UC San Diego
UC San Diego Electronic Theses and Dissertations

Title
Private group communication : two perspectives and a unifying solution

Permalink
https://escholarship.org/uc/item/83t4k4nb

Author
Panjwani, Saurabh Kumar

Publication Date
2007

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA, SAN DIEGO

PRIVATE GROUP COMMUNICATION: TWO PERSPECTIVES AND A UNIFYING SOLUTION

A Dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in

Computer Science

by

Saurabh Kumar Panjwani

Committee in charge:

Professor Daniele Micciancio, Chair
Professor Mihir Bellare
Professor Sam Buss
Professor Alex Snoeren
Professor Alexander Vardy

2007

Copyright

Saurabh Kumar Panjwani, 2007

All rights reserved.

The Dissertation of Saurabh Kumar Panjwani is approved, and it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

2007

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

My years at UCSD have been the most fruitful years of my life. Although I started my research in computer science on a rather negative note, and went into a long phase of depression in the very first year of graduate school, I was fortunate to find the right people and the right problems to help me recover from it pretty soon. Just when I had made up my mind to quit my PhD, I happened to meet my (then "to-be") advisor, Daniele Micciancio, and he introduced me to a problem which he felt could be of interest to me. For some reason, I immediately got hooked on to that problem and wouldn't want to leave it till I had found a solution to it that satisfied me (and Daniele). I did end up finding such a solution and, along the way, I also found an excellent advisor and plenty more interesting problems to work on.

It is perhaps an under-statement to say that this thesis wouldn't be what it is without the supervision I received from Daniele. My very motivation to do research in cryptography came from the initial interactions I had with him and if ever I felt this motivation waning later on, it would be his advice that would put me back on track. His capacity for abstract thinking always inspired me and even though my discussions with him often left me feeling like a dolt, in the big picture of things, these discsussions layed the very foundation for doing good research within me. I always knew that Daniele cared as much for my research problems as I did (sometimes, I felt he cared even more than me!) and I can say with full confidence that he belongs to that rare league of advisors who like to get "personally" involved in their students' work. *Thanks much, Daniele. There's little in this thesis that I could have done without your guidance.*

Besides Daniele, there are a number of people who have contributed towards the shaping up of this thesis, and to my overall development as a researcher, in many small, yet important, ways. I am grateful to Mihir Bellare, Sam Buss, Alex Snoeren and Alexander Vardy for having served on my thesis committee and for giving me valuable feedback during the candidacy exam as well as the final defense. Mihir and Alex, in particular, made some important remarks during my candidacy exam, which influenced the subsequent course of this thesis. I also thank Russell Impagliazzo for encouragement and mentorship throughout my stay in UCSD and especially, for giving me detailed feedback on several of my practice talks, prior to the research conferences I attended.

Much of what I learnt in graduate school was derived from interactions with my peers, including, in particular, my colleagues from the cryptography and theory labs. I was fortunate to meet some of the sharpest minds in computer science at my university and to engage in insightful discussions with them on a variety of topics, ranging from the mating patterns of Californian ravens to the complexity of concurrent black-box zero-knowledge proof systems. I extend my gratitude to Michel Abdalla, Alexandra Boldyreva (Sasha), Tadayoshi Kohno (Yoshi), Alejandro Hevia, Ragesh Jaiswal, Eike Kiltz, Kirill Levchenko, Vadim Lyubashevsky, Jia Mao, Anton Mityagin, Jean Monnerat, Gregory Neven, Adriana Palacio, Barath Raghavan, Thomas Ristenpart, Todor Ristov, Sarah Shoup, Bogdan Warinschi and Scott Yilek for being the wonderful colleagues that they are and for making graduate school the enjoyable experience that it (normally) isn't. Special thanks to Jean (for reading and commenting on the introduction of this thesis), to Bogdan (for discussing with me the details of his work on computational soundness, and also for feedback on some of my own results, in particular, the results included in Chapter 8 of this thesis), and to Kirill (for his enthusiastic involvement in all my presentations I gave during the theory seminar, and for the "peace" lunches he organized to increase collaboration between the theory and crypto labs). Special thanks also to Eike, Anton, and Barath for being such great collaborators in the research projects we worked on together; I daresay that those projects were the most fun-packed projects I did during my entire stay at UCSD.

I thank my parents for their love, support and encouragement and for giving me the freedom to follow the career path of my liking even though it conflicted with their own ideas and preferences in various ways. I thank my younger brother, Pratyush, for his affection and adoration, and for fulfilling many of the dreams that I had seen for myself (but could never pursue sincerely because of my strong inclination towards academics).

Finally, I would like to thank my friends at Udai, a voluntary student organization I was a member of, for the engaging discussions on social development I had with them, and for making graduate school such a “meaningful” experience for me. Although my activities at Udai had very little to do with the research I did in computer science, they did influence my critical thinking in several ways and improved my capacity to reason rationally and with a human touch. It is through these activities that I have discovered a new direction in life, which I shall now set about pursuing.

Chapters 4 and 8, in part, are reprints of the material as it appears in 33rd Internation Colloquium on Automata, Languages and Programming (ICALP), July 2006, Micciancio, Daniele; Panjwani, Saurabh. The dissertation author was the primary investigator and author of this paper.

Chapter 6, in part, is a reprint of the material to be published in IEEE/ACM Transactions on Networking, October 2008, Micciancio, Daniele; Panjwani, Saurabh. The dissertation author is the primary investigator and author of this paper.

VITA

2007	Doctor of Philosophy in Computer Science University of California, San Diego San Diego, CA, USA
2002	Bachelor of Technology Indian Institute of Technology (IIT), Bombay, Mumbai, India

PUBLICATIONS

D. Micciancio and S. Panjwani. Optimal communication complexity of generic multicast key distribution. *IEEE/ACM Transactions in Networking*, 2008. To appear.

B. Raghavan, S. Panjwani, and A. Mityagin. Analysis of the spv secure routing protocol: weaknesses and lessons. *ACM SIGCOMM Computer Communication Review*, 37(2):29–38, 2007.

S. Panjwani. Tackling adaptive corruptions in multicast encryption protocols. In S. Vadhan, editor, *Theory of Cryptography Conference, TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 21–40. Springer-Verlag, Berlin, Germany, February 2007.

D. Micciancio and S. 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, July 2006.

E. Kiltz, A. Mityagin, S. Panjwani, and B. Raghavan. Append-only signatures. In A. de Santis, editor, *Automata, Languages and Programming, 32nd International Colloquium, ICALP Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 434–445, Lisboa, Portugal, July 2005. Springer-Verlag, Berlin, Germany.

D. Micciancio and S. 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.

D. Micciancio and S. Panjwani. Optimal communication complexity of generic multicast key distribution. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 153–170, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.

FIELDS OF STUDY

Major Field: Computer Science

Studies in Cryptography
Professor Daniele Micciancio

ABSTRACT OF THE DISSERTATION

PRIVATE GROUP COMMUNICATION: TWO PERSPECTIVES AND A UNIFYING SOLUTION

by

Saurabh Kumar Panjwani
Doctor of Philosophy in Computer Science
University of California, San Diego, 2007
Professor Daniele Micciancio, Chair

Private communication in groups of users is a security problem that is relevant in a host of real-world applications like pay-per-view, secure online conferencing and the protection of content on digital media. Despite a long history of research on the topic, there exists a fundamental dichotomy in the literature in the manner in which the problem is modeled and security analysis of protocols conducted. Most of the existing protocols for the problem have been analyzed using a simple “symbolic” model of computation, one in which cryptographic primitives are treated as ideal objects and adversarial behavior defined using a fixed set of inference rules. Some others have been approached using a more detailed computational model, wherein security is based on complexity assumptions (like the existence of one-way functions) and proven using very careful probabilistic analysis. The two approaches have their individual merits but are in conflict with each other: while the method of the computational approach is more realistic, security proofs in the symbolic approach are far easier to produce and to verify.

This thesis reconciles the two approaches and proposes a methodology for analyzing protocols that is symbolic in nature, and still powerful enough to guarantee security against arbitrary computationally-bounded entities. We define a large class of protocols for private group communication, and provide syntactic conditions on protocols in this class such that any protocol that satisfies these conditions, and meets the

symbolic definition of security, is guaranteed to be secure in the computational model as well. Such an implication enables us to conduct security analysis of protocols (falling within our class) symbolically, and simultaneously reap the benefits of computational security analysis. As an illustration, we apply our methodology to the security analysis of four existing protocols for group privacy, two of which were not known to be computationally secure prior to our work.

An important contribution of this thesis is a new technique to prove security of group privacy protocols in the presence of computational adversaries who can corrupt protocol participants in an *adaptive* manner. We show that if one suitably restricts the length of encryption chains (sequences of ciphertexts of the form $\mathbf{E}_{K_1}(K_2)$, $\mathbf{E}_{K_2}(K_3), \mathbf{E}_{K_3}(K_4), \ldots$) generated in protocol executions, then security of a protocol against adaptive corruptions follows almost immediately from its security in the symbolic model. Prior to our work, all techniques to proving adaptive security of privacy protocols involved either restricting the security model severely (for example, by requiring all users to be stateless) or using non-standard, and inefficient, ways to implement the encryption operation.

Chapter 1

Introduction

Consider a situation in which a set of users $\mathcal{S}$ wish to communicate as a group over a public broadcast channel. The number of potential users of the channel is quite large and users in $\mathcal{S}$ would like to ensure that no user outside this set is able to decipher the information being communicated. The set $\mathcal{S}$ changes with time: users can leave and/or join it at different instants and privacy of group information needs to be maintained even when such changes take place arbitrarily, and in an a-priori-unknown manner. For example, if Alice is in $\mathcal{S}$ at time t_1 but not in it at time t_2 (for any $t_2 \neq t_1$), then she should be able to recover information sent at t_1 but should learn nothing about what is sent at t_2. If Bob is *not* in $\mathcal{S}$ at time t_1 but *is* so at time t_2, the converse should hold for him. Can we design cryptographic mechanisms that achieve such an objective, and do so in an *efficient* and *provably secure* manner?

The above problem often arises when dealing with protocols involving multi-user communication and is relevant in a lot of real-world scenarios. One such scenario is that of pay-per-view services. Providers of pay-per-view services distribute content to a dynamic set of clients using a broadcast medium (like a satellite link), and their primary concern is to ensure that at every point in time, all and only the receivers who subscribe to their service *at that instant* can recover the information being broadcast. Another scenario where the problem is applicable is that of DVD content protection. Content stored and distributed on DVDs must be encoded in a manner such that only certain "compliant" devices are able to decode it. Over time, the decoding rights of some of the compliant devices may be revoked (perhaps due to their involvement in piracy), and suitable mechanisms must be designed to preserve privacy even after such revocations occur. A third application scenario is that of securing multicast communication over the Internet. Today, quite a few Internet-based applications (like video conferences, online games) are implemented using IP multicast [17] and privacy is an important security concern in some of these applications (for example, applications run over enterprise networks). Since Internet users can join and leave multicast groups at will, privacy in multicast must be achievable even in the face of arbitrary group membership changes.

1.1 Privacy via Group Keys

In point-to-point communication, privacy can easily be ensured using conventional, symmetric-key cryptographic techniques: If Alice wishes to communicate privately with Bob, she first shares a cryptographic key K with him and then "encrypts" every message that she wishes to send using an encryption function $\mathbf{E}_K(\cdot)$. That is, for any message M, instead of sending M to Bob, she sends a transformed version of it, $\mathbf{E}_K(M)$, referred to as the *encryption of* M *under* K. The transformed message has the desirable property that only a user who knows K (in this case, Bob) is capable of recovering M from it.

A similar approach could be used for achieving privacy in multi-user communication as well: all the communicating users could share a common cryptographic key and encrypt every message exchanged within the group under that key. If the group is dynamic, this key must also be updated with time and distributed in a manner such that at each instant, all and only the current group members are able to recover it. In order to solve the problem of privacy in groups, it thus suffices to solve the problem of securely distributing group keys over a broadcast channel.

Assuming the presence of a central trusted authority, one way in which the key distribution problem can be solved is as follows: have the central authority share a unique long-lived key with every user of the channel, and at each instant, generate a fresh value for the group key and transmit it encrypted under the long-lived keys of the current group members. So, if K_i denotes the long-lived key of user i, and $\mathcal{S}^{(t)}$ denotes the group at time t, then at every instant t, the center generates a fresh key $K^{(t)}$, and transmits the encrypted message $\mathbf{E}_{K_i}(K^{(t)})$ for each $i \in \mathcal{S}^{(t)}$. $K^{(t)}$ thus becomes the group key for time t, and can be used to guarantee privacy of all information exchanged at that instant.

The problem with the above protocol is that it incurs a communication overhead that is linear in the size of the group: even for a single addition to or deletion from a group of size n, the number of messages that need to be transmitted to distribute the group key securely is $O(n)$. This does not scale well with the size of the group and a better solution is clearly desirable. Much research has gone into designing group key distribution protocols with sub-linear communication complexity and currently, the best known protocol for the problem has communication complexity that is logarithmic in the number of group members [10].

1.2 A Tale of Two Models

Despite the progress made in the direction of designing efficient protocols for group key distribution, there exists a fundamental dichotomy in the literature in the way security analysis of these protocols is conducted. Most of the existing work on group key distribution adopts a "symbolic" approach to model the problem and to describe and analyze solutions to it. Under this approach, all information generated and used by a protocol is modeled using abstract data types and security primitives (like encryption functions) are treated as symbolic operations performed on such data types. Every protocol message is thus an expression formed by applying one or more symbolic operations on basic data elements like keys and constants. It is assumed that all primitives behave ideally and that adversarial entities know nothing about their underlying implementation. So, for example, if an adversary acquires the encryption of a message M

under a key K and does not know K, one deduces that no information, whatsoever, is leaked about M by such an acquisition.

Such a model leads to simple, tractable proofs of security and typically, one can prove security of protocols within the model using straightforward inductive arguments. However, the model itself is quite unrealistic in that it makes extreme assumptions on the behavior of adversaries and on the security features offered by cryptographic primitives. In real implementations, primitives cannot be expected to behave ideally nor can adversaries be guaranteed to abide by the symbolic rules of information recovery. As such, the conclusions one derives from symbolic security arguments alone are of limited practical benefit.

The computational model addresses this limitation and provides a more rigorous framework for conducting security analysis of protocols. In this model, adversaries are modeled as computational entities that are completely unconstrained in the manner in which they can attack protocols as long as they can do so “efficiently” (for example, in time that is polynomial in the size of cryptographic keys). Primitives are not assumed to be ideal objects; rather, their security is defined with respect to computationally-bounded attacks only. Proving computational security of a protocol involves arguing, carefully, that an attack on the protocol by *any* computational adversary must imply an attack on at least one of the primitives used to construct it. A fair number (though still a minority) of group key distribution protocols in the literature have been analyzed using such an approach.

Even with its obvious benefits over the symbolic model, the computational model has its share of shortcomings. Proofs of security in the model are complex, hard to write down and hard to verify. They contain intricate arguments involving probabilities and complexity assumptions, and these arguments are too prone to human errors. The situation is particularly grim for multi-user protocols (like the ones we are considering), for which an arbitrary subset of users can be corrupted during protocol execution, possibly in an adversarial manner, and changes in the execution flow of the protocol (for example, membership dynamics in the case of group key distribution) can also be made adversarially. Analyzing protocols in the presence of computational entities who perform such manipulations can, expectedly, be quite cumbersome. This probably explains why most of the existing protocols for group key distribution have not been subjected to computational analysis at all, and even those that have been so are supported with security proofs that are either too opaque, or incomplete or, in some situations, even incorrect. (See [31] for a survey.)

1.3 The Central Question

The existence of two, seemingly-conflicting approaches to addressing the same problem in the literature is a rather unsettling state of affairs. Ideally, one would like to be able to prove security of all known protocols for group key distribution in the computational model or else modify them suitably so they are rendered computationally secure. In the current situation, it is not even clear if the majority of protocols, that are known only to be symbolically secure, *can* be proven secure in the computational model at all. Analyzing each of these protocols (at least seven are known [31]) against computational attacks from scratch is a tremendous task for any security analyst.

Is there an easy way out of this situation? Can we somehow guarantee security of protocols in the computational sense, while still keeping proofs simple and elegant, as they are in the symbolic model? It would be ideal if we could show that symbolic and computational notions of security are closely related and that one follows from the other always (that is, for every protocol). But this would be too much to ask for. Given the idealistic treatment of security in the symbolic model, it is natural to expect that there be protocols that are symbolically secure but are still insecure against computational attackers. (Indeed, we will see some examples of these later on.)

In order to establish connections between symbolic and computational security notions, we must thus impose some restrictions on the protocols first. We ask:

Is it possible to find reasonable conditions on group key distribution protocols, such that for any protocol satisfying these conditions, security in the symbolic model also *implies security against computationally-bounded adversaries?*

Here, by "reasonable", we mean that the conditions we obtain be simple (that is, it should be easy to check if a protocol satisfies them or not) and not overly restrictive (existing protocols for the problem should either satisfy them or should be modifiable to do so easily).

1.4 Contributions of This Thesis

In this thesis, we provide an affirmative answer to the question raised above. We show that for a fairly general class of group key distribution protocols, it is possible to argue about security in purely symbolic terms and simultaneously obtain strong guarantees on computational security of the protocol. We focus on protocols that rely on symmetric-key cryptographic techniques only and in particular, those that make arbitrary black-box usage of symmetric-key encryption schemes and pseudo-random generators. (Most protocols for the problem in the literature are symmetric-key in nature, primarily for efficiency reasons, and almost all of these rely on encryption and pseudo-random generation only.) For this class of protocols, we introduce simple syntactic conditions such that any protocol satisfying the conditions and also satisfying the symbolic notion of security for group key distribution, is provably secure against computational adversaries for *any* computationally-secure implementation of the primitives.

