Tackling Adaptive Corruptions in Multicast Encryption Protocols

Saurabh Panjwani*

January 30, 2007

Abstract

We prove a computational soundness theorem for symmetric-key encryption protocols that can be used to analyze security against adaptively corrupting adversaries (that is, adversaries who corrupt protocol participants *during* protocol execution). Our soundness theorem shows that if the encryption scheme used in the protocol is semantically secure, and encryption cycles are absent, then security against adaptive corruptions is achievable via a reduction factor of $O(n \cdot (2n)^l)$, with n and l being (respectively) the *size* and *depth* of the key graph generated during any protocol execution. Since, in most protocols of practical interest, the depth of key graphs (measured as the longest chain of ciphertexts of the form $\mathcal{E}_{k_1}(k_2), \mathcal{E}_{k_2}(k_3), \mathcal{E}_{k_3}(k_4), \cdots$) is much smaller than their size (the total number of keys), this gives us a powerful tool to argue about the adaptive security of such protocols, without resorting to non-standard techniques (like non-committing encryption).

We apply our soundness theorem to the security analysis of multicast encryption protocols and show that a variant of the Logical Key Hierarchy (LKH) protocol is adaptively secure (with its security being quasi-polynomially related to the security of the underlying encryption scheme). This settles (in part) an open question posed by Micciancio and Panjwani in TCC 2005.

Keywords: Adaptive Corruptions, Multicast, Encryption, Selective Decryption

*Department of Computer Science and Engineering, University of California, San Diego, Email: `panjwani@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/spanjwan`. This material is based upon work supported by the National Science Foundation under the Grant CNS-0430595

Contents

1 Introduction

Imagine a large group of users engaged in a private virtual conversation over the Internet. The group is monitored by a group manager who ensures that at all points in time, users share a common secret key which is used for secure communication within the group (e.g., for encrypting all data that is exchanged between group members). Over time, the composition of the group changes—users can leave and/or join it at various (a priori unknown) instants—and, accordingly, the manager sends "update" messages to the group which enable all and only current participants to acquire the common secret. At some calamitous hour, a large number of user terminals get hijacked (e.g., an Internet worm infects half the Windows users in the group) and all information possessed by these users gets compromised. Clearly, this results in the compromise of group data that was exchanged while these ill-fated participants were part of the group. The question is—can one be sure that the data for other instants (that is, instants when affected participants were all outside the group) is still secure?

Answering such a question in the affirmative, even for simple security protocols (based on conventional, symmetric-key encryption alone) is often beset with tough challenges. The possibility of user corruptions occurring during protocol execution, and in a manner that is *adaptively* controlled by the attacker, increases the threat to a protocol's security and makes the task of *proving* protocols secure an unnerving task. It is known that, in general, protocols proven secure against non-adaptive attacks may actually turn insecure once an adversary is allowed to corrupt participants adaptively. (See [CFGN96] for a simple separation result for protocols based on secret sharing.) The situation is especially annoying for protocols that make use of encryption—adversaries can spy on ciphertexts exchanged between two honest parties, and later, at will, corrupt one of the parties, acquire its internal state, and use such information to "open" all ciphertexts which were previously sent or received by that party. While trying to prove security of such a protocol, one must argue that all "unopened" ciphertexts (those that cannot be decrypted trivially using the compromised keys) leak essentially no information to the adversary (that is, appear as good as encryptions of random bitstrings). The heart of the problem lies in the fact that one does not a priori know *which* ciphertexts are going to be opened by the adversary since these decisions are made only as the protocol proceeds. Besides, every ciphertext is a binding commitment to the plaintext it hides—one cannot hope to "fool" the adversary by sending encryptions of random bitstrings every time and then, when he corrupts a party, somehow convince him that the ciphertexts he saw earlier on (and which he can now open) were, in fact, encryptions of real data.

Previous Approaches. In the past, security analysis of encryption-based multiparty protocols against adaptive adversaries has largely been conducted using three approaches. The first (and the simplest one) involves bypassing adaptive security altogether—if you cannot prove a protocol adaptively secure, then so be it. (That is, rest your minds with non-adaptive security.) For example, many recent papers on broadcast encryption [AKI03, BGW05, DC06] introduce protocols which improve upon previous proposals significantly (typically in terms of efficiency), but leave the question of adaptive security of their protocols unresolved. This is rather unfortunate. We believe that adaptive corruptions model a particularly interesting attack scenario (and a practical one, too!), and it is worthwhile to explore solutions that address the threats posed by them before improving upon other metrics (like efficiency).

The second approach to adaptive security has worked around the problem by studying it in restricted models where strict rules are imposed on the behaviour of honest participants. The most common imposition is that of *erasure* [BH92]—all honest parties should erase their past state the moment they enter a new state configuration (wherein keys are generated afresh). Intuitively, such an imposition (rather, honest abidance by it) enables us to achieve adaptively secure encryption protocols because adversaries can no longer "open" previously-sent ciphertexts even *after* corrupting the involved parties; doing so requires the keys used to create the ciphertexts in the first place, which, we trust, have been diligently erased from the system. However, investing such a level of trust in honest parties is an unrealistic proposition—an honest party could simply forget to erase its previous states, or else, internally deviate from the rules of the game (that is, purposely store past keys and behave in an "honest-but-curious" manner). Besides, some cryptographic protocols, for the sake of improving efficiency, *require* users to store keys received in the past and such protocols (an example will be discussed in this paper) would need to be re-designed in order to comply with the model.

The third approach, and perhaps the most compelling one, to adaptive security has been to develop non-standard notions of security of an encryption scheme. This corresponds to a line of research initiated by Canetti *et al.* [CFGN96], who introduced a cryptographic primitive, called *non-committing encryption*, specifically to address the problem of adaptive corruptions in multi-party protocols. Non-committing encryption schemes have the unusual property that ciphertexts created using them need not behave as binding commitments on the corresponding plaintexts (hence the name "non-committing"). That is, it is possible that an encryption of '0' collide with an encryption of '1' (or, more generally, encryption of real data be the same as encryption of a random bitstring). However, such collisions occur with only negligible probability—the chances of encrypting '0' and obtaining a ciphertext which can later be opened as '1' are very small. At the same time, these schemes allow to sample "ambiguous" ciphertexts (those that can be opened as either '0' or '1') efficiently and to *convince* an adversary of such a ciphertext being an encryption of '0' or of '1', as the situation demands. Encryption protocols implemented with non-committing encryption can be proven to achieve adaptive security quite easily—in the security proof, one just simulates the real protocol by transmitting ambiguous ciphertexts and upon corruption of a party, convinces the adversary that the ciphertexts he saw earlier were indeed the encryptions of the revealed data. Non-committing encryption schemes, though interesting in their own right, have their share of limitations—they are typically too inefficient for practical applications, and require bounding (a priori) the number of message bits that can be encrypted using any single key (usually, the number of bits that can be encrypted with a key cannot be more than the size of the key itself, which is highly prohibitive for real applications)[1].

Our Contribution. In this paper, we show that it is possible to argue about the adaptive security of a large class of encryption protocols, without requiring erasures and without resorting to primitives like non-committing encryption, while simultaneously achieving efficiency that meets practical requirements. We focus on protocols built generically from symmetric-key encryption (no other primitives are involved) and where every ciphertext is created by encrypting a key or a data element, with a single other key (no nesting of the encryption operation). We show that for a large variety of such protocols if keys are generated randomly and independently of each other, then protocols can be proven adaptively secure, *even under the assumption that the encryption scheme is semantically secure*, with very reasonable assurances on the strength of the protocol against adaptive corruptions.

Our main contribution is a general computational soundness theorem for encryption pro-

[1]As shown by Nielsen [**?**], any non-committing encryption scheme that has a non-interactive encryption procedure must use a decryption key that is at least as long as the total number of bits to be decrypted. Some non-committing encryption schemes [CHK05] circumvent this impossibility result by studying the problem in a restricted model where bounds on the frequency of communication between parties are placed.

tocols which works as follows. Consider an abstract game played between an adversary and a challenger, both being given a security parameter and access to a semantically secure symmetric-key encryption algorithm $\mathcal{E}$. Initially, the challenger generates n random, independent keys $k_1, k_2, \cdots, k_n$ and keeps them secret from the adversary. During the game, the adversary gradually and adaptively builds a directed graph G over n nodes labeled 1 through n. He arbitrarily introduces edges into the graph and for each such edge $i \rightarrow j$ he asks the challenger to provide an encryption of the key k_j under the key k_i, that is, $\mathcal{E}_{k_i}(k_j)$. (Thus, creation of the edge $i \rightarrow j$ in G depicts the fact that given k_i, the adversary can recover k_j, via the decryption operation corresponding to $\mathcal{E}$.) The adversary can also (again adaptively) decide to "corrupt" some nodes in the graph—from time to time, he instructs the challenger to reveal the key associated with the ith node in G (for any arbitrary i) and the challenger must answer with k_i in such a situation. We refer to G as the *key graph* generated by the adversary and the nodes in G that correspond to the revealed keys are called *corrupt nodes*. Note that any node i' in G that is reachable from a corrupt node i is also effectively corrupt; the adversary can recover the corresponding key using successive decryptions along the path from i to i'. The question is—can we prove that, at the end of the game, keys corresponding to nodes that are *not* reachable from any of the corrupt nodes, are still pseudorandom?

This simple game (formalized further in Section 2) provides an effective abstraction for many of the challenges a security analyst can expect to face when proving protocols secure against adaptive corruptions. The power to corrupt nodes in an adaptive fashion models the ability of attackers to compromise keys of users during the execution of the protocol. The power to decide the structure of all ciphertexts abstracts the fact that the execution flow of the protocol is indeterminable at design time and can potentially be influenced by the adversary during run-time. (A slight variant of the game would be one in which the adversary can also acquire ciphertexts formed by encrypting arbitrary messages of his choice. We will discuss this variant further in Section 2.) Note that we allow the creation of ciphertexts even after nodes have been corrupted (that is, the compromise of a key at some point in the protocol should not hamper security of ciphertexts created using future uncompromised keys). Likewise, the security of keys transmitted in the past must be preserved even if other keys are compromised in the future[2].

A naive first step to proving security in the game we just described would be to *guess*, a priori, the set of nodes that the adversary is going to corrupt during the execution and for every edge issuing from such a node, reply with a real ciphertext (while for the other edges reply with encryptions of random bitstrings). Any security reduction seeded with an idea that involves guessing a subset of nodes in this manner would give us a reduction factor that is exponential in n (that is, we would end up proving a statement like *"if the encryption scheme is ϵ-secure then security in the game is guaranteed with probability $2^n\epsilon$"*). Such a reduction would be completely impractical; in most applications, n would be of the order of the number of protocol participants, which can be extremely large.

In this paper, we prove security in this game using a significantly different approach, and one that is of much better practical value. We show that if the key graph G generated by the adversary is acyclic[3] and if its *depth* (defined as the length of the longest path in G) is upper

[2] Note, however, that our abstract game does not capture active attacks on protocols—the adversary cannot ask for decryptions of ciphertexts (under unrevealed keys) and/or send malleated ciphertexts to honest parties. Extending the results of this paper to a setting where active attacks are also allowed is postponed to future work.

[3] Acyclicity of key graphs is an almost-inescapable criterion required in security proofs of protocols based on encryption. For example, it was noted in the seminal work of Goldwasser and Micali [GM84] that semantically secure encryption schemes may not remain secure if keys are allowed to encrypt themselves and the ciphertext given to the adversary.

bounded by a parameter l, then security in our game can be proven via a reduction factor of $O(n \cdot (2n)^l)$. Here, by "security in our game" we mean that keys that (a) cannot be trivially recovered by the adversary (that is, are not reachable from corrupt nodes in G) and (b) are not used to encrypt other keys[4], remain pseudorandom at the end of it. That is, we prove that the security of a semantically secure encryption scheme can degrade in the face of adaptive attacks (as those captured by our game) by a factor of at most $O(n \cdot (2n)^l)$ but not by worse.

An Application. So what is this reduction good for? At first glance, it would appear that it is much worse than the naive solution—l could potentially be of the order of n and $n \cdot (2n)^n$ is obviously no more consoling than 2^n. Well, for arbitrary key graphs, this is indeed the case. However, in practice, key graphs are much smaller (in fact, orders of magnitude smaller) in depth than in total size. For example, the key graphs generated in the execution of most broadcast encryption protocols (those falling under the subset-cover framework introduced by Naor *et al.* [NNL01]) have depth 1 and their depth remains fixed for arbitrarily long runs of the protocol. All encryption-based group key distribution protocols (designed for secure multicast over the Internet, and also called *multicast encryption* protocols) generate key graphs that have depth at most logarithmic in the total number of users in the system (again, the depth remains fixed for arbitrarily long runs of the protocol, once the total space of users has been ascertained). In general, in all encryption protocols, the depth of key graphs created in any execution is likely to be related to the number of decryptions performed by users (in order to be able to recover certain keys or messages) while their total size to the number of users themselves; it is reasonable to expect that protocol designers, for the purpose of efficiency, would strive to keep the former smaller than the latter.

We exemplify the power of our soundness result by applying it to the security analysis of the Logical Key Hierarchy (LKH) protocol [WGL00]. LKH is a protocol originally developed for secure communication in multicast groups on the Internet (applications of the form we discussed in the first paragraph) and has since then attracted a lot of interest from both cryptographers and researchers in the networking community. Surprisingly, even though the protocol gets mentioned in several papers on cryptography, there has been little effort from within our community towards analyzing its security (adaptive or otherwise) rigorously or to make any claims to the contrary.

The original LKH protocol has a security flaw in it [MP06]. Although this flaw is quite easy to spot, we are not aware of any work (prior to ours) that rectifies this flaw in a provably secure manner. (In [MP06], a fix is suggested but not proven secure.) In Section 3 of this paper, we present a variant of LKH which is not only as efficient as the original protocol, but also enjoys strong guarantees of security *against adaptive adversaries.* In particular, we use our soundness theorem to show that the security of the improved protocol is related to the semantic security of the underlying encryption scheme via a reduction factor that is quasi-polynomial in the number of protocol participants. Concretely, our reduction factor is of the order of $\tilde{n}^{\log_2(n)+2}$, where n is the number of users in the protocol and $\tilde{n} = O(n)$.

This reduction factor, though not strictly polynomial in n, is still quite reasonable from a practical perspective. For example, in a system with 128 users, one is guaranteed that an execution of our protocol provides at least 65 bits of adaptive security when implemented with 128-bit AES in counter mode (for a run with upto 64 key updates)[5]. Our result practically eliminates the need for using expensive techniques like non-committing encryption to build adaptively secure

[4]This is a necessary criterion if our goal is to guarantee pseudorandomness of these keys.

[5] These numbers are computed assuming the protocol is implemented using a binary hierarchy of keys; for non-binary hierarchies, the security guarantee is actually better. Also, the security bound can be considerably improved by using 192-bit or 256-bit AES.

multicast encryption protocols, and it does this while matching the efficiency of existing schemes.

Relation with Selective Decryption. The abstract game used in our soundness theorem is reminiscent of the well-studied (though largely unresolved) problem of *selective decryption.* In this problem (like in ours), an adversary interacts with a challenger who initially generates a set of plaintexts $m_1, \cdots, m_n$ and a corresponding set of keys $k_1, \cdots, k_n$. (We stress here that the plaintexts are not chosen by the adversary, but generated by the challenger using some fixed distribution.) The adversary first wants to see the encryptions of all the plaintexts, $\{\mathcal{E}_{k_i}(m_i)\}_{i=1}^n$, and later "open" some of them adaptively; that is, he queries an arbitrary set $I \subseteq [n]$ and the challenger replies with $\{k_i\}_{i \in I}$. The question now is to show that plaintexts corresponding to all unopened ciphertexts are still "safe", in the sense that the adversary cannot learn any more information about them than what he could learn from the revealed plaintexts. In our soundness theorem, we are essentially generalizing this game to a setting in which the adversary can ask for not only *single* ciphertexts but *chains* of ciphertexts of the form $\mathcal{E}_{k_1}(k_2), \mathcal{E}_{k_2}(k_3), \mathcal{E}_{k_3}(k_4), \cdots$ and he is also allowed to open such chains adaptively (as above). Plus, we allow the adversary to interleave his "encrypt" and "open" queries arbitrarily. (Indeed, the fact that ciphertexts can be asked for in an adaptive manner, possibly depending upon past corruptions, is responsible for much of the complication in our proof.) It is for this reason that we refer to our game (detailed in Section 2) as the *generalized selective decryption (GSD)* game.

Does this paper solve the selective decryption problem? Not really. A crucial ingredient of that problem is the distribution from which the plaintexts $m_1, \cdots, m_n$ are drawn by the challenger. It has been shown [DNRS03] that if this distribution is such that each plaintext can be generated independently of the others then the unopened ciphertexts indeed remain secure and the adversary learns essentially no partial information about the plaintexts they hide from his interaction with the challenger. In the GSD game, too, we require all keys, even those which are not used for further encryption, to be generated independently of each other, and this "independence property" is crucial in our proof[6]. Our soundness theorem essentially builds up on this positive result for selective decryption and extends it to the more general scenario of arbitrarily (and adaptively) generated key graphs. The question of solving selective decryption without the independence assumption on plaintexts still remains open.

We remark here that independence of all keys is not just a simplifying assumption in our theorem; it is almost a requirement for the security of the protocols we are interested in analyzing. A multicast encryption protocol that uses related keys across key updates may not guarantee good security at all.

Related Work. The notion of computational soundness theorems was introduced by Abadi and Rogaway [AR02], and has since then found applications in the security analysis of various cryptographic tasks, including key exchange [DDMW06, CH06], mutual authentication [MW04, CH06], XML security [AW05] and multicast key distribution [MP05, MP06]. Although most of the literature on computational soundness theorems deals with protocols that make use of encryption as the fundamental primitive, to the best of our knowledge, none of these works prove soundness in the presence of adaptively corrupting adversaries. Recently, Gupta and Shmatikov [GS06] developed a symbolic logic that allows reasoning about a weak variant of adaptive security for the case of key exchange protocols; however, the protocols they analyze, do not make use of encryption (and instead use Diffie-Hellman exponentiation coupled with

[6] Jumping ahead, we remark that even in the variant in which the adversary can acquire encryptions of arbitrary messages of his choice, we need only keys to be independent of each other, and *not* the messages.

signatures).

The soundness result of this paper is of a very different flavor than those in previous works in the area. The protocol model we use is relatively simpler—in the protocols we consider, every message generated during an execution is either a key or an encryption of a key under a key or else, a sequence of values with one of these types[7]. Symbolic analysis of such protocols can be effectively conducted using graph-theoretic terminology: keys can be interpretted as nodes, ciphertexts as edges, and Dolev-Yao attacks on protocols can be expressed in terms of reachability from adversarial nodes (corresponding to corrupted keys). As such, all discussions on symbolic analysis in this paper take place within a graph-theoretic framework (as illustrated by the GSD game). This simplifies our presentation considerably and brings us quickly to the crux of the matter.

Lastly, a few words comparing the result of this paper with our previous work, joint with Micciancio [MP05, MP06], on the computationally sound analysis of encryption protocols are in order. Although both our works address adaptive attacks on encryption protocols in general, the adversarial model used in the current work is stronger: we not only allow the adversary to adaptively modify the execution flow of the protocol (as in our past work) but also to corrupt participants in an adaptive manner. Tackling the latter type of attacks is significantly more non-trivial, and is the central theme of this paper. Another important difference is that our previous soundness results applied only to protocols that satisfied certain syntactic conditions *besides* acyclicity of key graphs. Informally, these conditions require protocols to use every key in two phases—a *distribution* phase in which keys are used as plaintexts, followed by a *deployment* phase in which the distributed keys are used for encrypting other keys or messages. Key distribution is not allowed to succeed key deployment. Such a condition, though not extremely restrictive, does raise some concerns. It is not hard to conceive scenarios in which one would want to distribute previously deployed keys. (In fact, some existing protocols do this, too; e.g., the protocol of [PST01] "re-distributes" keys regularly in order to compensate for packet losses.) Our new result, while incorporating adaptive corruptions, also does away with this restriction. The downside, however, is that this result provides security guarantees in a manner that is dependent on the depth of protocol key graphs, and it is not meaningful for protocols that generate arbitrary-depth key graphs. We believe that improving the result of this paper to overcome this limitation is non-trivial, but a worthy direction for future research; in particular, obtaining an analogous result with a reduction factor smaller than $\Theta(n^l)$ would be quite remarkable, and could lead to even newer techniques to address adaptive corruptions in security protocols.

2 The Main Result

Let $\Pi = (\mathcal{E}, \mathcal{D})$ be a symmetric-key encryption scheme[8]. We use the standard notion of indistinguishability against chosen plaintext attacks (Ind-CPA) for encryption schemes as defined by Bellare *et al.* [BDJR97]. Let $\mathcal{O}^{\Pi}_{k,b}$ denote a left-or-right oracle for Π which first generates a key k uniformly at random from $\{0,1\}^\eta$ (η being the security parameter) and subsequently,

[7]We remark that extending our result to protocols that use nested encryption is also possible, but the soundness theorem and the corresponding proof become much more complex. We avoid nested encryption largely for the sake of simplicity (and partly because most existing multicast encryption protocols don't use nesting).

[8]In this paper, we consider encryption schemes where key generation is defined by picking a uniformly random bitstring from the set $\{0,1\}^\eta$ with η being the security parameter. Thus, the key generation algorithm is implicit in the definition of encryption schemes. We also assume that the encryption scheme allows to encrypt arbitrary bitstrings; so, keys themselves can always be used as plaintexts.

responds to every query of the form $(m_0, m_1) \in \{0,1\}^* \times \{0,1\}^*$ (such that $|m_0| = |m_1|$) with $\mathcal{E}_k(m_b)$—the encryption of m_b under key k. For any adversary (that is, any arbitrary probabilistic Turing machine) A, let $\mathsf{A}^{\mathcal{O}^{\Pi}_{k,b}}$ denote the random variable corresponding to the output of A when interacting with such an oracle.

Definition 2.1 Let $t \in I\!N^+$ and $0 < \epsilon < 1$. An encryption scheme Π is called (t, ϵ)-Ind-CPA secure if for every adversary A running in time t, we have

$$|\mathbf{P}[\mathsf{A}^{\mathcal{O}^{\Pi}_{k,b}} = 1 \mid b = 0] - \mathbf{P}[\mathsf{A}^{\mathcal{O}^{\Pi}_{k,b}} = 1 \mid b = 1]| \leq \epsilon$$

The GSD game. Consider the following game, which we call the *generalized selective decryption (GSD)* game, played between an adversary A and a challenger B, both being given blackbox access to a symmetric-key encryption scheme $\Pi = (\mathcal{E}, \mathcal{D})$. The game is parameterized by an integer n, assumed to be known to both A and B. In the beginning, the challenger generates a set of keys, $k_1, k_2, \cdots, k_n$, each key being sampled independently and uniformly at random from the set $\{0,1\}^\eta$ (where η is the security parameter). B also generates a *challenge* bit b, uniformly at random from $\{0,1\}$, which A is required to guess in the end. It stores the generated values for the rest of the game, and uses them to answer all of A's queries.

A can make three types of queries to B:

1. `encrypt`: At any point, A can make a query of the form `encrypt`(i, j), in response to which B creates a ciphertext $c \leftarrow \mathcal{E}_{k_i}(k_j)$ (using fresh coins for the encryption operation each time) and returns c to A.

2. `corrupt`: A can also ask for the value of any key initially generated by B; it does this by issuing a query of the form `corrupt`(i), in response to which it receives k_i.

3. `challenge`: Finally, A can issue a query of the form `challenge`(i). The response for such a query is decided based on the bit b: if $b = 0$, B returns the key k_i to A, whereas if $b = 1$, it generates a value r_i uniformly at random from $\{0,1\}^\eta$, and sends r_i to A[9].

