that the guessed node must well-divide the tree below. We thus obtain $\mathsf{Game}^{(0,j,0)}_{(j-1,d)}$ and $\mathsf{Game}^{(0,j,0)}_{(j,d)}$ from $\mathsf{Game}^{(0,j)}_{(j-1,d)}$ and $\mathsf{Game}^{(0,j)}_{(j,d)}$, respectively, where in addition we guess a node v' well-dividing the tree $\mathsf{T}_z \setminus \mathsf{T}_v$. In any case we obtain a pair that is $(\delta/((2d+1)^3dn^4)$-distinguishable and the differing edge has fewer than $\lceil s/8 \rceil$ sources.

Since, every time we guess and embed intermediate games, we halve the number of sources, after $\lceil \log s \rceil$ such steps we have two games that differ in one edge which has only 1 source, (meaning we have found **Game** and **Game′**). These games are thus $(\delta/d(2d+1)^{\lceil \log s \rceil} n^{\lceil \log s \rceil + 1})$-distinguishable. We now switch to the technique from Section 4, guessing the middle node of the path and embedding 2 intermediate hybrids, losing $1/3n$ in every step. After $\lceil \log \ell \rceil$ (where ℓ is the maximum length of any path), we arrive at two games from which we can construct an IND-CPA distinguisher. We lost another factor $1/(3n)^{\lceil \log \ell \rceil}$. In total, starting from a GSD adversary with success probability δ, we obtain an IND-CPA distinguisher with probability

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

Game_I^P is defined as follows:

- P is a list $(a_1, \ldots, a_{m_P})$, where $a_1 = 0$ and $a_i \in [d]_0$.
- I is a list $(e_1, \ldots, e_{m_I})$ with $m_I \leq m_P$ and $e_i \in [d]_0$
- B starts with guessing nodes $v_1, v_2, \ldots, v_{m_P}$.
- For all $1 \leq i \leq m_I$ and every $e_i = I[i]$, B returns fake responses to the first e_i queries $\mathsf{encrypt}(*, v_i)$. All other queries are responded with real answers. In the end A outputs bit $b \in \{0,1\}$.
- Define $\mathsf{T}^{(1)}, \ldots, \mathsf{T}^{(m_P)}$ from $P, v_1, \ldots, v_{m_P}$ as in Equation (7).
- If v_1 is the challenge node and for $i > 1$, v_i well-divides $\mathsf{T}^{(i)}$, then return b; otherwise return 0.

Figure 5: Definition of Game_I^P for the general-tree case.

To formalize this approach, in Figure 5 we formally define a hybrid game Game_I^P, where P determines the nodes we guess in the graph and I determines the queries (involving the guessed nodes) that are replied to with a fake answer. In contrast to Section 4, P and I are ordered *lists* rather than sets. We explain our definition in the following.

Checking whether we've guessed correctly. We start with explaining how in Game_I^P we check whether our guesses were well-dividing nodes. As an example, consider Figure 3. H_0 corresponds to $\mathsf{Game}_{(0,0)}^{(0,1)}$ (the superscripts mean that we must guess the challenge v_1 (the 0 in $(0,1)$) and a splitting node v_2 for the tree $\mathsf{T}_{v_1[1]}$ that ends in the *first* (denoted by 1 in $(0,1)$) edge going into the challenge; the subscripts mean that there are 0 fake edges going into v_1 and 0 fake nodes going into v_2. Guesses $v_1 = 7$ and $v_2 = 9$ would be correct, since A queried $\mathsf{challenge}(7)$ and $v_2 = 9$ well-divides the first subtree going into the challenge, that is T_{v_1}.

Now consider H_3 and H_4, corresponding to $\mathsf{Game}_{(0,3)}^{(0,1)}$ and $\mathsf{Game}_{(1,3)}^{(0,1)}$, respectively. The next guess v_3 should split the tree $\mathsf{T}_{v_1} \setminus \mathsf{T}_{v_2}$ (obtaining games $\mathsf{Game}_{(0,3)}^{(0,1,0)}$ and $\mathsf{Game}_{(1,3)}^{(0,1,0)}$). If we were to further recurse between these games, we thus never have to consider T_{v_2} anymore, because in all games all edges going into v_2 are fake edges.