The syntactic conditions we introduce are fairly easy to verify and some of these are, in fact, even necessary to prove security of protocols in the computational model. The conditions fall into two categories. Those in the first category restrict the *order* in which messages are transmitted in protocols and more specifically, the order in which keys are used in protocol messages. Intuitively, these conditions require that every key be used in a protocol in two distinct phases: a *distribution* phase, in which it is sent encrypted under one or more keys (or possibly transmitted unencrypted), followed by a *deployment* phase, in which it is used to encrypt other keys or plaintexts. Key distribution is not allowed to succeed key deployment. Most known protocols for group key distribution indeed comply with this requirement, that is, they distribute keys only prior to their being deployed for any cryptographic purposes.

One disadvantage of the ordering constraint is that it does not permit users' internal states to be revealed in the midst of a protocol since the state of any user could contain keys that are used for encryption. As a result, under this constraint, one cannot prove computational security of protocols against adversaries who corrupt users *during* protocol execution, possibly in an adaptive manner (that is, based on the protocol history). Proving computational security of multi-user encryption protocols against adaptive adversaries, in general, is known to be an extremely challenging problem in cryptography. Currently, the only known approaches to solving the problem involve using either non-standard models (for example, the erasure model [3]) for doing security proofs or non-standard primitives (for example, "non-committing" encryption schemes [9]) to implement the encryption operation. The first approach has the drawback that it makes unrealistic assumptions on the behavior of honest users while the second one suffers from being too inefficient to implement in practice.

An important contribution of this thesis is a new technique to argue about adaptive security of encryption protocols (and of group key distribution protocols, in particular) which neither restricts the security model in any way nor requires the use of encryption schemes with special properties. We demonstrate that the "amount" of adaptive security that can be provably achieved in any encryption protocol is related to the length of the longest encryption chain of the form $\mathbf{E}_{K_1}(K_2), \mathbf{E}_{K_2}(K_3), \mathbf{E}_{K_3}(K_4), \ldots,$ (**E** being a symmetric-key encryption operation) created by it and the shorter the encryption chains generated in any execution, the better is the guarantee on the adaptive security of the protocol. In particular, we show that if the longest encryption chain created by a protocol has length at most logarithmic in the number of protocol users, and, if the protocol is secure in the symbolic model, then security against adaptively-corrupting computational adversaries follows automatically[1]. Thus, to prove adaptive security of any encryption protocol that is known to be symbolically secure, it suffices to check that

[1]We remark that the above restriction on encryption chains allows to prove security via a *quasi*-polynomial reduction, as opposed to a fully polynomial one. See Chapter 9 for details.

the protocol does not create messages which could lead to the formation of arbitrarily long encryption chains.

ANALYSIS OF PROTOCOLS. We apply our results to the security analysis of two important types of group key distribution protocols—*logical key hierarchy (*LKH*) protocols* [46, 45] and *subset cover protocols* [34]. Most known protocols for group key distribution belong to one of these two types. The LKH protocols were designed for the problem of securing multicast communication over the Internet and they have exponentially better communication efficiency than the trivial protocol we described earlier on. The subset cover protocols are relatively less efficient but have the advantage of being stateless: users need to maintain a fixed, small set of keys as their internal state and their state does not change with time. This property makes subset cover protocols applicable in several scenarios besides that of multicast communication (for example, in the scenario of DVD content protection).

We analyze two protocols belonging to each of these classes [46, 45, 10, 34] (four protocols in all). In the process of analyzing these protocols, we uncover a weakness in the LKH protocols of [46, 45, 10] and show that both these protocols are insecure in the computational model, even against non-adaptive adversaries. We then fix these protocols in a manner such that they become compliant with our syntactic conditions and then use our results to establish computational security for each of them. (For the protocol of [10], computational security is proven against non-adaptive adversaries while for [46, 45], adaptive security is also proven.) For the two subset cover protocols we study, computational security was already proven in [34] but our techniques provide a much simpler proof of security for both the protocols.

A LOWER BOUND. Besides conducting security analysis of protocols, we also investigate the issue of designing group key distribution protocols that are more efficient than those already known to exist. In this context, our results are negative. We prove that in any secure (symmetric-key) group key distribution protocol, the minimum number of messages (each message being the encryption of a key under another key) required to be transmitted at each instant, on the average, is at least logarithmic in the number of group members. This lower bound on communication complexity matches the upper bound of [10]—the most communication-efficient protocol known—up to *sub-constant* additive terms and is, thus, extremely tight. Not only this, we show that the same bound also applies to protocols that use secret sharing schemes [42] besides encryption and pseudo-random generation. Thus, even the use of secret sharing cannot help in reducing the communication complexity of known protocols by any reasonable measure.

Our lower bound on group key distribution holds for protocols that are secure against *collusions* of malicious users; that is, it is proven with respect to a security definition in which the adversary can corrupt multiple users (possibly adaptively), put together their long-lived keys and use such information to circumvent the protocol. Indeed, such a bound cannot be proven for protocols that are secure only against solitary malicious users (as is illustrated by known protocols that satisfy this weaker definition and achieve constant communication complexity). Collusion-resistance is considered an essential security criterion for multi-user protocols and almost all group key distribution protocols in the literature have been designed with this objective in mind. We prove our lower bound with respect to the symbolic definition of collusion-resistance but since this definition is weaker than its computational counterparts, the same bound easily applies to computationally-secure protocols as well.

1.5 Thesis Organization

The rest of this thesis is divided into two parts. In the first part, we consider the group key distribution problem purely in symbolic terms. We first define a symbolic model for arbitrary symmetric-key encryption protocols, develop various definitions of security for group key distribution within this model, and then analyze the LKH protocols and the subset cover protocols of [46, 45, 10, 34] with respect to these definitions.

We also prove our lower bound on communication complexity in this part of the thesis.

The second part of the thesis contains our most important contributions. In this part, we provide computational definitions of security for group key distribution protocols and relate these definitions to the symbolic definitions developed earlier on using two very general technical results. The first result provides the syntactic conditions under which symbolic security implies computational security with respect to non-adaptive adversaries, while the second one does the same for computational security against adaptive adversaries. Once the relations between symbolic and computational definitions are established, we extend our security results from the first part of the thesis to the computational setting and prove computational security of all the protocols studied therein.

Part I

The Symbolic Model

In this part of the thesis, we develop a symbolic model for studying encryption protocols and, subsequently, use this model to define security notions for the specific task of group key distribution. Our model is symbolic in the sense that it treats all keys and messages generated by a protocol as abstract data types and cryptographic primitives as abstract functions over such data types. How these data types and functions are implemented is left unspecified, and even adversarial entities are assumed to know nothing about their implementation. Such an abstraction leads to simple, tractable proofs of security and as we illustrate, protocols can be proven secure within the model using straightforward inductive arguments.

A symbolic model like ours was first proposed in [18] for analyzing security of encryption protocols against impersonation attacks and is often referred to as the *Dolev-Yao security model* after the authors of [18]. Our model differs from the Dolev-Yao model in two ways: on the one hand, we generalize the original model to incorporate the use of pseudo-random generators, which are frequently used in conjunction with encryption schemes in the design of security protocols; on the other, we consider a weaker attack model, one in which the adversary only eavesdrops on the communication of all parties but may not have the power to disrupt or modify any such communication.

The symbolic model provides a convenient framework to analyze security of protocols but given its idealistic nature, the conclusions one derives from such security analysis are of limited practical benefit. This is an issue we discuss in detail, and suitably address, in the next part of the thesis.

Chapter 2

The Model

Let $\mathcal{R} = \{R_1, R_2, \ldots,\}$ be an arbitrary, infinite alphabet. We refer to the symbols in this alphabet as *purely random keys*[1] or *fresh keys*; these can be used directly for performing cryptographic operations like encryption. Oftentimes, though, protocols also make use of keys that are not purely random but "pseudo" random in the sense that no efficient observer can distinguish them from purely random values. Such keys can be obtained by using a *pseudo-random generator (PRG)*, a cryptographic primitive that takes a purely random key as input and "expands" it into a sequence of pseudo-random ones.

2.1 Pseudo-random Generators

In symbolic terminology, a pseudo-random generator is a function $\mathbf{G}$ that maps a key K to a sequence of ℓ seemingly random keys $(\mathbf{G}_0(K), \cdots, \mathbf{G}_{\ell-1}(K))$ for some $\ell > 1$. We refer to ℓ as the *expansion factor* of $\mathbf{G}$[2]. The keys that are output by a pseudo-random generator, when given a random (or pseudo-random) key K as input, are considered as good as purely random ones; in particular, encryption schemes remain secure when such keys are deployed in them. Pseudo-random generators have

[1] In the symbolic model, random keys are just constant expressions without any probability distribution associated with them. The word "random" refers only to the intended implementation of such expressions.

[2] The notion of expansion factor is different from that of the *stretch* of a PRG [7], more commonly used in the literature; the latter is defined for a setting in which the PRG is modeled as a map over binary strings.

been extensively used in the design of encryption protocols, in general, (and of group key distribution protocols, in particular), typically to improve protocol efficiency, and sometimes even to add new functionality; as such, incorporating them in our model is paramount.

For the rest of this thesis, we consider a fixed pseudo-random generator $\mathbf{G}$ with expansion factor equal to 2—on input a key K, such a generator outputs two keys, which we denote by $\mathbf{G}_0(K)$ and $\mathbf{G}_1(K)$. This is without loss of generality because PRGs with larger expansion factors can be quite easily built from such generators using standard techniques in the literature [49, 7]. Indeed, all protocols we are interested in use PRGs with expansion factor two only.

2.2 Encryption

Besides pseudo-random generators, the other primitive we are concerned with is *encryption* (more precisely, *symmetric-key* encryption). In the symbolic world, an encryption scheme is modeled as a function $\mathbf{E}$ defined over symbolic expressions: it takes, as input, an expression M, called the *message*, and a key K, and maps these to another expression $\mathbf{E}_K(M)$, called the *ciphertext* corresponding to M. Intuitively, the ciphertext hides the message completely and the latter can be recovered only if the encryption key K is known; that is, given $\mathbf{E}_K(M)$, M is recoverable if and only if K is recoverable.

We allow encryption to be performed in a "nested" fashion: ciphertexts can themselves act as messages and every message can be encrypted iteratively under multiple keys. For example, $\mathbf{E}_{R_1}(\mathbf{E}_{R_2}(R_3))$ is a valid ciphertext in our model, and so is $\mathbf{E}_{R_1}(\mathbf{E}_{R_1}(\mathbf{E}_{R_1}(R_2)))$. Nested encryption has, in fact, been deployed in security protocols in the past, sometimes with the purpose of enhancing their security [28] and sometimes even to better the efficiency of known schemes [14, 21].

2.3 Protocol Messages

Messages, in our model, are expressions created by applying the functions $\mathbf{E}$ and $\mathbf{G}$ iteratively on elements of $\mathcal{R}$. Formally, we define a *protocol message* as an expression that is derivable from the variable M in the following context-free grammar:

$$\begin{aligned} M &\rightarrow K \mid \mathbf{E}_K(M) \\ K &\rightarrow \mathcal{R} \mid \mathbf{G}_0(K) \mid \mathbf{G}_1(K) \end{aligned} \tag{2.1}$$

Here, the variable K models keys generated during the protocol, which can either be purely random (contained in the set $\mathcal{R}$) or pseudo-random (derived from purely random keys using $\mathbf{G}$). Note that a pseudo-random key K can be obtained via multiple, iterative applications of $\mathbf{G}$ on the same purely random key; the latter is referred to as the *root* of K and the number of iterations of $\mathbf{G}$ required to generate K from its root is called the *depth* of K. We denote these by $\mathsf{root}(K)$ and $\mathsf{depth}(K)$ respectively. As an example, if $K = \mathbf{G}_0(\mathbf{G}_1(R_1))$, then $\mathsf{root}(K) = R_1$ and $\mathsf{depth}(K) = 2$.

Let $\mathbf{Msgs}$ denote the set of all messages that can be derived from grammar (2.1) and $\mathbf{Keys}$ the set of all keys derivable from it. Note that both these sets are infinite and that $\mathbf{Keys} \subset \mathbf{Msgs}$. (That is, every key derivable from our grammar is also a message.) Some examples of expressions contained in $\mathbf{Msgs}$ are:

- $\mathbf{G}_0(\mathbf{G}_1(R_1))$, the pseudo-random key obtained by first applying $\mathbf{G}_1$ on the purely random key R_1, and then applying $\mathbf{G}_0$ on the resulting key.
- $\mathbf{E}_{R_2}(\mathbf{G}_0(\mathbf{G}_1(R_1)))$, the ciphertext corresponding to the encryption of $\mathbf{G}_0(\mathbf{G}_1(R_1))$ under a purely random key R_2;
- $\mathbf{E}_{\mathbf{G}_1(R_2)}(\mathbf{E}_{\mathbf{G}_0(R_1)}(R_3))$, the ciphertext corresponding to the "double" encryption of a key R_3 under the keys $\mathbf{G}_0(R_1)$ and $\mathbf{G}_1(R_2)$.

In practice, protocols can also generate ciphertexts formed by encrypting arbitrary data, and not necessarily keys; we use the above model because our interest is primarily in “key” distribution protocols, where keys are the only objects to be encrypted. It is possible to extend many of the results of this thesis to protocols that encrypt arbitrary data as well and in the sequel, we discuss how this can be done.

One important remark regarding our grammar is the following. Although we allow ciphertexts to be created by iterative applications of the encryption function and the PRG in a fairly general manner, we do not allow them to be used as keys, either for encrypting messages or as seeds to the pseudo-random generator. (That is, we do not incorporate rules of the form $M \rightarrow \mathbf{E}_M(M)$ or $K \rightarrow \mathbf{G}_0(M) \mid \mathbf{G}_1(M)$.) This is because, in practice, ciphertexts need not possess the pseudo-randomness properties that keys do (or rather, are assumed to do); indeed, using them to play the role of keys is considered injudicious cryptographic practice and could lead to the design of insecure protocols *even for secure implementations of* $\mathbf{G}$ *and* $\mathbf{E}$.

2.4 An Entailment Relation

Given a set of protocol messages $\mathcal{M} \subset \mathbf{Msgs}$, what information about the protocol can be recovered from this set? We formalize this notion using an entailment relation $\mathcal{M} \vdash M$ (symbolizing the assertion “M is recoverable from $\mathcal{M}$”), and define it recursively using the following rules:

$$M \in \mathcal{M} \implies \mathcal{M} \vdash M \qquad \text{(Rule 0)}$$

$$\mathcal{M} \vdash K \implies \mathcal{M} \vdash \mathbf{G}_0(K) \ \wedge \ \mathcal{M} \vdash \mathbf{G}_1(K) \qquad \text{(Rule 1)}$$

$$\mathcal{M} \vdash \mathbf{E}_K(M) \ \wedge \ \mathcal{M} \vdash K \implies \mathcal{M} \vdash M \qquad \text{(Rule 2)}$$

Rule 0 is trivial: every message that is contained in $\mathcal{M}$ is clearly recoverable from it. Rule 1 is the correctness condition associated with the pseudo-random generator—if one can recover a key K, then one can also recover all pseudo-random keys derived from K. The last rule, Rule 2, formalizes the notion of decryption associated with any encryption function—if one possesses a ciphertext $\mathbf{E}_K(M)$ and also the key K used to create that ciphertext, then one can "open" the ciphertext to recover M.[3]

We say that a message M is recoverable from $\mathcal{M}$ *in i steps* (for some $i \geq 0$) if M can be derived from $\mathcal{M}$ using i applications of the non-trivial rules, Rule 1 and Rule 2. (If $M \in \mathcal{M}$, then it is said to be recoverable from $\mathcal{M}$ in 0 steps.) For example, consider the set

$$\mathcal{M} = \{R_2, \mathbf{E}_{\mathbf{G}_1(R_2)}(\mathbf{E}_{\mathbf{G}_0(R_1)}(R_3)), \mathbf{E}_{R_2}(\mathbf{G}_1(R_1)), \mathbf{E}_{\mathbf{G}_0(\mathbf{G}_1(R_1))}(R_4)\}$$

From the trivial rule, we know that

$$\begin{aligned} & \mathcal{M} \vdash R_2, \\ & \mathcal{M} \vdash \mathbf{E}_{\mathbf{G}_1(R_2)}(\mathbf{E}_{\mathbf{G}_0(R_1)}(R_3)), \\ & \mathcal{M} \vdash \mathbf{E}_{R_2}(\mathbf{G}_1(R_1)), \\ \text{and,} \quad & \mathcal{M} \vdash \mathbf{E}_{\mathbf{G}_0(\mathbf{G}_1(R_1))}(R_4) \end{aligned}$$

The key R_4 can now be recovered from $\mathcal{M}$ in 3 steps:

$$\mathcal{M} \vdash R_2 \wedge \mathcal{M} \vdash \mathbf{E}_{R_2}(\mathbf{G}_1(R_1)) \implies \mathcal{M} \vdash \mathbf{G}_1(R_1) \qquad \text{(Using Rule 2)}$$

$$\mathcal{M} \vdash \mathbf{G}_1(R_1) \implies \mathcal{M} \vdash \mathbf{G}_0(\mathbf{G}_1(R_1)) \qquad \text{(Using Rule 1)}$$

$$\mathcal{M} \vdash \mathbf{G}_0(\mathbf{G}_1(R_1)) \wedge \mathcal{M} \vdash \mathbf{E}_{\mathbf{G}_0(\mathbf{G}_1(R_1))}(R_4) \implies \mathcal{M} \vdash R_4 \qquad \text{(Using Rule 2)}$$

while the ciphertext $\mathbf{E}_{\mathbf{G}_0(R_1)}(R_3)$ can be recovered in 2 steps:

$$\mathcal{M} \vdash R_2 \implies \mathcal{M} \vdash \mathbf{G}_1(R_2) \qquad \text{(Using Rule 1)}$$

$$\mathcal{M} \vdash \mathbf{G}_1(R_2) \wedge \mathcal{M} \vdash \mathbf{E}_{\mathbf{G}_1(R_2)}(\mathbf{E}_{\mathbf{G}_0(R_1)}(R_3)) \implies \mathcal{M} \vdash \mathbf{E}_{\mathbf{G}_0(R_1)}(R_3) \qquad \text{(Using Rule 2)}$$

For any message-set $\mathcal{M}$, we use $\mathbf{Rec}(\mathcal{M})$ to denote the set of *all* messages that are recoverable from it (irrespective of the number of steps required to do so); that is,

[3] For more generality, one could also incorporate a rule for "encryption", formalizing the idea that given a key K, and a message M, it is easy to construct the ciphertext $\mathbf{E}_K(M)$. However, such a rule would be of no benefit to us. For one, ciphertexts themselves cannot be used to recover anything other than the message that they encrypt. For two, we analyze protocols in terms of what *keys* (as opposed to ciphertexts) can be recovered from protocol messages; for this purpose, the rules that we have listed above suffice.