Multiple queries of each type can be made, interleavingly and adaptively. We stress here that A can make more than one `challenge` queries in the game and it can choose to interleave its `challenge` queries with the other two types of queries. (This is a slight generalization of the setting described in the introduction.) Giving the adversary the power to make multiple `challenge` queries models the requirement that keys linked with challenge nodes be "jointly" pseudorandom (as opposed to individual keys being pseudorandom by themselves). Allowing it to interleave `challenge`'s with other queries means that such keys are required to retain their pseudorandomness even after more corruptions or ciphertext transmissions have occurred.

We think of the queries of A as creating a directed graph over n nodes (labeled $1, 2, \cdots, n$), edge by edge, and in an adaptive fashion. Each query `encrypt`(i, j) corresponds to creating an edge from i to j, denoted $i \to j$, in this graph. For any adversary A, the graph created by its queries in this manner is called the *key graph* generated by A and is denoted $G(\mathsf{A})$. A node i in $G(\mathsf{A})$ for which A issues a query `corrupt`(i) is called a *corrupt node* while one for which A issues a query `challenge`(i) is referred to as a *challenge node*. The set of all corrupt nodes is denoted $V^{\mathsf{corr}}(\mathsf{A})$ and that of all challenge nodes is denoted $V^{\mathsf{chal}}(\mathsf{A})$. Note that $G(\mathsf{A}), V^{\mathsf{corr}}(\mathsf{A})$ and $V^{\mathsf{chal}}(\mathsf{A})$ are all random variables depending on the coins used by both A and B. Without loss of generality, we assume that $V^{\mathsf{chal}}(\mathsf{A})$ is always non-empty and in any execution of A, every

[9]If A issues multiple `challenge` queries with argument i and if b equals 1, B must return the same value r_i everytime.

node $i \in V^{\mathsf{chal}}(\mathsf{A})$ has at least one edge incident upon it. (Put differently, this means that A always makes at least one query of the form $\texttt{challenge}(i)$ and for each such query, it makes at least one query of the form $\texttt{encrypt}(x, i)$.)

Legitimate Adversaries. There is a trivial way in which any adversary can win in the GSD game—by corrupting a node i in $G(\mathsf{A})$ and making a query $\texttt{challenge}(j)$ for any j that is reachable from i, A can easily compute the challenge bit b. The interesting case to consider is, thus, one in which A is constrained *not* to issue queries of this form, that is, where A is restricted to make queries in a manner such that no challenge node is reachable from a corrupt node in $G(\mathsf{A})$.

Our intuition suggests that if the encryption scheme is secure (in the Ind-CPA sense), then the chances of such an adversary being able to decipher b correctly are no better than half. However, translating this intuition into a proof is far from easy. For one, it is not even possible to do this without further restrictions on the adversary's queries: if a key k_j is used to encrypt other keys (that is, there exists an edge issuing from j in $G(\mathsf{A})$), then k_j cannot be guaranteed to remain pseudorandom, even if j is not reachable from the corrupt nodes. In other words, we can hope to prove pseudorandomness of keys associated with challenge nodes only as long as these nodes have no outgoing edge in $G(\mathsf{A})$. Secondly, arguing about the security of encryption schemes in the presence of key cycles is a gruelingly hard problem; in particular, it is currently not known whether an arbitrary Ind-CPA-secure encryption scheme can be proved to retain its security in a situation where ciphertexts of the form $\mathcal{E}_{k_1}(k_2), \mathcal{E}_{k_2}(k_3), \cdots, \mathcal{E}_{k_{t-1}}(k_t), \mathcal{E}_{k_t}(k_1)$, for some $t > 1$, are created using it. Standard techniques do not allow to prove such statements and counterexamples are not known either. Given this state of affairs, our only hope to prove security in the GSD game is to forbid the creation of key cycles altogether.

We formalize all our requirements from the adversary in the following definition:

Definition 2.2 An adversary A is called *legitimate* if in any execution of A in the GSD game, the values of $G(\mathsf{A}), V^{\mathsf{corr}}(\mathsf{A})$ and $V^{\mathsf{chal}}(\mathsf{A})$ are such that:

1. For any $i \in V^{\mathsf{corr}}(\mathsf{A})$ and any $j \in V^{\mathsf{chal}}(\mathsf{A})$, j is unreachable from i in $G(\mathsf{A})$.
2. $G(\mathsf{A})$ is a DAG and every node in $V^{\mathsf{chal}}(\mathsf{A})$ is a sink in this DAG.

The first condition above is essentially a symbolic security criterion—what it means for the challenge nodes (or keys) to be secure against a Dolev-Yao adversary. The second condition spells out the syntactic restrictions that are required to prove the soundness of this symbolic criterion with respect to the actual (computational) adversary A we are considering.

The Result. Let A be any legitimate adversary playing the GSD game. We say that A is an (n, e, l)-adversary if in any execution, the number of nodes and edges in the key graph generated by A are bounded from above by n and e respectively and the *depth* of the graph (the length of the longest path in it) is at most l. We denote the random variable corresponding to the output of A in the game by $\mathsf{A}^{\mathsf{B}_b^\Pi}$.

Definition 2.3 Let $t, n, e, l \in I\!N^+$ and $0 < \epsilon < 1$. An encryption scheme Π is called (t, ϵ, n, e, l)-GSD secure if for every legitimate (n, e, l)-adversary A running in time t, we have

$$|\mathbf{P}[\mathsf{A}^{\mathsf{B}_b^\Pi} = 1 \mid b = 0] - \mathbf{P}[\mathsf{A}^{\mathsf{B}_b^\Pi} = 1 \mid b = 1]| \leq \epsilon$$

Here, probabilities are taken over the random choices made by both A and B (including the randomness used by B in creating ciphertexts). The following is the main result of this paper:

Theorem 2.4 Let $t, n, e, l \in I\!N^+$ and $0 < \epsilon < 1$. If an encryption scheme Π is (t, ϵ)-Ind-CPA secure, then it is (t', ϵ', n, e, l)-GSD secure for quantities t' and ϵ' defined as:

$$\epsilon' = \epsilon \cdot \frac{3n^2}{2} \cdot (2n+1)^{l-1}$$

$$t' = t - (O(n) \cdot t_{\text{GenKey}} + e \cdot t_{\text{Encrypt}})$$

where t_{GenKey} (resp. t_{Encrypt}) denotes the time taken to perform key generation (resp. encryption) in Π.

Overview of the Proof. The starting point of the proof of our theorem is the positive result on the selective decryption problem (more precisely, the selective *decommitment* problem) due to Dwork *et al.* [DNRS03]. Consider first the GSD game for the case $l = 1$. The graph $G(\mathsf{A})$ in this case is a directed bipartite graph mapping a set of sources to a set of sinks. (In the problem studied in [DNRS03], the map from sources to sinks is one-to-one. In our case, it could be many-to-many; plus, it could be adaptively generated based on previous corruptions.) How can we argue about security in this case? Intuitively, an attacker's ability to differentiate between real and random values for *all* nodes in $V^{\mathsf{chal}}(\mathsf{A})$ translates into its ability to differentiate between the two values for *some* node (say the jth one) in $V^{\mathsf{chal}}(\mathsf{A})$; that is, such an adversary can effectively differentiate between two worlds, one in which the reply to each of the first $j-1$ queries of the form $\texttt{challenge}(i)$ is r_i (and for the rest, it is k_i), and the other in which the reply to each of the first j queries of this form is r_i (and that for the rest is k_i).

Let us call these worlds $\mathsf{World}_j(0)$ and $\mathsf{World}_j(1)$ respectively. Let us assume that the argument specified in A's jth $\texttt{challenge}$ query is known a priori (it can be guessed with success probability $1/n$) and equals i_j. Let $I(i_j)$ denote the set of nodes i_s for which there exists an edge $i_s \to i_j$ in $G(\mathsf{A})$. Now consider this modified version of the GSD game: While generating keys in the beginning, B also generates a random key $\tilde{k}_{i_j}$, independently of all other keys. It replies to the adversary's queries in one of two worlds again, but now the worlds are defined as follows. Each query of the form $\texttt{encrypt}(i_s, i_j)$ is replied to with the real ciphertext $\mathcal{E}_{k_{i_s}}(k_{i_j})$ in the first world, $\mathsf{World}'_j(0)$, but with a *fake* one, namely $\mathcal{E}_{k_{i_s}}(\tilde{k}_{i_j})$, in the other one, $\mathsf{World}'_j(1)$. All other $\texttt{encrypt}$ queries are replied to with real ciphertexts in both worlds. For the $\texttt{challenge}$ queries the replies always have the same distribution—r_i for the first $j-1$ $\texttt{challenge}$ queries and k_i for the rest. (In particular, the reply for $\texttt{challenge}(i_j)$ is always k_{i_j}.)

It is easy to see that the distribution on the challenger's replies in $\mathsf{World}'_j(0)$ is exactly the same as in $\mathsf{World}_j(0)$. (The replies to all $\texttt{encrypt}$, $\texttt{corrupt}$ and $\texttt{challenge}$ queries are decided using the same procedure.) The key observation to make here is that the distribution on the replies in $\mathsf{World}'_j(1)$ is also the same as that in $\mathsf{World}_j(1)$! This is true because the keys $k_{i_j}, \tilde{k}_{i_j}$ and r_{i_j} are generated by the challenger independently of each other, and so, replying to $\texttt{encrypt}(i_s, i_j)$ with $\mathcal{E}_{k_{i_s}}(k_{i_j})$ and $\texttt{challenge}(i_j)$ with r_{i_j} (as done in $\mathsf{World}_j(1)$) produces the same distribution as replying to the former with $\mathcal{E}_{k_{i_s}}(\tilde{k}_{i_j})$ and the latter with k_{i_j} (as done in $\mathsf{World}'_j(1)$). Thus, our adversary can differentiate between $\mathsf{World}_j(0)$ and $\mathsf{World}_j(1)$ with the same probability as it can differentiate between $\mathsf{World}'_j(0)$ and $\mathsf{World}'_j(1)$.

Why are the two worlds $\mathsf{World}'_j(0)$ and $\mathsf{World}'_j(1)$ indistinguishable? Because the encryption scheme is Ind-CPA-secure. If the adversary can distinguish between two sets of ciphertexts $\{\mathcal{E}_{k_{i_s}}(k_{i_j})\}_{i_s \in I(i_j)}$ (the real ones) and $\{\mathcal{E}_{k_{i_s}}(\tilde{k}_{i_j})\}_{i_s \in I(i_j)}$ (the fake ones) then it must be able to tell the difference between $\mathcal{E}_{k_{i_s}}(k_{i_j})$ and $\mathcal{E}_{k_{i_s}}(\tilde{k}_{i_j})$ for *some* node $i_s \in I(i_j)$. (A standard hybrid argument applies here[10].) This goes against the Ind-CPA-security of Π.

Going beyond $l = 1$. In the general setting, a node i_s, pointing at any node $i_j \in V^{\mathsf{chal}}(\mathsf{A})$ need not be a source—there could be other edges incident upon each such i_s and extending the above argument to this general setting requires more work. In order to be able to make a statement like *"the ciphertext $\mathcal{E}_{k_{i_s}}(k_{i_j})$ is indistinguishable from $\mathcal{E}_{k_{i_s}}(\tilde{k}_{i_j})$"*, one must first argue that every ciphertext of the form $\mathcal{E}_{k_{i'_s}}(k_{i_s})$ (where $i'_s \to i_s$ is an edge in $G(\mathsf{A})$) looks the same as one of the form $\mathcal{E}_{k_{i'_s}}(\tilde{k}_{i_s})$ (a fake ciphertext). But every such $k_{i'_s}$ could, in turn, be encrypted under other keys (that is, the node i'_s could have other edges incident on it). There could be a lot of nodes ($O(n)$, in general) from which i_j is reachable in $G(\mathsf{A})$ and at some point or the other, we would need to argue that replying with real ciphertexts created under each of these nodes is the same as replying with fake ones. Worse still, we do not a priori know the set of nodes from which i_j can be reached in $G(\mathsf{A})$ since the graph is created adaptively; so we must make guesses in the process.

It is easy to come up with an argument where the amount of guesswork involved is exponential in n (simply guess the entire set of nodes from which there is a path to i_j). In our proof, however, we take a radically different approach. We first define a sequence of hybrid distributions on the replies given to A such that in each of the distributions, the replies corresponding to some of the edges in the key graph are fake, *and* these "faked" edges are such that their end-points lie on *a single path* ending in i_j. (Henceforth, we will refer to every edge for which the corresponding reply is fake, as a *faked* edge.) The extreme hybrid distributions are defined as in the two worlds $\mathsf{World}'_j(0)$ and $\mathsf{World}'_j(1)$ for $l = 1$: in one extreme, the replies corresponding to all edges are real, and in the other extreme, the replies corresponding to all edges incident on i_j are fake (while the rest of the replies are still real). Intermediate to these extremes, however, are several distributions in which edges other than those incident on i_j are faked. For any two adjacent distributions in the sequence of distributions, the following properties are always satisfied:

(a) The distributions differ in the reply corresponding to a *single* edge $i_s \to i_t$; the reply is real in one distribution while fake in the other.

(b) In both distributions, for every $i_r \in I(i_s)$, the edge $i_r \to i_s$ is faked.

(c) There exists a path from i_t to i_j in the key graph and in both distributions, "some" of the edges incident upon this path are faked, the faked edges being the same in both distributions.

(d) No other edge in the key graph is faked in either of the distributions.

Properties (a) and (b) are meant to ensure that any two adjacent hybrids can be simulated using a single left-or-right encryption oracle (and so, A's capability to distinguish between them would imply that the encryption scheme is not Ind-CPA-secure); properties (c) and (d) enable the simulation to be carried out by guessing a path (that goes from i_s to i_t to i_j) as opposed to guessing all the nodes from which i_j is reachable. (This partly explains why our reduction factor is exponential in the depth, rather than the size, of the key graph.) In order to simultaneously achieve all these properties, we order the hybrid distributions such that *(i)* when the reply for any edge $i_s \to i_t$ is changed (from real to fake or vice versa) in moving from one hybrid to another, all edges of the form $i_r \to i_s$ have already been faked in previous hybrids; and *(ii)* after

[10]The reduction factor in this hybrid argument would be at most n. This combined with the guessing probability $(1/n)$ associated with the node i_j defined above gives us a gross reduction factor of $O(n^2)$, as desired for $l = 1$.

changing the reply for $i_s \to i_t$, there is a sequence of hybrids in which the replies for all edges $i_r \to i_s$ are, step by step, *changed back from fake to real*. This is done in order to satisfy property (d) above (particularly, to make sure that it is satisfied when the replies for edges issuing from i_t are changed in a subsequent hybrid).

Thus, if we scan the sequence of hybrid distributions from one extreme to the other, we observe both "real-to-fake" and "fake-to-real" transitions in the replies given to A, taking place in an oscillating manner. The oscillations have a recursive structure—for every oscillation in replies (transition from real to fake and back to real) for an edge $i_s \to i_t$, there are two oscillations (transition from real to fake to real to fake to real) for every edge $i_r \to i_s$ incident upon i_s. Simulating these hybrid distributions (using a left-or-right oracle) and subsequently, arguing that the simulation works correctly is the most challenging part of the proof. After developing an appropriate simulation strategy, we prove its correctness using an inductive argument—assuming that, for some $l' \leq l$, the simulation behaves correctly whenever i_s is at depth smaller than l' in the key graph, we show that the simulation is correct also when i_s is at depth smaller than $l'+1$; this simplifies our analysis considerably. The details of the proof are given in the appendix of the paper.

Other Variants. A natural variant of the GSD game would be one in which the adversary is allowed to acquire encryptions of messages of its choice (besides receiving encryptions of keys, as in the original game). Consider the following modified version of the game: A issues `encrypt` and `corrupt` queries, as before, but instead of making `challenge` queries, it makes queries of the form $\texttt{encrypt\_msg}(i, m_0, m_1)$ (such that $m_0, m_1 \in \{0,1\}^*$ and $|m_0| = |m_1|$). In return for each such query, the challenger sends it the ciphertext $\mathcal{E}_{k_i}(m_b)$. A legitimate adversary in this modified game would be one whose key graph is always a DAG and for whom every query $\texttt{encrypt\_msg}(i, m_0, m_1)$ is such that i is unreachable from the corrupt nodes in the DAG. (*Note: i need not* be a sink in the DAG.) We remark that a result analogous to Theorem 2.4 can also be proven for this modified game, and with only a slight modification to the proof of that theorem. Specifically, we can show that if Π is (t, ϵ)-Ind-CPA secure, then for any t'-time (n, e, l) adversary A (t' as defined in Theorem 2.4),

$$|\mathbf{P}[\mathsf{A}^{\mathsf{B}_b^\Pi} = 1 \mid b = 0] - \mathbf{P}[\mathsf{A}^{\mathsf{B}_b^\Pi} = 1 \mid b = 1]| \leq \epsilon \cdot \frac{3n}{2} \cdot (2n+1)^l$$

A different variant of our game would be one in which A is provided encryptions of messages, but these messages are sampled by the *challenger* using some fixed distribution known to A. In this variant, the messages themselves can be thought of as nodes (more specifically, sinks) in the key graph, whose values are hidden from A but whose probability distribution is defined differently from that of the keys. The goal now would be to argue that from A's perspective, all "unopened" messages (that is, messages that are not reachable from corrupt nodes in the key graph) appear as good as fresh samples from the same message space. If we assume that messages are sampled independently of each other, then security in this variant can also be proven, and with almost the same reduction factor as in Theorem 2.4. (Specifically, the reduction factor would be $(3/2) \cdot Mn(2n+1)^l$, where M is an upper bound on the total number of messages that are encrypted.) However, in the absence of this assumption, it becomes considerably more challenging to prove the same claim. The techniques developed in this paper do not allow us to argue about security in such a setting, not even in the case where the key graph has depth 0 (only messages, and not keys, are used as plaintexts)[11].

[11]Here, by "argue about security" we mean the following: Consider an adversary A who makes only `encrypt` and `corrupt` queries in the above variant of the GSD game. At the end of the game, give the adversary one out

3 The Application

In this section, we illustrate how our result from Section 2 applies to the security analysis of multicast encryption protocols.

Multicast Encryption. A group of n users, labeled $U_1, \cdots, U_n$, share a broadcast channel and wish to use it for secure communication with each other. At any point in time t, only a subset of users, labeled S_{t}, are "logged in" to the network, that is, are authorized to receive information sent on the channel. We would like to ensure that for all t, only the users in S_{t} (called *group members*) be able to decipher the broadcasts. We assume the existence of a central group manager C who shares a unique long-lived key k_{U_i} with each user U_i[12] and runs a key distribution program, KD, in order to accomplish the said task. The manager (or, equivalently, the program KD) receives user `login` and `logout` requests[13] and for the request at time t, sends out a set of *rekey messages*, $\mathcal{M}_{\mathsf{t}}$, on the channel. These rekey messages carry information about a key $k[\mathsf{t}]$ (the *group key* for t), and are such that only the group members can decipher them (and, subsequently, recover $k[\mathsf{t}]$). The key $k[\mathsf{t}]$ can then be used to carry out all group-specific security tasks until the next `login/logout` request arrives, which, we assume, happens at time $\mathsf{t}+1$. For example, it can be used for ensuring privacy of all data sent between time t and $\mathsf{t}+1$ and/or guaranteeing "group authenticity" of data (that is, enabling members to verify that the sender of the data is a group member at time t, and not an outsider). To ensure security of any such task, it is important to guarantee that $k[\mathsf{t}]$ appears pseudorandom to users *not in* S_{t} (the non-members) for all instants t, even when such users can collude with each other and share all their information. The problem is to design the program KD in a manner such that this guarantee is achieved.

Fiat and Naor [FN93] were the first to define this problem formally and they introduced it under the title of *broadcast* encryption—a formulation in which all users are assumed to be stateless and group members are required to be able to recover $k[\mathsf{t}]$, given only $\mathcal{M}_{\mathsf{t}}$ and their long-lived keys. Subsequent work (for example, [Mit97, WGL00]) lifted the problem to the more general setting of stateful users, and studied it in the context of ensuring privacy in multicast groups on the Internet (hence the name *multicast* encryption). LKH is a protocol that relies on the statefulness assumption.

The Protocol. A trivial approach to multicast key distribution would be to have the center generate a new, purely random key $k[\mathsf{t}]$ for every group membership change, and to let $\mathcal{M}_{\mathsf{t}}$ (the rekey messages for time t) be the set of ciphertexts obtained by encrypting $k[\mathsf{t}]$ individually under the long-lived keys of *every* user in S_{t}, that is, the set $\{\mathcal{E}_{k_{U_i}}(k[\mathsf{t}])\}_{U_i \in S_{\mathsf{t}}}$. This, however, is an unscalable solution since it involves a linear communication overhead per membership change, which is prohibitive for most applications that use multicast.

The LKH protocol betters the above trivial approach by distributing to users, in addition to the group key, a set of *auxiliary* keys, with each auxiliary key being given to some subset of the

of two sets of values—in one world, reveal the real values of all unopened messages, and in the other, provide an equal number of messages that are sampled from the probability space of the unopnened messages, *conditioned* on the values of the already-opened messages. Now show that A cannot distinguish between these two worlds. This problem is essentially the same as the version of the selective decryption problem where plaintexts are not assumed to be independent of each other. We don't know of any solution to the problem yet.

[12] In practice, such long-lived keys could be established during the first `login` request made by users using, say, public-key based approaches.

[13] In some scenarios, the `logout` operations may be "forced"; that is, the manager may proactively revoke some user(s) (as a punishment for not paying subscription fees in time and/or indulging in piracy).

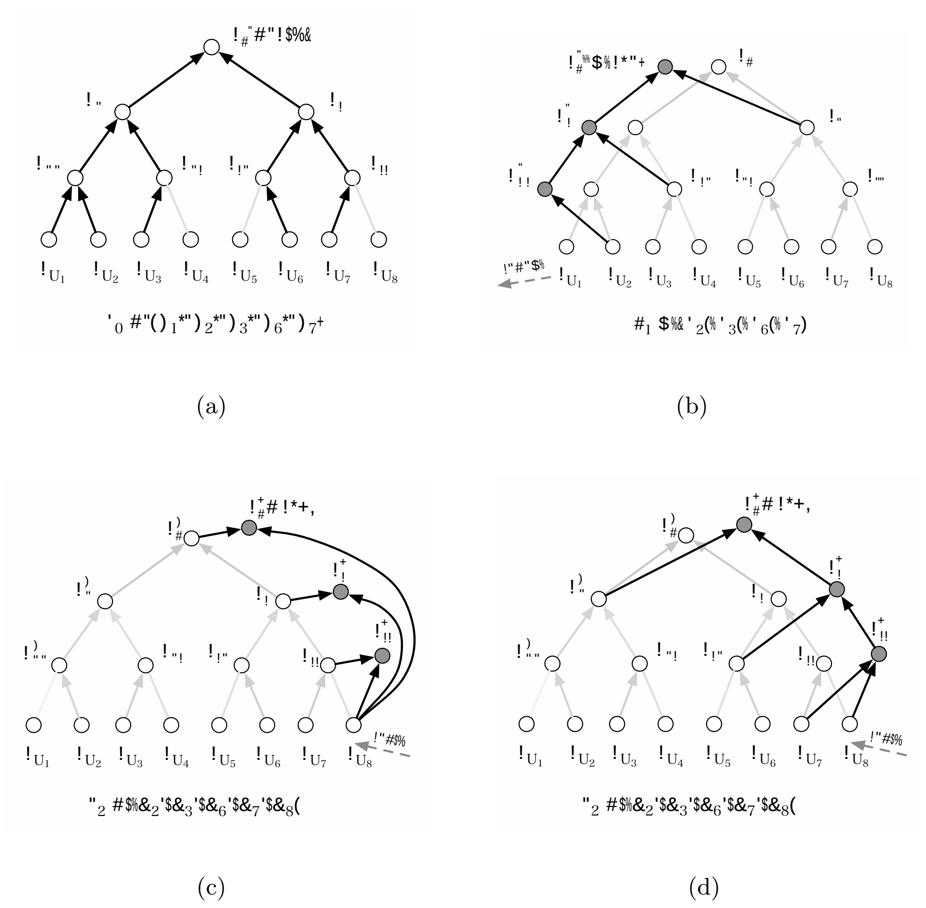

Figure 1: **LKH and rLKH:** Figure 1(a) shows how key distribution to the initial set of users S_0 is performed while figure 1(b) demonstrates the rekeying process for user `logout` (both these procedures are the same in LKH and rLKH). Figure 1(c) shows how rekeying for user `login` works in LKH and fig. 1(d) illustrates the same for rLKH.

current group members. All keys in the system are organized in the form of a *hierarchy*—the group key is associated with the root node in the hierarchy, the long-lived keys of users with the leaves, and the auxiliary keys with internal nodes. At each point in time $\mathtt{t}$, a user $U_i \in S_{\mathtt{t}}$ knows all keys on the path from the leaf node corresponding to k_{U_i} to the root node (which corresponds to $k[\mathtt{t}]$). The protocol maintains this property as an invariant across membership changes.