The tree $\mathsf{T}^{(i)}$, which v_i should well-divide is thus defined for $i = 1, \ldots$ as follows: as long as there have not been any $a_i = P[i]$ with $a_i = 0$ (that is, we considered a subtree below the splitting node), $\mathsf{T}^{(i)}$ is the tree ending in the a_i-th parent of v_{i-1} (that is, $\mathsf{T}_{v_{i-1}[a_i]}$). When we have $a_i = 0$ for the first time then the lower subtree is chosen, that is $\mathsf{T}^{(i)} = \mathsf{T}^{(i-1)} \setminus \mathsf{T}_{v_{i-1}}$. In any further recursion, we need not consider $\mathsf{T}_{v_{i-1}}$ anymore, so we store it as $\mathsf{R}^{(i)}$ (for "removed"). If $a_{i+1} \geq 1$ then v_{i+1} must well-divide $\mathsf{T}_{v_i[a_{i+1}]} \setminus \mathsf{R}^{(i)}$. We need to apply the same method for the rest of the guessed nodes as well. Meaning we let $\mathsf{R}^{(i)}$ be the set of all the removed edges in the first ith steps and we first remove $\mathsf{R}^{(i)}$ from $\mathsf{T}_{v_{i-1}[a_i]}$ or $\mathsf{T}^{(i)}$ for $a_i \geq 1$ and $a_i = 0$ respectively, and what is left must be well-divided by the new guessed node. Formally,

$$\mathsf{R}^{(1)} = \emptyset \qquad \mathsf{T}^{(1)} = \mathsf{T}_{v_1} \qquad \begin{array}{lll} \text{if } a_i = 0 \text{ then} & \mathsf{R}^{(i)} = \mathsf{R}^{(i-1)} \cup \mathsf{T}_{v_{i-1}}, & \mathsf{T}^{(i)} = \mathsf{T}^{(i-1)} \setminus \mathsf{R}^{(i)}, \\ \text{if } a_i \geq 1 \text{ then} & \mathsf{R}^{(i)} = \mathsf{R}^{(i-1)}, & \mathsf{T}^{(i)} = \mathsf{T}_{v_{i-1}[a_i]} \setminus \mathsf{R}^{(i)} \end{array} \tag{7}$$

Set I and the hybrid games. In Section 4 the set I was a subset of P and for every x in I the query $\mathsf{encrypt}(*, x)$ was responded with a fake answer. Unlike before, here we can have multiple queries of form $\mathsf{encrypt}(*, x)$, therefore I needs to determine which queries are fake for each node in P. To do so, the i-th element in I, e_i, determines that the first e_i queries $\mathsf{encrypt}(*, v_i)$ are responded with fake answers and the rest of queries are responded with real ones. If $e_i = 0$ then all such queries are replied with real answers.

Previously, for each new element z added to P, and for any game Game_I^P, we defined two new hybrid games, one with a fake response and one with a real response to the query $\mathsf{encrypt}(*, z)$. Thus for any pair of games Game_I^P and $\mathsf{Game}_{I'}^P$, with $|I \triangle I'| = 1$ we ended up with 4 games which built up 3 pairs of games satisfying property 1. Here, for a newly guessed node v_i, we can define many new games with $P_{new} = P \parallel (a_i)$ and $I_{new} = I \parallel (e_i)$, for $a_i, e_i \in [d]_0$. However not all of such games are needed. First consider two games Game_I^P and $\mathsf{Game}_{I'}^P$ with $|P| = |I| = m-1$ such that they satisfy Property 1, meaning

- I and I' are identical except at one position i, we have $I[i] = e_i$ and $I'[i] = e'_i$, and $|I| = |I'| = m-1$
- $e'_i < d$ and $e_i = e'_i + 1$, where d is the indegree of v_i,
- If $i < m-1$ then, $a_{i+1} = e_i,\ e_{i+1} = d$ and $\forall j\ , i+1 < j < m,\ e_j = 0,\ a_j = d$.