$\mathbf{Rec}(\mathcal{M}) = \{M \mid \mathcal{M} \vdash M\}$. In our example, both R_4 and $\mathbf{E}_{\mathbf{G}_0(R_1)}(R_3)$ are in $\mathbf{Rec}(\mathcal{M})$. Notice that $\mathbf{Rec}(\mathcal{M})$ is typically infinite; for example, if any key $R_i \in \mathbf{Rec}(\mathcal{M})$ then for any $d > 0$ and any $b_1, b_2, \ldots, b_l \in \{0, 1\}$, the key $\mathbf{G}_{b_l}(\mathbf{G}_{b_{l-1}}(\cdots(\mathbf{G}_{b_1}(R_i))\cdots))$ is also in $\mathbf{Rec}(\mathcal{M})$. For singleton message-sets $\mathcal{M} = \{M\}$, we write $\mathbf{Rec}(M)$ for $\mathbf{Rec}(\mathcal{M})$.

The entailment relation, besides formalizing the functionality of the operations $\mathbf{E}$ and $\mathbf{G}$, implicitly defines their security semantics as well. In particular, we assume that information that *cannot* be recovered using this relation is completely "hidden" from the point of view of any adversarial observer. In our example, the keys R_3 and $\mathbf{G}_0(R_1)$ cannot be recovered from $\mathcal{M}$ (even though they both appear in it) and so, it is assumed that an attacker, when given access to all of $\mathcal{M}$, has no knowledge, whatsoever, either about R_3 or about $\mathbf{G}_0(R_1)$. (In effect, $\mathcal{M}$ conceals these two keys completely.) Such a line of reasoning reflects the basic philosophy of the symbolic model, namely, that cryptographic operations are ideal objects and information that is not obtainable using certain fixed rules (modeling just the correctness properties of these objects) is secure by default.

Chapter 3

Defining Group Key Distribution

Consider a set of $\mathbf{n}$ users, labeled $1, 2, \ldots, \mathbf{n}$, sharing a broadcast communication channel. Suppose that $\mathbf{n}$ is very large (say, of the order of the number of hosts on the Internet). At any time t, users in a specific set $\mathcal{S}^{(t)} \subset \{1, \ldots, \mathbf{n}\}$, referred to as the *members* at that time, are authorized to receive information sent on the channel and in order to enable private communication amongst these users, we would like to enable them to share a *group key* $K^{(t)}$ and to encrypt all transmitted information using that key. The set $\mathcal{S}^{(t)}$ changes with time and accordingly, the users should also be able to update $K^{(t)}$ in such a manner that at every instant t, all and only the users in $\mathcal{S}^{(t)}$ can recover $K^{(t)}$.

This, in essence, is the problem of group key distribution. The problem can easily be seen to be equivalent to that of privacy in group communication: on one hand, given a protocol for distributing group keys securely, one can use it to ensure privacy in the group (by suitably performing encryption under the group key at every instant) and on the other, given a protocol for private group communication, one can distribute group keys securely using it (by simply generating a fresh key at every instant and transmitting it securely to everyone in the group). As such, throughout this thesis, we identify the problem of private group communication with the group key distribution problem. We remark that group key distribution protocols can find applications in multiple contexts (besides that of ensuring privacy); for example, when coupled with a message authentication scheme, they can be used to guarantee integrity of all messages exchanged within the group and to identify the sender of any message as being a member of the group.

In this thesis, we are interested in *centralized* protocols for group key distribution; that is, we assume that there exists a (physical or logical) central trusted authority C who shares a unique, long-lived key K_i with every user i of the underlying broadcast channel, and uses these keys to communicate the group key $K^{(t)}$ securely to all members at time t. We also assume that the channel is reliable (every message sent by C is received by every user of the channel) and authenticated (every user can verify that the message was indeed sent by C). Most known protocols for group key distribution in the literature are based on the assumption of centralized trust (primarily for reasons of efficiency).

3.1 Protocol Correctness

A group key distribution (GKD) protocol Π for $\mathbf{n}$ users has two components: a setup program S, and a key distribution program C. (The latter models all activities of the center.) The setup program assigns long-lived keys to all users and decides the initial state of C. For all i, the long-lived key of user i, denoted K_i, is an element of $\mathbf{Keys}$ subject to the requirement that it is not recoverable from any other long-lived key. In other words, there must exist no two keys K_i, K_j such that $K_j \in \mathbf{Rec}(K_i)$ (that is, $K_j = \mathbf{G}_{b_1}(\mathbf{G}_{b_2}(\cdots \mathbf{G}_{b_l}(K_i)\cdots))$ for some $l \geq 0$ and $b_1, \cdots, b_l \in \{0,1\}$). The initial state $\mathcal{Z}^{(0)}$ of the program C is a set of keys with the property that for every $i \in \{1, \dots, \mathbf{n}\}$, $K_i \in \mathbf{Rec}(\mathcal{Z}^{(0)})$.

We remark that the setup program is just an abstraction used for convenience of protocol description. In practice, long-lived keys of users need not be generated all at once before the protocol begins, but only when the respective user joins the group. (Once generated, the key can be shared between the user and the center using standard techniques based on, say, public-key cryptography.) Furthermore, to reduce storage requirements at the center, all long-lived keys could be derived using a single seed key K_0, via multiple applications of the PRG $\mathbf{G}$, while satisfying the above-stated requirement. (For greater efficiency, a pseudo-random *function* [22] could also be used.)

At every instant $t > 0$, the program C is given, as input, a description of the current set of members $\mathcal{S}^{(t)}$ and a set of keys $\mathcal{Z}^{(t-1)}$ corresponding to its state at the previous instant. It outputs a set of messages $\mathcal{M}^{\Pi}_{\mathcal{S}^{(t)}}$ (to be transmitted on the broadcast channel) and its updated state $\mathcal{Z}^{(t)}$. Each message in $\mathcal{M}^{\Pi}_{\mathcal{S}^{(t)}}$ is an element of $\mathbf{Msgs}$ (that is, it is derived from the variable M in grammar (2.1)). These messages are referred to as the *rekey* messages for time t since they are used to establish a fresh secret key—the group key—amongst all members at that time.

Let $[\mathbf{n}]$ denote the set $\{1, \ldots, \mathbf{n}\}$ and let $2^{[\mathbf{n}]}$ denote the power set of $[\mathbf{n}]$. For any sequence of member-sets $\overrightarrow{\mathcal{S}}^{(t)} := (\mathcal{S}^{(1)}, \cdots, \mathcal{S}^{(t)}) \in (2^{[\mathbf{n}]})^t$, let $\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ denote the set of all the rekey messages output by C when given this sequence as input; that is,

$$\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}} = \bigcup_{1 \le t' \le t} \mathcal{M}^{\Pi}_{\mathcal{S}^{(t')}}$$

Definition 3.1.1 An $\mathbf{n}$-user GKD protocol Π is called correct if for all $t > 0$, for all sequences $\overrightarrow{\mathcal{S}}^{(t)} \in (2^{[\mathbf{n}]})^t$ there exists a key K such that

$$\forall i \in \mathcal{S}^{(t)} \ : \ K \in \mathbf{Rec}(\{K_i\} \bigcup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}) \tag{3.1}$$

We assume that for every instant t, there is a distinguished key K that satisfies the above criterion and that is also used for applications (like the encryption of group data); this key is the group key for time t and we denote it by $K^{(t)}$. We remark that our correctness definition is quite liberal in the sense that members are allowed to be able to recover the group key using *all* the messages transmitted by the center up to the current instant. In practice, one could be more stringent and require instead that the group key be recoverable by a member using only the messages transmitted *since the instant the member joined the group*. Indeed, all protocols we consider in this thesis have that property.

3.2 Security Definitions

A natural first step in defining security of group key distribution would be to require that for each instant t, no non-member be able to recover, individually, the group key $K^{(t)}$ using the messages sent up to that instant. This essentially involves negating criterion (3.1) for non-members as follows:

Definition 3.2.1 An $\mathbf{n}$-user GKD protocol Π is called *secure against single-user attacks* (in the symbolic model) if for all $t > 0$, and for all sequences $\overrightarrow{\mathcal{S}}^{(t)} \in (2^{[\mathbf{n}]})^t$

$$\forall i \notin \mathcal{S}^{(t)} \;:\; K^{(t)} \notin \mathbf{Rec}(\{K_i\} \bigcup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}) \tag{3.2}$$

The above definition is quite intuitive and easy to use, but it has one severe shortcoming: it does not allow non-members to *collude* with each other and to *mount coordinated attacks* on the protocol. In practice, coordinated attacks could be quite feasible to implement—an intruder could gain illegitimate access to several hosts on a network (by, say, cracking their administrative passwords in parallel), and could potentially combine all information available on these hosts and recover group keys that none of the hosts can recover individually.

Let us now strengthen this definition to incorporate collusion attacks. For any instant t, let $\overline{\mathcal{S}}^{(t)}$ denote the set of non-members in the protocol at that instant; that is, $\overline{\mathcal{S}}^{(t)} = [\mathbf{n}] \setminus \mathcal{S}^{(t)}$.

Definition 3.2.2 An $\mathbf{n}$-user GKD protocol Π is called *secure against collusion attacks*, or simply *collusion-resistant* (in the symbolic model), if for all $t > 0$, for all sequences $\overrightarrow{\mathcal{S}}^{(t)} \in (2^{[\mathbf{n}]})^t$

$$K^{(t)} \notin \mathbf{Rec}(\{K_i\}_{i \in \overline{\mathcal{S}}^{(t)}} \bigcup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}) \tag{3.3}$$

Note that condition (3.3) is equivalent to requiring that the key $K^{(t)}$ not be in $\mathbf{Rec}(\{K_i\}_{i \in \overline{\mathcal{S}}} \bigcup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}})$ for *any subset* $\overline{\mathcal{S}}$ of $\overline{\mathcal{S}}^{(t)}$. This follows from the observation that the function $\mathbf{Rec}(\cdot)$ is monotone with respect to the set inclusion relation; that is, for every two sets of messages $\mathcal{M}_1, \mathcal{M}_2$ such that $\mathcal{M}_1 \subseteq \mathcal{M}_2$, $\mathbf{Rec}(\mathcal{M}_1) \subseteq \mathbf{Rec}(\mathcal{M}_2)$.

Thus, $K^{(t)}$ is protected from all non-members at time t if and only if it is protected from every malicious subset of them.

The new definition appears reasonable but it is still lacking in one aspect: what if non-members at time t cannot recover $K^{(t)}$ immediately but can do so *later on*, after viewing more messages sent by the protocol? For example, consider a protocol that sets $K^{(t+1)}$ to be the same as $K^{(t)}$ whenever a new user joins the group at time $t+1$. This means that, when running the protocol, a user who is not a member at time t but becomes one at time $t+1$ would be able to acquire the key $K^{(t)}$, absolutely for free! *Should such a protocol be called secure?*

The answer depends on the application one is considering though it is reasonable to believe that in most situations, such an occurrence would count as a security violation. For example, in many applications (like pay-per-view services), every piece of information has a price associated with it and often, this price is independent of when the information is actually communicated. In such a setting, preserving secrecy of key material communicated in the past would be very important.

Thus, a natural way to strengthen the above security definitions would be to require that non-members at any instant t not be able to recover the group key $K^{(t)}$, *even when they are given access to future protocol messages*. Such a requirement is usually referred to as *backward secrecy* in the literature (see, for example, [43]) because it ensures that no group member can recover group keys of past instants when it was not part of the group.

Definition 3.2.3 An $\mathbf{n}$-user GKD protocol Π is called *strongly secure against single-user attacks* (in the symbolic model) if for all $t > 0$, for all sequences $\overrightarrow{\mathcal{S}}^{(t)} \in (2^{[\mathbf{n}]})^t$, we have

$$\forall \tilde{t} \leq t,\ \ \forall i \notin \mathcal{S}^{(\tilde{t})}\ :\ K^{(\tilde{t})} \notin \mathbf{Rec}(\{K_i\} \bigcup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}) \tag{3.4}$$

Definition 3.2.4 An $\mathbf{n}$-user GKD protocol Π is called *strongly collusion-resistant* (in the symbolic model) if for all $t > 0$, for all sequences $\overrightarrow{\mathcal{S}}^{(t)} \in (2^{[\mathbf{n}]})^t$, we have

$$\forall \tilde{t} \leq t\ :\ K^{(\tilde{t})} \notin \mathbf{Rec}(\{K_i\}_{i \in \overline{\mathcal{S}}^{(\tilde{t})}} \bigcup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}) \tag{3.5}$$

Definition 3.2.4 is the strongest symbolic security definition we consider in this thesis. As in the case of Definition 3.2.2, we could re-phrase condition (3.5) by requiring that for all $\tilde{t} \leq t$, $K^{(\tilde{t})}$ not be in $\mathbf{Rec}(\{K_i\}_{i \in \overline{\mathcal{S}}} \bigcup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}})$ for any subset $\overline{\mathcal{S}}$ of $\overline{\mathcal{S}}^{(\tilde{t})}$. (Again, the equivalence of the reformulated definitions with the above ones would follow from the monotonicity of the function $\mathbf{Rec}(\cdot)$.) Alternatively, we could say that a protocol is strongly collusion-resistant if for all sets $\overline{\mathcal{S}} \subseteq [\mathbf{n}]$, for all sequences $\overrightarrow{\mathcal{S}}^{(t)}$, and for all instants $\tilde{t} \leq t$ such that $\mathcal{S}^{(\tilde{t})} \cap \overline{\mathcal{S}} = \emptyset$, $K^{(\tilde{t})} \notin \mathbf{Rec}(\{K_i\}_{i \in \overline{\mathcal{S}}} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}})$.

In past work on group key distribution, security notions of the above kind have already been considered but to the best of our knowledge, our work is the first to formalize these notions within a general symbolic model of computation. Such a formalization is essential both from the perspective of proving security of protocols (as we do in Chapter 5) and also for the purpose of proving robust lower bounds on the efficiency of protocols (as done in Chapter 6).

Chapter 4

An Equivalence Theorem

4.1 Single Encryption Protocols

Although the protocol language we defined in Chapter 2 is fairly general and encompasses all (symmetric-key) group key distribution protocols in the literature, it turns out that most protocols of interest can be captured by a more restrictive model. Specifically, our grammar of protocol messages (equation (2.1)) allows protocols to use *nested encryption* in generating ciphertexts whereas most GKD protocols (nine out of eleven surveyed in [31]) do not exploit this possibility at all. In other words, messages in such protocols are derivable from the following modified version of grammar (2.1):

$$\begin{aligned} M &\rightarrow K \mid \mathbf{E}_K(K) \\ K &\rightarrow \mathcal{R} \mid \mathbf{G}_0(K) \mid \mathbf{G}_1(K) \end{aligned} \tag{4.1}$$

Notice that the rule $M \rightarrow \mathbf{E}_K(M)$ has been replaced with $M \rightarrow \mathbf{E}_K(K)$ in the above grammar.

We refer to protocols that generate messages according to grammar (4.1) as *single encryption protocols* and group key distribution protocols that fall within this class are called *single encryption* GKD *protocols*, or simply, S-GKD *protocols*. Interestingly, even with this minor restriction, security analysis of protocols can be greatly simplified, as is illustrated by the following theorem:

Theorem 4.1.1 An S-GKD protocol Π is (strongly) secure against single-user attacks if and only if it is (strongly) collusion-resistant. In other words, Π satisfies Definition 3.2.1 (resp. Definition 3.2.3) if and only if it satisfies Definition 3.2.2 (resp. Definition 3.2.4).

Thus, if one is interested in analyzing the security of an S-GKD protocol against collusion attacks, it suffices to prove it secure against single-user attacks only; collusion-resistance would follow from this *automatically!* We illustrate the usefulness of this result by applying it to the security analysis of various protocols in Chapter 5. First, let us prove the result.

4.2 Key Graphs

The key insight underlying the proof of Theorem 4.1.1 (and of several other results in subsequent chapters) is the observation that security analysis of S-GKD protocols can be conducted, quite conveniently, using a graph-theoretic abstraction. We first describe this abstraction in detail.