Rekey Messages. For simplicity, we illustrate the protocol using an example where $n = 8$ and the key hierarchy is binary. (So the height of the hierarchy is $\log_2(8) = 3$.) We assume that all parties (including the center) have blackbox access to a symmetric-key encryption scheme $\Pi = (\mathcal{E}, \mathcal{D})$ with key space $\{0,1\}^\eta$ for some fixed security parameter η. In our description, we use the terms "keys" and "nodes" interchangeably (the relation between them is obvious in the current context) and depict transmission of a ciphertext $\mathcal{E}_{k_1}(k_2)$ with an edge $k_1 \rightarrow k_2$ in the figures.

Suppose that initially ($\mathtt{t} = 0$), the set of group members $S_0 = \{U_1, U_2, U_3, U_6, U_7\}$ as shown in Figure 1(a). The center's key distribution program KD generates the initial group key $k[0] = k_\epsilon$

(the root node) and all auxiliary keys (internal nodes) which are supposed to be given to users in S_0. For example, since k_{00} and k_0 lie on the path from k_{U_1} to $k[0]$, these keys must be generated afresh and sent securely to U_1. KD transmits the keys to the designated users by sending the ciphertexts shown by dark edges in the figure. So, for example, user U_1 can obtain all the keys it is supposed to know $(k_{00}, k_0, k_\epsilon)$ by decrypting, in order, the ciphertexts $\mathcal{E}_{k_{U_1}}(k_{00}), \mathcal{E}_{k_{00}}(k_0)$ and $\mathcal{E}_{k_0}(k_\epsilon)$.

Now suppose that at time $\mathtt{t} = 1$, user U_1 logs out of the group. That is, $S_1 = \{U_2, U_3, U_6, U_7\}$. The program KD should re-generate the group key k_ϵ, and the auxiliary keys which were known to U_1 at $\mathtt{t} = 0$ (k_{00} and k_0) and distribute the new values in a manner such that U_1 cannot recover them but other users who are required to do so (according to the protocol invariant) still can. Specifically, it generates new keys k^1_{00}, k^1_0 and $k^1_\epsilon =: k[1]$ (independently and uniformly at random) and sends out the ciphertexts shown in figure 1(b). Thus, every rekey operation for a user `logout` requires sending logarithmically many (specifically, $2\log_2(n) - 1$) ciphertexts; in our example, this number is 5.

The Flaw and the Fix. The flaw in the original LKH protocol lies in the way it implements rekeying for user `login` operations. Suppose U_8 sends a `login` request at time $\mathtt{t} = 2$. The center must now re-generate keys k_{11}, k_1, k_ϵ and send them securely to all the designated users (including U_8). The protocol does this by transmitting the ciphertexts shown in figure 1(c). ($k^2_{11}, k^2_1, k^2_\epsilon$ denote the newly generated keys.) Sending the old values of *any* of these keys, even the auxiliary ones, to U_8 (that is, encrypted under the key k_{U_8}) could potentially allow U_8 to recover past group keys, which must be prevented. (For example, if only the group key was generated afresh and the keys k_1, k_{11} were transmitted to U_8 as is (though encrypted under k_{U_8}), U_8 could use k_1 and the transmissions at $\mathtt{t} = 1$ to recover $k[1]$.) For this reason, LKH generates fresh values for k_{11} and k_1 in our example.

Note that the group key at $\mathtt{t} = 1$, $k[1] = k^1_\epsilon$, is used to encrypt the group key at $\mathtt{t} = 2$, k^2_ϵ. This is a problem since our initial goal was to guarantee pseudorandomness of all group keys but deploying $k[1]$ in this manner clearly fails that purpose. In principle, if $k[1]$ is used in keying other applications (for example, in a message authentication scheme) at $\mathtt{t} = 1$, and is also used for rekeying in the manner shown, then the protocol could be completely subverted (both the keys $k[1]$ and $k[2]$ fully recovered) *even by a passive eavesdropper on the channel.* Of course, this does not mean that the protocol is broken for *any* secure implementation of the encryption scheme; but for *some* (albeit contrived ones), it is.

We propose to fix the LKH protocol by changing the rekeying procedure for user `login`s as shown in Figure 1(d). (We remark that this fix is different from the one suggested in [MP06].) Notice that the communication cost incurred is the same as in the original protocol ($2\log_2(n)$ ciphertexts for a user space of size n). Notice also that the structure of the rekey messages is now similar to that of the messages sent upon a user `logout` request (figure 1(b)). We refer to this modified version of LKH as "rLKH" (the r stands for "repaired"). The protocol can be easily generalized to work with arbitrary hierarchies (as opposed to the binary one in our example); in particular, when the key hierarchy is a d-ary tree (so its height equals $\lceil\log_d(n)\rceil$), the communication complexity (number of ciphertexts transmitted) of rekeying would be $d\lceil\log_d(n)\rceil$ for user `login`s and $d\lceil\log_d(n)\rceil - 1$ for user `logout`s. An implementation of rLKH with n users and a d-ary hierarchy is referred to as the (n, d)*-instance* of the protocol.

One could conceive other ways of fixing the user `login` process of LKH (possibly as secure and as efficient as the one we propose). We prefer this fix for various reasons: (a) the key hierarchy in rLKH has the nice property that at all instants, every auxiliary key (and even the group key) is transmitted to the legitimate recipients by encrypting it under its two children only (and no other keys). This property could potentially simplify implementation of the protocol in practice. (b) rekeying is possible using encryption alone (without requiring other cryptographic primitives); thus, the security of the protocol rests upon the secure implementation of a single primitive, which, we believe, makes for good cryptographic design; (c) most importantly, our fix ensures that the depth of the key graph generated by all ciphertexts remains logarithmic in the size of the group (does not grow with time); this property is useful in arguing about the protocol's adaptive security.

Adaptive Security. Let KD be an n-user multicast key distribution program. We define adaptive security of KD using the following game (which we call the MKD game) played between an adversary A and a challenger B. Initially, B generates the long-lived keys of all users $k_{U_1}, \cdots, k_{U_n}$ (randomly, independently from the underlying key space) and also generates a random challenge bit b. A specifies the initial set of group members, S_0, in response to which KD is invoked and the initial key distribution messages, $\mathcal{M}_0$, returned to A. Subsequently, A issues multiple queries to B, each query being either:

1. a *rekey* query—at any instant t, A can issue a query of the form $\mathtt{rekey}(\mathtt{command}, U_i)$ where $\mathtt{command}$ is either $\mathtt{login}$ or $\mathtt{logout}$. In response, B runs KD based on the membership change command specified and returns the set of rekey messages $\mathcal{M}_{\mathsf{t}}$ to A; OR
2. a *corrupt* query—A can also issue queries of the form $\mathtt{corrupt}(U_j)$, in return for which B sends it the key k_{U_j}; OR
3. a *challenge* query—finally, A can issue a $\mathtt{challenge}$ query at any instant t; in response, it is given the key $k[\mathsf{t}]$ if $b = 0$, or a fresh key $r[\mathsf{t}]$ (sampled independently and uniformly at random from $\{0,1\}^{\eta}$) if $b = 1$.

All queries can be issued interleavingly and adaptively. Let $U^{\mathsf{corr}}(\mathsf{A})$ be the set of all users corrupted by A during the game. Let $T^{\mathsf{chal}}(\mathsf{A})$ be the set of instants t at which A issues a $\mathtt{challenge}$ query. We say that A is *legitimate* if in every execution of A in the MKD game, its queries satisfy:

$$\left(\bigcup_{\mathsf{t} \in T^{\mathsf{chal}}(\mathsf{A})} S_{\mathsf{t}} \right) \cap U^{\mathsf{corr}}(\mathsf{A}) = \emptyset$$

Let $\mathsf{A}^{\mathsf{B}_b^{\mathsf{KD}}}$ denote the random variable corresponding to the output of A in the game, conditioned on the event that B selects b as the challenge bit.

Definition 3.1 Let $t, r \in \mathbb{N}^+$ and $0 < \epsilon < 1$. A multicast key distribution program KD is (t, r, ϵ)-secure against adaptive adversaries if for every legitimate adversary A that runs in time t, and makes r $\mathtt{rekey}$ queries:

$$|\mathbf{P}[\mathsf{A}^{\mathsf{B}_b^{\mathsf{KD}}} = 1 \mid b = 0] - \mathbf{P}[\mathsf{A}^{\mathsf{B}_b^{\mathsf{KD}}} = 1 \mid b = 1]| \leq \epsilon$$

On the lines of the above definition, one can also define the problem of multicast encryption (or, for that matter, any security task based on multicast key distribution). For example, consider a multicast encryption protocol ME constructed using a key distribution program KD and an encryption scheme $\overline{\Pi} = (\overline{\mathcal{E}}, \overline{\mathcal{D}})$ as follows: the protocol distributes rekey messages for every group membership change just as KD but besides this, it also encrypts arbitrary messages—upon receiving a message m to encrypt at time t, the protocol outputs $\overline{\mathcal{E}}_{k[\mathsf{t}]}(m)$. Security of such a scheme can be defined using a game similar to the MKD game, but with one change—every time the adversary issues a `challenge` query, it also specifies two messages (m_0, m_1) $(m_0, m_1 \in \{0,1\}^*$, and $|m_0| = |m_1|)$ and the challenger replies with $\overline{\mathcal{E}}_{k[\mathtt{t}]}(m_b)$ ($k[\mathtt{t}]$ being the current group key). It is possible to show that if KD is (t, r, ϵ)-secure against adaptive adversaries, and $\overline{\Pi}$ is (t, ϵ')-Ind-CPA secure, then ME is $(O(t), r, 2\epsilon + \epsilon')$-secure against adaptive adversaries.

In general, the problems of multicast key distribution and multicast encryption are equivalent to each other but studying the key distribution problem is more natural since it allows to generically build protocols for any security task (not necessarily multicast encryption) that can be accomplished using shared group keys. For this reason, we have focussed our attention on the key distribution problem alone, and discuss the security of rLKH in the same context.

Theorem 3.2 Let $n, d, t, r' \in I\!N^+$ such that $1 < d \leq n$. Let $0 < \epsilon < 1$. The (n, d)-instance of rLKH, when implemented using a (t, ϵ)-Ind-CPA secure encryption scheme Π, is (t', r', ϵ')-secure against adaptive adversaries for

$$\epsilon' = \epsilon \cdot \frac{3\tilde{n}^2}{2} \cdot (2\tilde{n} + 1)^{\lceil \log_d(n) \rceil - 1}$$

$$t' = t - (O(\tilde{n}) \cdot t_{\text{GenKey}} + (r'd\lceil \log_d(n) \rceil) \cdot t_{\text{Encrypt}})$$

Here, $\tilde{n} = \max\{n, d^{\lceil \log_d(n) \rceil - 1} + r'\}$ and t_{GenKey} (resp. t_{Encrypt}) is the time taken to perform key generation (resp. encryption) in Π.

The proof of this theorem follows almost immediately from our soundness result of Section 2, given that (a) the key graph generated by any execution of rLKH is acyclic; (b) all group keys correspond to sinks in the protocol key graph; (c) the depth of the graph remains $\lceil \log_d(n) \rceil$ throughout; and (d) for any r'-round execution of the protocol, and for all $\mathtt{t} \leq r'$, the group key $k[\mathtt{t}]$ can be reached from a long-lived key k_{U_i} if and only if $U_i \in S_{\mathtt{t}}$. (The last part can be proven using a straightforward inductive argument, with the induction being performed on r'.) The reduction factor given in the theorem is slightly better than what one gets using a direct invocation of Theorem 2.4: this is achieved using the fact that in any r'-round execution of the rLKH protocol, (a) a key at depth i in the key graph (that is, at distance i from some source) is encrypted only by keys at depth $i - 1$ and (b) there are at most $d^{\lceil \log_d(n) \rceil - 1} + r'$ keys at any depth in the graph (and at most n sources in it). Note that our reduction factor is exponential in $\lceil \log_d(n) \rceil$ which is independent of the number of rounds the protocol is executed for. That is, the adaptive security of rLKH degrades polynomially (and not exponentially) with the number of rounds in the protocol execution.

Changing the hierarchy structure in rLKH involves a natural trade-off between efficiency and security: If we increase the arity d of the hierarchy (and correspondingly, reduce the height), the communication efficiency of the protocol suffers, but we get a better guarantee on its adaptive security. The extreme case is the n-ary hierarchy that has a linear rekeying communication complexity but provides adaptive security via a reduction factor of only $O(\tilde{n}^2)$. (Note that this is exactly the trivial approach to key distribution we discussed earlier on.) Whether or not one can further improve this trade-off between efficiency and security across different instances of rLKH, and, in particular, prove its adaptive security via a reduction factor smaller than the one given in Theorem 3.2, assuming only the semantic security of Π, is a question left open by this work.

Acknowledgements

Many thanks to my advisor, Daniele Micciancio, for providing detailed feedback on the paper and pointing out a bug in an earlier version of the proof of Theorem 2.4. Thanks also to the anonymous reviewers of TCC and to my colleagues, Thomas Ristenpart and Scott Yilek, for commenting on an earlier draft.

References

[AKI03] Nuttapong Attrapadung, Kazukuni Kobara, and Hideki Imai. Broadcast encryption with short keys and transmissions. In M. Yung, editor, *Proceedings of the 2003 ACM workshop on Digital rights management (DRM)*, pages 55–66, Washington, DC, USA., October 2003. ACM Press, New York, NY, USA.

[AR02] Martin Abadi and Philip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[AW05] Martin Abadi and Bogdan Warinschi. Security analysis of cryptographically controlled access to xml documents. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems (PODS)*, pages 108–117, Baltimore, Maryland, June 2005. ACM.

[BDJR97] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Society Press, October 1997.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In V. Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275, Santa Barbara, CA, USA, August 2005. Springer Verlag, Berlin, Germany.

[BH92] Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In Rainer A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT'92*, volume 658 of *Lecture Notes in Computer Science*, pages 307–323. Springer-Verlag, May 1992.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multiparty computation. In *28th Annual ACM Symposium on Theory of Computing*, pages 639–648. ACM Press, May 1996.

[CH06] Ran Canetti and Jonathan Herzog. Universally composable symbolic analysis of mutual authentication and key exchange protocols. In Shai Halevi and Tal Rabin, editors, *TCC '06: Third Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 380–403. Springer-Verlag, 2006.

[CHK05] Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively secure non-interactive public key encryption. In Joe Kilian, editor, *TCC '05: Second Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 150–168. Springer-Verlag, February 2005.

[DC06] Yitao Duan and John Canny. How to construct multicast cryptosystems provably secure against adaptive chosen ciphertext attacks. In David Pointcheval, editor, *CT-RSA'06: RSA Conference, Cryptographers' Track*, volume 3860 of *Lecture Notes in Computer Science*, pages 244–261. Springer-Verlag, February 2006.

[DDMW06] Anupam Datta, Ante Derek, John Mitchell, and Bogdan Warinschi. Computationally sound compositional logic for key exchange protocols. In *19th IEEE Computer Security Foundations Workshop (CSFW '06)*, pages 321–334. IEEE Computer Society, 2006.

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions. *Journal of the ACM*, 50(6):852–921, 2003.

[FN93] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer-Verlag, August 1993.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[GS06] Prateek Gupta and Vitaly Shmatikov. Key confirmation and adaptive corruptions in the protocol security logic. In *FCS-ARSPA 2006 (Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis)*, 2006.

[Mit97] Suvo Mittra. Iolus: A framework for scalable secure multicasting. In *Proceedings of ACM SIGCOMM*, pages 277–288, Cannes, France, September 14–18, 1997.

[MP05] Daniele Micciancio and Saurabh Panjwani. Adaptive security of symbolic encryption. In J. Kilian, editor, *Theory of Cryptography Conference, TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 169–187, Cambridge, MA, USA, February 2005. Springer-Verlag, Berlin, Germany.

[MP06] Daniele Micciancio and Saurabh Panjwani. Corrupting one vs. corrupting many: The case of broadcast and multicast encryption. In *Automata, Languages, and Programming: 33rd International Colloquium, ICALP 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*. Springer-Verlag, January 2006.

[MW04] Daniele Micciancio and Bogdan Warinschi. Soundness of formal encryption in the presence of active adversaries. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer-Verlag, February 2004.

[NNL01] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer-Verlag, August 2001.

[PST01] Adrian Perrig, Dawn Song, and Doug Tygar. ELK, a new protocol for efficient large-group key distribution. In *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2001. IEEE Computer Society Press.

[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 Theorem 2.4

Let A be a legitimate (n, e, l)-adversary playing the GSD game, and suppose that the encryption scheme Π in the game is (t, ϵ)-Ind-CPA secure. Suppose that A runs in time t' and is able to win the game with advantage greater than ϵ' (t' and ϵ' as defined in Theorem 2.4); that is, suppose that

$$\Delta_{\text{GSD}}(\mathsf{A}) \;:=\; |\mathbf{P}[\mathsf{A}^{\mathsf{B}_b^\Pi} = 1 \mid b = 0] - \mathbf{P}[\mathsf{A}^{\mathsf{B}_b^\Pi} = 1 \mid b = 1]| \;>\; \epsilon' \tag{1}$$

Given such an adversary, we will construct another adversary A' that runs in time t and, using blackbox access to A, is able to break the encryption scheme Π with advantage greater than ϵ. That is, we will construct an A' with the property that

$$\Delta_{\text{Ind-CPA}}(\mathsf{A}') \;:=\; |\mathbf{P}[\mathsf{A}'^{\mathcal{O}_{k,b}^\Pi} = 1 \mid b = 0] - \mathbf{P}[\mathsf{A}'^{\mathcal{O}_{k,b}^\Pi} = 1 \mid b = 1]| \;>\; \epsilon \tag{2}$$

This would contradict the (t, ϵ) security of Π, hence implying that our assumption about A was incorrect.

A.1 The Reduction

For any positive integer i, we use $[i]$ to denote the set $\{1, \cdots, i\}$. Any path in the graph $G(\mathsf{A})$ generated by A can be represented using a sequence of length $l+1$ as follows: First, write down the nodes in the path in the order of their occurrence from start to end. Then, if the path is of length smaller than l (has fewer than $l+1$ nodes), *prepend* this sequence with a 0 as many times as is required to make its length equal $l+1$. For example, a path $i_1 \to i_2 \to i_3$ (with 2 edges only) would be represented under this convention as:

$$(\underbrace{0, 0, \cdots, 0}_{(l-2)\text{ times}}, i_1, i_2, i_3)$$

We say that a sequence of values from $\{0, 1, \cdots, n\}$ is a *valid* path in $G(\mathsf{A})$ if it is a representation (defined as above) of a path that exists in $G(\mathsf{A})$.

Our construction of the adversary A' is organized in two parts. In the first part, called the *setup* phase (Figure 2), A' generates all keys required to reply to A's queries and some other random values which are used to form the replies. (The italicized comments embedded in Figure 2 give some intuition about the semantics of these random values.) The second part of A'—the *execution* phase—which is shown in Figure 3, is the one in which A' runs A (in a blackbox manner) and simulates replies to all its queries; the replies to some of the queries (namely, queries of the form $\texttt{encrypt}(s, x)$ where s is as decided in the setup phase) are given using the left-or-right oracle $\mathcal{O}_{k,b}^\Pi$.

It is easy to check that the running time of A' is bounded from above by $t' + O(n) \cdot t_{\text{GenKey}} + e \cdot t_{\text{Encrypt}}$, which is the same as the quantity t in Theorem 2.4.

<u>PHASE I: SETUP</u>

(Generating keys and preparing to reply to A*'s queries)*

001. Sample $l-1$ numbers $u_0, u_1, \cdots, u_{l-2}$ independently from the set $\{0, 1, 2, \cdots, n\}$ such that for each $j \in \{0, 1, \cdots, l-2\}$, the following holds:
$\forall i \in [n]\colon\ \mathbf{P}[u_j = i] = \frac{2}{2n+1}$; and
$\mathbf{P}[u_j = 0] = \frac{1}{2n+1}$
Sample u_{l-1} and u_l independently and uniformly at random from $[n]$.
(The sequence $(u_0, u_1, \cdots, u_l)$ *is* A'*'s "guess" for a path; a successful execution of* A' *will be one in which this sequence is a valid path in* $G(\mathsf{A})$*. Note that we do not rule out repetitions amongst the* u_j*'s—it is possible that* $u_j = u_{j'}$ *for some* $j \neq j'$*—even though the resulting sequences are bound to be invalid. This is done only for the sake of simplicity. (The reduction factor is not significantly improved by avoiding this triviality.) The choice for the specific distribution of the* u_j*'s defined above will be clearer after seeing the analysis of* A'*.)*

002. Let u_s be the first non-zero value in the sequence $(u_0, u_1, \cdots, u_{l-1}, u_l)$.
(u_s *is the* **s***tart node of the path guessed by* A'*. While replying to* A*'s queries,* A' *will associate the key* k *used by its encryption oracle* $\mathcal{O}^{\Pi}_{k,b}$*, with* u_s*. Note that* $s \leq l-1$ *always.)*

003. Sample $l-s-1$ values $b_s, b_{s+1}, \cdots, b_{l-2}$ independently and uniformly at random from $\{0,1\}$. Let $b_{l-1} = 0$.
(Roughly, these bit values determine which of the edges in the path $(u_s, u_{s+1}, \cdots, u_l)$ *are replied to with "fake" ciphertexts and which are not. For their exact semantics, see the execution phase of* A'*.)*

004. Generate keys $k_1, \cdots, k_{u_s-1}, k_{u_s+1}, \cdots, k_n$ randomly and independently from $\{0,1\}^\eta$. Also generate (random, independent) keys $r_{u_s}, r_{u_{s+1}}, \cdots, r_{u_l}$ from the same space.
(The r_i*'s are used in creating "fake" ciphertexts in the execution phase.)*

Figure 2: The first phase of the adversary A'.

A.2 The Analysis.