The differentiating step in these two games is $\mathsf{encrypt}(v_i[e_i], v_i)$. Therefore the new element (to be guessed) must be in the subtree that ends with $v_i[e_i]$. That's why if the $(i+1)$-th element is already in the set P then it must be $a_{i+1} = e_i$, and to add a new element in P we must set $a_m = 0$. On the other hand, if a_i is the last element in S, the new element must be $a_m = e_i$. Thus we only need to define $2(d+1)$ games which create $2d+1$ pairs of games satisfying Property 1, as listed below.

$$\left(\mathsf{Game}_{I\parallel(e_m)}^{P\parallel(a_m)}, \mathsf{Game}_{I\parallel(e_m+1)}^{P\parallel(a_m)}\right) \text{ for } 0 \le e_m < d,$$
$$\left(\mathsf{Game}_{I'\parallel(e_m)}^{P\parallel(a_m)}, \mathsf{Game}_{I'\parallel(e_m+1)}^{P\parallel(a_m)}\right) \text{ for } 0 \le e_m < d,$$
$$\left(\mathsf{Game}_{I\parallel(d)}^{P\parallel(a_m)}, \mathsf{Game}_{I'\parallel(d)}^{P\parallel(a_m)}\right) \ .$$

Finally, one of the pairs of games satisfying property 1, must be $\delta/n(2d+1)$-distinguishable, if Game_I^P and $\mathsf{Game}_{I'}^P$ are δ-distinguishable.

In Lemma 3 we formalize this argument. Let us first show that well-dividing nodes always exist. The proof of the following is in Appendix B.

Lemma 2 *Let T be a tree of n nodes, s sources, $s \ge 2$ and let x be a sink in T with strictly more than one source reaching x. Then there exists a node x^* in T such that x^* well-divides T_x.*

Lemma 3 *Let P and I be sets of integers as defined in Figure 5 and $|P| = |I| = m-1 \ge 0$, and I' be identical to I except for i^*-th element, we have $I[i^*] = e_{i^*}$ and $I'[i^*] = e'_{i^*} = e_{i^*} - 1$. Then there are 2 cases: either*

- *$i^* = m-1$, then let $a_m := e_{i^*}$; or*
- *$i^* < m-1$, $a_{i^*+1} = e_{i^*}$, $e_{i^*+1} = d$ and $\forall j,\ i^*+1 < j < m,\ a_j = 0,\ e_j = d$, then let $a_m := 0$.*

If Game_I^P and $\mathsf{Game}_{I'}^P$ are (t, ϵ)-distinguishable then

- *if $m = \lceil \log s \rceil + 1$ then, $(\mathsf{Enc}, \mathsf{Dec})$ is not $(t + Q_{\mathsf{Adv}} T_{\mathsf{Enc}} + \tilde{O}(Q_{\mathsf{Adv}}), \epsilon/(3n)^{\lceil \log \ell \rceil})$-IND-CPA-secure.*
- *Otherwise, at least one of the following pairs of games is $(t, \epsilon/n(2d+1))$-distinguishable*

$$\mathsf{Game}_{I\parallel(e_m)}^{P\parallel(a_m)} \text{ and } \mathsf{Game}_{I\parallel(e_m+1)}^{P\parallel(a_m)} \quad \text{for } 0 \le e_m < d$$
$$\mathsf{Game}_{I\parallel(d)}^{P\parallel(a_m)} \text{ and } \mathsf{Game}_{I'\parallel(d)}^{P\parallel(a_m)}$$
$$\mathsf{Game}_{I'\parallel(e_m)}^{P\parallel(a_m)} \text{ and } \mathsf{Game}_{I'\parallel(e_m+1)}^{P\parallel(a_m)} \quad \text{for } 0 \le e_m < d$$

See Appendix C for the proof. Guessing the challenge node and defining games $\mathsf{Game}_{(0)}^{(0)}, \ldots, \mathsf{Game}_{(d)}^{(0)}$ loses a factor $n \cdot d$. Applying Lemma 3 repeatedly for $\lceil \log s \rceil$ times, we get the proof of Theorem 2.

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.

[BH92] 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 1992.

[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.

[BRS02] 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 2002.

[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 *ASIACRYPT 2014*, volume ???? of *LNCS*, pages ???–??? Springer, 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.

[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 2000.

A Proof of Lemma 1

If $z = y + 1$ then the pair $(\mathsf{Game}_I^P, \mathsf{Game}_{I\cup\{z\}}^P)$ satisfies Properties 1 and 2. As we'll explain below, this allows us to construct distinguisher D that breaks the IND-CPA security of the encryption scheme with the same advantage, in time t that it takes to run A plus the time that it takes to simulate the game Game_I^P, which consists of at most Q_{Adv} encryption or challenge queries, each taking time T_{Enc}, plus some $\tilde{O}(Q_{\mathsf{Adv}})$ bookkeeping overhead. Thus we get a total running time of of $t + Q_{\mathsf{Adv}}T_{\mathsf{Enc}} + \tilde{O}(Q_{\mathsf{Adv}})$.