Consider the execution of an S-GKD protocol Π given any sequence of member sets $\overrightarrow{\mathcal{S}}^{(t)} = (\mathcal{S}^{(1)}, \cdots, \mathcal{S}^{(t)})$ as input. With any such execution, let us associate a directed graph $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$, called the *key graph* for that execution of Π, defined as follows:

- The set of nodes in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ is equal to $\mathbf{Keys}$ (the set of all keys derivable from grammar (2.1)).
- For any $K, K' \in \mathbf{Keys}$, there is an edge from K to K' in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ if and only if either of the following is true:
  - $K' = \mathbf{G}_b(K)$ for some $b \in \{0, 1\}$;
  - $\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ contains the message $\mathbf{E}_K(K')$.

  We refer to an edge of the form $K \to \mathbf{G}_b(K)$ as a *given edge* or, simply, a g-edge and all other edges are referred to as *ciphertext edges* or c-edges.

The intuition underlying both types of edges is the same: Given the key K (and the ciphertexts transmitted by the protocol), if we can recover another key K' in exactly one step (via the entailment relation $\vdash$ of Section 2.4), then—and only then—we introduce an edge from K to K' in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$. Note that the creation of c-edges is dependent on the protocol whereas g-edges are given to us for free, independently of which protocol we are running (hence the name "given edges").

Since every edge in a key graph corresponds to a single step in the computation of the function $\mathbf{Rec}(\cdot)$, a sequence of edges (that is, a *path* in the graph) would naturally correspond to a sequence of such steps. As a result, all keys that can be *reached* from a key K in the graph are exactly those that can be recovered from it (and the ciphertexts sent by the protocol).

The following lemma formalizes this relationship between reachability and recoverability. For any key graph $\mathcal{G}$ and any set of keys $\mathcal{K}$, let $\mathbf{Reach}_{\mathcal{G}}(\mathcal{K})$ denote the set of all keys that are reachable from $\mathcal{K}$ in $\mathcal{G}$. Formally, $\mathbf{Reach}_{\mathcal{G}}(\mathcal{K})$ is the smallest set of keys such that $\mathcal{K} \subseteq \mathbf{Reach}_{\mathcal{G}}(\mathcal{K})$ and for every $K \in \mathbf{Reach}_{\mathcal{G}}(\mathcal{K})$, and every edge from K to another key K', K' is also in $\mathbf{Reach}_{\mathcal{G}}(\mathcal{K})$. For any set of message $\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ transmitted by a protocol (when given the sequence $\overrightarrow{\mathcal{S}}^{(t)}$ as input), let $\mathcal{K}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ denote the set of unencrypted keys contained in $\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$.

Lemma 4.2.1 For any $\mathbf{n}$-user S-GKD protocol Π, for any set of keys $\mathcal{K}$, for any integer $t > 0$, and for any sequence of sets $\overrightarrow{\mathcal{S}}^{(t)} \in (2^{[\mathbf{n}]})^t$

$$\mathbf{Rec}(\mathcal{K} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}) \cap \mathbf{Keys} = \mathbf{Reach}_{\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}}(\mathcal{K} \cup \mathcal{K}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}})$$

Proof: We prove, using induction, that, for any $i \geq 0$, a key K is recoverable from $\mathcal{K} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ in i steps if and only if there exists a path of length i in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ that starts from some key in $\mathcal{K} \cup \mathcal{K}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ and ends in K. From this, the lemma would follow immediately.

For the base case, observe that a key K is recoverable from $\mathcal{K} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ in 0 steps if and only if it is contained in $\mathcal{K} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$, which is true if and only if $K \in \mathcal{K} \cup \mathcal{K}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$. The latter is equivalent to saying that there exists a path of length 0 from a node in $\mathcal{K} \cup \mathcal{K}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ to K.

Now suppose that the claim is true for some arbitrary $\tilde{i} \geq 0$. Using this, we prove the claim for $\tilde{i}+1$. K is recoverable from $\mathcal{K} \cup \mathcal{M}^{\Pi}_{\vec{\mathcal{S}}^{(t)}}$ in $\tilde{i}+1$ steps if and only if there exists another key K' that is recoverable (from the same set) in $\tilde{i}$ steps and either (a) $K = \mathbf{G}_b(K')$ for some $b \in \{0,1\}$ or else, (b) $\mathbf{E}_{K'}(K) \in \mathcal{M}^{\Pi}_{\vec{\mathcal{S}}^{(t)}}$. (In the former case, we apply Rule 1 of Section 2.4 to get K, and in the latter, we use Rule 2.) From the inductive hypothesis, we know that there exists a path of length $\tilde{i}$ in $\mathcal{G}^{\Pi}_{\vec{\mathcal{S}}^{(t)}}$ from some key $K'' \in \mathcal{K} \cup \mathcal{K}^{\Pi}_{\vec{\mathcal{S}}^{(t)}}$ to K'. From the definition of $\mathcal{G}^{\Pi}_{\vec{\mathcal{S}}^{(t)}}$, we know that either (a) or (b) is true if and only if there is an edge from K' to K in that graph. Joining the former path with the latter edge gives us a path from K'' to K of length $\tilde{i}+1$. ▌

For the rest of this thesis, most of the discussion on S-GKD protocols takes place in terms of key graphs. In particular, we describe and analyze protocols (within this class) in graph-theoretic terms, and, later on, use the concept of key graphs to establish lower bounds as well. As a first step, let us see how this concept helps us give a simple proof of Theorem 4.1.1.

4.3 Proof of Theorem 4.1.1

We first prove equivalence between definitions 3.2.3 and 3.2.4; the proof of equivalence between definitions 3.2.1 and 3.2.2 is very similar and we only sketch the difference between the two proofs in-line.

Clearly, Definition 3.2.4 implies Definition 3.2.3—if a protocol is strongly collusion-resistant then it must be strongly secure against single-user attacks as well. (This follows from the monotonicity of the key recovery function $\mathbf{Rec}(\cdot)$.) So we just need to prove the converse.

Let Π be any S-GKD protocol that is *not* strongly collusion-resistant, that is, one that fails to meet Definition 3.2.4. Then, there must exist some sequence $\vec{\mathcal{S}}^{(t)} = (\mathcal{S}^{(1)}, \cdots, \mathcal{S}^{(t)})$ and some instant $\tilde{t} \leq t$ for which:

$$K^{(\tilde{t})} \in \mathbf{Rec}(\{K_i\}_{i \in \overline{\mathcal{S}}^{(\tilde{t})}} \cup \mathcal{M}^{\Pi}_{\vec{\mathcal{S}}^{(t)}})$$

Using Lemma 4.2.1 we can express the above condition in terms of key graphs as below:

$$K^{(\tilde{t})} \in \mathbf{Reach}_{\mathcal{G}^{\Pi}_{\vec{\mathcal{S}}(t)}}(\{K_i\}_{i \in \overline{\mathcal{S}}^{(\tilde{t})}} \cup \mathcal{K}^{\Pi}_{\vec{\mathcal{S}}(t)}) \tag{4.2}$$

We claim that for any key graph $\mathcal{G}$, any set of users $\overline{\mathcal{S}} \subseteq [\mathbf{n}]$, and any set of keys $\mathcal{K}$ in $\mathcal{G}$,

$$\mathbf{Reach}_{\mathcal{G}}(\{K_i\}_{i \in \overline{\mathcal{S}}} \cup \mathcal{K}) = \bigcup_{i \in \overline{\mathcal{S}}} \mathbf{Reach}_{\mathcal{G}}(\{K_i\} \cup \mathcal{K}) \tag{4.3}$$

Note that given this claim, condition (4.2) on protocol Π can be written equivalently as:

$$\begin{aligned} & K^{(\tilde{t})} \in \bigcup_{i \in \overline{\mathcal{S}}^{(\tilde{t})}} \mathbf{Reach}_{\mathcal{G}^{\Pi}_{\vec{\mathcal{S}}(t)}}(\{K_i\} \cup \mathcal{K}^{\Pi}_{\vec{\mathcal{S}}(t)}) \\ \Longrightarrow \quad & \exists i \in \overline{\mathcal{S}}^{(\tilde{t})} \ : \ K^{(\tilde{t})} \in \mathbf{Reach}_{\mathcal{G}^{\Pi}_{\vec{\mathcal{S}}(t)}}(\{K_i\} \cup \mathcal{K}^{\Pi}_{\vec{\mathcal{S}}(t)}) \\ \Longrightarrow \quad & \exists i \in \overline{\mathcal{S}}^{(\tilde{t})} \ : \ K^{(\tilde{t})} \in \mathbf{Rec}(\{K_i\} \cup \mathcal{M}^{\Pi}_{\vec{\mathcal{S}}(t)}) \cap \mathbf{Keys} \end{aligned}$$

which implies that Π fails to meet Definition 3.2.3, and the theorem follows from this. (For the proof of equivalence between Definition 3.2.1 and Definition 3.2.2, the argument would be the same except that $\tilde{t} \leq t$ would get replaced by $\tilde{t} = t$.)

What we have claimed above—equation (4.3)—is actually a simple consequence of the definition of reachability in graphs, formalized in the succeeding lemma. (Below, we use the notation $\mathbf{Reach}_{\mathcal{G}}(\mathcal{K})$ for an arbitrary graph $\mathcal{G}$ just as we did for key graphs, that is, to denote the set of nodes reachable from a node-set $\mathcal{K}$ in $\mathcal{G}$.)

Lemma 4.3.1 Let $\mathcal{G}$ be any directed graph and for any positive integer s, let $\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_s$ be arbitrary sets of nodes in it. Then,

$$\mathbf{Reach}_{\mathcal{G}}(\cup_{i=1}^{s} \mathcal{K}_i) = \bigcup_{i=1}^{s} \mathbf{Reach}_{\mathcal{G}}(\mathcal{K}_i)$$

Note that by selecting $s = |\overline{\mathcal{S}}|$ and $\mathcal{K}_i = \mathcal{K} \cup \{K_i\}$ for each $i \in \{1, \cdots, s\}$ in the above lemma, we obtain equation (4.3).

Proof: The proof uses an inductive argument. Let us first enhance the definition of reachability to incorporate the number of "hops" required to reach a node from a given set of nodes. For any set of nodes $\mathcal{V}$ in $\mathcal{G}$, the set of nodes that are reachable from $\mathcal{V}$ *in h hops*, denoted $\mathbf{Reach}^h_{\mathcal{G}}(\mathcal{V})$, is defined recursively as follows:

(a) $\mathbf{Reach}^0_{\mathcal{G}}(\mathcal{V}) = \mathcal{V}$; and

(b) For any $h \geq 0$, a node v' is contained in $\mathbf{Reach}^{h+1}_{\mathcal{G}}(\mathcal{V})$ if and only if $v' \in \mathbf{Reach}^h_{\mathcal{G}}(\mathcal{V})$ or there exists a node $v \in \mathbf{Reach}^h_{\mathcal{G}}(\mathcal{V})$ for which $v \to v'$ is an edge in $\mathcal{G}$.

Clearly, for any $\mathcal{V}$, $\mathbf{Reach}_{\mathcal{G}}(\mathcal{V}) = \bigcup_{h=0}^{\infty} \mathbf{Reach}^h_{\mathcal{G}}(\mathcal{V})$. Thus, to prove the lemma, it suffices to show that for any sequence of node-sets $\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_s$ and any $h \geq 0$, $\mathbf{Reach}^h_{\mathcal{G}}(\cup_{i=1}^s \mathcal{K}_i) = \bigcup_{i=1}^s \mathbf{Reach}^h_{\mathcal{G}}(\mathcal{K}_i)$. From the monotonicity of the function, we know that $\mathbf{Reach}^h_{\mathcal{G}}(\cup_{i=1}^s \mathcal{K}_i) \supseteq \bigcup_{i=1}^s \mathbf{Reach}^h_{\mathcal{G}}(\mathcal{K}_i)$, so we only need to prove containment in the other direction.

The statement is trivially true for $h = 0$: by definition, $\mathbf{Reach}^0_{\mathcal{G}}(\cup_{i=1}^s \mathcal{K}_i) = \cup_{i=1}^s \mathcal{K}_i = \bigcup_{i=1}^s \mathbf{Reach}^0_{\mathcal{G}}(\mathcal{K}_i)$. Suppose that the statement is true for any arbitrary $h \geq 0$; that is, for any $\mathcal{K}_1, \cdots, \mathcal{K}_s$, $\mathbf{Reach}^h_{\mathcal{G}}(\cup_{i=1}^s \mathcal{K}_i) \subseteq \bigcup_{i=1}^s \mathbf{Reach}^h_{\mathcal{G}}(\mathcal{K}_i)$. Consider any node v' in $\mathbf{Reach}^{h+1}_{\mathcal{G}}(\cup_{i=1}^s \mathcal{K}_i)$. For any such node, there exists a node $v \in \mathbf{Reach}^h_{\mathcal{G}}(\cup_{i=1}^s \mathcal{K}_i)$ such that either $v = v'$ or else there exists an edge from v to v' in $\mathcal{G}$. From the inductive hypothesis, $v \in \bigcup_{i=1}^s \mathbf{Reach}^h_{\mathcal{G}}(\mathcal{K}_i)$; that is, v belongs to $\mathbf{Reach}^h_{\mathcal{G}}(\mathcal{K}_i)$ for *some* $i \in \{1, \cdots, s\}$. For such an i, v' must be in $\mathbf{Reach}^{h+1}_{\mathcal{G}}(\mathcal{K}_i)$. (This is true whether $v = v'$ or there exists an edge $v \to v'$ in $\mathcal{G}$.) Thus, $v' \in \bigcup_{i=1}^s \mathbf{Reach}^{h+1}_{\mathcal{G}}(\mathcal{K}_i)$, which means $\mathbf{Reach}^{h+1}_{\mathcal{G}}(\cup_{i=1}^s \mathcal{K}_i) \subseteq \bigcup_{i=1}^s \mathbf{Reach}^{h+1}_{\mathcal{G}}(\mathcal{K}_i)$. ▌

4.4 A Cautionary Note

A natural question to ask is whether our equivalence theorem can be proven for GKD protocols that exploit the possibility of nesting the encryption operation. It turns out that this is impossible—even for protocols that make use of double encryption (one level of nesting) only, we can establish a separation between the notion of collusion-resistance and that of security against single-user attacks.

Theorem 4.4.1 There exists a GKD protocol that uses double encryption, is strongly secure against single-user attacks (that is, satisfies Definition 3.2.3) but is not collusion-resistant (that is, does not satisfy Definition 3.2.2).

Since collusion-resistance (Definition 3.2.2) is weaker than strong collusion-resistance (Definition 3.2.4), the above theorem also implies a separation between strong collusion-resistance and strong security against single-user attacks. Similarly, the notions of plain collusion-resistance and plain security against single-user attacks (definitions 3.2.2 and 3.2.1 respectively) are also separated by the theorem.

The intuition behind the separation is quite simple: a key K that is doubly encrypted requires the knowledge of *two* decryption keys to be recovered (as opposed to *one* such key in the case of single encryption). Now, if a protocol distributes this doubly encrypted key and assigns the other keys in a manner such that the decryption keys required to recover K are known to two different users, collusion can be a boon: the users cannot recover K by themselves but, by cooperating and putting their keys together, they can! If K is the group key and the two users are malicious non-members, this would mean that the protocol is ruined against collusions.

We now present a concrete protocol that formalizes this intuition. Our protocol is very similar to one due to Pinkas [39], which was designed for the purpose of performing state updates in GKD protocols. (In [39], an improvement of the same protocol that provides collusion-resistance is also given.)

Proof: The separation protocol involves the use of, what we refer to in this thesis, *fully-pseudo-random chains (FPCs)* of keys, a notion similar to that of forward-secure PRGs already studied in the literature [6]. An FPC of length $\mathbf{n}$, built from a purely random key $K_0 \in \mathcal{R}$ is a sequence of $\mathbf{n}$ key pairs $((K_i, K'_i))_{i \in [\mathbf{n}]}$ such that for every $i \in [\mathbf{n}]$, $K_i = \mathbf{G}_b(K_{i-1})$ and $K'_i = \mathbf{G}_{1-b}(K_i)$ (where $b \in \{0, 1\}$ is fixed for all i). The K'_i's in this chain are "fully" pseudo-random in the sense that it is (computationally) infeasible to distinguish between them and a sequence of $\mathbf{n}$ independently random keys.

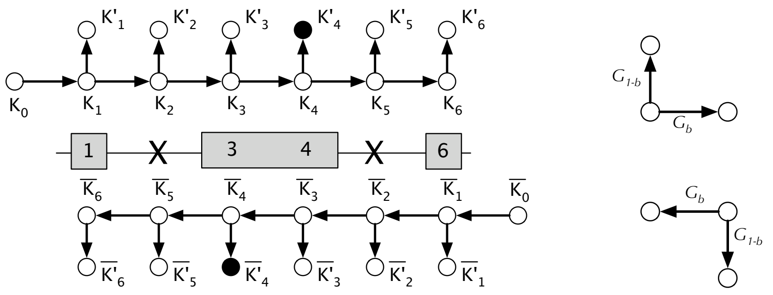

Figure 4.1: An illustration for the proof of Theorem 4.4.1.

In our protocol, the setup program S creates two FPCs of length $\mathbf{n}$ (the total number of users) using two different purely random keys K_0 and $\overline{K}_0$, one called the *forward chain* and the other called the *backward chain* (See the figure for an example with $\mathbf{n} = 6$), and gives the keys $(K_i, \overline{K}_{\mathbf{n}-i+1})$ to user i[1]. Note that given this, user i can recover the key pairs $(K_i, K'_i), \cdots, (K_{\mathbf{n}}, K'_{\mathbf{n}})$ in the forward chain and the key pairs $(\overline{K}_{\mathbf{n}-i+1}, \overline{K}'_{\mathbf{n}-i+1}), \cdots, (\overline{K}_{\mathbf{n}}, \overline{K}'_{\mathbf{n}})$ in the backward chain.