For any execution of A, the *transcript* of that execution is the sequence of queries made by and replies given to A; formally, it is the sequence $(\mathsf{q}_1, \mathsf{r}_1, \mathsf{q}_2, \mathsf{r}_2, \cdots, \mathsf{q}_f, \mathsf{r}_f)$, where f is the total number of queries made by A in that execution, and for every $i \in \{1, \cdots, f\}$, q_i is the ith query of A and r_i is the reply received for q_i. The *view* of A, given a fixed procedure for deciding replies to its queries, is the distribution over all possible transcripts that can be generated by executing A and replying to it using the said procedure. The variables $G(\mathsf{A}), V^{\mathsf{chal}}(\mathsf{A})$ and $V^{\mathsf{corr}}(\mathsf{A})$ are all functions of the view of A, that is, their distribution depends not only on the coins used by A but also on the procedure used to reply to A's queries. For the most of our analysis, we will be concerned with the view of A in its interaction with A', that is, when the procedure for replying to A's queries is as shown in figures 2 and 3. Thus, unless otherwise specified, $G(\mathsf{A}), V^{\mathsf{chal}}(\mathsf{A}), V^{\mathsf{corr}}(\mathsf{A})$ should be treated as random variables defined for this particular view.

```
PHASE II: EXECUTION
(Running A and simulating replies to its queries)

100. Initialize an array of boolean values $seen[s], seen[s+1], \ldots, seen[l-1]$; set each to $false$.
200. Run A. When A issues a query encrypt$(x, y)$, do the following—
210.     If $x = u_s \wedge y = u_{s+1}$,
             Set $seen[s]$ to be $true$.
             If $b_s = 0$, reply with $\mathcal{O}^{\Pi}_{k,b}(k_{u_{s+1}}, r_{u_{s+1}})$.
             If $b_s = 1$, reply with $\mathcal{O}^{\Pi}_{k,b}(r_{u_{s+1}}, k_{u_{s+1}})$.
220.     If $x = u_s \wedge y \notin \{u_{s+1}, \cdots, u_{l-1}, u_l\}$,
             Reply with $\mathcal{O}^{\Pi}_{k,b}(k_y, k_y)$.
230.     If $y = u_s$,
             Reply with $\mathcal{E}_{k_x}(r_{u_s})$. (Since A′ does not know the value of $k$—the key associated
             with $u_s$—its reply to every edge $x \to u_s$ created by A is a fake ciphertext.)
240.     For every $j \in \{s, \cdots, l-1\}$, do the following:
241.         If $x \neq u_j \wedge y = u_{j+1} \wedge \neg seen[j]$,
                 If $x \neq u_s$, reply with $\mathcal{E}_{k_x}(r_{u_{j+1}})$; else, reply with $\mathcal{O}^{\Pi}_{k,b}(r_{u_{j+1}}, r_{u_{j+1}})$.
242.         If $x \neq u_j \wedge y = u_{j+1} \wedge seen[j]$,
                 If $x \neq u_s$, reply with $\mathcal{E}_{k_x}(k_{u_{j+1}})$; else, reply with $\mathcal{O}^{\Pi}_{k,b}(k_{u_{j+1}}, k_{u_{j+1}})$.
243.         If $x = u_j \wedge y = u_{j+1} \wedge j > s$, (Note: The case $x = u_s, y = u_{s+1}$ is addressed above)
                 Set $seen[j]$ to be $true$.
                 If $b_j = b_{j-1}$, reply with $\mathcal{E}_{k_{u_j}}(k_{u_{j+1}})$.
                 If $b_j \neq b_{j-1}$, reply with $\mathcal{E}_{k_{u_j}}(r_{u_{j+1}})$.
250.     If none of the above conditions are satisfied, reply with $\mathcal{E}_{k_x}(k_y)$.

300. When A issues a query corrupt$(x)$, do the following—
         If $x \neq u_s$, return $k_x$ to A.
         If $x = u_s$, output a random bit. Halt!

400. When A issues a query challenge$(x)$, do the following—
         Reply with $k_x$ if either of the following is true:
             (a) $x = u_l$.
             (b) $x \neq u_l$ but the query challenge$(u_l)$ has been made (and replied to) already.
         Otherwise, reply with a fresh, random sample from $\{0,1\}^{\eta}$.
         (We stress that the reply for the query challenge$(u_l)$ is always $k_{u_l}$. Also, until the
         query challenge$(u_l)$ is made, the reply to every query of the from challenge$(x)$ is a
         random value, sampled independently of $k_x$.)

     (Following are some "bad" conditions under which A′ fails in its simulation.)
500. If at any point during or after the execution of A, it is found that:
             (a) $(u_0, \cdots, u_l)$ is not a valid path in $G(\mathsf{A})$; OR
             (b) $u_l \notin V^{\mathsf{chal}}(\mathsf{A})$,
         Then output a random bit. Halt!

600. In the end, output whatever A outputs.
```

Figure 3: The second phase of the adversary A′.

Let Bad be the event that A''s simulation of replies to A's queries is unsuccessful, that is, the values $u_0, \cdots, u_l$ that it selects are such that:

(a) $u_l \notin V^{\mathsf{chal}}(\mathsf{A})$; OR

(b) $(u_0, u_1, \cdots u_{l-1}, u_l)$ is not a valid path in $G(\mathsf{A})$.

Roughly speaking, the occurrence of Bad is computationally independent of the choice of the bit b for otherwise we would be contradicting the Ind-CPA security of Π. This is formalized in the following proposition:

Proposition A.1

$$\Delta_{\mathsf{Bad}} := |\mathbf{P}[\mathsf{Bad} \mid b = 0] - \mathbf{P}[\mathsf{Bad} \mid b = 1]| \leq \epsilon$$

Proof Sketch: The proof uses a straightforward reduction argument. Suppose that the statement is false, that is, $\Delta_{\mathsf{Bad}} > \epsilon$. Modify the code of A' slightly such that if at any point during the simulation of A the event Bad occurs, the code outputs 1 (instead of a purely random bit). The resulting code gives us an adversary that defies the (t, ϵ) security of Π. ▌

Let $\mathsf{O}_{\mathsf{A}'}$ denote the event that A' outputs 1 when given access to a left-or-right oracle $\mathcal{O}^{\Pi}_{k,b}$. Our goal is to bound the Ind-CPA advantage of A', which can now be re-written as

$$\Delta_{\text{Ind-CPA}}(\mathsf{A}') = |\mathbf{P}[\mathsf{O}_{\mathsf{A}'} \mid b = 0] - \mathbf{P}[\mathsf{O}_{\mathsf{A}'} \mid b = 1]|$$

Let us expand this quantity based on the occurrence/non-occurrence of event Bad. Below and for the rest of the proof, we use the shorthand $A; B$ to denote $A \wedge B$ for any two events A and B.

$$\begin{aligned}
\Delta_{\text{Ind-CPA}}(\mathsf{A}') &= \big| \big(\mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}} \mid b = 0] - \mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}} \mid b = 1]\big) \\
&\qquad + \ (\mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \mathsf{Bad} \mid b = 0] - \mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \mathsf{Bad} \mid b = 1]) \big| \\
&\geq |\mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}} \mid b = 0] - \mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}} \mid b = 1]| \\
&\qquad - \ |\mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \mathsf{Bad} \mid b = 0] - \mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \mathsf{Bad} \mid b = 1]| \\
&= |\mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}} \mid b = 0] - \mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}} \mid b = 1]| \\
&\qquad - \ |\underbrace{\mathbf{P}[\mathsf{O}_{\mathsf{A}'} \mid b = 0; \mathsf{Bad}]}_{=\frac{1}{2}} \cdot \mathbf{P}[\mathsf{Bad} \mid b = 0] - \underbrace{\mathbf{P}[\mathsf{O}_{\mathsf{A}'} \mid b = 1; \mathsf{Bad}]}_{=\frac{1}{2}} \cdot \mathbf{P}[\mathsf{Bad} \mid b = 1]| \\
&= |\mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}} \mid b = 0] - \mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}} \mid b = 1]| - \frac{1}{2} \cdot \Delta_{\mathsf{Bad}} \\
&\geq |\mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}} \mid b = 0] - \mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}} \mid b = 1]| - \frac{\epsilon}{2} \qquad (3) \\
&\quad \text{(Follows from Prop. A.1)}
\end{aligned}$$

Let $\Delta := \mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}}; b = 0] - \mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}}; b = 1]$. Inequality 3 can be re-written in terms of Δ as follows:

$$\begin{aligned}
\Delta_{\text{Ind-CPA}}(\mathsf{A}') &\geq |\mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}} \mid b = 0] - \mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}} \mid b = 1]| - \frac{\epsilon}{2} \\
&= \left| \frac{\mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}}; b = 0]}{\mathbf{P}[b = 0]} - \frac{\mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}}; b = 1]}{\mathbf{P}[b = 1]} \right| - \frac{\epsilon}{2} \\
&= \left| \frac{\mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}}; b = 0]}{1/2} - \frac{\mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}}; b = 1]}{1/2} \right| - \frac{\epsilon}{2} \\
&= 2 \cdot \big|\mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}}; b = 0] - \mathbf{P}[\mathsf{O}_{\mathsf{A}'}; \overline{\mathsf{Bad}}; b = 1]\big| - \frac{\epsilon}{2} \\
&= 2 \cdot |\Delta| - \frac{\epsilon}{2}
\end{aligned}$$

We will now focus our attention on bounding the quantity Δ. Let

$$\alpha = \frac{1}{n^2 \cdot (2n+1)^{l-1}} \tag{4}$$

We will aim to show that:

Lemma A.2 $|\Delta| > \frac{\alpha\epsilon'}{2}$

From this, the theorem would follow immediately using the following chain of inequalities:

$$\begin{aligned}
\Delta_{\text{Ind-CPA}}(\mathsf{A}') &\geq 2 \cdot |\Delta| - \epsilon/2 \\
&> 2 \cdot \frac{\alpha\epsilon'}{2} - \frac{\epsilon}{2} \qquad \text{(Plugging in Lemma A.2)} \\
&= \alpha\epsilon' - \frac{1}{2} \cdot \frac{2\alpha\epsilon'}{3} = \alpha\epsilon' - \frac{\alpha\epsilon'}{3} = \frac{2\alpha\epsilon'}{3} = \epsilon
\end{aligned}$$

which is what our initial goal—inequality 2—was.

A.2.1 Proof of Lemma A.2

Let O_A be the event that A' completes the execution of A successfully (event Bad does not occur) and the latter outputs 1 after termination. Observe that if Bad is known *not* to occur, the events O_A and $\mathsf{O}_{\mathsf{A}'}$ are exactly the same; that is, $\mathbf{P}[\mathsf{O}_{\mathsf{A}'} \mid \overline{\mathsf{Bad}}] = \mathbf{P}[\mathsf{O}_\mathsf{A} \mid \overline{\mathsf{Bad}}]$. Using this observation, we can re-write Δ as

$$\Delta = \mathbf{P}[\mathsf{O}_\mathsf{A}; \overline{\mathsf{Bad}}; b = 0] - \mathbf{P}[\mathsf{O}_\mathsf{A}; \overline{\mathsf{Bad}}; b = 1]$$

Let G denote the event $\overline{\mathsf{Bad}}$. (G stands for Good.) We will first break up this event into l mutually exclusive events $\mathsf{G}_0, \mathsf{G}_1, \cdots, \mathsf{G}_{l-1}$ as follows: for each $j \in \{0, \cdots, l-1\}$, define G_j as the event that the values $u_0, u_1, \cdots, u_l$ selected by A' satisfy the following three conditions:

(a) $u_l \in V^{\mathsf{chal}}(\mathsf{A})$;

(b) $(u_0, u_1, \cdots, u_l)$ is a valid path in $G(\mathsf{A})$;

(c) $u_0 = u_1 = \cdots = u_{j-1} = 0$ but $u_j \neq 0$. (In other words, the value of s decided in line 002 of A''s setup phase is j.)

Condition (c), together with (b), implies that for all $j' \in \{j, j+1, \cdots, l\}$, $u_{j'} \neq 0$. It is fairly obvious that for any distinct j and j' (in $\{0, \cdots, l-1\}$), G_j and $\mathsf{G}_{j'}$ are mutually exclusive and that $\mathsf{G} = \bigvee_{j=0}^{l-1} \mathsf{G}_j$. For each $j \in \{0, \cdots, l-1\}$, define the following quantity

$$\Delta_j := \mathbf{P}[\mathsf{O}_\mathsf{A}; \mathsf{G}_j; b = 0] - \mathbf{P}[\mathsf{O}_\mathsf{A}; \mathsf{G}_j; b = 1]$$

Δ can be expressed in terms of these quantities as follows:

$$\begin{aligned}\Delta &= \mathbf{P}[\mathsf{O}_\mathsf{A}; \overline{\mathsf{Bad}}; b=0] - \mathbf{P}[\mathsf{O}_\mathsf{A}; \overline{\mathsf{Bad}}; b=1]\\ &= \mathbf{P}[\mathsf{O}_\mathsf{A}; \bigvee_{j=0}^{l-1} \mathsf{G}_j; b=0] - \mathbf{P}[\mathsf{O}_\mathsf{A}; \bigvee_{j=0}^{l-1} \mathsf{G}_j; b=1]\\ &= \sum_{j=0}^{l-1} (\mathbf{P}[\mathsf{O}_\mathsf{A}; \mathsf{G}_j; b=0] - \mathbf{P}[\mathsf{O}_\mathsf{A}; \mathsf{G}_j; b=1])\\ &= \sum_{j=0}^{l-1} \Delta_j \qquad (5)\end{aligned}$$

We will now work towards breaking down the events $\mathsf{G}_0, \cdots, \mathsf{G}_{l-1}$ further and correspondingly, expressing the Δ_j's as summations of more detailed terms. For this, we need some more definitions.

Consider A's interaction with A'. For any of the values u_j selected by A', we denote (the random variable corresponding to) the in-degree of u_j in the graph $G(\mathsf{A})$ created during this interaction by *Indegree*(u_j). (If $u_j = 0$, we define *Indegree*(u_j) to be 0.) We think of nodes in $V^{\mathsf{chal}}(\mathsf{A})$ to be ordered according to their occurrence as arguments of `challenge` queries; so, in the sequel, whenever we say that u_l is "the i_lth node in $V^{\mathsf{chal}}(\mathsf{A})$", we imply that `challenge`$(u_l)$ is the i_lth among all `challenge` queries received by A' from A. Likewise, for any two values u_{j-1}, u_j, whenever we say that u_{j-1} is "the i_jth node pointing at u_j", we mean that `encrypt`(u_{j-1}, u_j) is the i_jth query of the form `encrypt`(x, u_j) received by A'.

For any $j \in [l]$, any $d, d_l, d_{l-1}, \cdots, d_{j+1}, d_j \in [n]$, and any $i, i_l, i_{l-1}, \cdots, i_{j+1}, i_j \in [n]$ such that $i \le d, i_l \le d_l, \cdots, i_j \le d_j$, let $\Psi^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}$ denote the event that

(a) G_{j-1} occurs; and

(b) $V^{\mathsf{chal}}(\mathsf{A})$ has size d and u_l is the ith node in it; and

(c) for each $j' \in \{j, j+1, \cdots, l\}$, *Indegree*$(u_{j'}) = d_{j'}$ and $u_{j'-1}$ is the $i_{j'}$th node pointing at $u_{j'}$ in $G(\mathsf{A})$.

The events $\mathsf{G}_0, \mathsf{G}_1, \cdots, \mathsf{G}_{l-1}$ can quite easily be expressed in terms of events of the above type. For any $j \in \{0, \cdots, l-1\}$, G_j occurs if and only if the size of $V^{\mathsf{chal}}(\mathsf{A})$ equals d for *some* $d \in [n]$, and u_l is the ith node in $V^{\mathsf{chal}}(\mathsf{A})$ for *some* $i \in [d]$ and has non-zero in-degree d_l for *some* d_l in $[n]$, and u_{l-1} is the i_lth node pointing at u_l in $G(\mathsf{A})$ for *some* $i_l \in [d_l]$, and so on, all the way upto u_j. Put succinctly, for any $j \in \{0, \cdots, l-1\}$:

$$\mathsf{G}_j = \bigvee_{d=1}^{n} \bigvee_{i=1}^{d} \bigvee_{d_l=1}^{n} \bigvee_{i_l=1}^{d_l} \cdots \bigvee_{d_{j+1}=1}^{n} \bigvee_{i_{j+1}=1}^{d_{j+1}} \Psi^{(d,d_l,\cdots,d_{j+1})}_{(i,i_l,\cdots,i_{j+1})}$$

Clearly, for any distinct pairs of vectors $(d, d_l, \cdots, d_j), (i, i_l, \cdots, i_j)$ and $(d', d'_l, \cdots, d'_{j'}), (i', i'_l, \cdots, i'_{j'})$, the events $\Psi^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}$ and $\Psi^{(d',d'_l,\cdots,d'_{j'})}_{(i',i'_l,\cdots,i'_{j'})}$ are mutually exclusive, and so

$$\begin{aligned}\Delta_j &= \mathbf{P}[\mathsf{O}_\mathsf{A}; \mathsf{G}_j; b=0] - \mathbf{P}[\mathsf{O}_\mathsf{A}; \mathsf{G}_j; b=1]\\ &= \sum_{\substack{d\in[n],\\ i\in[d]}} \sum_{\substack{d_l\in[n],\\ i_l\in[d_l]}} \cdots \sum_{\substack{d_{j+1}\in[n],\\ i_{j+1}\in[d_{j+1}]}} \left[\begin{array}{r} \mathbf{P}[\mathsf{O}_\mathsf{A}; \Psi^{(d,d_l,\cdots,d_{j+1})}_{(i,i_l,\cdots,i_{j+1})}; b=0] \\ - \mathbf{P}[\mathsf{O}_\mathsf{A}; \Psi^{(d,d_l,\cdots,d_{j+1})}_{(i,i_l,\cdots,i_{j+1})}; b=1] \end{array} \right] \qquad (6)\end{aligned}$$

We will now show how to sum up this quantity with j ranging from 0 through $l-1$ and to express the sum (which is the same as Δ) in terms of $\Delta_{\mathrm{GSD}}(\mathsf{A})$. Towards this, we first define another type of event, similar to (but slightly more involved than) the Ψ's.

Let j be any arbitrary number in $[l]$. For any sequence of bits $\nu_j, \nu_{j+1}, \cdots, \nu_{l-1}$, let $\overrightarrow{\nu}_j$ denote the bitvector $(\nu_j, \nu_{j+1}, \cdots, \nu_{l-1})$. For any $d, d_l, \cdots, d_j \in [n]$ any $i \in [d], i_l \in [d_l], \cdots, i_j \in [d_j]$ and any bitvector $\overrightarrow{\nu}_{j-1} \in \{0,1\}^{l-j+1}$, let $\Theta^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(\overrightarrow{\nu}_{j-1})$ denote the event that:

(a) For some $\tilde{j} \in \{0, 1, \cdots, j-1\}$, the event $\mathsf{G}_{\tilde{j}}$ occurs and *Indegree*$(u_{\tilde{j}}) = 0$

(b) $V^{\mathsf{chal}}(\mathsf{A})$ has size d and u_l is the ith node in it;

(c) For each $j' \in \{j, j+1, \cdots, l\}$, *Indegree*$(u_{j'}) = d_{j'}$, $u_{j'-1}$ is the $i_{j'}$th node pointing at $u_{j'}$ in $G(\mathsf{A})$ *and* the following holds:

- If $\nu_{j'-1} = 0$, A receives the real reply for $\mathtt{encrypt}(u_{j'-1}, u_{j'})$, that is, $\mathcal{E}_{k_{u_{j'-1}}}(k_{u_{j'}})$.
- If $\nu_{j'-1} = 1$, A receives a fake reply for the same query, that is, $\mathcal{E}_{k_{u_{j'-1}}}(r_{u_{j'}})$.

(d) For each $j' \in \{\tilde{j}+1, \cdots, j-1\}$, $u_{j'-1}$ is the *first* node pointing at $u_{j'}$ and A receives the real reply for the query $\mathtt{encrypt}(u_{j'-1}, u_{j'})$. (That is, it receives a ciphertext $c \leftarrow \mathcal{E}_{k_{u_{j'-1}}}(k_{u_{j'}})$.)

The last condition is equivalent to saying that A receives the real reply for *every* query of the form $\mathtt{encrypt}(x, u_{j'})$ where $j' < j$. (To see this, observe line 242 in A''s code—A''s reply to each query of this form succeeding $\mathtt{encrypt}(u_{j'-1}, u_{j'})$ is always real.) We use Θ_{first} and Θ_{last} to denote the following events:

$$\begin{aligned} \Theta_{\text{first}} &= \bigvee_{d=1}^{n} \bigvee_{d_l=1}^{n} \Theta^{(d,d_l)}_{(1,1)}((0)) \\ \Theta_{\text{last}} &= \bigvee_{d=1}^{n} \bigvee_{d_l=1}^{n} \Theta^{(d,d_l)}_{(d,d_l)}((1)) \end{aligned}$$

In words, Θ_{first} (resp. Θ_{last}) is the event that for some $\tilde{j} \in \{0, \cdots, l-1\}$, $\mathsf{G}_{\tilde{j}}$ occurs and *Indegree*$(u_{\tilde{j}}) = 0$, that u_l is the *first* (resp. *last*) node in $V^{\mathsf{chal}}(\mathsf{A})$, that u_{l-1} is the *first* (resp. *last*) node pointing at u_l in $G(\mathsf{A})$, that the reply A receives for the query $\mathtt{encrypt}(u_{l-1}, u_l)$ is a real ciphertext (resp. a fake one) and, finally, that the reply A receives for every query of the form $\mathtt{encrypt}(x, u_j)$ $(\tilde{j} < j < l)$ is a real ciphertext. The view of A under the occurrence of either Θ_{first} or Θ_{last} are, in fact, quite similar to the replies the challenger B provides to A in the GSD game.

Claim A.3 $|\mathbf{P}[\mathsf{O}_{\mathsf{A}} \mid \Theta_{\text{first}}] - \mathbf{P}[\mathsf{O}_{\mathsf{A}} \mid \Theta_{\text{last}}]| = \Delta_{\mathrm{GSD}}(\mathsf{A})$

In the following claim, we relate the probabilities of the events Θ_{first} and Θ_{last} to the quantity α defined in equation 4.

Claim A.4 $\mathbf{P}[\Theta_{\text{first}}] = \mathbf{P}[\Theta_{\text{last}}] = \alpha/2$

The proof of these claims are postponed to Sections A.2.2 and A.2.3 respectively. We will now illustrate how the two types of events we have defined hitherto—the Ψ's and the Θ's—are related to each other. This will help us sum up the Δ_j's (as defined in equation 6), express the sum in terms of Θ_{first} and Θ_{last}, and thus relate it to $\Delta_{\mathrm{GSD}}(\mathsf{A})$.

Before we explain the relation between the Ψ's and the Θ's, we need one last set of notations. For any bitvector $\overrightarrow{\nu}_j$ and any bit value $\nu \in \{0,1\}$, let $\nu \cdot \overrightarrow{\nu}_j$ denote the bitvector formed by *prepending* ν to $\overrightarrow{\nu}_j$, and let $\mathsf{XOR}(\nu, \overrightarrow{\nu}_j)$ be defined as follows:

$$\mathsf{XOR}(\nu, \overrightarrow{\nu}_j) \quad = \quad (\nu \oplus \nu_j, \nu_j \oplus \nu_{j+1}, \nu_{j+1} \oplus \nu_{j+2}, \cdots, \nu_{l-3} \oplus \nu_{l-2}, \nu_{l-2} \oplus \nu_{l-1})$$

(If $j = l-1$, $\mathsf{XOR}(\nu, \overrightarrow{\nu}_j)$ equals $(\nu \oplus \nu_j)$.) Let $\Theta^{(d,d_l,\cdots,d_j,0)}_{(i,i_l,\cdots,i_j,0)}(1 \cdot \overrightarrow{\nu}_{j-1})$ denote the event that $\Theta^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(\overrightarrow{\nu}_{j-1})$ occurs and *Indegree*$(u_{j-1}) = 0$.

For any two events E_1 and E_2, we use $E_1 = E_2$ to denote an assertion that E_1 and E_2 are identical events (that is, E_1 occurs if and only if E_2 occurs) and $E_1 \simeq E_2$ to denote that the view of A in its interaction with A' given E_1 occurs is identically distributed as its view given E_2 occurs. Let $E_1 \equiv E_2$ denote that $E_1 \simeq E_2$ and $\mathbf{P}[E_1] = \mathbf{P}[E_2]$. It is easy to check that if $E_1 \equiv E_2$, then $\mathbf{P}[\mathsf{O}_\mathsf{A}; E_1] = \mathbf{P}[\mathsf{O}_\mathsf{A}; E_2]$.