In both games the query $\mathsf{encrypt}(p_{y-1}, p_y)$ is answered with $\mathsf{Enc}_{k_{p_{y-1}}}(r_{p_y})$, meaning that k_{p_y} is never encrypted (as there cannot be other queries $\mathsf{encrypt}(x, p_y)$, since A is a $\mathcal{G}_1$ adversary). Moreover, note that p_y cannot be queried to $\mathsf{corrupt}$ either. The query $\mathsf{encrypt}(p_y, p_z)$ is responded to with a real answer $\mathsf{Enc}_{k_{p_y}}(k_{p_z})$ in Game_I^P and with a fake answer $\mathsf{Enc}_{k_{p_y}}(r_{p_z})$ in $\mathsf{Game}_{I\cup\{z\}}^P$.

We now let our distinguisher D against Enc simulate the game Game_I^P implicitly replacing k_{p_y} with its challenger's key (but sampling all remaining real and random keys itself). D answers queries $\mathsf{encrypt}(p_y, x)$, for $x \neq p_z$ by using the encryption oracle. When A queries $\mathsf{encrypt}(p_y, p_z)$, D outputs the challenge $(k_{p_z}), r_{p_z})$ and forwards the answer to A. Depending on D's challenger's bit, D either simulates Game_I^P or $\mathsf{Game}_{I\cup\{z\}}^P$, and thus has the same advantage as A.

For the case $z > y+1$ we have that the node z' is roughly in the middle of y and z on the path. If Game_I^P and $\mathsf{Game}_{I\cup\{z\}}^P$ are (t, ϵ)-distinguishable then games $\mathsf{Game}_I^{P\cup\{z'\}}$ and $\mathsf{Game}_{I\cup z}^{P\cup\{z'\}}$ are $(t, \epsilon/n)$-distinguishable: the guess of $z' \leftarrow [n]$ does not influence A's behavior (it only influences B's output after A has stopped), and the probability of guessing correctly is $1/n$. We therefore have $\Pr[\mathsf{Game}_J^P = 1] = 1/n \cdot \Pr[\mathsf{Game}_J^{P\cup\{z'\}} = 1]$ for any $J \subseteq P$ and thus

$$\Delta_t\Big(\mathsf{Game}_I^{P\cup\{z'\}}, \mathsf{Game}_{I\cup\{z\}}^{P\cup\{z'\}}\Big) = \frac{1}{n} \cdot \Delta_t\Big(\mathsf{Game}_I^P, \mathsf{Game}_{I\cup\{z\}}^P\Big) > \frac{\epsilon}{n} .$$

By the triangle equality we have (recall that $P' = P \cup \{z'\}$):

$$\Delta_t\big(\mathsf{Game}_I^{P'}, \mathsf{Game}_{I\cup\{z'\}}^{P'}\big) + \Delta_t\big(\mathsf{Game}_{I\cup\{z'\}}^{P'}, \mathsf{Game}_{I\cup\{z',z\}}^{P'}\big) + \Delta_t\big(\mathsf{Game}_{I\cup\{z',z\}}^{P'}, \mathsf{Game}_{I\cup\{z\}}^{P'}\big) \geq$$
$$\Delta_t\big(\mathsf{Game}_I^{P'}, \mathsf{Game}_{I\cup\{z\}}^{P'}\big) > \frac{\epsilon}{n} .$$

Thus at least one of the summands in the first line must be greater than $\epsilon/3n$, meaning one of the pairs must be $(t, \epsilon/3n)$-distinguishable.