The key distribution program C works as follows: given a set $\mathcal{S}^{(t)}$ as input, it first divides the sequence $(1, \cdots, \mathbf{n})$ into the smallest possible set of intervals such that every $i \in \mathcal{S}^{(t)}$ belongs to exactly one interval and no $i \in \overline{\mathcal{S}}^{(t)}$ belongs to any of the intervals. For example, if $\mathbf{n} = 6$ and the target set is $\{1, 3, 4, 6\}$, these intervals would be $(1), (3, 4), (6)$, as shown in the figure. The number of such intervals is at most the size of $\overline{\mathcal{S}}^{(t)}$, plus 1.

Let $I_1, \cdots, I_{r+1}$ denote these intervals. C generates a fresh (purely random) key $K^{(t)}$ and for each interval $I_j = (j_1, \cdots, j_m)$, it creates a ciphertext $\mathbf{E}_{\overline{K}'_{\mathbf{n}-j_1+1}}(\mathbf{E}_{K'_{j_m}}(K^{(t)}))$. Note that this ciphertext can be decrypted by the users who know both K'_{j_m} and $\overline{K}'_{\mathbf{n}-j_1+1}$, which is exactly the users in I_j. (In the figure, the darkened keys denote the keys used to encrypt $K^{(t)}$ so as to transmit it to users in the interval $(3, 4)$.) The set $\mathcal{M}^{\Pi}_{\mathcal{S}^{(t)}}$ includes all ciphertexts created in this manner.

[1]Such a setup procedure does not directly fit our model for group key distribution but can easily be made to do so by having the center transmit, at time $t = 1$, the keys in the two FPCs suitably encrypted under the unique long-lived keys of the users.

It is easy to verify that this protocol is strongly secure against single-user attacks (satisfies Definition 3.2.3) but is not collusion-resistant (fails Definition 3.2.2). In fact, a malicious coalition of size two is enough to break the protocol. For example, in the figure, users 2 and 5 can collude and recover $K^{(t)}$: user 2 knows K'_4 (but not $\overline{K}'_4$) and user 5 knows $\overline{K}'_4$ (but not K'_4); so, even though neither of them can recover $K^{(t)}$ by himself, they can do so easily by colluding.

∎

4.5 Acknowledgement

Chapter 4, in part, is a reprint of the material as it appears in 33rd International Colloquium on Automata, Languages and Programming (ICALP), July 2006, Micciancio, Daniele; Panjwani, Saurabh. The dissertation author was the primary investigator and author of this paper. The presentation of the proof of Theorem 4.1.1 is different and more modular in the current work.

Chapter 5

Upper Bounds

In this chapter, we present several group key distribution protocols and analyze them with respect to the symbolic notions of security defined in Chapter 3. The protocols we present fall into two categories: *logical key hierarchy protocols* and *subset cover protocols*.

5.1 Logical Key Hierarchy Protocols

The *Logical Key Hierarchy (*LKH*)* protocols are a class of GKD protocols devised by Wong *et al.* [46] and independently by Wallner *et al.* [45], in the context of implementing secure multicast over the Internet. LKH protocols are highly efficient, both in terms of the amount of computation performed by protocol users and the communication costs incurred by rekey operations, and require very little state to be maintained by users. In this thesis, we consider two protocols from the LKH class: the first protocol, which we refer to as plain-LKH, is the original protocol due to [46, 45] and uses encryption as the only cryptographic building-block. The second protocol, called improved-LKH here, is a protocol due to Canetti, Garay, Itkis, Micciancio, Naor and Pinkas [10]; it uses both encryption and pseudo-random generators and betters plain-LKH in terms of communication efficiency. Both protocols are single encryption protocols, that is, neither of them uses nested encryption in the process of generating rekey messages. In this section, we present plain-LKH and improved-LKH (rather, a more secure variant of each of the protocols), and analyze security of both these protocols in the symbolic model.

5.1.1 Protocol state

In any LKH protocol (and in plain-LKH and improved-LKH in particular), the center's state at time t consists of a set of keys that are organized in the form of a tree, denoted $\mathcal{T}r^{(t)}$. This tree represents a hierarchy of keys, with the group key $K^{(t)}$ being at the top of the hierarchy (as the root), and the long-lived keys of users in $\mathcal{S}^{(t)}$ being at the bottom (as leaves of the tree). The goal of the protocol, roughly, is to ensure that at every instant t, each user $i \in \mathcal{S}^{(t)}$ can recover all and only the keys lying *on the path from K_i to $K^{(t)}$* in $\mathcal{T}r^{(t)}$, and that non-members cannot recover any key in it. This, in turn, guarantees that the group key is recoverable by all and only the members at every instant.

For simplicity, we present the protocols assuming that the tree $\mathcal{T}r^{(t)}$ is always binary and its height is fixed at $\lceil \log_2(\mathbf{n}) \rceil$; generalizing to trees with arbitrary structure (in particular, trees that have variable depth and width) is easy, and we omit the details of how this is done. Let $\mathcal{L}$ denote the set of all binary strings of length at most $\lceil \log_2(\mathbf{n}) \rceil$. For any string $s \in \mathcal{L}$ and any value $b \in \{0, 1\}$, let $s \cdot b$ denote the string formed by appending b to s. For any $s \in \mathcal{L}$, let $|s|$ denotes the length of s.

For any instant t during the execution of an LKH protocol, we denote the keys in $\mathcal{T}r^{(t)}$ using symbols of the form $K_s^{(t)}$, with s being a string in $\mathcal{L}$. The root of $\mathcal{T}r^{(t)}$ (the group key) is denoted $K_\epsilon^{(t)}$ and for any key denoted $K_s^{(t)}$, its left (resp. right) child is denoted $K_{s\cdot 0}^{(t)}$ (resp. $K_{s\cdot 1}^{(t)}$). For any set $S \subseteq \mathcal{L}$, let $\mathcal{K}_S^{(t)} = \{K_s^{(t)} \mid s \in S\}$. Let $\mathcal{K}^{(t)}$ denote the set of all keys in $\mathcal{T}r^{(t)}$. Initially (at $t = 0$), $\mathcal{K}^{(t)} = \emptyset$ and for all $s \in \mathcal{L}$, the key $K_s^{(0)} = \perp$ (which stands for "undefined"). For any $t > 0$, $K_\epsilon^{(t)}$ is the group key at time t and for every $i \in \mathcal{S}^{(t)}$, there exists a unique string $s_t(i) \in \{0,1\}^{\lceil \log_2(\mathbf{n}) \rceil}$ such that $K_{s_t(i)}^{(t)} = K_i$. We denote the set of all prefixes of $s_t(i)$ by $\mathbf{pre}_t(i)$. So, $\mathcal{K}_{\mathbf{pre}_t(i)}^{(t)}$ is the set of keys in $\mathcal{T}r^{(t)}$ that lie on the path from K_i to $K_\epsilon^{(t)}$.

Besides the set of keys in $\mathcal{T}r^{(t)}$, the center also stores the long-lived keys of all

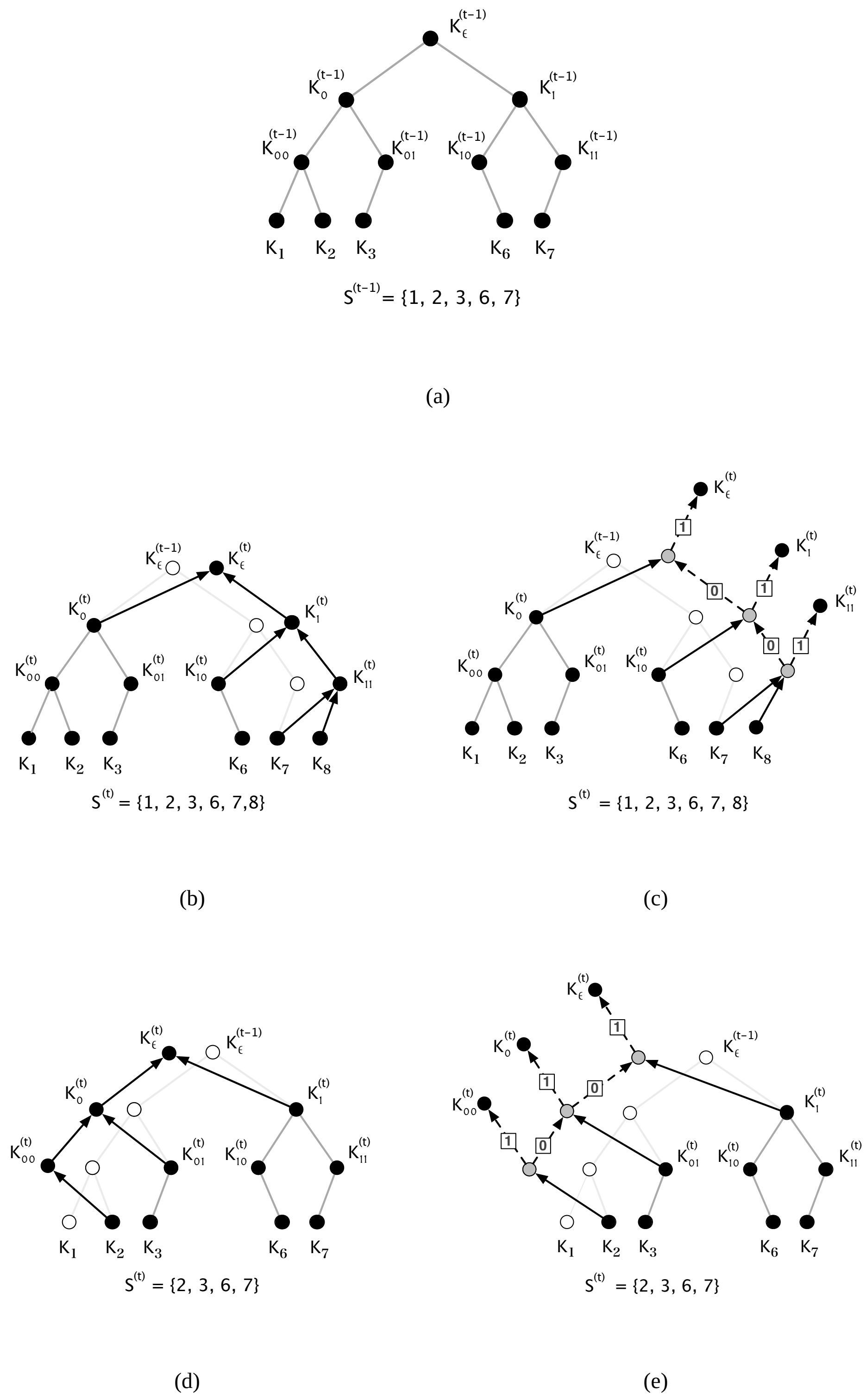
$K_\epsilon^{(t-1)}$
$K_0^{(t-1)}$
$K_1^{(t-1)}$
$K_{00}^{(t-1)}$
$K_{01}^{(t-1)}$
$K_{10}^{(t-1)}$
$K_{11}^{(t-1)}$
K_1 K_2 K_3 K_6 K_7
$S^{(t-1)} = \{1, 2, 3, 6, 7\}$
(a)
$S^{(t)} = \{1, 2, 3, 6, 7, 8\}$
(b)
$S^{(t)} = \{1, 2, 3, 6, 7, 8\}$
(c)
$S^{(t)} = \{2, 3, 6, 7\}$
(d)
$S^{(t)} = \{2, 3, 6, 7\}$
(e)

Figure 5.1: The plain-LKH$^+$ and improved-LKH$^+$ protocols.

users $K_1, \ldots, K_{\mathbf{n}}$ as part of its internal state. In particular, its state at time $t = 0$ contains only these keys (or a succinct representation of the same).

5.1.2 Rekey Messages

As in [46, 45, 10], we assume that at each instant t, the size of the group $\mathcal{S}^{(t)}$ changes by at most one and that either a single member leaves the group or else a single non-member joins it. A sequence of sets $\overrightarrow{\mathcal{S}}^{(t)} = (\mathcal{S}^{(1)}, \cdots, \mathcal{S}^{(t)})$ created in this manner is referred to as a *simple* sequence; formally, $\overrightarrow{\mathcal{S}}^{(t)}$ is simple if $\mathcal{S}^{(1)}$ has exactly one user and for all $1 \leq \tilde{t} < t$, $\mathcal{S}^{(\tilde{t}+1)} = \mathcal{S}^{(\tilde{t})} \cup \{i\}$ for some $i \in \overline{\mathcal{S}}^{(\tilde{t})}$ or $\mathcal{S}^{(\tilde{t}+1)} = \mathcal{S}^{(\tilde{t})} \setminus \{i\}$ for some $i \in \mathcal{S}^{(\tilde{t})}$. It is clear that arbitrary group membership updates can be simulated using simple sequences only.

Suppose that at some instant $t > 0$, a user $i \in \overline{\mathcal{S}}^{(t-1)}$ is added to the group; that is, $\mathcal{S}^{(t)} = \mathcal{S}^{(t-1)} \cup \{i\}$. (We think of $\mathcal{S}^{(0)}$ as being empty, so $\overline{\mathcal{S}}^{(0)} = [\mathbf{n}]$.) In response, the center first assigns a value to $s_t(i)$ from $\{0,1\}^{\lceil \log_2(\mathbf{n}) \rceil}$ such that there is no $j \in \mathcal{S}^{(t-1)}$ for which $s_t(i) = s_{t-1}(j)$. For each $j \in \mathcal{S}^{(t-1)}$, the center sets $s_t(j)$ to be equal to $s_{t-1}(j)$. Subsequently, the key hierarchy is updated from $\mathcal{T}r^{(t-1)}$ to $\mathcal{T}r^{(t)}$ as follows: $K^{(t)}_{s_t(i)}$ is set to be equal to K_i and for all $s \notin \mathbf{pre}_t(i)$, $K^{(t)}_s$ is set to $K^{(t-1)}_s$. Keys in the set $\mathcal{K}^{(t)}_{\mathbf{pre}_t(i) \setminus \{s_t(i)\}}$ are assigned new values, as described below.

Let $\mathbf{pre}'_t(i)$ denote the set $\mathbf{pre}_t(i) \setminus \{s_t(i)\}$. In both plain-LKH and improved-LKH, every key in $\mathcal{K}^{(t)}_{\mathbf{pre}'_t(i)}$ is set to be a fresh symbol from $\mathcal{R}$—a symbol that has not been used in the protocol before time t. The center then distributes each newly generated key $K^{(t)}_s$ (for $s \in \mathbf{pre}'_t(i)$) by outputting two ciphertexts for each such key:

(a) the encryption of $K^{(t)}_s$ under $K^{(t-1)}_s$, that is, $\mathbf{E}_{K^{(t-1)}_s}(K^{(t)}_s)$.

(b) the encryption of $K^{(t)}_s$ under K_i, that is, $\mathbf{E}_{K_i}(K^{(t)}_s)$.

Intuitively, the first set of ciphertexts allows all members in the set $\mathcal{S}^{(t-1)}$ who are authorized to recover $K^{(t)}_s$ (that is, for whom $K^{(t)}_s$ lies on the path from their long-lived key to the root of $\mathcal{T}r^{(t)}$) to be able to do so, and the second set of ciphertexts allows user i to be able to recover it. We argue (in Chapter 8) that this procedure for rekeying is insecure when considering security against computational attackers since it requires the group key at time $t-1$ to be used for key distribution; in particular, it is used for encrypting the new group key $K_{\epsilon}^{(t)}$ and distributing it. As such, we modify the procedure and define two new protocols plain-LKH^{+} and improved-LKH$^{+}\dot{}$The rekeying procedure for both these protocols is illustrated in the figure.

Suppose again that user i is added to the group at time t. In plain-LKH^{+}, the center generates values for keys in $\mathcal{K}_{\mathbf{pre}'_t(i)}^{(t)}$ just like in plain-LKH but distributes each freshly generated key by encrypting it under its two "children" in the updated key hierarchy. Specifically, for each $s \in \mathbf{pre}'_t(i)$, and for each $b \in \{0,1\}$, if $K_{s\cdot b}^{(t)} \neq \perp$, then the center outputs the rekey message $\mathbf{E}_{K_{s\cdot b}^{(t)}}(K_s^{(t)})$. In terms of key graphs, this corresponds to creating a c-edge from $K_{s\cdot b}^{(t)}$ to its parent $K_s^{(t)}$ in the hierarchy. For example, consider the situation shown in Figure Figure 5.1(a) where $\mathbf{n}=8$ and $\mathcal{S}^{(t-1)} = \{1,2,3,6,7\}$. Suppose that user 8 joins the group at time t ($\mathcal{S}^{(t)} = \{1,2,3,6,7,8\}$) and is assigned the label $s_t(8) = 111$. New values for keys with labels in $\mathbf{pre}'_t(8) = \{11,1,\epsilon\}$ are generated, and distributed using the c-edges shown (as solid arrows) in Figure Figure 5.1(b). (Keys that are in $\mathcal{K}^{(t)}$ are shown in black while those that are in $\mathcal{K}^{(t-1)} \setminus \mathcal{K}^{(t)}$ are shown in white.)