Lemma A.5 (Hybrid Cancellation Lemma)

1. For all $d \in \{2, \cdots, n\}$ and $i \in \{1, \cdots, d-1\}$,

$$\bigvee_{d_l=1}^{n} \Theta^{(d,d_l)}_{(i,d_l)}((1)) \equiv \bigvee_{d_l=1}^{n} \Theta^{(d,d_l)}_{(i+1,1)}((0))$$

2. For all $j \in \{1, \cdots, l\}$, for all $d, d_l, \cdots, d_j \in [n]$ and $i, i_l, \cdots, i_j$ such that $1 \le i \le d, 1 \le i_l \le d_l, \cdots, 1 \le i_j \le d_j$, and for any bitvector $\overrightarrow{\nu}_j = (\nu_j, \cdots, \nu_{l-1}) \in \{0,1\}^{l-j}$:

$$\Theta^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(1 \cdot \overrightarrow{\nu}_j) \equiv \Theta^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1},i_j+1)}(0 \cdot \overrightarrow{\nu}_j)$$

3. For all $j \in \{1, \cdots, l\}$, for all $d, d_l, \cdots, d_j \in [n]$ and $i, i_l, \cdots, i_j$ such that $1 \le i \le d, 1 \le i_l \le d_l, \cdots, 1 \le i_j \le d_j$, and for any bitvector $\overrightarrow{\nu}_{j-1} = (\nu_{j-1}, \cdots, \nu_{l-1}) \in \{0,1\}^{l-j+1}$:

$$\Theta^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(\overrightarrow{\nu}_{j-1}) \quad = \quad \left[\Theta^{(d,d_l,\cdots,d_j,0)}_{(i,i_l,\cdots,i_j,0)}(1 \cdot \overrightarrow{\nu}_{j-1})\right] \vee \left[\bigvee_{d_{j-1}=1}^{n} \Theta^{(d,d_l,\cdots,d_j,d_{j-1})}_{(i,i_l,\cdots,i_j,1)}(0 \cdot \overrightarrow{\nu}_{j-1})\right]$$

4. Fix $\nu_{l-1} = 0$. Let ν be any arbitrary bit value and let $(b = \nu)$ denote the event that the oracle $\mathcal{O}^{\Pi}_{k,b}$ (provided to A') selects ν to be the value of b. Then, for all $j \in \{1, 2, \cdots, l\}$, for all $d, d_l, \cdots, d_j \in [n]$ and $i, i_l, \cdots, i_j$ such that $1 \le i \le d, 1 \le i_l \le d_l, \cdots, 1 \le i_j \le d_j$, the following is true:

   (a) If $j = 1$, then

$$\Psi^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)} \wedge (b = \nu) \quad = \quad \bigvee_{\nu_{j-1},\nu_j,\cdots,\nu_{l-2} \in \{0,1\}} \Theta^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(\mathsf{XOR}(\nu, \overrightarrow{\nu}_{j-1}))$$

   (b) If $j > 1$, then

$$\Psi^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)} \wedge (b = \nu) \quad \equiv \quad \bigvee_{d_{j-1}=0}^{n} \left(\bigvee_{\nu_{j-1},\nu_j,\cdots,\nu_{l-2} \in \{0,1\}} \Theta^{(d,d_l,\cdots,d_j,d_{j-1})}_{(i,i_l,\cdots,i_j,d_{j-1})}(1 \cdot \mathsf{XOR}(\nu, \overrightarrow{\nu}_{j-1})) \right)$$

The proof of this lemma is given in Section A.2.4. The next lemma invokes the above lemma and uses it to sum up the Δ_j's, step by step, in an inductive manner. For any $j \in \{0, \cdots, l-1\}$, let

$$\overline{\Delta}_j = \sum_{j'=0}^{j} \Delta_{j'}$$

Lemma A.6 (Telescoping Sums Lemma) For all $j \in \{0, 1, \cdots, l-1\}$,

$$\overline{\Delta}_j = \sum_{\substack{d \in [n],\\ i \in [d]}} \sum_{\substack{d_l \in [n],\\ i_l \in [d_l]}} \cdots \sum_{\substack{d_{j+1} \in [n],\\ i_{j+1} \in [d_{j+1}]}} \left[\sum_{\substack{\nu_j, \cdots, \nu_{l-2} \in \{0,1\},\\ \nu_{l-1}=0}} \left(\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{j+1})}_{(i,i_l,\cdots,i_{j+1})}(\mathsf{XOR}(0, \overrightarrow{\nu}_j))] \\ \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{j+1})}_{(i,i_l,\cdots,i_{j+1})}(\mathsf{XOR}(1, \overrightarrow{\nu}_j))] \end{array} \right) \right]$$

The proof of this lemma is given in Section A.2.5. Given these two lemmas and Claims A.3 and A.4, the final result (Lemma A.2) is quite easy to prove. Using Lemma A.6, equation 5 can be re-written as:

$$\begin{aligned}
\Delta &= \overline{\Delta}_{l-1} \\
&= \sum_{\substack{d \in [n],\\ i \in [d]}} \sum_{\substack{d_l \in [n],\\ i_l \in [d_l]}} \sum_{\nu_{l-1}=0} \Big(\mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l)}_{(i,i_l)}(\mathsf{XOR}(0, \overrightarrow{\nu}_{l-1}))] - \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l)}_{(i,i_l)}(\mathsf{XOR}(1, \overrightarrow{\nu}_{l-1}))] \Big) \\
&= \sum_{\substack{d \in [n],\\ i \in [d]}} \sum_{\substack{d_l \in [n],\\ i_l \in [d_l]}} \Big(\mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l)}_{(i,i_l)}((0))] - \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l)}_{(i,i_l)}((1))] \Big) \\
&= \sum_{\substack{d \in [n],\\ i \in [d]}} \sum_{d_l \in [n]} \Big(\mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l)}_{(i,1)}((0))] - \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l)}_{(i,d_l)}((1))] \Big) \\
&\quad \text{(Follows from the hybrid cancellation lemma, Lemma A.5, part 2)} \\
&= \sum_{d=1}^{n} \sum_{d_l=1}^{n} \left[\sum_{i=1}^{d} \Big(\mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l)}_{(i,1)}((0))] - \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l)}_{(i,d_l)}((1))] \Big) \right] \\
&= \sum_{d=1}^{n} \sum_{d_l=1}^{d} \Big(\mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l)}_{(1,1)}((0))] - \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l)}_{(d,d_l)}((1))] \Big) \\
&\quad \text{(Follows from the hybrid cancellation lemma, part 1)} \\
&= \mathbf{P}[\mathsf{O_A}; \Theta_{\text{first}}] - \mathbf{P}[\mathsf{O_A}; \Theta_{\text{last}}] \ = \ \mathbf{P}[\mathsf{O_A} \mid \Theta_{\text{first}}] \cdot \mathbf{P}[\Theta_{\text{first}}] - \mathbf{P}[\mathsf{O_A} \mid \Theta_{\text{last}}] \cdot \mathbf{P}[\Theta_{\text{last}}]
\end{aligned}$$

Now invoking Claims A.3 and A.4, we get

$$\begin{aligned}
|\Delta| &= |\mathbf{P}[\mathsf{O_A} \mid \Theta_{\text{first}}] \cdot \mathbf{P}[\Theta_{\text{first}}] - \mathbf{P}[\mathsf{O_A} \mid \Theta_{\text{last}}] \cdot \mathbf{P}[\Theta_{\text{last}}]| \\
&= \frac{\alpha}{2} \cdot |\mathbf{P}[\mathsf{O_A} \mid \Theta_{\text{first}}] - \mathbf{P}[\mathsf{O_A} \mid \Theta_{\text{last}}]| \\
&= \frac{\alpha}{2} \cdot \Delta_{\text{GSD}}(\mathsf{A}) \\
&> \frac{\alpha\epsilon'}{2} \quad \text{(Follows from our initial assumption, inequality 1)}
\end{aligned}$$

In the remainder of the appendix, we will give the proofs of all the lemmas and claims used to prove Lemma A.2.

A.2.2 Proof of Claim A.3

We will argue that A's view in its interaction with A' given the occurrence of Θ_{first} (resp. Θ_{last}) is distributed identically as its view in the GSD game when the challenge bit b (chosen by B) is 0 (resp. 1). From this, the claim will follow immediately.

When Θ_{first} occurs, *every* query of A of the form $\texttt{encrypt}(x, y)$ is replied to with a real ciphertext ($\mathcal{E}_{k_x}(k_y)$) and that of the form $\texttt{challenge}(x)$ is replied to with k_x—exactly the manner in which replies are created when $b = 0$ in the GSD game. Why is this? To see why the former is true, note that under the occurence of Θ_{first}, the only queries that could be replied to with fake ciphertexts are those of the form $\texttt{encrypt}(x, u_l)$ and either the same as or preceding the query $\texttt{encrypt}(u_{l-1}, u_l)$. But, given Θ_{first} occurs, u_{l-1} is the first node pointing at u_l and the reply to $\texttt{encrypt}(u_{l-1}, u_l)$ is real, thus implying that the replies for all $\texttt{encrypt}$ queries of A are real. To see why the latter is true, note that the reply to every query $\texttt{challenge}(x)$, including or succeeding $\texttt{challenge}(u_l)$ is always k_x (line 400 of A''s code), and when Θ_{first} occurs, $\texttt{challenge}(u_l)$ is the first $\texttt{challenge}$ query.

When Θ_{last} happens, things are a bit less straightforward. As above, the only queries whose replies could be faked by A' are those of the form $\texttt{encrypt}(x, u_l)$, either including or preceding $\texttt{encrypt}(u_{l-1}, u_l)$. However, under the occurrence of Θ_{last}, $\texttt{encrypt}(u_{l-1}, u_l)$ is the *last* query of this form, and even the reply for $\texttt{encrypt}(u_{l-1}, u_l)$ is fake, thus implying that every query of the form $\texttt{encrypt}(x, u_l)$ is replied to in a fake manner (specifically, with $\mathcal{E}_{k_x}(r_{u_l})$). Note also that given Θ_{last}, $\texttt{challenge}(u_l)$ is the last $\texttt{challenge}$ query, and thus, the reply to all $\texttt{challenge}(x)$ queries *except* $\texttt{challenge}(u_l)$ is a random key, sampled independently of k_x. For $\texttt{challenge}(u_l)$, the reply is k_{u_l}. Now comes the crucial part: Replying to the query $\texttt{challenge}(u_l)$ with k_{u_l} and to every query of the form $\texttt{encrypt}(x, u_l)$ with $\mathcal{E}_{k_x}(r_{u_l})$—where r_{u_l} is independent of (but identically distributed as) k_{u_l}—produces the same distribution on the replies as replying to $\texttt{challenge}(u_l)$ with r_{u_l} and to every query of the form $\texttt{encrypt}(x, u_l)$ with $\mathcal{E}_{k_x}(k_{u_l})$. Plus, we know that neither k_{u_l} nor r_{u_l} can be used in creating any other replies for A (other than those of the form $\texttt{encrypt}(x, u_l)$ or $\texttt{challenge}(u_l)$). In effect, the replies that A' provides to A conditioned on event Θ_{last}, are distributed exactly as B's replies when $b = 1$ in the GSD game—real ciphertexts for all $\texttt{encrypt}$ queries and a random key (independent of k_x) for every query of the form $\texttt{challenge}(x)$. ▮

A.2.3 Proof of Claim A.4

We will prove that $\mathbf{P}[\Theta_{\text{first}}] = \alpha/2$; the proof for the other part of the claim (namely, $\mathbf{P}[\Theta_{\text{last}}] = \alpha/2$) is quite similar and is omitted.

Some notations first. Recall that in any execution of A, the graph $G(\mathsf{A})$ and the set $V^{\mathsf{chal}}(\mathsf{A})$ are random variables dependent on the coins used by A and those used in the procedure for replying to A's queries. We define here some more random variables related to $G(\mathsf{A})$ and $V^{\mathsf{chal}}(\mathsf{A})$ and the manner in which these are created in any execution.

- Let *fchal* be the random variable corresponding to the first node in $V^{\mathsf{chal}}(\mathsf{A})$.
- For any fixed $w \in [n]$, let *fnode*(w) be the random variable corresponding to the first node pointing at w in $G(\mathsf{A})$.
- For any fixed $w \in [n]$, let *fpath*(w) be the random variable corresponding to the path in $G(\mathsf{A})$ that ends in w, that starts in a source (of $G(\mathsf{A})$) and is such that for every edge $x \to y$ in the path, $x =$ *fnode*(y). Let *flen*(w) be the length of *fpath*(w) (that is, the number of edges in it).

Since $G(\mathsf{A})$ is always acyclic, *fpath*(w) and *flen*(w) are well-defined (and uniquely so) for every value of $w \in [n]$ and every value assigned to $G(\mathsf{A})$.

Let us now consider the event Θ_{first} and re-phrase it in terms of the above definitions. Θ_{first} occurs if and only if in A's interaction with A', the variables $G(\mathsf{A}), V^{\mathsf{chal}}(\mathsf{A}), u_0, \cdots, u_l, s, b_s, \cdots, b_{l-2}$ and the bit b (chosen by $\mathcal{O}^{\Pi}_{k,b}$) are such that:

1. $\mathit{fchal} = u_l$;
2. $\mathit{fnode}(u_l) = u_{l-1}$;
3. $s = l - \mathit{flen}(u_l) = l - \mathit{flen}(u_{l-1}) - 1$ and $\mathit{fpath}(u_{l-1}) = (u_s, u_{s+1}, \cdots, u_{l-1})$[14]
4. For all $j \in \{0, 1, \cdots, s-1\}$, $u_j = 0$.
5. $b = 0$ and $b_s = b_{s+1} = \cdots = b_{l-2} = 0$.[15]

The last condition ensures that the replies that A' provides to A for all queries of the form $\texttt{encrypt}(u_j, u_{j+1})$ ($j \geq s$) are real ciphertexts. (How so? For this, first observe that the reply for any query $\texttt{encrypt}(u_j, u_{j+1})$, for $j > s$, is real if and only if $b_j = b_{j-1}$ and that for the query $\texttt{encrypt}(u_s, u_{s+1})$ is real if and only if $b = b_s$. Then notice that $b_{l-1} = 0$ always, which means that for Θ_{first} to occur, all the other b_j's and the bit b must also be zero.)

The probability that Θ_{first} occurs is thus:

$$\begin{aligned}
\mathbf{P}[\Theta_{\text{first}}] &= \sum_{l'=0}^{l-1} \mathbf{P}\left[\begin{array}{c} \mathit{fchal} = u_l;\ \mathit{fnode}(u_l) = u_{l-1};\ \mathit{flen}(u_{l-1}) = l';\ \mathit{fpath}(u_{l-1}) = (u_{l-l'-1}, \cdots, u_{l-1}); \\ u_0 = \cdots = u_{l-l'-2} = 0;\ b = 0;\ b_{l-l'-1} = \cdots = b_{l-2} = 0 \end{array}\right] \\
&= \sum_{l'=0}^{l-1} \sum_{w_l, \cdots, w_{l-l'-1} \in [n]} \mathbf{P}\left[\begin{array}{c} \mathit{fchal} = w_l;\ \mathit{fnode}(w_l) = w_{l-1};\ \mathit{flen}(w_{l-1}) = l'; \\ \mathit{fpath}(w_{l-1}) = (w_{l-l'-1}, \cdots, w_{l-2}, w_{l-1}); \\ u_l = w_l;\ \ \cdots;\ u_{l-l'-1} = w_{l-l'-1};\ u_{l-l'-2} = \cdots = u_0 = 0; \\ b = 0;\ b_{l-l'-1} = \cdots = b_{l-2} = 0 \end{array}\right]
\end{aligned}$$

Let $\overrightarrow{w}$ denote the sequence $(w_{l-l'-1}, w_{l-l'}, \cdots, w_{l-1})$. For any $l' \in \{0, \cdots, l-1\}$ and $\overrightarrow{w} \in [n]^{l'+2}$, let $E_1^{(l',\overrightarrow{w})}$ and $E_2^{(l',\overrightarrow{w})}$ be events defined as follows:

$$E_1^{(l',\overrightarrow{w})} = \Big(\mathit{fchal} = w_l \wedge \mathit{fnode}(w_l) = w_{l-1} \wedge \mathit{flen}(w_{l-1}) = l' \wedge \mathit{fpath}(w_{l-1}) = (w_{l-l'-1}, \cdots, w_{l-1})\Big)$$

$$E_2^{(l',\overrightarrow{w})} = \Big(u_l = w_l \wedge \cdots \wedge u_{l-l'-1} = w_{l-l'-1} \wedge u_{l-l'-2} = \cdots = u_0 = 0 \wedge b = 0 \wedge b_{l-l'-1} = \cdots = b_{l-2} = 0\Big)$$

$\mathbf{P}[\Theta_{\text{first}}]$ can now be written succinctly as follows:

$$\begin{aligned}
\mathbf{P}[\Theta_{\text{first}}] &= \sum_{l'=0}^{l-1} \sum_{\overrightarrow{w} \in [n]^{l'+2}} \mathbf{P}[E_1^{(l',\overrightarrow{w})};\ E_2^{(l',\overrightarrow{w})}] \\
&= \sum_{l'=0}^{l-1} \sum_{\overrightarrow{w} \in [n]^{l'+2}} \mathbf{P}[E_1^{(l',\overrightarrow{w})} \mid E_2^{(l',\overrightarrow{w})}] \cdot \mathbf{P}[E_2^{(l',\overrightarrow{w})}]
\end{aligned}$$

[14] Throughout the proof of Claim A.4 and Lemma A.5, we denote paths in $G(\mathsf{A})$ as sequences of nodes, but without any zeroes prepended to the first node.

[15] While expressing Θ_{last} in terms of *fchal*, *fnode*$(\cdot)$, *fpath*$(\cdot)$ etc., the first two conditions and the last condition must be suitably modified (u_l must be the *last* node in $V^{\mathsf{chal}}(\mathsf{A})$, u_{l-1} must be the *last* node pointing at u_l and $b, b_s, \cdots, b_{l-2}$ must all equal 1) while the other conditions remain the same as above.

Let us first compute $\mathbf{P}[E_2^{(l',\overrightarrow{w})}]$ for any arbitrary l' and $\overrightarrow{w}$. Notice that the values $u_0, \cdots, u_l, b_s, \cdots b_{l-2}$ are all generated by A' independently of each other and of the bit b chosen by A''s oracle $\mathcal{O}_{k,b}^{\Pi}$. Using this fact, computing $\mathbf{P}[E_2^{(l',\overrightarrow{w})}]$ is quite straightforward:

$$\begin{aligned}
\mathbf{P}[E_2^{(l',\overrightarrow{w})}] &= \mathbf{P}\left[\begin{array}{c} u_l = w_l;\ u_{l-1} = w_{l-1};\ \cdots;\ u_{l-l'-1} = w_{l-l'-1};\ u_{l-l'-2} = \cdots = u_0 = 0; \\ b = 0;\ b_{l-l'-1} = \cdots = b_{l-2} = 0 \end{array}\right] \\
&= \mathbf{P}[u_l = w_l] \cdot \mathbf{P}[u_{l-1} = w_{l-1}] \cdot \mathbf{P}[u_{l-2} = w_{l-2}] \cdots \mathbf{P}[u_{l-l'-1} = w_{l-l'-1}] \\
&\qquad \times\ \mathbf{P}[u_{l-l'-2} = 0] \cdot \mathbf{P}[u_{l-l'-3} = 0] \cdots \mathbf{P}[u_0 = 0] \\
&\qquad\qquad \times\ \mathbf{P}[b = 0] \cdot \mathbf{P}[b_{l-l'-1} = 0] \cdots \mathbf{P}[b_{l-2} = 0] \\
&= \left\{\frac{1}{n} \cdot \frac{1}{n} \cdot \left(\frac{2}{2n+1}\right)^{l'}\right\} \times \left\{\left(\frac{1}{2n+1}\right)^{l-l'-1}\right\} \times \left\{\frac{1}{2} \cdot \left(\frac{1}{2}\right)^{l'}\right\} \\
&\qquad \text{(To understand this step, observe the probability distribution associated} \\
&\qquad \text{with the } u_j\text{'s in line 001 of figure 2)} \\
&= \frac{1}{2n^2} \cdot \left(\frac{1}{2n+1}\right)^{l'} \cdot \left(\frac{1}{2n+1}\right)^{l-l'-1} \;=\; \frac{1}{2n^2(2n+1)^{l-1}} \;=\; \frac{\alpha}{2}
\end{aligned}$$

Plugging this into the expression for $\mathbf{P}[\Theta_{\text{first}}]$, we get:

$$\mathbf{P}[\Theta_{\text{first}}] \;=\; \frac{\alpha}{2} \cdot \sum_{l'=0}^{l-1} \sum_{\overrightarrow{w} \in [n]^{l'+2}} \mathbf{P}[E_1^{(l',\overrightarrow{w})} \mid E_2^{(l',\overrightarrow{w})}]$$

We will now focus on the quantity $\mathbf{P}[E_1^{(l',\overrightarrow{w})} \mid E_2^{(l',\overrightarrow{w})}]$. Let us denote this quantity by p. Our goal will be to show that the sum

$$S_p := \sum_{l'=0}^{l-1} \sum_{\overrightarrow{w} \in [n]^{l'+2}} p$$

is equal to one. From this, the claim will follow immediately.

Proving $S_p = 1$: Fix l' and $\overrightarrow{w}$. A transcript of A's execution is called a *good* transcript if the event $E_1^{(l',\overrightarrow{w})}$ occurs during that execution; it is called *bad* otherwise. (*Note:* "Goodness" is a concept well-defined for any execution of A, not necessarily one involving interaction with A'.) Given a transcript of A's execution, it is easy to tell whether it is good or not—simply check if the queries contained in it satisfy the following conditions: *(i)* $\mathit{fchal} = w_l$; *(ii)* $\mathit{fnode}(w_l) = w_{l-1}$; *(iii)* $\mathit{flen}(w_{l-1}) = l'$; and *(iv)* $\mathit{fpath}(w_{l-1}) = (w_{l-l'-1}, \cdots, w_{l-1})$. This essentially boils down to verifying that:

(a) `challenge`(w_l) is the first `challenge` query made by A;

(b) for every $j \in \{l-l', \cdots, l-1\}$, `encrypt`$(w_{j-1}, w_j)$ is the first query of the form `encrypt`(x, w_j) made by A; and

(c) no query of the form `encrypt`$(x, w_{l-l'-1})$ is ever made by A.

A *bad query* in a transcript t is one that violates any of the above conditions; that is, a query $\mathsf{q}_i \in \mathsf{t}$ is bad if

- $\mathsf{q}_i = \texttt{challenge}(x)$ for some $x \neq w_l$ and the query $\texttt{challenge}(w_l)$ does not occur before it in t; or
- $\mathsf{q}_i = \texttt{encrypt}(x, w_j)$ (for some $j \in \{l - l', \cdots, l-1\}$ and $x \neq w_{j-1}$) and the query $\texttt{encrypt}(w_{j-1}, w_j)$ does not occur before it in t; or
- q_i is of the form $\texttt{encrypt}(x, w_{l-l'-1})$.[16]

Clearly, every bad transcript contains at least one bad query. The *longest good prefix* of a bad transcript $\mathsf{t} = (\mathsf{q}_1, \mathsf{r}_1, \cdots, \mathsf{q}_f, \mathsf{r}_f)$ is the sequence $(\mathsf{q}_1, \mathsf{r}_1, \cdots, \mathsf{q}_i, \mathsf{r}_i)$ $(i < f)$ such that the queries $\mathsf{q}_1, \cdots, \mathsf{q}_i$ are all good but the query q_{i+1} is bad. (*Note:* The longest good prefix of a bad transcript could possibly be empty—the very first query in the transcript could be bad.)

For any transcript t, the *relevant* transcript corresponding to t, $rel(\mathsf{t})$, is defined as follows: if t is good, $rel(\mathsf{t}) = \mathsf{t}$, else $rel(\mathsf{t})$ is the longest good prefix of t. Note that relevant transcripts contain only good queries. For any multiset of transcripts, $\mathcal{T}$, let $rel(\mathcal{T})$ denote the multiset of relevant transcripts corresponding to $\mathcal{T}$; that is, $rel(\mathcal{T}) = \{rel(\mathsf{t}) \mid \mathsf{t} \in \mathcal{T}\}$. An execution of A is called relevant if it yields a relevant transcript. (Here, when we say "execution", we also refer to *partial* executions of A, that is, executions upto a certain query, e.g. the first bad query, made by A.)