B Proof of Lemma 2

We proof Lemma 2, by giving an algorithm that finds x^* in T_x in Figure 6.

```
Well-Dividing Node($\mathsf{T}, x$)
$m \leftarrow 0$
$s_{\mathsf{T}} \leftarrow S(x)$
found $\leftarrow$ false
while found = false do
    if $\exists\, i$ with $S(x[i]) > s_{\mathsf{T}}/2$ then
        $m \leftarrow s_{\mathsf{T}} - S(x[i])$
        $x \leftarrow x[i]$
    else
        found $\leftarrow$ true
    end if
end while
output $x$
```

Figure 6: Algorithm finding a well-dividing node in a tree

Let $S(x)$ denote the number of sources in the subtree T_x and let $x[1], \ldots, x[d]$ be all nodes where there is an edge $x[i] \to x$ in T. As long as $S(x) \geq 2$, the following algorithm will output a node x^* in tree T_x such that x^* well-divides T_x

Termination. In each iteration of the while loop either x is replace with one of its parents, which can happen at most n times before reaching a source, or the algorithm terminates. Therefore there can be at most be n iterations of the while loop.

Correctness. The algorithm starts from the sink x of the tree and it traverses the tree up until it finds a well-dividing node. In each iteration, the variable m keeps track of the number of sources in the subtree $\mathsf{T} - \mathsf{T}_x$, where x is the node at hand and T is the tree it started with. In other words, at each iteration of the **while** loop, $m + S(x)$ is the total number of sources in the tree. Note that at most one of the parents of x can satisfy the predicate of the **if** statement and if such a parent exists then that node has more than half of the sources of the entire tree, so we update m and x accordingly and search for the well-dividing node in the new T_x. On the other hand, if no such a parent exists, it means that after removing all the ingoing edges of x from the tree all the subtrees ending in one of the parents of x have less sources than half of the sources of the entire tree. Also note that m is never greater than $s_{\mathsf{T}}/2$, since m is changed only when $S(x[i]) > s_{\mathsf{T}}/2$. Therefore the tree $\mathsf{T} \setminus \mathsf{T}_x$ has less than $s_{\mathsf{T}}/2$ sources as well, meaning x is the well-dividing node.

C Proof of Lemma 3

According to P, I and I', the two games only differ in the response to e_{i^*}-th query of the form $\mathsf{encrypt}(v_{i^*}[e_{i^*}], v_{i^*})$. If $m < \log s$, we add another element to P. The m-th guessed node must well-divide subtree $\mathsf{T}^{(m)} := \mathsf{T}_{v_{i^*}[e_{i^*}]} - \sum_{i \in \{2,3,\cdots,m-1\}: P[i]=0} \mathsf{T}_{v_{i-1}}$ in both games. If such a node exists, the probability of guessing the node correctly is $1/n$, thus $\mathsf{Game}_I^{P\|(a_m)}$ and $\mathsf{Game}_{I'}^{P\|(a_m)}$ are $(t, \epsilon/n)$-distinguishable. Moreover, by the triangle inequality we have

$$\sum_{e_m=0}^{d-1} \Delta_t \left(\mathsf{Game}_{I\|(e_m)}^{P\|(a_m)}, \mathsf{Game}_{I\|(e_m+1)}^{P\|(a_m)} \right) + \Delta_t \left(\mathsf{Game}_{I\|(d)}^{P\|(a_m)}, \mathsf{Game}_{I'\|(d)}^{P\|(a_m)} \right) + \sum_{e_m=0}^{d-1} \Delta_t \left(\mathsf{Game}_{I'\|(e_m)}^{P\|(a_m)}, \mathsf{Game}_{I'\|(e_m+1)}^{P\|(a_m)} \right) \ \geq \ \Delta_t \left(\mathsf{Game}_{I}^{P\|(a_m)}, \mathsf{Game}_{I'}^{P\|(a_m)} \right) > \frac{\epsilon}{n} \ .$$

Consequently at least one of the $2d+1$ pairs of games, on the left hand side of the inequality above, must be $(t, \epsilon/n(2d+1))$-distinguishable. On the other hand if no such a node exists we simply ignore the rest of the guessed nodes in the game. Then $\mathsf{Game}_{I\|(d)}^{P\|(a_m)}$ and $\mathsf{Game}_{I'\|(d)}^{P\|(a_m)}$ are as distinguishable as Game_{I}^{P} and $\mathsf{Game}_{I'}^{P}$. Therefore the claim of the lemma holds in this case as well.

Since every guessing step reduces the number sources in the subtree reaching the challenge into at most half of what it was before, after $m-1 = \lceil \log s \rceil$ steps, we know for certain that there is at most one source left in the subtree reaching the challenge, in which case we can apply Lemma A at most $\lceil \log \ell \rceil$ times to conclude the proof.