In improved-LKH^{+}, keys in $\mathcal{K}_{\mathbf{pre}'_t(i)}^{(t)}$ are assigned pseudo-random values. For any integer l and any $R_j \in \mathcal{R}$, let $\mathbf{G}_0^l(R_j)$ denote the pseudo-random key formed by applying the function $\mathbf{G}_0$ on R_j l times, iteratively. When user i is added to the group in the protocol improved-LKH^{+}, the center first picks a fresh (unused) symbol R_j from $\mathcal{R}$ and for every $s \in \mathbf{pre}'_t(i)$, defines a value $\tilde{K}_s^{(t)} = \mathbf{G}_0^{(\lceil \log_2(\mathbf{n}) \rceil - 1) - |s|}(R_j)$. Then, it sets $K_s^{(t)}$ to be equal to $\mathbf{G}_1(\tilde{K}_s^{(t)})$. Note that such a key assignment ensures that for any $s, s' \in \mathbf{pre}'_t(i)$ for which s' is a prefix of s, $K_{s'}^{(t)}$ is recoverable from $\tilde{K}_s^{(t)}$ (that is, $K_{s'}^{(t)} \in \mathbf{Rec}(\tilde{K}_s^{(t)})$). In particular, the group key $K_{\epsilon}^{(t)}$ is recoverable from $\tilde{K}_s^{(t)}$ for every $s \in \mathbf{pre}'_t(i)$.

Finally, for each $s \in \mathbf{pre}'_t(i)$, and for each $b \in \{0,1\}$, if $K_{s\cdot b}^{(t)} \notin \mathcal{K}_{\mathbf{pre}'_t(i)}^{(t)} \cup \{\perp\}$, the center outputs the rekey message $\mathbf{E}_{K_{s\cdot b}^{(t)}}(\tilde{K}_s^{(t)})$. Figure Figure 5.1(c) shows an example for the case when user 8 is added to the group $\mathcal{S}^{(t-1)} = \{1,2,3,6,7\}$ of Figure

Figure 5.1(a); here, the newly created g-edges are shown using dashed arrows (g-edges corresponding to $\mathbf{G}_0$ are labeled with 0 and the others with 1), while c-edges are shown using solid arrows as before.

We remark that the procedure for rekeying in response to user addition in improved-LKH$^+$ is very similar to that for user deletion in improved-LKH, described below. We also remark that a simpler procedure in which the step $K_s^{(t)} \leftarrow \mathbf{G}_1(\tilde{K}_s^{(t)})$ (assignment of $\mathbf{G}_1(\tilde{K}_s^{(t)})$ to $K_s^{(t)}$) is replaced with $K_s^{(t)} \leftarrow \tilde{K}_s^{(t)}$ could also be considered. However, the resulting protocol would not be secure in the computational model, which is why we prefer the above presentation.

USER DELETION. The rekeying procedure for user deletion in both plain-LKH$^+$ and improved-LKH$^+$ is the same as in plain-LKH and improved-LKH respectively, and is, in fact, quite similar to the process of rekeying for user addition. When a user $i \in \mathcal{S}^{(t-1)}$ leaves the group at time t (or is deleted from it on purpose), that is, $\mathcal{S}^{(t)} = \mathcal{S}^{(t-1)} \setminus \{i\}$, the center generates new keys corresponding to strings in $\mathbf{pre}_{t-1}(i)$ and distributes them securely to the legitimate members. First, for each $j \in \mathcal{S}^{(t)}$, it sets $s_t(j)$ to be equal to $s_{t-1}(j)$, and for each $s \notin \mathbf{pre}_{t-1}(i)$, it sets $K_s^{(t)}$ to $K_s^{(t-1)}$. Then, it computes the longest string $s_i \in \mathbf{pre}_{t-1}(i)$ such that there exists $b \in \{0,1\}$ for which

(a) $s_i \cdot b \notin \mathbf{pre}_{t-1}(i)$; and

(b) $K_{s_i \cdot b}^{(t)} \neq \perp$.

Let $\mathbf{pre}'_t(i)$ be the set of all prefixes of s_i.

New values are generated only for the keys labeled by strings in $\mathbf{pre}'_t(i)$. (So, for all $s \in \mathbf{pre}_{t-1}(i) \setminus \mathbf{pre}'_t(i)$, $K_s^{(t)}$ equals $\perp$.) The new values are generated and distributed differently in plain-LKH and in improved-LKH. In plain-LKH, a fresh (unused) symbol from $\mathcal{R}$ is assigned to $K_s^{(t)}$ for each $s \in \mathbf{pre}'_t(i)$, and for each $b \in \{0,1\}$, if $K_{s\cdot b}^{(t)} \neq \perp$, the rekey message $\mathbf{E}_{K_{s\cdot b}^{(t)}}(K_s^{(t)})$ is output. In improved-LKH, a fresh symbol R_j from $\mathcal{R}$ is picked and for each $s \in \mathbf{pre}'(i)$, the center defines the value $\tilde{K}_s^{(t)} = \mathbf{G}_0^{|s_i|-|s|}(R_j)$ and sets $K_s^{(t)}$ to be equal to $\mathbf{G}_1(\tilde{K}_s^{(t)})$. For each $s \in \mathbf{pre}'_t(i)$, if there exists $b \in \{0,1\}$ such that $K_{s\cdot b}^{(t)} \notin \mathcal{K}_{\mathbf{pre}'_t(i)}^{(t)} \cup \{\perp\}$, the center outputs the rekey message $\mathbf{E}_{K_{s\cdot b}^{(t)}}(\tilde{K}_s^{(t)})$. Figure Figure 5.1(d) shows how rekeying is done in plain-LKH when user 1 is deleted from the group $\{1,2,3,6,7\}$ of Figure Figure 5.1(a), and Figure Figure 5.1(e) shows the same for improved-LKH.

It is easy to show that both the protocols satisfy the correctness definition for GKD protocols (Definition 3.1.1): at every instant t, every member $i \in \mathcal{S}^{(t)}$ knows all the keys in $\mathcal{K}_{\mathbf{pre}_t(i)}^{(t)}$ and this set includes the group key $K_\epsilon^{(t)}$. A more formal argument appears as part of the proof of Theorem 5.1.1 below.

5.1.3 Efficiency

The LKH protocols are much more communication-efficient than the trivial protocol discussed in the introduction. For each addition or deletion of a member from the group, the number of ciphertexts output as rekey messages by the center is at most $2\lceil\log_2(\mathbf{n})\rceil$ in the case of plain-LKH$^+$ and at most $\lceil\log_2(\mathbf{n})\rceil$ in the case of improved-LKH$^+$. (Contrast this with $O(\mathbf{n})$ ciphertexts in the case of the trivial protocol.) This decrease in communication overhead comes at the cost of slightly increased computation cost ($O(\lceil\log_2(\mathbf{n})\rceil)$ decryptions and/or PRG computations per member) and increased storage requirements ($O(\lceil\log_2(\mathbf{n})\rceil)$ keys per member besides the long-lived key). We remark that it is easy to modify both the protocols such that the key hierarchy is maintained as a *complete* binary tree at every instant and its depth varied based on the number of members in the group. The communication complexity of the resulting protocols would be $2\lceil\log_2(n)\rceil$ in the case of plain-LKH$^+$ and $\lceil\log_2(n)\rceil$ in improved-LKH$^+$, for n being the size of the group (rather than the total number of users in the protocol).

Both plain-LKH$^+$ and improved-LKH$^+$ can also be modified to use d-ary trees (for arbitrary $d \in \{2,3,\ldots,\mathbf{n}\}$) instead of a binary one. In the modified protocols, the key hierarchy would be labeled using strings from the alphabet $\{0,1,\ldots,d-1\}^*$ and every newly generated key would be transmitted by encrypting it under all its children in the hierarchy (in the case of plain-LKH$^+$) or under all but one of its children (in the case of improved-LKH$^+$). We refer to the modified protocols as the *d-ary instances* of plain-LKH$^+$ and improved-LKH$^+$ respectively. The communication complexity of the d-ary instance of plain-LKH$^+$ is $d \cdot \lceil \log_d(\mathbf{n}) \rceil$ while that of improved-LKH$^+$ is $(d-1) \cdot \lceil \log_d(\mathbf{n}) \rceil$. Thus, an increase in the value of d affects the communication efficiency of the protocols negatively but, as we show later (in Chapter 9), it also leads to stronger computational security guarantees.

5.1.4 Security Analysis

Although both plain-LKH and improved-LKH have existed in the literature for a long time, a formal security analysis of either of these protocols has not been conducted prior to the current work. Indeed, as we discuss in Chapter 8, neither of these protocols are provably secure in the computational model, *even against passive eavesdropping attacks*; as such, their security analysis in the symbolic model alone is of little value. In the following theorems, we establish security of the modified protocols, plain-LKH$^+$ and improved-LKH$^+$, against collusion attacks in the symbolic model. In Chapter 8, we use these results to prove security of the same protocols in the computational model as well.

Theorem 5.1.1 Both plain-LKH$^+$ and improved-LKH$^+$ are strongly secure against collusion attacks in the symbolic model.

Proof: We analyze security of plain-LKH$^+$ and improved-LKH$^+$ with respect to simple sequences only, since the protocol accepts only such sequences as input. (As already mentioned, execution with non-simple sequences is easily simulatable using executions with simple ones.)

We use p-LKH$^+$ and i-LKH$^+$ as shorthand for plain-LKH$^+$ and improved-LKH$^+$. For any $\Pi \in \{\text{p-LKH}^+, \text{i-LKH}^+\}$, any $i \in [\mathbf{n}]$, and any simple sequence $\overrightarrow{\mathcal{S}}^{(t)}$, let $\mathbf{Reach}_{\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}}(i)$ denote the set of keys reachable from K_i in the key graph created when Π is executed on input $\overrightarrow{\mathcal{S}}^{(t)}$. Given Lemma 4.2.1, and the fact that no rekey message ever output by either p-LKH$^+$ or i-LKH$^+$ is an unencrypted key, this set is the same as $\mathbf{Rec}(\{K_i\} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}) \cap \mathbf{Keys}$.

Claim 5.1.2 Let $\Pi \in \{\textsf{p-LKH}^+, \textsf{i-LKH}^+\}$. For all $t \geq 0$, and for all simple sequences $\overrightarrow{\mathcal{S}}^{(t)} \in (2^{[\mathbf{n}]})^t$, the following is true:

(a) $\forall i \in \mathcal{S}^{(t)} \; : \; \mathbf{Reach}_{\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}}(i) \cap \mathcal{K}^{(t)} = \mathcal{K}^{(t)}_{\mathbf{pre}_t(i)}$;

(b) $\forall i \in \overline{\mathcal{S}}^{(t)} \; : \; \mathbf{Reach}_{\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}}(i) \cap \mathcal{K}^{(t)} = \emptyset$

Note that this claim implies two important facts about $\textsf{p-LKH}^+$ and $\textsf{i-LKH}^+$. First, that both are correct (in the sense of Definition 3.1.1) and second, that both are secure against single-user attacks (in the sense of Definition 3.2.1).

NOTATIONS. Before we prove the claim, we develop some notation. Let $\Pi \in \{\textsf{p-LKH}^+, \textsf{i-LKH}^+\}$. For any set of keys $\mathcal{K}$ and any $\mathcal{S}^{(t)} \subseteq [\mathbf{n}]$, let $\mathbf{reach}^{\Pi}_{\mathcal{S}^{(t)}}(\mathcal{K})$ denote the set of keys reachable from $\mathcal{K}$ via the $\textsf{c}$-edges and $\textsf{g}$-edges created in the protocol key graph for Π *at* time t, given that the input at that time is $\mathcal{S}^{(t)}$. For any $\mathcal{S}^{(t)} \subseteq [\mathbf{n}]$ and any $i \in \mathcal{S}^{(t)}$, let $\tilde{\mathcal{K}}^{(t)}_{\mathbf{pre}_t(i)}$ be a set defined as follows: if Π is $\textsf{p-LKH}^+$, this set is empty, and if it is $\textsf{i-LKH}^+$, it contains all keys of the form $\tilde{K}^{(t)}_s$ for which $s \in \mathbf{pre}_t(i)$. (Note that this definition is valid even if i is not the user added to the group at time t.)

For each instant $t > 0$ during the execution of either $\textsf{p-LKH}^+$ or $\textsf{i-LKH}^+$, the following is true:

- For each $i \in \mathcal{S}^{(t)} \cap \mathcal{S}^{(t-1)}$, $\mathbf{reach}^{\Pi}_{\mathcal{S}^{(t)}}(\mathcal{K}^{(t-1)}_{\mathbf{pre}_{t-1}(i)}) = \mathcal{K}^{(t-1)}_{\mathbf{pre}_{t-1}(i)} \cup \mathcal{K}^{(t)}_{\mathbf{pre}_t(i)} \cup \tilde{\mathcal{K}}^{(t)}_{\mathbf{pre}_t(i)}$
- For each $i \in \mathcal{S}^{(t-1)} \setminus \mathcal{S}^{(t)}$, $\mathbf{reach}^{\Pi}_{\mathcal{S}^{(t)}}(\mathcal{K}^{(t-1)}_{\mathbf{pre}_{t-1}(i)}) = \mathcal{K}^{(t-1)}_{\mathbf{pre}_{t-1}(i)}$
- For each $i \in \mathcal{S}^{(t)} \setminus \mathcal{S}^{(t-1)}$, $\mathbf{reach}^{\Pi}_{\mathcal{S}^{(t)}}(\{K_i\}) = \mathcal{K}^{(t)}_{\mathbf{pre}_t(i)} \cup \tilde{\mathcal{K}}^{(t)}_{\mathbf{pre}_t(i)}$
- For each $i \notin \mathcal{S}^{(t)} \cup \mathcal{S}^{(t-1)}$, $\mathbf{reach}^{\Pi}_{\mathcal{S}^{(t)}}(\{K_i\}) = \{K_i\}$.

The proof of the above four assertions follows simply from inspection of $\textsf{p-LKH}^+$ and $\textsf{i-LKH}^+$.

Proof of Claim 5.1.2. Clearly, the claim is true for $t = 0$: $\mathcal{S}^{(0)} = \emptyset$ and $\mathcal{K}^{(0)} = \emptyset$ for both protocols and so, $\mathbf{Reach}_{\mathcal{G}^{\Pi}_{\vec{\mathcal{S}}^{(0)}}}(i) \cap \mathcal{K}^{(0)} = \emptyset$ for all i. We argue that if the claim is true for some $t - 1 \geq 0$, then it is true for t as well; this suffices to prove the claim in general.

Consider any simple sequence $\overrightarrow{\mathcal{S}}^{(t)} = (\mathcal{S}^{(1)}, \cdots, \mathcal{S}^{(t-1)}, \mathcal{S}^{(t)})$ and any $i \in [\mathbf{n}]$. Let $\mathcal{R}each_1 := \mathbf{Reach}_{\mathcal{G}^{\Pi}_{\vec{\mathcal{S}}^{(t)}}}(i)$. This set can alternatively be expressed as follows:

$$\begin{aligned} \mathcal{R}each_1 &= \mathbf{reach}^{\Pi}_{\mathcal{S}^{(t)}}(\mathbf{Reach}_{\mathcal{G}^{\Pi}_{\vec{\mathcal{S}}^{(t-1)}}}(i)) \\ &= \mathbf{reach}^{\Pi}_{\mathcal{S}^{(t)}}((\mathbf{Reach}_{\mathcal{G}^{\Pi}_{\vec{\mathcal{S}}^{(t-1)}}}(i) \cap \mathcal{K}^{(t-1)}) \cup \{K_i\}) \end{aligned}$$

Both these equalities follow from the fact that in p-LKH^{+} and i-LKH^{+}, every c-edge created in the protocol key graph at time t issues from a key in $\mathcal{K}^{(t-1)} \cup \mathcal{K}^{(t)}$ or else a long-lived key K_i, and is always incident upon a key in $\mathcal{K}^{(t)}$ (or else, upon a key of the form $\tilde{K}_s^{(t)}$ in the case of i-LKH^{+}). Let $\mathcal{R}each_2$ denote the set $(\mathbf{Reach}_{\mathcal{G}^{\Pi}_{\vec{\mathcal{S}}^{(t-1)}}}(i) \cap \mathcal{K}^{(t-1)}) \cup \{K_i\}$; so $\mathcal{R}each_1 = \mathbf{reach}^{\Pi}_{\mathcal{S}^{(t)}}(\mathcal{R}each_2)$.

- *Case 1 ($i \in \mathcal{S}^{(t-1)}, i \in \mathcal{S}^{(t)}$):* From the inductive hypothesis, we know that, in this case, $\mathcal{R}each_2 = \mathcal{K}^{(t-1)}_{\mathbf{pre}_{t-1}(i)}$; so, $\mathcal{R}each_1 = \mathbf{reach}^{\Pi}_{\mathcal{S}^{(t)}}(\mathcal{R}each_2) = \mathcal{K}^{(t-1)}_{\mathbf{pre}_{t-1}(i)} \cup \mathcal{K}^{(t)}_{\mathbf{pre}_t(i)} \cup \tilde{\mathcal{K}}^{(t)}_{\mathbf{pre}_t(i)}$, and so, $\mathcal{R}each_1 \cap \mathcal{K}^{(t)} = \mathcal{K}^{(t)}_{\mathbf{pre}_t(i)}$.

- *Case 2 ($i \in \mathcal{S}^{(t-1)}, i \notin \mathcal{S}^{(t)}$):* In this case, $\mathcal{R}each_2 = \mathcal{K}^{(t-1)}_{\mathbf{pre}_{t-1}(i)}$ again, but this time $\mathcal{R}each_1 = \mathbf{reach}^{\Pi}_{\mathcal{S}^{(t)}}(\mathcal{R}each_2) = \mathcal{K}^{(t-1)}_{\mathbf{pre}_{t-1}(i)}$ (since user i gets deleted from $\mathcal{S}^{(t-1)}$ at time t); thus, $\mathcal{R}each_1 \cap \mathcal{K}^{(t)} = \mathcal{K}^{(t-1)}_{\mathbf{pre}_{t-1}(i)} \cap \mathcal{K}^{(t)} = \emptyset$.