Now let us consider A's interaction with A'. Suppose that the event $E_2^{(l',\vec{w})}$ is known to have occurred, which means that s (computed in line 003 of A') is equal to $l - l' - 1$. Let $\mathcal{T}_{\mathsf{A} \mid E_2^{(l',\vec{w})}}$ denote the multiset of all possible transcripts that can be generated given $E_2^{(l',\vec{w})}$; thus, the size of $\mathcal{T}_{\mathsf{A} \mid E_2^{(l',\vec{w})}}$, say T, is equal to the number of possible assignments to the random variables $k_1, \cdots, k_{u_s - 1}, k_{u_s+1}, \cdots, k_n$ and $r_{u_s}, \cdots, r_{u_l}$ (generated by A' in line 004), the key k (generated by $\mathcal{O}^{\Pi}_{k,b}$), the randomness, $r_{\mathcal{E}}$, used in performing all encryption operations, and, finally, the random coins, r_{A}, used by A. Some of the transcripts in $\mathcal{T}_{\mathsf{A} \mid E_2^{(l',\vec{w})}}$ are good while others are bad, and the quantity p we defined earlier on is the ratio of the number of good transcripts in $\mathcal{T}_{\mathsf{A} \mid E_2^{(l',\vec{w})}}$ to T.

Consider the multiset $rel(\mathcal{T}_{\mathsf{A} \mid E_2^{(l',\vec{w})}})$. There are two interesting features of this multiset:

(a) First, notice that, for each transcript in $rel(\mathcal{T}_{\mathsf{A} \mid E_2^{(l',\vec{w})}})$, none of the replies given to the adversary involve the random variables $r_{u_s}, r_{u_s+1}, \cdots, r_{u_l}$ at all. Why is this? Given that $E_2^{(l',\vec{w})}$ occurs, these variables could be used in replying to queries either of the form $\texttt{encrypt}(x, w_s)$, or else of the form $\texttt{encrypt}(x, w_j)$ $(j \in \{s+1, \cdots, l\})$ *provided the latter type of queries are made before* $\texttt{encrypt}(w_{j-1}, w_j)$. But any such query would be a bad query, and, by definition, transcripts in $rel(\mathcal{T}_{\mathsf{A} \mid E_2^{(l',\vec{w})}})$ don't contain such queries!

Thus, every relevant execution of A in its interaction with A' given $E_2^{(l',\vec{w})}$ is completely determined by an assignment to $k_1, \cdots, k_{u_s-1}, k_{u_s} := k, k_{u_s+1}, \cdots, k_n$, and to $r_{\mathcal{E}}$ and r_{A}. Let $\hat{\mathcal{T}}_{\mathsf{A} \mid E_2^{(l',\vec{w})}}$ denote the multiset of transcripts containing one transcript for each such execution. (*Note:* $\hat{\mathcal{T}}_{\mathsf{A} \mid E_2^{(l',\vec{w})}}$ contains the same transcripts as in $rel(\mathcal{T}_{\mathsf{A} \mid E_2^{(l',\vec{w})}})$ but is smaller than it, because we ignore the r_{u_i}'s while enumerating these transcripts.)

(b) Now observe that in every transcript in $\hat{\mathcal{T}}_{\mathsf{A} \mid E_2^{(l',\vec{w})}}$, the replies given for a query of the form $\texttt{encrypt}(x, y)$ is a real ciphertext (that is, $\mathcal{E}_{k_x}(k_y)$) and that of the form $\texttt{challenge}(x)$ is

[16]One could consider adding another requirement to the definition of bad queries, namely that $\mathsf{q}_i \neq \texttt{corrupt}(w_j)$ for any $j \in \{l - l' - 1, \cdots l\}$; the proof remains essentially the same even without this extra condition.

a real key (that is, k_x). This is exactly the manner in which replies to queries are created in the GSD game, conditioned on the event that the challenge bit b equals 0! Thus, if we were to consider only relevant executions of A, the view of A in the GSD game given $b = 0$ has the same distribution as its view in its interaction with A' given $E_2^{(l',\overrightarrow{w})}$ occurs.

It follows that the quantity p is equal to the probability that event $E_1^{(l',\overrightarrow{w})}$ takes place when $b = 0$ in the GSD game. S_p simply sums this probability over all possible values of l' and $\overrightarrow{w}$ and must thus be equal to 1. ▌

A.2.4 Proof of the Hybrid Cancellation Lemma (Lemma A.5)

Let us first set up some notation that will be useful throughout the lemma. We think of the interaction between A' and A as a game in which A makes multiple queries and A' replies to these queries in some prescribed manner based on the random variables $u_0, \cdots, u_l, b_s, \cdots, b_{l-2}$ (and other randomness involved in generating keys, and forming ciphertexts). We denote this game by $\mathbf{Game}_0$. Throughout the proof of the lemma, we will be considering various modifications of this game and making statements of the sort: *"the probability that event E occurs in* $\mathbf{Game}_0$ *is the same as the probability that E occurs in some modified version of* $\mathbf{Game}_0$*"*. To formalize such statements, we adopt the following convention: For any event E, $\mathbf{P}[E]$ denotes the probability that E occurs in $\mathbf{Game}_0$, while for any modification of $\mathbf{Game}_0$, say $\mathbf{Game}'$, the probability that E occurs in $\mathbf{Game}'$ is denoted by $\mathbf{P}_{\mathbf{Game}'}[E]$.

Proof of Part 1. Fix d and i such that $d \in \{2, \cdots, n\}$ and $i \in \{1, \cdots, d-1\}$. Define events $E_{d,i}^{(0)}$ and $E_{d,i}^{(1)}$ as follows:

$$E_{d,i}^{(0)} = \bigvee_{d_l=1}^{n} \Theta_{(i,d_l)}^{(d,d_l)}((1)) \qquad \text{and} \qquad E_{d,i}^{(1)} = \bigvee_{d_l=1}^{n} \Theta_{(i+1,1)}^{(d,d_l)}((0))$$

Now consider the following modified version of $\mathbf{Game}_0$, which we denote by $\mathbf{Game}_i$. In this modification, A' first generates keys $k_1, \cdots, k_n$ (*Note: All* keys are generated) and, subsequently, replies to *all* `encrypt` queries using real ciphertexts, just as in the GSD game. The responses to all `corrupt` queries are also just as they are in the GSD game. For the `challenge` queries, however, A' does the following—it replies to the first i queries of the form `challenge`(x) with a random element of $\{0,1\}^\eta$, sampled independently of k_x, while to the other queries of this form, its reply is k_x.

We claim that the view of A in $\mathbf{Game}_0$ given $E_{d,i}^{(0)}$ occurs or given $E_{d,i}^{(1)}$ occurs is the same as its view in $\mathbf{Game}_i$ given that $V^{\mathsf{chal}}(\mathsf{A})$ has size d. It is easy to see why this is true for the case when $E_{d,i}^{(1)}$ occurs (follows almost immediately from the definition of event $\Theta_{(i+1,d_l)}^{(d,d_l)}((0))$). The other part is somewhat more non-trivial and we will prove it in greater detail.

Given that event $E_{d,i}^{(0)}$ occurs, the replies that A' provides to A (in $\mathbf{Game}_0$) are decided as follows:

(a) for each query of the form `challenge`(x) that is issued *before* the ith `challenge` query—which is the same as `challenge`(u_l)—the reply is a random key from $\{0,1\}^\eta$, sampled independently of k_x, and for each query `challenge`(x) issued *after* the ith `challenge` query, the reply is k_x;

(b) for the ith `challenge` query, namely `challenge`(u_l), the reply is k_{u_l}.

(c) for each query of the form $\texttt{encrypt}(x, u_l)$, the reply is $\mathcal{E}_{k_x}(r_{u_l})$. (This is because, given $E^{(0)}_{d,i}$ occurs, the reply for the last query of the form $\texttt{encrypt}(x, u_l)$—which is the same as $\texttt{encrypt}(u_{l-1}, u_l)$—is a fake ciphertext $\mathcal{E}_{k_{u_{l-1}}}(r_{u_l})$. This means that the reply for *every* query of that form must be fake, too.)

(d) for every other $\texttt{encrypt}$ (and $\texttt{corrupt}$) query, the reply is just as in the GSD game.

Notice that k_{u_l} and r_{u_l} are generated by A' independently of each other and are not used in replying to any query other than that of the form $\texttt{encrypt}(x, u_l)$ or $\texttt{challenge}(u_l)$. So, if we modify conditions (b) and (c) as below

(b') for the ith $\texttt{challenge}$ query, namely $\texttt{challenge}(u_l)$, the reply is r_{u_l}; and

(c') for each query of the form $\texttt{encrypt}(x, u_l)$, the reply is $\mathcal{E}_{k_x}(k_{u_l})$.

the distribution of the replies given to A remains unmodified. Conditions (a), (b'), (c') and (d) can now succinctly be written as follows:

(a") for each query $\texttt{challenge}(x)$ that is issued *before the* $(i+1)$*th* $\texttt{challenge}$ *query* the reply is a random key from $\{0,1\}^\eta$, sampled independently of k_x, and for each query $\texttt{challenge}(x)$ issued *after* the $(i+1)$th $\texttt{challenge}$ query (and including it), the reply is k_x;

(b") for each $\texttt{encrypt}$ and $\texttt{corrupt}$ query, the reply is just as in the GSD game.

This is exactly the manner in which replies to A's queries are decided in $\mathbf{Game}_i$. As such, the view of A given $E^{(0)}_{d,i}$ occurs is the same as its view in $\mathbf{Game}_i$ given $V^{\mathsf{chal}}(\mathsf{A})$ has size d.

We are left with proving that $\mathbf{P}[E^{(0)}_{d,i}] = \mathbf{P}[E^{(1)}_{d,i}]$. The proof for this statement uses techniques similar to those used to prove Claim A.4; we will, thus, omit many details.

For any $w \in [n]$, let $\mathit{fnode}(w), \mathit{fpath}(w)$ and $\mathit{flen}(w)$ be random variables as defined in Section A.2.3. Let $\mathit{lnode}(w)$ be the random variable corresponding to the last node pointing at w in $G(\mathsf{A})$. Let $\Phi^{(d)}$ denote the event that the size of $V^{\mathsf{chal}}(\mathsf{A})$ equals d and $\Phi^{(d)}_{(i)}(w)$ ($w \in [n]$) the event that $\Phi^{(d)}$ occurs and the ith node in it is w. As with $\mathit{fnode}(w), \mathit{fpath}(w)$ and $\mathit{flen}(w)$, the events $\Phi^{(d)}, \Phi^{(d)}_{(i)}(w)$ and the random variable $\mathit{lnode}(w)$ are well-defined for any execution of A, not necessarily one involving interaction with A'.

The probability that event $E^{(0)}_{d,i}$ occurs can be written in terms of these random variables as follows:

$$\begin{aligned}
\mathbf{P}[E^{(0)}_{d,i}] &= \sum_{d_l=1}^{n} \mathbf{P}[\Theta^{(d,d_l)}_{(i,d_l)}((1))] \\
&= \sum_{d_l=1}^{n}\sum_{l'=0}^{l-1} \mathbf{P}\left[\begin{array}{c} \Phi^{(d)}_{(i)}(u_l);\ \mathit{Indegree}(u_l) = d_l;\ \mathit{lnode}(u_l) = u_{l-1};\ \mathit{flen}(u_{l-1}) = l'; \\ \mathit{fpath}(u_{l-1}) = (u_{l-l'-1}, \cdots, u_{l-1}); \\ u_{l-l'-2} = \cdots = u_0 = 0;\ b = 1;\ b_{l-l'-1} = \cdots = b_{l-2} = 1 \end{array}\right]
\end{aligned}$$

To understand the last part of this step (that is, why we require $b, b_{l-l'-1}, \cdots b_{l-2}$ all to be equal to 1), observe that under the occurrence of $E^{(0)}_{d,i}$, (a) the reply to the query $\texttt{encrypt}(u_{l-1}, u_l)$ must be fake, which means that $b_{l-1} \oplus b_{l-2} = 1$ and so $b_{l-2} = 1$; and (b) the reply to all queries of the form $\texttt{encrypt}(u_{j-1}, u_j)$ ($j < l$ and $j > l - 1 - \mathit{flen}(u_{l-1})$) must be real, which means that $b_{l-2} \oplus b_{l-3} = 0, b_{l-3} \oplus b_{l-4} = 0, \cdots, b_{l-\mathit{flen}(u_{l-1})} \oplus b_{l-1-\mathit{flen}(u_{l-1})} = 0, b_{l-1-\mathit{flen}(u_{l-1})} \oplus b = 0$.

For any $l' \in \{0, \cdots, l-1\}$, and any vector of values $\overrightarrow{w} = (w_{l-l'-1}, \cdots, w_l) \in [n]^{l'+2}$, let $E_1^{(l',\overrightarrow{w})}$ and $E_2^{(l',\overrightarrow{w})}$ be events defined as follows:

$$E_1^{(l',\overrightarrow{w})} = \left(\begin{array}{c} \Phi_{(i)}^{(d)}(w_l) \wedge \mathit{Indegree}(w_l) = d_l \wedge \mathit{lnode}(w_l) = w_{l-1} \wedge \mathit{flen}(w_{l-1}) = l' \wedge \\ \mathit{fpath}(w_{l-1}) = (w_{l-l'-1}, \cdots, w_{l-1}) \end{array} \right)$$

$$E_2^{(l',\overrightarrow{w})} = \Big(u_l = w_l \wedge \cdots \wedge u_{l-l'-1} = w_{l-l'-1} \wedge u_{l-l'-2} = \cdots = u_0 = 0 \wedge b = 1 \wedge b_{l-l'-1} = \cdots = b_{l-2} = 1 \Big)$$

$\mathbf{P}[E_{d,i}^{(0)}]$ can now be expressed in terms of these events as follows:

$$\begin{aligned} \mathbf{P}[E_{d,i}^{(0)}] &= \sum_{d_l=1}^{n} \sum_{l'=0}^{l-1} \sum_{\overrightarrow{w} \in [n]^{l'+2}} \mathbf{P}[E_1^{(l',\overrightarrow{w})};\ E_2^{(l',\overrightarrow{w})}] \\ &= \sum_{d_l=1}^{n} \sum_{l'=0}^{l-1} \sum_{\overrightarrow{w} \in [n]^{l'+2}} \mathbf{P}[E_1^{(l',\overrightarrow{w})} \mid E_2^{(l',\overrightarrow{w})}] \cdot \mathbf{P}[E_2^{(l',\overrightarrow{w})}] \end{aligned}$$

As in the proof of Claim A.4, we can show that $\mathbf{P}[E_2^{(l',\overrightarrow{w})}]$ is equal to $\alpha/2$ for every choice of l' and $\overrightarrow{w}$, and so

$$\mathbf{P}[E_{d,i}^{(0)}] = \frac{\alpha}{2} \cdot \sum_{d_l=1}^{n} \sum_{l'=0}^{l-1} \sum_{\overrightarrow{w} \in [n]^{l'+2}} \mathbf{P}[E_1^{(l',\overrightarrow{w})} \mid E_2^{(l',\overrightarrow{w})}]$$

We now claim that the conditional probability $\mathbf{P}[E_1^{(l',\overrightarrow{w})} | E_2^{(l',\overrightarrow{w})}]$ is equal to the probability of occurrence of $E_1^{(l',\overrightarrow{w})}$ in $\mathbf{Game}_i$. The proof of this claim uses the same ideas as used in the proof of Claim A.4 (specifically, one needs to carefully establish a one-to-one correspondance between transcripts—complete ones as well as partial ones—that do not violate $E_1^{(l',\overrightarrow{w})}$ in $\mathbf{Game}_0$ and those that do not violate it in $\mathbf{Game}_i$); details of the proof of the claim are omitted. Using the claim, we can re-write $\mathbf{P}[E_{d,i}^{(0)}]$ as

$$\begin{aligned} \mathbf{P}[E_{d,i}^{(0)}] &= \frac{\alpha}{2} \cdot \sum_{d_l=1}^{n} \sum_{l'=0}^{l-1} \sum_{\overrightarrow{w} \in [n]^{l'+2}} \mathbf{P}_{\mathbf{Game}_i}[E_1^{(l',\overrightarrow{w})}] \\ &= \frac{\alpha}{2} \cdot \sum_{d_l=1}^{n} \sum_{l'=0}^{l-1} \sum_{\overrightarrow{w} \in [n]^{l'+2}} \mathbf{P}_{\mathbf{Game}_i} \left[\begin{array}{c} \Phi_{(i)}^{(d)}(w_l);\ \mathit{Indegree}(w_l) = d_l;\ \mathit{lnode}(w_l) = w_{l-1}; \\ \mathit{flen}(w_{l-1}) = l';\ \mathit{fpath}(w_{l-1}) = (w_{l-l'-1}, \cdots, w_{l-1}) \end{array} \right] \\ &= \frac{\alpha}{2} \cdot \sum_{w_l=1}^{n} \sum_{d_l=1}^{n} \mathbf{P}_{\mathbf{Game}_i}[\Phi_{(i)}^{(d)}(w_l);\ \mathit{Indegree}(w_l) = d_l] \;=\; \frac{\alpha}{2} \cdot \mathbf{P}_{\mathbf{Game}_i}[\Phi^{(d)}] \end{aligned}$$

Using essentially the same approach, we can also equate $\mathbf{P}[E_{d,i}^{(1)}]$ to $(\alpha/2)\mathbf{P}_{\mathbf{Game}_i}[\Phi^{(d)}]$. First, define two events $\tilde{E}_1^{(l',\overrightarrow{w})}$ and $\tilde{E}_2^{(l',\overrightarrow{w})}$ as follows:

$$\tilde{E}_1^{(l',\overrightarrow{w})} = \left(\begin{array}{c} \Phi_{(i+1)}^{(d)}(w_l) \wedge \mathit{Indegree}(w_l) = d_l \wedge \mathit{fnode}(w_l) = w_{l-1} \wedge \mathit{flen}(w_{l-1}) = l' \wedge \\ \mathit{fpath}(w_{l-1}) = (w_{l-l'-1}, \cdots, w_{l-1}) \end{array} \right)$$

$$\tilde{E}_2^{(l',\vec{w})} = \Big(u_l = w_l \wedge \cdots \wedge u_{l-l'-1} = w_{l-l'-1} \wedge u_{l-l'-2} = \cdots = u_0 = 0 \wedge b = 0 \wedge b_{l-l'-1} = \cdots = b_{l-2} = 0\Big)$$

(Notice how $\tilde{E}_1^{(l',\vec{w})}$ differs from $E_1^{(l',\vec{w})}$: We require w_l to be the $(i+1)$th node in $V^{\mathsf{chal}}(\mathsf{A})$ and w_{l-1} to be the *first* node pointing at w_l. Also notice that in $\tilde{E}_2^{(l',\vec{w})}$, we require the b_i's to be equal to 0, not 1.) Now express $\mathbf{P}[E_{d,i}^{(1)}]$ in terms of these events:

$$\begin{aligned}
\mathbf{P}[E_{d,i}^{(1)}] &= \sum_{d_l=1}^{n}\sum_{l'=0}^{l-1}\sum_{\vec{w}\in[n]^{l'+2}} \mathbf{P}[\tilde{E}_1^{(l',\vec{w})};\ \tilde{E}_2^{(l',\vec{w})}] \\
&= \sum_{d_l=1}^{n}\sum_{l'=0}^{l-1}\sum_{\vec{w}\in[n]^{l'+2}} \mathbf{P}[\tilde{E}_1^{(l',\vec{w})} \mid \tilde{E}_2^{(l',\vec{w})}]\cdot\mathbf{P}[\tilde{E}_2^{(l',\vec{w})}] \\
&= \frac{\alpha}{2}\cdot\sum_{d_l=1}^{n}\sum_{l'=0}^{l-1}\sum_{\vec{w}\in[n]^{l'+2}} \mathbf{P}[\tilde{E}_1^{(l',\vec{w})} \mid \tilde{E}_2^{(l',\vec{w})}]
\end{aligned}$$

Again, $\mathbf{P}[\tilde{E}_1^{(l',\vec{w})} \mid \tilde{E}_2^{(l',\vec{w})}]$ can be shown to be equal to $\mathbf{P}_{\mathbf{Game}_i}[\tilde{E}_1^{(l',\vec{w})}]$, using which the desired expression for $\mathbf{P}[E_{d,i}^{(1)}]$ is easily obtained:

$$\begin{aligned}
\mathbf{P}[E_{d,i}^{(1)}] &= \frac{\alpha}{2}\cdot\sum_{d_l=1}^{n}\sum_{l'=0}^{l-1}\sum_{\vec{w}\in[n]^{l'+2}} \mathbf{P}_{\mathbf{Game}_i}[\tilde{E}_1^{(l',\vec{w})}] \\
&= \frac{\alpha}{2}\cdot\sum_{d_l=1}^{n}\sum_{l'=0}^{l-1}\sum_{\vec{w}\in[n]^{l'+2}} \mathbf{P}_{\mathrm{GSD}(i)}\left[\begin{array}{l}\Phi_{(i+1)}^{(d)}(w_l);\ \mathit{Indegree}(w_l) = d_l;\ \mathit{fnode}(w_l) = w_{l-1};\\ \mathit{flen}(w_{l-1}) = l';\ \mathit{fpath}(w_{l-1}) = (w_{l-l'-1},\cdots,w_{l-1})\end{array}\right] \\
&= \frac{\alpha}{2}\cdot\sum_{w_l=1}^{n}\sum_{d_l=1}^{n}\mathbf{P}_{\mathbf{Game}_i}[\Phi_{(i+1)}^{(d)}(w_l);\ \mathit{Indegree}(w_l) = d_l] \;=\; \frac{\alpha}{2}\cdot\mathbf{P}_{\mathbf{Game}_i}[\Phi^{(d)}] \qquad\blacksquare
\end{aligned}$$

Proof of Part 2. Fix $j, d, d_l, \cdots, d_j$ and $i, i_l, \cdots, i_j$ such that $i \in [d], i_l \in [d_l], \cdots, i_j \in [d_j]$ and fix a bitvector $\vec{\nu}_j \in \{0,1\}^{l-j}$. Consider the following modified version of $\mathbf{Game}_0$, which we call $\mathbf{Game}_{j,i_j}$. In the setup phase, A' first generates the values $u_j, u_{j+1}, \cdots, u_l$ and $b_j, \cdots, b_{l-2}, b_{l-1}$ and it does so exactly as in the original version: u_l and u_{l-1} are sampled uniformly at random from $[n]$; for each $j' \in \{j, \cdots, l-2\}$, $\mathbf{P}[u_{j'} = 0] = 1/(2n+1)$ and $\mathbf{P}[u_{j'} = x]$ (for any $x \in [n]$) equals $2/(2n+1)$; $b_{l-1} = 0$; and, $b_j, \cdots, b_{l-2}$ are all sampled uniformly at random from $\{0,1\}$. A' then sets $b_{j-1} = \nu_j \oplus \nu_{j+1} \oplus \cdots \oplus \nu_{l-1}$. Finally, A' generates keys $k_1, \cdots, k_n$ (*Note again: All* keys are generated!) and also some other random values $r_{u_j}, \cdots, r_{u_l}$ (each sampled independently and uniformly at random from $\{0,1\}^\eta$).

In the execution phase, the replies to A's queries are decided as follows: For any `corrupt` query, `corrupt`(x), the reply is simply k_x; for every query of the form `challenge`(x) made before `challenge`(u_l), the reply is a random bitstring, generated independently of k_x, and for every such query made after `challenge`(u_l) (and including it as well), the reply is k_x. For every query of the form `encrypt`$(u_{j'}, u_{j'+1})$ such that $j' \in \{j, \cdots, l-1\}$, the reply is real, that is $\mathcal{E}_{k_{u_{j'}}}(k_{u_{j'+1}})$, if and only if $b_{j'} = b_{j'-1}$ (and is fake, that is $\mathcal{E}_{k_{u_{j'}}}(r_{u_{j'+1}})$ otherwise); for every query of the form `encrypt`$(x, u_{j'})$ made before (resp. after) `encrypt`$(u_{j'-1}, u_{j'})$, the reply is fake (resp. real). For all other `encrypt` queries, except those of the form `encrypt`(x, u_l), the reply is always real. For queries of the form $\texttt{encrypt}(x, u_j)$, the matter is a bit tricky: *the first i_j queries of this form (i_j as fixed earlier on) are replied to with fake ciphertexts ($\mathcal{E}_{k_x}(r_{u_j})$)* while the rest with real ones ($\mathcal{E}_{k_x}(k_{u_j})$).

For any $w \in [n]$, and any execution of A either in $\mathbf{Game}_0$ or in $\mathbf{Game}_{j,i_j}$, we define the following event, denoted $\Phi^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(\overrightarrow{\nu}_j, w)$. This event occurs if and only if

- $V^{\mathsf{chal}}(\mathsf{A})$ has size d and u_l is the ith node in it; and
- For all $j' \in \{j+1, \cdots, l\}$, *Indegree*$(u_{j'}) = d_{j'}$, $u_{j'-1}$ is the $i_{j'}$th node pointing at $u_{j'}$ in $G(\mathsf{A})$ and the reply given for query $\texttt{encrypt}(u_{j'-1}, u_{j'})$ is real if and only if $\nu_{j'-1} = 0$; and
- *Indegree*$(u_j) = d_j$ and w is the i_jth node pointing at u_j in $G(\mathsf{A})$.

Let $\Phi^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1})}(\overrightarrow{\nu}_j)$ denote the event that $\Phi^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(\overrightarrow{\nu}_j, w)$ occurs for some $w \in [n]$; that is $\Phi^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1})}(\overrightarrow{\nu}_j) = \bigvee_{w\in[n]} \Phi^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(\overrightarrow{\nu}_j, w)$. Let $E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)$ and $E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)$ be defined as follows:

$$E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j) = \Theta^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(1\cdot\overrightarrow{\nu}_j) \qquad \text{and} \qquad E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j) = \Theta^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1},i_j+1)}(0\cdot\overrightarrow{\nu}_j)$$

Our task is to show that $E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j) \equiv E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)$, that is, (a) $E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j) \simeq E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)$ and (b) $\mathbf{P}[E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)] = \mathbf{P}[E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)]$. Proving part (a) is relatively simple—one only needs to argue that the view of A in $\mathbf{Game}_0$ given $E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)$ occurs or given $E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)$ occurs is the same as its view in $\mathbf{Game}_{j,i_j}$ given that the event $\Phi^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1})}(\overrightarrow{\nu}_j)$ occurs in that game. This follows almost immediately from the definitions of $E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j), E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)$ and of $\Phi^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1})}(\overrightarrow{\nu}_j)$. (The details of the proof are omitted.) Proving $\mathbf{P}[E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)] = \mathbf{P}[E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)]$ is the hard part and we will sketch the proof for this here. The proof, as in the proof of part 1, involves expressing each of these probabilities as a sum of various conditional probabilities; in the current proof, though, each such conditional probability expression $\mathbf{P}[E_1 \,|\, E_2]$ will be equated to an expression of the form $\mathbf{P}_{\mathbf{Game}_{j,i_j}}[E_1]$, and this will then be used to perform the summation.

First, focus on $\mathbf{P}[E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)]$. Notice that when $E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)$ occurs in $\mathbf{Game}_0$, u_{j-1} must be the i_jth node pointing at u_j and the reply to the query $\texttt{encrypt}(u_{j-1}, u_j)$ must be fake. Since the replies to all queries of the form $\texttt{encrypt}(u_{j'-1}, u_{j'})$ for $j' > j$ are ascertained by the vector $\overrightarrow{\nu}_j$ and since, for $s < j' < j$, every such query must be replied to with a real ciphertext, there is exactly one assignment to the variables $b_{j-1}, b_{j-2}, \cdots, b_s, b$ (where b is selected by A''s left-or-right oracle) for which $E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)$ can occur. A careful analysis of the code of A' reveals that this assigment is as follows:

$$\begin{aligned} b_{j-1} &= \nu_j \oplus \nu_{j+1} \oplus \cdots \oplus \nu_{l-1} \\ b = b_s = \cdots = b_{j-3} = b_{j-2} &= 1 \oplus \nu_j \oplus \nu_{j+1} \oplus \cdots \oplus \nu_{l-1} \end{aligned}$$

Using this observation, we can write $\mathbf{P}[E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)]$ as follows:

$$\mathbf{P}[E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)] \;=\; \sum_{l'=0}^{j-1} \mathbf{P}\left[\begin{array}{c} \Phi^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(\overrightarrow{\nu}_j, u_{j-1});\; \mathit{flen}(u_{j-1}) = l';\; \mathit{fpath}(u_{j-1}) = (u_{j-l'-1}, \cdots, u_{j-1}); \\ u_{j-l'-2} = \cdots = u_0 = 0;\; b_{j-1} = \nu_j \oplus \nu_{j+1} \oplus \cdots \oplus \nu_{l-1}; \\ b = b_{j-l'-1} = \cdots = b_{j-2} = 1 \oplus \nu_j \oplus \nu_{j+1} \oplus \cdots \oplus \nu_{l-1} \end{array}\right]$$

Now, for any $l' \in \{0, \cdots, j-1\}$ and any vector $\overrightarrow{w} = (w_{j-l'-1}, \cdots, w_{j-1}) \in [n]^{l'+1}$, let us define the following events:

$$E_{j,1}^{(l',\overrightarrow{w})} = \left(\Phi_{(i,i_l,\cdots,i_j)}^{(d,d_l,\cdots,d_j)}(\overrightarrow{\nu}_j, w_{j-1}) \;\wedge\; \mathit{flen}(w_{j-1}) = l' \;\wedge\; \mathit{fpath}(w_{j-1}) = (w_{j-l'-1}, \cdots, w_{j-1})\right)$$

$$E_{j,2}^{(l',\overrightarrow{w})} = \begin{pmatrix} u_{j-1} = w_{j-1} \;\wedge\; u_{j-2} = w_{j-2} \;\wedge\; \cdots \;\wedge\; u_{j-l'-1} = w_{j-l'-1} \;\wedge \\ u_{j-l'-2} = \cdots = u_0 = 0 \;\wedge\; b_{j-1} = \nu_j \oplus \nu_{j+1} \oplus \cdots \oplus \nu_{l-1} \;\wedge \\ b = b_{j-l'-1} = \cdots = b_{j-2} = 1 \oplus \nu_j \oplus \nu_{j+1} \oplus \cdots \oplus \nu_{l-1} \end{pmatrix}$$

$\mathbf{P}[E_{j,\mathbf{d},\mathbf{i}}^{(0)}(\overrightarrow{\nu}_j)]$ can now be written succinctly as:

$$\begin{aligned} \mathbf{P}[E_{j,\mathbf{d},\mathbf{i}}^{(0)}(\overrightarrow{\nu}_j)] &= \sum_{l'=0}^{j-1} \sum_{\overrightarrow{w} \in [n]^{l'+1}} \mathbf{P}[E_{j,1}^{(l',\overrightarrow{w})};\; E_{j,2}^{(l',\overrightarrow{w})}] \\ &= \sum_{l'=0}^{j-1} \sum_{\overrightarrow{w} \in [n]^{l'+1}} \mathbf{P}[E_{j,1}^{(l',\overrightarrow{w})} \mid E_{j,2}^{(l',\overrightarrow{w})}] \cdot \mathbf{P}[E_{j,2}^{(l',\overrightarrow{w})}] \end{aligned}$$

Let $\alpha' = \mathbf{P}[E_{j,2}^{(l',\overrightarrow{w})}]$. It can be checked easily that for any $l' \in \{0, \cdots, j-1\}$ and any $\overrightarrow{w} \in [n]^{l'+1}$,

$$\alpha' = \begin{cases} \frac{1}{2(2n+1)^j} & \text{if } j < l \\ \frac{1}{2n(2n+1)^{l-1}} & \text{if } j = l \end{cases}$$

Thus,

$$\mathbf{P}[E_{j,\mathbf{d},\mathbf{i}}^{(0)}(\overrightarrow{\nu}_j)] = \alpha' \cdot \sum_{l'=0}^{j-1} \sum_{\overrightarrow{w} \in [n]^{l'+1}} \mathbf{P}[E_{j,1}^{(l',\overrightarrow{w})} \mid E_{j,2}^{(l',\overrightarrow{w})}]$$

We claim that the probability $\mathbf{P}[E_{j,1}^{(l',\overrightarrow{w})} \mid E_{j,2}^{(l',\overrightarrow{w})}]$ is equal to the probability that $E_{j,1}^{(l',\overrightarrow{w})}$ occurs in $\mathbf{Game}_{j,i_j}$, that is, $\mathbf{P}[E_{j,1}^{(l',\overrightarrow{w})} \mid E_{j,2}^{(l',\overrightarrow{w})}] = \mathbf{P}_{\mathbf{Game}_{j,i_j}}[E_{j,1}^{(l',\overrightarrow{w})}]$. The intuition behind this is the following—given that $E_{j,2}^{(l',\overrightarrow{w})}$ occurs in $\mathbf{Game}_0$, none of the random values $r_{j-l'-1}, \cdots, r_{j-1}$ are used in any execution of A unless and until one or more conditions under $E_{j,1}^{(l',\overrightarrow{w})}$ are violated. Thus, if we consider transcripts of this interaction—given $E_{j,2}^{(l',\overrightarrow{w})}$ occurs—only upto the point where a violation of $E_{j,1}^{(l',\overrightarrow{w})}$ is found (that is, truncate the "bad" transcripts at the point where they start violating $E_{j,1}^{(l',\overrightarrow{w})}$), the transcripts would look exactly like one in the interaction of A with A' in $\mathbf{Game}_{j,i_j}$. In effect, there is a one-to-one correspondance between "non-violating" transcripts in these two interactions. The probability $\mathbf{P}[E_{j,1}^{(l',\overrightarrow{w})} \mid E_{j,2}^{(l',\overrightarrow{w})}]$ is the ratio of the *good* (that is, not truncated) transcripts to the total number of transcripts in $\mathbf{Game}_0$, which, in the setting of $\mathbf{Game}_{j,i_j}$, is the same as $\mathbf{P}_{\mathbf{Game}_{j,i_j}}[E_{j,1}^{(l',\overrightarrow{w})}]$. The details of the proof are similar to those in the proof of Claim A.4 and are omitted.

Using the above claim, we can write:

$$\begin{aligned}
\mathbf{P}[E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)] &= \alpha' \cdot \sum_{l'=0}^{j-1} \sum_{\overrightarrow{w}\in[n]^{l'+1}} \mathbf{P}_{\mathbf{Game}_{j,i_j}}[E^{(l',\overrightarrow{w})}_{j,1}] \\
&= \alpha' \cdot \sum_{l'=0}^{j-1} \sum_{\overrightarrow{w}\in[n]^{l'+1}} \mathbf{P}_{\mathbf{Game}_{j,i_j}} \left[\begin{array}{c} \Phi^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(\overrightarrow{\nu}_j, w_{j-1});\ \mathit{flen}(w_{j-1}) = l'; \\ \mathit{fpath}(w_{j-1}) = (w_{j-l'-1},\cdots,w_{j-1}) \end{array} \right] \\
&= \alpha' \cdot \sum_{w_{j-1}\in[n]} \sum_{l'=0}^{j-1} \mathbf{P}_{\mathbf{Game}_{j,i_j}}[\Phi^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(\overrightarrow{\nu}_j, w_{j-1});\ \mathit{flen}(w_{j-1}) = l'] \\
&= \alpha' \cdot \mathbf{P}_{\mathbf{Game}_{j,i_j}}[\Phi^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1})}(\overrightarrow{\nu}_j)]
\end{aligned}$$

And now using essentially the same approach one can also equate $\mathbf{P}[E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)]$ to $\alpha' \cdot \mathbf{P}_{\mathbf{Game}_{j,i_j}}[\Phi^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1})}(\overrightarrow{\nu}_j)]$. We briefly sketch here how this is done, highlighting only the parts where the proof differs from that for the case of $E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)$.

When $E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j) = \Theta^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1},i_j+1)}(0 \cdot \overrightarrow{\nu}_j)$ occurs in $\mathbf{Game}_0$, u_{j-1} is the (i_j+1)th node pointing at u_j and the reply to the query `encrypt`(u_{j-1}, u_j) is real. The replies to all queries `encrypt`$(u_{j'-1}, u_{j'})$ for $j' > j$ are ascertained by the vector $\overrightarrow{\nu}_j$, and for $s < j' < j$, every such query is replied to with a real ciphertext. This implies that there is exactly one assignment to the variables $b_{j-1}, b_{j-2}, \cdots, b_s, b$ for which $E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)$ can occur, which is as follows:

$$b = b_s = \cdots = b_{j-3} = b_{j-2} = b_{j-1} \quad = \quad \nu_j \oplus \nu_{j+1} \oplus \cdots \oplus \nu_{l-1}$$

Now, for any $l' \in \{0,\cdots,j-1\}$ and any vector $\overrightarrow{w} = (w_{j-l'-1},\cdots,w_{j-1}) \in [n]^{l'+1}$, let us define the following events:

$$\tilde{E}^{(l',\overrightarrow{w})}_{j,1} = \Big(\Phi^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1},i_j+1)}(\overrightarrow{\nu}_j, w_{j-1}) \ \wedge\ \mathit{flen}(w_{j-1}) = l' \ \wedge\ \mathit{fpath}(w_{j-1}) = (w_{j-l'-1},\cdots,w_{j-1}) \Big)$$

$$\tilde{E}^{(l',\overrightarrow{w})}_{j,2} = \left(\begin{array}{c} u_{j-1} = w_{j-1} \ \wedge\ u_{j-2} = w_{j-2} \ \wedge\ \cdots \ \wedge\ u_{j-l'-1} = w_{j-l'-1} \ \wedge \\ u_{j-l'-2} = \cdots = u_0 = 0 \ \wedge\ b = b_{j-l'-1} = \cdots = b_{j-2} = b_{j-1} = \nu_j \oplus \nu_{j+1} \oplus \cdots \oplus \nu_{l-1} \end{array} \right)$$

Notice how $\tilde{E}^{(l',\overrightarrow{w})}_{j,1}$ differs from $E^{(l',\overrightarrow{w})}_{j,1}$: we require w_{j-1} to be the (i_j+1)th node (and not the i_jth one) pointing at u_j. $\mathbf{P}[E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)]$ can be written in terms of these events as:

$$\begin{aligned}
\mathbf{P}[E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)] &= \sum_{l'=0}^{j-1} \sum_{\overrightarrow{w}\in[n]^{l'+1}} \mathbf{P}[\tilde{E}^{(l',\overrightarrow{w})}_{j,1};\ \tilde{E}^{(l',\overrightarrow{w})}_{j,2}] \\
&= \sum_{l'=0}^{j-1} \sum_{\overrightarrow{w}\in[n]^{l'+1}} \mathbf{P}[\tilde{E}^{(l',\overrightarrow{w})}_{j,1} \mid \tilde{E}^{(l',\overrightarrow{w})}_{j,2}] \cdot \mathbf{P}[\tilde{E}^{(l',\overrightarrow{w})}_{j,2}]
\end{aligned}$$

which can (using the same techniques as used in the case of $E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)$) be shown to be equal to

$$\begin{aligned}
&= \alpha' \cdot \sum_{l'=0}^{j-1} \sum_{\overrightarrow{w} \in [n]^{l'+1}} \mathbf{P}_{\mathbf{Game}_{j,i_j}}[\tilde{E}^{(l',\overrightarrow{w})}_{j,1}] \\
&= \alpha' \cdot \sum_{l'=0}^{j-1} \sum_{\overrightarrow{w} \in [n]^{l'+1}} \mathbf{P}_{\mathbf{Game}_{j,i_j}} \left[\begin{array}{c} \Phi^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1},i_j+1)}(\overrightarrow{\nu}_j, w_{j-1});\ \mathit{flen}(w_{j-1}) = l'; \\ \mathit{fpath}(w_{j-1}) = (w_{j-l'-1}, \cdots, w_{j-1}) \end{array} \right] \\
&= \alpha' \cdot \sum_{w_{j-1} \in [n]} \sum_{l'=0}^{j-1} \mathbf{P}_{\mathbf{Game}_{j,i_j}}[\Phi^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1},i_j+1)}(\overrightarrow{\nu}_j, w_{j-1});\ \mathit{flen}(w_{j-1}) = l'] \\
&= \alpha' \cdot \mathbf{P}_{\mathbf{Game}_{j,i_j}}[\Phi^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1})}(\overrightarrow{\nu}_j)] \qquad \blacksquare
\end{aligned}$$

<u>Proof of Part 3</u>. The proof of this part follows from the very definition of $\Theta^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(\overrightarrow{\nu}_{j-1})$. When $\Theta^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(\overrightarrow{\nu}_{j-1})$ occurs, the in-degree of u_{j-1} in $G(\mathsf{A})$ must either be zero or in the range of 1 through n. If the in-degree is zero, this is the same as the occurrence of $\Theta^{(d,d_l,\cdots,d_j,0)}_{(i,i_l,\cdots,i_j,0)}(1\cdot\overrightarrow{\nu}_{j-1})$. If the in-degree is non-zero (say, it is equal to d_{j-1}), then u_{j-2} must be the first node pointing at u_{j-1} in $G(\mathsf{A})$ and the event $\Theta^{(d,d_l,\cdots,d_j,d_{j-1})}_{(i,i_l,\cdots,i_j,1)}(0 \cdot \overrightarrow{\nu}_{j-1})$ must occur. This gives us the desired expression for $\Theta^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}(\overrightarrow{\nu}_{j-1})$. $\blacksquare$

<u>Proof of Part 4</u>. The proof of part (a) is relatively straightforward and follows immediately from the definition of $\Psi^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)}$ and by inspection of the code of A'. Notice that when $\Psi^{(d,d_l,\cdots,d_1)}_{(i,i_l,\cdots,i_1)}$ occurs, the event G_0 must occur (that is, $u_0, u_1, \cdots, u_l$ must all be non-zero and must form a valid path in $G(\mathsf{A})$), and since the depth of $G(\mathsf{A})$ is at most l, $\mathit{Indegree}(u_0)$ must equal 0. Also, for each $j' \in \{0, \cdots, l-2\}$, $b_{j'}$ must equal $\nu_{j'}$ for "some" $\nu_{j'} \in \{0,1\}$ and for each such assignment to the $b_{j'}$'s, the reply given for the query $\texttt{encrypt}(u_{j'-1}, u_{j'})$, when $j' > 1$, is real if and only if $\nu_{j'-1} \oplus \nu_{j'-2} = 0$; when $j' = 1$, the reply is real if and only if $\nu_0 \oplus b = 0$. Thus, the occurrence of $\Psi^{(d,d_l,\cdots,d_1)}_{(i,i_l,\cdots,i_1)}$ is equivalent to the occurrence of $\Theta^{(d,d_l,\cdots,d_1)}_{(i,i_l,\cdots,i_1)}(\mathsf{XOR}(b, \overrightarrow{\nu}_0))$ for "some" choice of $\nu_0, \nu_1, \cdots, \nu_{l-2} \in \{0,1\}$. From this, the desired result follows.

The proof of part 4(b) is similar to the proof of part 2 we already did. As in that proof, we first define a new game played between A' and A which is the same as $\mathbf{Game}_{j,i_j}$ but with two differences: *(i)* In the setup phase, A' selects b_{j-1} to be equal to ν (as opposed to $\nu_j \oplus \cdots \oplus \nu_{l-1}$ as in $\mathbf{Game}_{j,i_j}$); and *(ii)* In the execution phase, A' replies to *all* queries of the form $\texttt{encrypt}(x, u_j)$ with fake ciphertexts (as opposed to just the first i_j queries as in $\mathbf{Game}_{j,i_j}$). We denote this modified game by $\mathbf{Game}_{j,\text{all}}$ and for any event E, we denote the probability that E occurs during $\mathbf{Game}_{j,\text{all}}$ by $\mathbf{P}_{\mathbf{Game}_{j,\text{all}}}[E]$. Note that the event $\Phi^{(d,d_l,\cdots,d_{j+1},d_j)}_{(i,i_l,\cdots,i_{j+1})}(\overrightarrow{\nu}_j)$ is well-defined for the game $\mathbf{Game}_{j,\text{all}}$.

$$\begin{aligned}
\text{Let}\quad E^{(0)}_{j,\mathbf{d},\mathbf{i},\nu} &= \Psi^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)} \wedge (b = \nu)\ ; \qquad \text{and} \\
E^{(1)}_{j,\mathbf{d},\mathbf{i},\nu} &= \bigvee_{d_{j-1}=1}^{n} \left(\bigvee_{\nu_{j-1},\cdots,\nu_{l-2} \in \{0,1\}} \Theta^{(d,d_l,\cdots,d_j,d_{j-1})}_{(i,i_l,\cdots,i_j,d_{j-1})}(1 \cdot \mathsf{XOR}(\nu, \overrightarrow{\nu}_{j-1})) \right)
\end{aligned}$$

Let $\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}} := \Phi^{(d,d_l,\cdots,d_j,d_{j-1})}_{(i,i_l,\cdots,i_j)}(\mathsf{XOR}(\nu,\overrightarrow{\nu}_{j-1}))$. It is easy to check that the view of A in $\mathbf{Game}_0$ given $\Psi^{(d,d_l,\cdots,d_j)}_{(i,i_l,\cdots,i_j)} \wedge (b=\nu)$ occurs is the same as its view in $\mathbf{Game}_{j-1,\text{all}}$ given $\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}}$ occurs for some $d_{j-1} \in [n] \cup \{0\}$ and some $\overrightarrow{\nu}_{j-1} \in \{0,1\}^{l-j}$ (such that $\nu_{l-1}=0$). (This is also the same as A's view in $\mathbf{Game}_0$ given $\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}}$ occurs for some d_{j-1} and $\overrightarrow{\nu}_{j-1}$.) At the same time, given that $\Theta^{(d,d_l,\cdots,d_j,d_{j-1})}_{(i,i_l,\cdots,i_j,d_{j-1})}(1\cdot\mathsf{XOR}(\nu,\overrightarrow{\nu}_{j-1}))$ occurs in $\mathbf{Game}_0$, the view of A is identically distributed as its view in $\mathbf{Game}_{j-1,\text{all}}$ given $\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}}$ occurs. In other words, A's view in $\mathbf{Game}_0$ given $E^{(1)}_{j,\mathbf{d},\mathbf{i},\nu}$ is the same as its view in $\mathbf{Game}_{j-1,\text{all}}$ given $\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}}$ occurs for some d_{j-1} and $\overrightarrow{\nu}_{j-1} \in \{0,1\}^{l-j}$ ($\nu_{l-1}=0$). From this, it follows that $E^{(0)}_{j,\mathbf{d},\mathbf{i},\nu} \simeq E^{(1)}_{j,\mathbf{d},\mathbf{i},\nu}$.

We are left with proving $\mathbf{P}[E^{(0)}_{j,\mathbf{d},\mathbf{i},\nu}] = \mathbf{P}[E^{(1)}_{j,\mathbf{d},\mathbf{i},\nu}]$. Fix $\nu_{l-1}=0$. For any $\nu_{j-1},\cdots,\nu_{l-2}$, let $\sum_{\nu_{j-1,\ldots,l-2}}$ denote the summation $\sum_{(\nu_{j-1},\cdots,\nu_{l-2})\in\{0,1\}^{l-j}}$.