- *Case 3 ($i \notin \mathcal{S}^{(t-1)}, i \in \mathcal{S}^{(t)}$):* From the inductive hypothesis, we know that $\mathcal{R}each_2 = \{K_i\}$, which means $\mathcal{R}each_1 = \mathbf{reach}^{\Pi}_{\mathcal{S}^{(t)}}(\mathcal{R}each_2) = \mathcal{K}^{(t)}_{\mathbf{pre}_t(i)} \cup \tilde{\mathcal{K}}^{(t)}_{\mathbf{pre}_t(i)}$ (since user i gets added to $\mathcal{S}^{(t-1)}$ at time t), and so, $\mathcal{R}each_1 \cap \mathcal{K}^{(t)} = \mathcal{K}^{(t)}_{\mathbf{pre}_t(i)}$.

- *Case 4 ($i \notin \mathcal{S}^{(t-1)}, i \notin \mathcal{S}^{(t)}$):* In this case, $\mathcal{R}each_2 = \{K_i\}$ as in Case 3, but $\mathbf{reach}^{\Pi}_{\mathcal{S}^{(t)}}(\mathcal{R}each_2) = \{K_i\}$ this time, and so, $\mathcal{R}each_1 \cap \mathcal{K}^{(t)} = \{K_i\} \cap \mathcal{K}^{(t)} = \emptyset$.

The claim, thus, follows from the induction principle. ▌

Next, we use Claim 5.1.2 to prove the following.

Claim 5.1.3 Let $\Pi \in \{\textsf{p-LKH}^+, \textsf{i-LKH}^+\}$. For all $t \geq 0$, for all $t' \geq t$, for every simple sequences $\overrightarrow{\mathcal{S}}^{(t')}$, and for every $i \in \overline{\mathcal{S}}^{(t)}$, $\mathbf{Reach}_{\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t')}}}(i) \cap \mathcal{K}^{(t)} = \emptyset$

Proof of Claim 5.1.3. We prove the claim using induction over t'. The claim is trivially true for $t' = t$ since we know from Claim 5.1.2 that $\mathbf{Reach}_{\overrightarrow{\mathcal{S}}^{(t)}}(i) \cap \mathcal{K}^{(t)} = \emptyset$ for every $i \in \overline{\mathcal{S}}^{(t)}$. Suppose that the claim is true for some arbitrary $t' \geq t$. We prove that it is also true for $t' + 1$ as well.

Let $\overrightarrow{\mathcal{S}}^{(t'+1)}$ be any simple sequence and consider any $i \notin \overline{\mathcal{S}}^{(t)}$. Let $\mathcal{R}each_3 = \mathbf{Reach}_{\mathcal{G}^{\Pi}_{\mathcal{S}^{(t'+1)}}}(i) \setminus \mathbf{Reach}_{\mathcal{G}^{\Pi}_{\mathcal{S}^{(t')}}}(i)$. Our goal is to show that $\mathcal{R}each_3 \cap \mathcal{K}^{(t)} = \emptyset$; this, combined with the inductive hypothesis, would immediately imply Claim 5.1.3. $\mathcal{R}each_3$ can be expressed as follows:

$$\begin{aligned}
\mathcal{R}each_3 &= \mathbf{reach}^{\Pi}_{\mathcal{S}^{(t'+1)}}(\mathbf{Reach}_{\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t')}}}(i)) \setminus \mathbf{Reach}_{\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t')}}}(i) \\
&= \mathbf{reach}^{\Pi}_{\mathcal{S}^{(t'+1)}}(\mathbf{Reach}_{\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t')}}}(i) \cap \mathcal{K}^{(t')}) \cup \{K_i\}) \setminus \mathcal{K}^{(t')}
\end{aligned}$$

These equalities follow from the fact that in $\textsf{p-LKH}^+$ and $\textsf{i-LKH}^+$, every $\textsf{c}$-edge created in the protocol key graph at time $t' + 1$ issues from a key in $\mathcal{K}^{(t')} \cup \mathcal{K}^{(t'+1)}$ or else a long-lived key K_i and is always incident upon a key in $\mathcal{K}^{(t'+1)}$ (or else, upon a key of the form $\tilde{K}_s^{(t'+1)}$ in the case of $\textsf{i-LKH}^+$). Let $\mathcal{R}each_4 = (\mathbf{Reach}_{\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t')}}}(i) \cap \mathcal{K}^{(t')}) \cup \{K_i\}$. As before, four possibilities arise.

- *Case 1 ($i \in \mathcal{S}^{(t')}, i \in \mathcal{S}^{(t'+1)}$):* From Claim 5.1.2, we know that, in this case, $\mathcal{R}each_4 = \mathcal{K}^{(t')}_{\mathbf{pre}_{t'}(i)}$, which means $\mathbf{reach}^{\Pi}_{\mathcal{S}^{(t'+1)}}(\mathcal{R}each_4) = \mathcal{K}^{(t')}_{\mathbf{pre}_{t'}(i)} \cup \mathcal{K}^{(t'+1)}_{\mathbf{pre}_{t'+1}(i)} \cup \tilde{\mathcal{K}}^{(t'+1)}_{\mathbf{pre}_{t'+1}(i)}$, and so $\mathcal{R}each_3 = \mathcal{K}^{(t'+1)}_{\mathbf{pre}_{t'+1}(i)} \cup \tilde{\mathcal{K}}^{(t'+1)}_{\mathbf{pre}_{t'+1}(i)}$. We know that $i \notin \mathcal{S}^{(t)}$ and so, $\mathcal{K}^{(t'+1)}_{\mathbf{pre}(i)}$ must contain keys that are generated *after* time t (possibly when i was first added to the group after time t or possibly even later on). As such, $\mathcal{K}^{(t'+1)}_{\mathbf{pre}_{t'+1}(i)}$ can have no overlap with $\mathcal{K}^{(t)}$. Further, all keys in $\tilde{\mathcal{K}}^{(t'+1)}_{\mathbf{pre}_{t'+1}(i)}$ are generated at time $t' + 1$ and are trivially not contained in $\mathcal{K}^{(t)}$. Therefore, $\mathcal{R}each_3 \cap \mathcal{K}^{(t)}$ must be empty.

- *Case 2 ($i \in \mathcal{S}^{(t')}, i \notin \mathcal{S}^{(t'+1)}$):* Again, $\mathcal{R}each_4 = \mathcal{K}^{(t')}_{\mathbf{pre}_{t'}(i)}$ and so, $\mathbf{reach}^{\Pi}_{\mathcal{S}^{(t'+1)}}(\mathcal{R}each_4) = \mathcal{K}^{(t')}_{\mathbf{pre}_{t'}(i)}$, implying that $\mathcal{R}each_3 = \emptyset$, which, in turn, implies $\mathcal{R}each_3 \cap \mathcal{K}^{(t)} = \emptyset$.

- *Case 3 ($i \notin \mathcal{S}^{(t')}, i \in \mathcal{S}^{(t'+1)}$):* From Claim 5.1.2, we know that, in this case, $\mathcal{R}each_4 = \{K_i\}$, which means $\mathbf{reach}^{\Pi}_{\mathcal{S}^{(t'+1)}}(\mathcal{R}each_4) = \mathcal{K}^{(t'+1)}_{\mathbf{pre}_{t'+1}(i)} \cup \tilde{\mathcal{K}}^{(t'+1)}_{\mathbf{pre}_{t'+1}(i)}$, and so $\mathcal{R}each_3$ equals $\mathcal{K}^{(t'+1)}_{\mathbf{pre}(i)} \cup \tilde{\mathcal{K}}^{(t'+1)}_{\mathbf{pre}_{t'+1}(i)}$ as well. As in case 1, both $\mathcal{K}^{(t'+1)}_{\mathbf{pre}(i)}$ and $\tilde{\mathcal{K}}^{(t'+1)}_{\mathbf{pre}_{t'+1}(i)}$ share no element with $\mathcal{K}^{(t)}$, thus implying that $\mathcal{R}each_3 \cap \mathcal{K}^{(t)} = \emptyset$.
- *Case 4 ($i \notin \mathcal{S}^{(t')}, i \notin \mathcal{S}^{(t'+1)}$):* As in case 3, $\mathcal{R}each_4$ equals $\{K_i\}$, but this time, $\mathbf{reach}^{\Pi}_{\mathcal{S}^{(t'+1)}}(\mathcal{R}each_4) = \{K_i\}$, so, $\mathcal{R}each_3 = \emptyset$, thus implying $\mathcal{R}each_3 \cap \mathcal{K}^{(t)} = \emptyset$.

This concludes the proof of Claim 5.1.3. ▌

Note that Claim 5.1.3 implies that p-LKH$^+$ and i-LKH$^+$ are both strongly secure against single-user attacks (that is, they satisfy Definition 3.2.3). Using Theorem 4.1.1 and the fact that both protocols are S-GKD protocols, we conclude directly that they are strongly collusion-resistant in the symbolic model. ▌

5.2 Subset Cover Protocols

The subset cover method is used for performing group key distribution in various *broadcast encryption* schemes, which in turn, find application in several contexts like secure pay-per-view and copyright protection for digital media. The method was proposed by Naor *et al.* [34], and has, since then, been deployed in multiple broadcast encryption schemes [25, 24, 15, 26]. Subset cover protocols, like LKH protocols, are based on symmetric-key techniques (in particular, they use symmetric-key encryption and PRGs only), but unlike the latter, these schemes implement group key distribution for *stateless* users; that is, legitimate users in a subset cover protocol can recover group keys using only their *initial* state and the instantaneous transmissions of the center. This property makes the protocols applicable in a larger context than LKH protocols (for example, they are better suited for copyright protection mechanisms) but it also reduces their efficiency: subset cover protocols have a greater communication complexity than that of LKH protocols.

At an abstract level, every subset cover protocol has the following structure: Suppose that the number of users in the protocol is $\mathbf{n}$. At the outset, the center forms a collection of sets $\mathcal{C} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_w\}$ such that for each $j \in [w]$, $\mathcal{S}_j$ is a subset of $[\mathbf{n}]$. A long-lived key $\hat{K}_j$ is associated with each subset $\mathcal{S}_j$. (These keys could be purely random keys drawn from $\mathcal{R}$ or else, related to each other via a pseudo-random generator $\mathbf{G}$.) Every user i is given a set of keys $\mathcal{K}_i$ as initial state, such that for each subset $\mathcal{S}_j \ni i$, $\hat{K}_j \in \mathbf{Rec}(\mathcal{K}_i)$.

To distribute a group key to a set $\mathcal{S}^{(t)}$, the center first picks a purely random key $K^{(t)} \in \mathcal{R}$ (such that $K^{(t)}$ has not been used in the protocol before time t) and then computes a set of disjoint subsets $\mathcal{S}_{j_1}, \mathcal{S}_{j_2}, \ldots, \mathcal{S}_{j_l}$ such that

$$\mathcal{S}^{(t)} = \bigcup_{i=1}^{l} \mathcal{S}_{j_i}$$

This set of subsets constitutes a "subset cover" for $\mathcal{S}^{(t)}$. (The collection $\mathcal{C}$ is such that every set $\mathcal{S}^{(t)} \subseteq [\mathbf{n}]$ can be covered using it.) The center then distributes the key $K^{(t)}$ to users in $\mathcal{S}^{(t)}$ by outputting the ciphertexts

$$\mathbf{E}_{\hat{K}_{j_1}}(K^{(t)}), \mathbf{E}_{\hat{K}_{j_2}}(K^{(t)}), \ldots, \mathbf{E}_{\hat{K}_{j_l}}(K^{(t)})$$

This gives us the set of rekey messages $\mathcal{M}^{\Pi}_{\mathcal{S}^{(t)}}$ for time t. Each user $i \in \mathcal{S}^{(t)}$ can recover $K^{(t)}$ by decrypting the ciphertext $\mathbf{E}_{\hat{K}_{j_{i'}}}(K^{(t)})$ for which $i \in \mathcal{S}_{j_{i'}}$ since for any such i', $\hat{K}_{j_{i'}} \in \mathbf{Rec}(\mathcal{K}_i)$. (In a broadcast encryption protocol based on such a protocol, some group data is also transmitted along with the above set of ciphertexts, by suitably encrypting such data under $K^{(t)}$.)

Note that a subset cover protocol does not directly fit the model of group key distribution defined in Chapter 3 since the initial state of every user is not a single (unique) long-lived key but a *set* of keys, and the elements of each set could be shared across users. However, any such protocol can be easily transformed into one that fits our model by including an initial key distribution step in which the center distributes all the long-lived keys; that is, if K_i denotes the individual long-lived key of user i (that corresponding to the subset $\{i\}$), the center in the transformed protocol would transmit the following ciphertexts at time $t = 0$:

$$\bigcup_{i \in [\mathbf{n}]} \{\mathbf{E}_{K_i}(K) \mid K \in \mathcal{K}_i \setminus \{K_i\}\}$$

To prove that a subset cover protocol is correct and secure in the symbolic model, it suffices to check that it satisfies the following two properties:

1. *For all* $i \in [\mathbf{n}]$, $\{\hat{K}_j \mid i \in \mathcal{S}_j\} \subseteq \mathbf{Rec}(\mathcal{K}_i)$. This property guarantees correctness of the protocol in the sense of Definition 3.1.1.

2. *For all* $i \in [\mathbf{n}]$, $\{\hat{K}_j \mid i \notin \mathcal{S}_j\} \cap \mathbf{Rec}(\mathcal{K}_i) = \emptyset$. This property guarantees security of the protocol against single-user attacks, both in the sense of Definition 3.2.1 and of Definition 3.2.3. Furthermore, given Theorem 4.1.1 and the fact that subset cover protocols don't use nested encryption, this also implies their security against collusion attacks, in the sense of definitions 3.2.2 and 3.2.4.

We now present two example subset cover protocols due to Naor *et al.* [34] and verify the above two properties for both of them. Based on this analysis, in Chapter 8, we prove security of the protocols in the computational model as well.

5.2.1 The Complete Subtree Protocol

In the complete subtree (CS) protocol [34], every user is associated with a leaf in a complete binary tree $\mathcal{Tr}$ of height $\lceil \log_2(\mathbf{n}) \rceil$. We label nodes in this tree using the letter u and denote the leaf corresponding to any user $i \in [\mathbf{n}]$ by $u(i)$. The subset collection $\mathcal{C}$ corresponds one-to-one with the set of all nodes in $\mathcal{Tr}$: there exists a subset $\mathcal{S}_u \in \mathcal{C}$ for every node $u \in \mathcal{Tr}$ and for any $i \in [\mathbf{n}]$, $i \in \mathcal{S}_u$ if and only if i is a leaf in the *subtree* of $\mathcal{Tr}$ rooted at u. The extreme subsets are the set $[\mathbf{n}]$ (corresponding to the root of $\mathcal{Tr}$) and the singleton sets $\{1\}, \{2\}, \ldots, \{\mathbf{n}\}$ (corresponding to its leaves). The subset $\mathcal{S}_u$ corresponding to an internal node u contains exactly $2^{h(u)}$ elements where $h(u)$ denotes the height of u in the tree.

The key distribution mechanism works as follows: *(i)* with each subset $\mathcal{S}_u \in \mathcal{C}$ a long-lived key $\hat{K}_u \in \mathcal{R}$ is associated in a manner such that for any $u_1 \neq u_2$, $\hat{K}_{u_1} \neq$

$\hat{K}_{u_2}$; and *(ii)* for any $i \in [\mathbf{n}]$, $\mathcal{K}_i$ contains $\hat{K}_u$ if and only if the subtree rooted at u contains $u(i)$. In other words, for each $i \in [\mathbf{n}]$, $\mathcal{K}_i = \{\hat{K}_u \mid i \in \mathcal{S}_u\}$. (In a sense, the CS protocol is a stateless analogue of the plain-LKH protocol: the keys in the key hierarchy of plain-LKH correspond to the subset keys in CS, but with the difference that these subset keys do not change with time.)

The correctness of the CS protocol (property 1) is obvious. Furthermore, since the long-lived keys are all distinct, there exists no subset $\mathcal{S}_u \not\ni i$ for which $\hat{K}_u \in \mathcal{K}_i$. This implies property 2 above. We conclude that:

Theorem 5.2.1 The CS protocol is strongly secure against collusion attacks in the symbolic model.

In terms of efficiency, the CS protocol incurs a communication cost of at most $q \cdot \lceil \log_2(\mathbf{n}/q) \rceil$ to transmit a group key to any set of size $\mathbf{n} - q$. This is because the smallest sub-collection of $\mathcal{C}$ required to cover any set $\mathcal{S}$ of size $\mathbf{n} - q$ has size at least $q \cdot \lceil \log_2(\mathbf{n}/q) \rceil$. (The optimal subset cover is formed as follows: consider the spanning tree $\mathcal{T}r'$ of the nodes in $\{u(i) \mid i \notin \mathcal{S}\}$ and the root of $\mathcal{T}r$. Consider the set of nodes $u \in \mathcal{T}r \setminus \mathcal{T}r'$ that "hang off" from this tree (that is, nodes for which u is not in $\mathcal{T}r'$ but u's parent node is). Include $\mathcal{S}_u$ in the desired sub-collection. It can be shown, using induction, that the sub-collection thus obtained has size $q \cdot \lceil \log_2(\mathbf{n}/q) \rceil$.) The protocol requires every user to store at most $\lceil \log_2(\mathbf{n}) \rceil$ keys, which is reasonable for most applications.

5.2.2 The Subset Difference Protocol

The subset difference (SD) protocol [34] uses a more involved procedure to form the subset collection $\mathcal{C}$. As in the CS protocol, it first associates every user i with a leaf $u(i)$ in a binary tree $\mathcal{T}r$ of height $\lceil \log_2(\mathbf{n}) \rceil$. For any node u in $\mathcal{T}r$, let $\mathcal{S}_u$ denote the set of all indices i such that $u(i)$ is a leaf in the subtree of $\mathcal{T}r$ rooted at u. For any pair of nodes (u, v) in $\mathcal{T}r$ let $\mathcal{S}_{u,v} = \mathcal{S}_u \setminus \mathcal{S}_v$.