First, focus on $\mathbf{P}[E^{(0)}_{j,\mathbf{d},\mathbf{i},\nu}]$. This probability can be written in terms of the event $\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}}$, as follows:

$$\begin{aligned}\mathbf{P}[E^{(0)}_{j,\mathbf{d},\mathbf{i},\nu}] &= \sum_{d_{j-1}=0}^{n}\sum_{\nu_{j-1,\ldots,l-2}} \mathbf{P}\left[\ \Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}};\ u_0=u_1=\cdots=u_{j-2}=0;\ b=\nu\ \right]\\ &= \frac{1}{2}\left(\frac{1}{2n+1}\right)^{j-1}\sum_{d_{j-1}=0}^{n}\sum_{\nu_{j-1,\ldots,l-2}} \mathbf{P}\left[\ \Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}} \mid u_0=u_1=\cdots=u_{j-2}=0;\ b=\nu\ \right]\end{aligned}$$

The probability that event $\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}} = \Phi^{(d,d_l,\cdots,d_j,d_{j-1})}_{(i,i_l,\cdots,i_j)}(\mathsf{XOR}(\nu,\overrightarrow{\nu}_{j-1}))$ occurs in $\mathbf{Game}_0$ given $u_0,\cdots,u_{j-2}$ are all zero and $b=\nu$ is simply equal to the probability that $\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}}$ occurs in $\mathbf{Game}_{j-1,\text{all}}$. So,

$$\mathbf{P}[E^{(0)}_{j,\mathbf{d},\mathbf{i},\nu}] = \frac{1}{2}\left(\frac{1}{2n+1}\right)^{j-1}\sum_{d_{j-1}=0}^{n}\sum_{\nu_{j-1,\ldots,l-2}} \mathbf{P}_{\mathbf{Game}_{j-1,\text{all}}}\left[\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}}\right]$$

The probability of occurrence of event $E^{(1)}_{j,\mathbf{d},\mathbf{i},\nu}$ (in $\mathbf{Game}_0$) can also be shown to be equal to the above quantity. First, let us split this event based on the in-degree of node u_{j-1} in $G(\mathsf{A})$ being zero or non-zero.

$$\begin{aligned}\mathbf{P}[E^{(1)}_{j,\mathbf{d},\mathbf{i},\nu}] &= \sum_{d_{j-1}=0}^{n}\sum_{\nu_{j-1,\ldots,l-2}} \mathbf{P}[\Theta^{(d,d_l,\cdots,d_j,d_{j-1})}_{(i,i_l,\cdots,i_j,d_{j-1})}(1\cdot\mathsf{XOR}(\nu,\overrightarrow{\nu}_{j-1}))]\\ &= \underbrace{\sum_{\nu_{j-1,\ldots,l-2}} \mathbf{P}\left[\Phi^{(d,d_l,\cdots,d_j,0)}_{(i,i_l,\cdots,i_j)}(\mathsf{XOR}(\nu,\overrightarrow{\nu}_{j-1}));\ u_0=\cdots=u_{j-2}=0;\ b=\nu\right]}_{=\,p_1}\\ &\quad \underbrace{+\sum_{d_{j-1}=1}^{n}\sum_{\nu_{j-1,\ldots,l-2}}\left(\sum_{l'=0}^{j-2}\sum_{\substack{w_{j-2}\in[n],\\ \vdots\\ w_{j-l'-2}\in[n]}} \mathbf{P}\left[\begin{array}{c}\Phi^{(d,d_l,\cdots,d_j,d_{j-1})}_{(i,i_l,\cdots,i_j,d_{j-1})}(\mathsf{XOR}(\nu,\overrightarrow{\nu}_{j-1}),w_{j-2});\\ \mathit{flen}(w_{j-2})=l';\\ \mathit{fpath}(w_{j-2})=(w_{j-l'-2},\cdots,w_{j-2});\\ u_{j-2}=w_{j-2};\ \cdots;\ u_{j-l'-2}=w_{j-l'-2};\\ u_{j-l'-3}=\cdots=u_0=0;\ b_{j-2}=\nu;\\ b=b_{j-l'-2}=\cdots=b_{j-3}=1\oplus\nu\end{array}\right]\right)}_{=\,p_2}\end{aligned}$$

The first term, p_1, in the right hand side can easily be shown to be equal to

$$\frac{1}{2}\left(\frac{1}{2n+1}\right)^{j-1} \sum_{\nu_{j-1,\dots,l-2}} \mathbf{P}_{\mathbf{Game}_{j-1,\mathrm{all}}}\left[\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}}\right]$$

The second term, p_2, is harder to tackle. For any $l' \in \{0, \cdots, j-2\}$ and $\overrightarrow{w} = (w_{j-l'-2}, \cdots, w_{j-2}) \in [n]^{l'+1}$, define the following two events:

$$E^{(l',\overrightarrow{w})}_{j,\nu,1} = \left(\begin{array}{c} \Phi^{(d,d_l,\cdots,d_j,d_{j-1})}_{(i,i_l,\cdots,i_j,d_{j-1})}(\mathsf{XOR}(\nu, \overrightarrow{\nu}_{j-1}), w_{j-2}); \\ \mathit{flen}(w_{j-2}) = l';\ \mathit{fpath}(w_{j-2}) = (w_{j-l'-2}, \cdots, w_{j-2}) \end{array}\right)$$

$$E^{(l',\overrightarrow{w})}_{j,\nu,2} = \left(\begin{array}{c} u_{j-2} = w_{j-2};\ \cdots;\ u_{j-l'-2} = w_{j-l'-2};\ u_{j-l'-3} = \cdots = u_0 = 0; \\ b_{j-2} = \nu;\ b = b_{j-l'-2} = \cdots = b_{j-3} = 1 \oplus \nu \end{array}\right)$$

$\mathbf{P}[E^{(l',\overrightarrow{w})}_{j,\nu,2}]$, for any valid choice of l' and $\overrightarrow{w}$, is equal to $\frac{1}{2(2n+1)^{j-1}}$. Thus,

$$\begin{aligned} p_2 &= \sum_{d_{j-1}=1}^{n} \sum_{\nu_{j-1,\dots,l-2}} \left(\sum_{l'=0}^{j-2} \sum_{\overrightarrow{w}\in[n]^{l'+1}} \mathbf{P}\left[E^{(l',\overrightarrow{w})}_{j,\nu,1};\ E^{(l',\overrightarrow{w})}_{j,\nu,2}\right]\right) \\ &= \sum_{d_{j-1}=1}^{n} \sum_{\nu_{j-1,\dots,l-2}} \left(\sum_{l'=0}^{j-2} \sum_{\overrightarrow{w}\in[n]^{l'+1}} \mathbf{P}\left[E^{(l',\overrightarrow{w})}_{j,\nu,1} \mid E^{(l',\overrightarrow{w})}_{j,\nu,2}\right] \cdot \mathbf{P}\left[E^{(l',\overrightarrow{w})}_{j,\nu,2}\right]\right) \\ &= \frac{1}{2(2n+1)^{j-1}} \sum_{d_{j-1}=1}^{n} \sum_{\nu_{j-1,\dots,l-2}} \left(\sum_{l'=0}^{j-2} \sum_{\overrightarrow{w}\in[n]^{l'+1}} \mathbf{P}\left[E^{(l',\overrightarrow{w})}_{j,\nu,1} \mid E^{(l',\overrightarrow{w})}_{j,\nu,2}\right]\right) \end{aligned}$$

For any fixed l' and $\overrightarrow{w}$, the conditional probability $\mathbf{P}\left[E^{(l',\overrightarrow{w})}_{j,\nu,1} \mid E^{(l',\overrightarrow{w})}_{j,\nu,2}\right]$ is equal to the probability that event $E^{(l',\overrightarrow{w})}_{j,\nu,1}$ occurs in $\mathbf{Game}_{j-1,\mathrm{all}}$ (again, this involves showing a one-to-one correspondance between transcripts in $\mathbf{Game}_0$ conditioned on event $E^{(l',\overrightarrow{w})}_{j,\nu,2}$ occurring and transcripts in $\mathbf{Game}_{j-1,\mathrm{all}}$; details are omitted) and using this fact, we get that p_2 equals

$$\begin{aligned} & \frac{1}{2(2n+1)^{j-1}} \sum_{d_{j-1}=1}^{n} \sum_{\nu_{j-1,\dots,l-2}} \left(\sum_{l'=0}^{j-2} \sum_{\overrightarrow{w}\in[n]^{l'+1}} \mathbf{P}_{\mathbf{Game}_{j-1,\mathrm{all}}}\left[E^{(l',\overrightarrow{w})}_{j,\nu,1}\right]\right) \\ = & \frac{1}{2(2n+1)^{j-1}} \sum_{d_{j-1}=1}^{n} \sum_{\nu_{j-1,\dots,l-2}} \sum_{l'=0}^{j-2} \sum_{\overrightarrow{w}\in[n]^{l'+1}} \mathbf{P}_{\mathbf{Game}_{j-1,\mathrm{all}}}\left[\begin{array}{c} \Phi^{(d,d_l,\cdots,d_j,d_{j-1})}_{(i,i_l,\cdots,i_j,d_{j-1})}(\mathsf{XOR}(\nu, \overrightarrow{\nu}_{j-1}), w_{j-2}); \\ \mathit{flen}(w_{j-2}) = l'; \\ \mathit{fpath}(w_{j-2}) = (w_{j-l'-2}, \cdots, w_{j-2}) \end{array}\right] \\ = & \frac{1}{2(2n+1)^{j-1}} \sum_{d_{j-1}=1}^{n} \sum_{\nu_{j-1,\dots,l-2}} \sum_{w_{j-2}\in[n]} \mathbf{P}_{\mathbf{Game}_{j-1,\mathrm{all}}}\left[\Phi^{(d,d_l,\cdots,d_j,d_{j-1})}_{(i,i_l,\cdots,i_j,d_{j-1})}(\mathsf{XOR}(\nu, \overrightarrow{\nu}_{j-1}), w_{j-2})\right] \\ = & \frac{1}{2(2n+1)^{j-1}} \sum_{d_{j-1}=1}^{n} \sum_{\nu_{j-1,\dots,l-2}} \mathbf{P}_{\mathbf{Game}_{j-1,\mathrm{all}}}\left[\Phi^{(d,d_l,\cdots,d_j,d_{j-1})}_{(i,i_l,\cdots,i_j)}(\mathsf{XOR}(\nu, \overrightarrow{\nu}_{j-1}))\right] \\ = & \frac{1}{2(2n+1)^{j-1}} \sum_{d_{j-1}=1}^{n} \sum_{\nu_{j-1,\dots,l-2}} \mathbf{P}_{\mathbf{Game}_{j-1,\mathrm{all}}}\left[\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}}\right] \end{aligned}$$

Thus,

$$\mathbf{P}[E^{(1)}_{j,\mathbf{d},\mathbf{i},\nu}] = p_1 + p_2 = \frac{1}{2(2n+1)^{j-1}} \sum_{d_{j-1}=0}^{n} \sum_{\nu_{j-1,\ldots,l-2}} \mathbf{P}_{\mathbf{Game}_{j-1,\mathrm{all}}} \left[\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}}\right] \qquad \blacksquare$$

A.2.5 Proof of the Telescoping Sums Lemma (Lemma A.6)

We will prove the lemma using induction over j. Recall the expression for Δ_j (equation 6):

$$\Delta_j = \sum_{\substack{d\in[n],\\ i\in[d]}} \sum_{\substack{d_l\in[n],\\ i_l\in[d_l]}} \cdots \sum_{\substack{d_{j+1}\in[n],\\ i_{j+1}\in[d_{j+1}]}} \left[\begin{array}{l} \mathbf{P}[\mathsf{O}_\mathsf{A}; \Psi^{(d,d_l,\cdots,d_{j+1})}_{(i,i_l,\cdots,i_{j+1})}; b=0] \\ \qquad - \mathbf{P}[\mathsf{O}_\mathsf{A}; \Psi^{(d,d_l,\cdots,d_{j+1})}_{(i,i_l,\cdots,i_{j+1})}; b=1] \end{array} \right]$$

When $j=0$, this becomes:

$$\Delta_0 = \sum_{\substack{d\in[n],\\ i\in[d]}} \sum_{\substack{d_l\in[n],\\ i_l\in[d_l]}} \cdots \sum_{\substack{d_1\in[n],\\ i_1\in[d_1]}} \left[\begin{array}{l} \mathbf{P}[\mathsf{O}_\mathsf{A}; \Psi^{(d,d_l,\cdots,d_1)}_{(i,i_l,\cdots,i_1)}; b=0] \\ \qquad - \mathbf{P}[\mathsf{O}_\mathsf{A}; \Psi^{(d,d_l,\cdots,d_1)}_{(i,i_l,\cdots,i_1)}; b=1] \end{array} \right]$$

Using the hybrid cancellation lemma, part 4, we can re-write this as

$$\Delta_0 = \sum_{\substack{d\in[n],\\ i\in[d]}} \sum_{\substack{d_l\in[n],\\ i_l\in[d_l]}} \cdots \sum_{\substack{d_1\in[n],\\ i_1\in[d_1]}} \left(\sum_{\substack{\nu_0,\nu_1,\cdots,\nu_{l-2}\in\{0,1\},\\ \nu_{l-1}=0}} \left[\begin{array}{l} \mathbf{P}[\mathsf{O}_\mathsf{A}; \Theta^{(d,d_l,\cdots,d_1)}_{(i,i_l,\cdots,i_1)}(\mathsf{XOR}(0,\overrightarrow{\nu}_0))] \\ \\ - \mathbf{P}[\mathsf{O}_\mathsf{A}; \Theta^{(d,d_l,\cdots,d_1)}_{(i,i_l,\cdots,i_1)}(\mathsf{XOR}(1,\overrightarrow{\nu}_0))] \end{array} \right] \right)$$

From this, and the fact that $\overline{\Delta}_0 = \Delta_0$, it follows that Lemma A.6 is true for $j=0$.

Suppose that for some $\hat{j} > 0$, the lemma is true for $j = \hat{j} - 1$, that is:

$$\overline{\Delta}_{\hat{j}-1} = \sum_{\substack{d\in[n],\\ i\in[d]}} \sum_{\substack{d_l\in[n],\\ i_l\in[d_l]}} \cdots \sum_{\substack{d_{\hat{j}}\in[n],\\ i_{\hat{j}}\in[d_{\hat{j}}]}} \left[\sum_{\substack{\nu_{\hat{j}-1},\cdots,\nu_{l-2}\in\{0,1\},\\ \nu_{l-1}=0}} \left(\begin{array}{l} \mathbf{P}[\mathsf{O}_\mathsf{A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}-1}))] \\ \\ - \mathbf{P}[\mathsf{O}_\mathsf{A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}-1}))] \end{array} \right) \right]$$

We will show that the lemma is also true for $j = \hat{j}$. In the sequel, we will denote any sequence of summations of the form

$$\sum_{\substack{d\in[n],\\ i\in[d]}} \sum_{\substack{d_l\in[n],\\ i_l\in[d_l]}} \cdots \sum_{\substack{d_j\in[n],\\ i_j\in[d_j]}}$$

by $\sum_{d,i,d_l,\cdots,d_j,i_j}$ and one of the form $\sum_{\substack{\nu_j,\nu_{j+1},\cdots,\nu_{l-2}\in\{0,1\},\\ \nu_{l-1}=0}}$ by $\sum_{\nu_j^+}$. From the inductive hypothesis, we have:

$$\begin{aligned}
\overline{\Delta}_{\hat{j}-1} &= \sum_{d,i,d_l,\cdots,d_{\hat{j}},i_{\hat{j}}} \sum_{\nu^+_{\hat{j}-1}} \left[\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(\mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}-1}))] \\ \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(\mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}-1}))] \end{array} \right] \\
&= \sum_{d,i,d_l,\cdots,d_{\hat{j}},i_{\hat{j}}} \sum_{\nu^+_{\hat{j}-1}} \left[\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}((\nu_{\hat{j}-1}, \nu_{\hat{j}-1} \oplus \nu_{\hat{j}}, \cdots, \nu_{l-2} \oplus \nu_{l-1}))] \\ \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}((\overline{\nu_{\hat{j}-1}}, \nu_{\hat{j}-1} \oplus \nu_{\hat{j}}, \cdots, \nu_{l-2} \oplus \nu_{l-1}))] \end{array} \right]
\end{aligned}$$

Let us now expand the innermost sequence of summations based on the value assigned to $\nu_{\hat{j}-1}$.

$$\begin{aligned}
\overline{\Delta}_{\hat{j}-1} &= \sum_{d,i,d_l,\cdots,d_{\hat{j}},i_{\hat{j}}} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}((0, \nu_{\hat{j}}, \nu_{\hat{j}} \oplus \nu_{\hat{j}+1}, \cdots, \nu_{l-2} \oplus \nu_{l-1}))] \\ +\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}((1, \overline{\nu_{\hat{j}}}, \nu_{\hat{j}} \oplus \nu_{\hat{j}+1}, \cdots, \nu_{l-2} \oplus \nu_{l-1}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}((1, \nu_{\hat{j}}, \nu_{\hat{j}} \oplus \nu_{\hat{j}+1}, \cdots, \nu_{l-2} \oplus \nu_{l-1}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}((0, \overline{\nu_{\hat{j}}}, \nu_{\hat{j}} \oplus \nu_{\hat{j}+1}, \cdots, \nu_{l-2} \oplus \nu_{l-1}))] \end{array} \right] \\
&= \sum_{d,i,d_l,\cdots,d_{\hat{j}},i_{\hat{j}}} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(0 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ +\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(1 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(1 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(0 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right] \\
&= \sum_{d,i,d_l,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{d_{\hat{j}} \in [n]} \sum_{i_{\hat{j}} \in [d_{\hat{j}}]} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(0 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(1 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ +\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(1 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(0 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right] \\
&= \sum_{d,i,d_l,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{d_{\hat{j}} \in [n]} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \sum_{i_{\hat{j}} \in [d_{\hat{j}}]} \left(\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(0 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(1 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right) \\ \\ +\ \sum_{i_{\hat{j}} \in [d_{\hat{j}}]} \left(\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(1 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}})}(0 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right) \end{array} \right]
\end{aligned}$$

And now let us apply the hybrid cancellation lemma (part 2) to the terms in the innermost summations.

$$\overline{\Delta}_{\hat{j}-1} = \sum_{d,i,d_l,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{d_{\hat{j}}\in[n]} \sum_{\nu_{\hat{j}}^+} \left[\begin{array}{l} \left(\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},1)}(0 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right) \\ \\ +\ \left(\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},1)}(0 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right) \end{array} \right] \tag{7}$$

Now, let us recall the expression for $\Delta_{\hat{j}}$ (equation 6), and let us re-write it in terms of the Θ's by invoking part 4 of the hybrid cancellation lemma:

$$\Delta_{\hat{j}} = \sum_{d,i,d_l,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{0\leq d_{\hat{j}}\leq n} \left(\sum_{\substack{\nu_{\hat{j}},\cdots,\nu_{l-2}\in\{0,1\},\\ \nu_{l-1}=0}} \left[\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right] \right)$$

$$\begin{aligned} = & \sum_{d,i,d_l,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{d_{\hat{j}}\in[n]} \sum_{\nu_{\hat{j}}^+} \left[\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right] \\ & + \sum_{d,i,d_l,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{\nu_{\hat{j}}^+} \left[\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},0)}_{(i,i_l,\cdots,i_{\hat{j}+1},0)}(1 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},0)}_{(i,i_l,\cdots,i_{\hat{j}+1},0)}(1 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right] \end{aligned} \tag{8}$$

Let us now use the above equation and equation 7 to express $\overline{\Delta}_{\hat{j}}$ in terms of the Θ's:

$$\begin{aligned} \overline{\Delta}_{\hat{j}} &= \overline{\Delta}_{\hat{j}-1} + \Delta_{\hat{j}} \\ &= \sum_{d,i,d_l,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{d_{\hat{j}}\in[n]} \sum_{\nu_{\hat{j}}^+} \left[\begin{array}{l} \left(\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},1)}(0 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right) \\ \\ +\ \left(\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},1)}(0 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right) \\ \\ +\ \left(\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right) \end{array} \right] \\ &\quad + \sum_{d,i,d_l,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{\nu_{\hat{j}}^+} \left[\begin{array}{l} \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},0)}_{(i,i_l,\cdots,i_{\hat{j}+1},0)}(1 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O_A}; \Theta^{(d,d_l,\cdots,d_{\hat{j}+1},0)}_{(i,i_l,\cdots,i_{\hat{j}+1},0)}(1 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right] \end{aligned}$$

Notice that two pairs of terms in the first sequence of summations (enclosed in the tall square braces $\left[\cdots\right]$) cancel out, leaving us with the following:

$$\begin{aligned}
\overline{\Delta}_{\hat{j}} &= \sum_{d,i,d_l,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{d_{\hat{j}}\in[n]} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \mathbf{P}[\mathsf{O}_\mathsf{A};\Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},1)}(0\cdot\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O}_\mathsf{A};\Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},1)}(0\cdot\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}}))] \end{array}\right] \\
&\quad + \sum_{d,i,d_l,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \mathbf{P}[\mathsf{O}_\mathsf{A};\Theta^{(d,d_l,\cdots,d_{\hat{j}+1},0)}_{(i,i_l,\cdots,i_{\hat{j}+1},0)}(1\cdot\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O}_\mathsf{A};\Theta^{(d,d_l,\cdots,d_{\hat{j}+1},0)}_{(i,i_l,\cdots,i_{\hat{j}+1},0)}(1\cdot\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}}))] \end{array}\right] \\
&= \sum_{d,i,d_l,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \left(\begin{array}{l} \mathbf{P}[\mathsf{O}_\mathsf{A};\Theta^{(d,d_l,\cdots,d_{\hat{j}+1},0)}_{(i,i_l,\cdots,i_{\hat{j}+1},0)}(1\cdot\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}}))] \\ +\ \sum_{d_{\hat{j}}\in[n]}\mathbf{P}[\mathsf{O}_\mathsf{A};\Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},1)}(0\cdot\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}}))] \end{array}\right) \\ \\ -\left(\begin{array}{l} \mathbf{P}[\mathsf{O}_\mathsf{A};\Theta^{(d,d_l,\cdots,d_{\hat{j}+1},0)}_{(i,i_l,\cdots,i_{\hat{j}+1},0)}(1\cdot\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}}))] \\ +\ \sum_{d_{\hat{j}}\in[n]}\mathbf{P}[\mathsf{O}_\mathsf{A};\Theta^{(d,d_l,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_l,\cdots,i_{\hat{j}+1},1)}(0\cdot\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}}))] \end{array}\right) \end{array}\right]
\end{aligned}$$

One final invocation of the hybrid cancellation lemma (this time, part 3) gives us the desired expression for $\overline{\Delta}_{\hat{j}}$:

$$\begin{aligned}
\overline{\Delta}_{\hat{j}} &= \sum_{d,i,d_l,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \mathbf{P}[\mathsf{O}_\mathsf{A};\Theta^{(d,d_l,\cdots,d_{\hat{j}+1})}_{(i,i_l,\cdots,i_{\hat{j}+1})}(\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O}_\mathsf{A};\Theta^{(d,d_l,\cdots,d_{\hat{j}+1})}_{(i,i_l,\cdots,i_{\hat{j}+1})}(\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}}))] \end{array}\right] \\
&= \sum_{\substack{d\in[n],\\ i\in[d]}} \sum_{\substack{d_l\in[n],\\ i_l\in[d_l]}} \cdots \sum_{\substack{d_{\hat{j}+1}\in[n],\\ i_{\hat{j}+1}\in[d_{\hat{j}+1}]}} \sum_{\substack{\nu_{\hat{j}},\cdots,\nu_{l-2}\in\{0,1\},\\ \nu_{l-1}=0}} \left[\begin{array}{l} \mathbf{P}[\mathsf{O}_\mathsf{A};\Theta^{(d,d_l,\cdots,d_{\hat{j}+1})}_{(i,i_l,\cdots,i_{\hat{j}+1})}(\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\mathsf{O}_\mathsf{A};\Theta^{(d,d_l,\cdots,d_{\hat{j}+1})}_{(i,i_l,\cdots,i_{\hat{j}+1})}(\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}}))] \end{array}\right] \quad \blacksquare
\end{aligned}$$