In the SD protocol, the set $\mathcal{C}$ contains all sets $\mathcal{S}_{u,v}$ for which u is a strict ancestor of v in $\mathcal{T}r$, that is, for which u lies on the path from v to the root of $\mathcal{T}r$ and is distinct from v. By definition, for any pair of such nodes (u, v), $\mathcal{S}_u$ is a strict superset of $\mathcal{S}_v$, so $\mathcal{S}_{u,v}$ is always non-empty. Besides sets of this form, the set $[\mathbf{n}]$ is also included in $\mathcal{C}$. It can be shown [34] that to cover any set $\mathcal{S} \subseteq [\mathbf{n}]$ of size $\mathbf{n} - q$, one needs to pick at most $2q - 1$ subsets from $\mathcal{C}$; this feature makes the communication complexity of the SD protocol better than that of the CS protocol.

KEY GENERATION. Long-lived keys in the SD protocol are generated using a pseudo-random generator $\mathbf{G}$. We describe here a slight variant of the process of key generation given in [34]: in [34], a PRG with expansion factor 3 is used, while we simulate the same using a PRG with expansion factor 2.

For any $b_1, b_2 \in \{0, 1\}$, let $\mathbf{G}_{b_1 b_2}(\cdot)$ denote the function $\mathbf{G}_{b_1}(\mathbf{G}_{b_2}(\cdot))$. First, the center generates a purely random key R_u for every node u in the tree $\mathcal{T}r$. Just like in the CS protocol, these keys are all distinct. For any node v and any distinct ancestor u of v in $\mathcal{T}r$, the long-lived key $\hat{K}_{u,v}$ corresponding to the subset $\mathcal{S}_{u,v}$ is derived from R_u. Towards this, every node v in $\mathcal{T}r_u$ (including u itself) is labeled with an "intermediate" key $L_{u,v}$ as follows: let $L_{u,u} = R_u$, and for any node v in $\mathcal{T}r_u$, label its left child $\mathbf{G}_{00}(L_{u,v})$ and its right child $\mathbf{G}_{01}(L_{u,v})$. For any node v in $\mathcal{T}r_u$ such that $v \neq u$, $\hat{K}_{u,v}$ is set to be equal to $\mathbf{G}_{10}(L_{u,v})$.

Note that every key $\hat{K}_{u,v}$ can be derived from R_u using at most $2\lceil \log_2(\mathbf{n}) \rceil$ PRG applications. The key $\hat{K}_{[\mathbf{n}]}$ corresponding to the set $[\mathbf{n}]$ is a purely random key that is distinct from R_u for every $u \in \mathcal{T}r$.

KEY ASSIGNMENT. Every user i is required to be able to recover a long-lived key $\hat{K}_{u,v}$ if and only if $i \in \mathcal{S}_{u,v}$, which, in turn, is true if and only if u is an ancestor of $u(i)$ but v is not. There are $O(\mathbf{n})$ such keys for every i but since some of them are related to each other via the PRG, the set $\mathcal{K}_i$ need not contain a direct representation of each of them.

The key assignment works as follows. For every user i, and every ancestor u

of $u(i)$ in $\mathcal{T}r$, $\mathcal{K}_i$ contains all "intermediate" keys $L_{u,v}$ such that v is not an ancestor of $u(i)$ in $\mathcal{T}r_u$ *but the parent of* v *is*. The key $\hat{K}_{[\mathbf{n}]}$ is also included in $\mathcal{K}_i$. For every user i, the size of the set $\mathcal{K}_i$ thus obtained is $O(\lceil \log_2(\mathbf{n}) \rceil^2)$. It is straightforward to check that for every i, $\mathbf{Rec}(\mathcal{K}_i)$ contains the long-lived key associated with every subset that contains i and is part of $\mathcal{C}$.

Security of the key assignment scheme follows from two important observations. Pick any $i \in [\mathbf{n}]$. Since the purely random keys R_u used to derive all long-lived keys are distinct for distinct values of u, $\mathbf{Rec}(\mathcal{K}_i)$ does not contain any key of the form $\hat{K}_{u,v}$ for which u is *not* an ancestor of $u(i)$. Second, recall that the inverses of the functions $\mathbf{G}_0$ and $\mathbf{G}_1$ cannot be computed via the symbolic recovery function $\vdash$. Given this fact, it is easy to show that $\mathbf{Rec}(\mathcal{K}_i)$ does not contain any key $\hat{K}_{u,v}$ for which u and v are both ancestors of $u(i)$. In effect, $\mathbf{Rec}(\mathcal{K}_i)$ contains no key $\hat{K}_{u,v}$ for which $i \notin \mathcal{S}_{u,v}$. Property 2 for subset cover protocols is thus verified, and we conclude that

Theorem 5.2.2 The SD protocol is strongly secure against collusion attacks in the symbolic model.

In terms of communication complexity, the SD protocol fairs better than the CS protocol: for any target set $\mathcal{S}^{(t)}$ of size $\mathbf{n} - q$, the number of rekey messages transmitted is at most $2q - 1$, and when $q \ll \mathbf{n}$ (which is typically the scenario in broadcast encryption), this is considerably smaller than the cost of rekeying in the CS protocol. However, the protocol requires every user to maintain $O(\lceil \log_2(\mathbf{n}) \rceil^2)$ keys as its state, which is greater than the amount of storage required in the CS protocol. The Layered Subset Difference (LSD) protocol [25] improves this trade-off between communication and storage costs for the SD protocol: it reduces the storage requirements to $O(\lceil \log_2(\mathbf{n}) \rceil^{3/2})$ keys per user while maintaining the cost of rekeying at $O(q)$.

Chapter 6

A Lower Bound

One advantage of using a symbolic model of computation is that it facilitates the task of proving lower bounds on the efficiency of protocols. In this chapter, we use the model to prove a lower bound on the communication complexity of GKD protocols which shows that in any GKD protocol, the number of rekey messages required to be transmitted per group update in order to maintain security against collusion attacks, is at least logarithmic in the group size. The bound applies not only to protocols that use pseudo-random generators and symmetric-key encryption (as modeled by grammar (2.1)), but also to those that employ secret sharing schemes [42], and combine such schemes with encryption and pseudo-random generation in an arbitrary fashion.

Our result establishes the optimality of the LKH protocols presented in the preceding chapter and in particular, it shows that the improved-LKH$^+$ protocol is the most communication-efficient GKD protocol that one can design, while also satisfying security against collusion attacks. In fact, our bound matches the communication efficiency of improved-LKH$^+$ *exactly*, modulo a small additive sub-constant term. Thus, even the use of techniques like nested encryption and secret sharing schemes (neither of which is used in LKH protocols) cannot improve the communication complexity of improved-LKH$^+$ by any reasonable measure.

6.1 Previous Lower Bounds

Lower bounds for the communication complexity of GKD protocols have been explored in several prior works. The first non-trivial communication lower bound for the problem was proven by Canetti, Malkin and Nissim in [13]. This bound applies to a restricted class of protocols, namely protocols where users have a bounded amount of memory, and the key distribution mechanism has a special "structure-preserving" property (as defined in [13]). A different, and seemingly tight, lower bound, for a more general class of protocols without memory or structure restrictions was later proven by Snoeyink, Suri and Varghese [43], who showed that in any GKD protocol (within a certain class), the group center can be forced to transmit at least $3\log_3(n)$ rekey messages per group update (on the average) in order to maintain security of against collusion attacks[1]. A similar result was independently proven by Yang and Lam [48], although the bound in [48] is slightly smaller, namely $\ln(n)$. Snoeyink *et al.* [43] also provide a simple variant of the plain-LKH protocol [46, 45] that meets the established lower bound.

It turns out that despite the close relationship between existing protocols and the lower bounds proven in [48, 43], the class of protocols considered in these works falls short of accommodating all known protocols. The protocol model used in [48, 43] mandates that the group center transmit rekey messages only of the form $\mathbf{E}_{R_1}(R_2)$—the encryption of a random key R_2 under another random key R_1. However, this rules out the possibility of using nested encryption and/or pseudo-random generators, two important cryptographic techniques which are deployed in various GKD protocols in the literature [19, 14, 10, 38, 34, 25, 21].

The inadequacy of the results of [48, 43] is clearly demonstrated by known protocols that "beat" the lower bound proven in [43]. An important example is the improved-LKH$^+$ protocol: it has a communication complexity of $\log_2(n)$, which is *strictly smaller than the* $3\log_3(n) \approx 1.89\log_2(n)$ *bound proven in [43].* This obser-

[1] In [43], security of protocols is analyzed with respect to simple sequences only; that is, it is assumed that at each instant, either a single non-member joins the group or else a single member leaves it.

vation highlights the limited applicability of the results of [48, 43], and leaves open the possibility of designing protocols with communication efficiency better than $\log_2(n)$ using cryptographic techniques not considered in either of those works. The question we are interested in is the following:

Can we design GKD *protocols that use nested encryption and/or pseudo-random generators, are secure against collusion attacks, and have communication complexity smaller than* $\log_2(n)$*?*

6.2 Our Result

We answer the above question in the negative. We show that all collusion-resistant GKD protocols that use symmetric-key encryption, pseudo-random generators, *and even secret sharing schemes* in generating rekey messages must incur a communication cost of at least $\log_2(n)$ per group update operation in the worst case. More precisely, we exhibit an adversarial strategy for performing group update operations that forces the center in any collusion-resistant GKD protocol to transmit at least $\log_2(n) - \delta$ messages per update on the average, with n being the size of the group and δ a quantity that tends to 0 as the number of updates increases.

The adversarial strategy we use involves *replacing* an existing member of the group with a non-member at every instant. In other words, we consider input sequences of the following form:

Definition 6.2.1 A sequence of member-sets $\overrightarrow{\mathcal{S}}^{(t)} = (\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(t)}) \in (2^{[\mathbf{n}]})^t$ is called a *replace-only* sequence if for each $t' \in \{1, \ldots, t-1\}$, $\mathcal{S}^{(t'+1)} = \mathcal{S}^{(t')} \cup \{i\} \setminus \{j\}$ for some $i \in \overline{\mathcal{S}}^{(t')}$ and $j \in \mathcal{S}^{(t')}$.

Analyzing protocols against such sequences simplifies the task of proving the lower bound since it ensures that the size of the group remains constant across time. It is easy to see that the improved-LKH$^+$ protocol can be modified suitably so that it incurs a communication cost of $\lceil \log_2(n) \rceil$ rekey messages (per update operation) against replace-only sequences.

Before we present our result, we first describe how we extend the model of group key distribution defined in Chapter 3 to incorporate secret sharing schemes.

6.2.1 An Extended Symbolic Model

SECRET SHARING. Let n be any integer. An n-wise secret sharing function is a function that takes a message M as input and outputs n expressions, denoted $\mathbf{S}_1(M), \ldots, \mathbf{S}_{\mathsf{n}}(M)$, which are referred to as the *shares* corresponding to M. With any such function is associated an *access structure* Γ which is a set of subsets of $[\mathsf{n}]$; that is $\Gamma \subseteq 2^{[\mathsf{n}]}$. The function $\mathbf{S}$ is defined in such a way that given the set of shares $\mathbf{S}_{\mathcal{I}}(M) = \{\mathbf{S}_i(M)\}_{i \in \mathcal{I}}$ for any $\mathcal{I} \in \Gamma$, the message M can be easily recovered. A typical example of secret sharing schemes is k-out-of-n secret sharing, where the access structure is the set of all subsets of $\{1, \ldots, \mathsf{n}\}$ of size at least k, that is, a secret message can be recovered if and only if at least k of its shares are known.

The efficiency of secret sharing schemes makes them excellent tools to be used in conjunction with symmetric-key encryption and pseudorandom generators. (For example, "n out of n" sharing, which is a special case of "k out of n" sharing, can be implemented using only the XOR operation.) Secret sharing schemes have been used in various security protocols of practical interest, and, in particular, they have been used in the design of some broadcast encryption protocols [35].

THE MODIFIED GRAMMAR. Let $\mathbf{S}$ be any n-wise secret sharing function with arbitrary access structure $\Gamma \subseteq 2^{[\mathsf{n}]}$. We extend the grammar of protocol messages defined in Chapter 2 as follows:

$$\begin{aligned} M &\rightarrow K \mid \mathbf{E}_K(M) \mid \mathbf{S}_1(M) \mid \cdots \mid \mathbf{S}_{\mathsf{n}}(M) \\ K &\rightarrow \mathcal{R} \mid \mathbf{G}_0(K) \mid \mathbf{G}_1(K) \end{aligned} \tag{6.1}$$

For example, expressions of the form $\mathbf{S}_1(\mathbf{S}_2(R_1))$ and $\mathbf{E}_{R_2}(\mathbf{E}_{\mathbf{G}_0(R_3)}(\mathbf{S}_2(R_5)))$ can be derived from the variable M above. Every expression derivable from M is referred to as a protocol message. Note that our grammar allows shares to be created by applying the sharing function iteratively on both keys and ciphertexts. Shares can be encrypted under multiple keys but cannot themselves be used to encrypt other messages or to generate pseudorandom keys. Although we restrict shares from being used in this manner, we remark that our lower bound also applies to protocols that do use shares for encryption and pseudorandom generation. We focus on the above class of protocols mainly for simplicity of exposition.

The semantics of the sharing function is defined by suitably modifying the definition of the entailment relation $\vdash$. Recall Rule 0, Rule 1 and Rule 2 used to define $\vdash$ in Section 2.4 of Chapter 2. Besides these three rules, we also include the following rule:

$$\exists \mathcal{I} \in \Gamma.\ \left[\forall i \in \mathcal{I}.\ \Big(\mathcal{M} \vdash \mathbf{S}_i(M) \Big) \right] \implies \mathcal{M} \vdash M \qquad \text{(Rule 3)}$$

For any message-set $\mathcal{M}$, $\mathbf{Rec}(\mathcal{M})$ now denotes the set of all messages that are recoverable from $\mathcal{M}$ using Rule 0, Rule 1, Rule 2 as well as Rule 3. A message M is said to be recoverable from $\mathcal{M}$ *in i steps* (for some $i \geq 0$) if M can be derived from $\mathcal{M}$ using i applications of Rule 1, Rule 2 or Rule 3.

6.2.2 The Result

We consider GKD protocols in which every rekey message transmitted by the center is an expression derived from grammar (6.1). For the rest of this chapter, every usage of the term "GKD protocol" refers to a protocol of this kind. For any set $\mathcal{S}$, we use $|\mathcal{S}|$ to denote the size of $\mathcal{S}$. The following is the main result of the chapter.

Theorem 6.2.2 For any $\mathbf{n}$-user GKD protocol Π that is secure against collusion attacks (satisfies Definition 3.2.2), and for any n, t such that $n + t \leq \mathbf{n}$, there exists a replace-only sequence $\overrightarrow{\mathcal{S}}^{(t)} = (\mathcal{S}^{(1)}, \mathcal{S}^{(2)}, \ldots, \mathcal{S}^{(t)}) \in (2^{[\mathbf{n}]})^t$ such that $|\mathcal{S}^{(1)}| = n$, and

$$|\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}| \ \geq \ (t-1) \cdot \lceil \log_2(n) \rceil$$

From the theorem, it follows that the amortized communication cost that any collusion-resistant GKD protocol incurs is at least $(\frac{t-1}{t}) \cdot \lceil \log_2(n) \rceil = (1 - \frac{1}{t}) \cdot \lceil \log_2(n) \rceil$

in the worst case. This quantity equals $\lceil \log_2(n) \rceil - o(1)$ for $t \gg \log_2(n)$, and so the asymptotic communication complexity of the protocol is bounded from below by $\lceil \log_2(n) \rceil$. Note that the theorem applies to protocols that are collusion-resistant in the sense of Definition 3.2.2, which is weaker than Definition 3.2.4. However, since we are proving a lower bound, the same result also holds for protocols that are secure in the stronger sense (those that satisfy Definition 3.2.4) as well. Some important remarks are in order:

- It is imperative to consider the amortized communication complexity of GKD protocols when proving a lower bound since in any protocol, the rekeying cost incurred for a single membership update can be distributed across multiple instants, while still maintaining security and correctness. Particularly, rekey messages required to be sent in response to the tth update command could instead be sent before time t, thus lowering the rekeying cost for time t arbitrarily. Previous lower bounds [48, 43] in the literature are also amortized lower bounds[2].

- Our lower bound, like that of [48, 43], is a *worst-case* lower bound for GKD protocols, proven with respect to an adversarially-chosen sequence of membership update operations. Worst-case analysis of these protocols (and of cryptographic protocols, in general) is essential from the perspective of security since we would like all protocols to be designed so that they remain secure in any possible execution environment. Analyzing protocols in the worst case is the standard approach to proving their security in the cryptography literature.

We present the proof of Theorem 6.2.2 in two steps. In the first step, we show that a similar result holds for S-GKD protocols (those that use single encryption and pseudo-random generation only and do not deploy secret sharing schemes). This step highlights the key idea underlying the proof of Theorem 6.2.2 and gives intuition about

[2] In [44], a lower bound of $\log_2(n)$ is proven on the rekeying cost for a specific class of GKD protocols (called *key graph protocols* in [44]), but without amortizing over sequences of update operations. However, the result of [44] is incomparable to ours: although it proves a non-amortized lower bound, it does so for a very restricted type of protocols. In particular, the protocol class considered in [44] is much smaller than that considered in all previous works on lower bounds on GKD protocols [13, 48, 43].

how the $\log_2(n)$ bound is reached. In the second step, we extend the result from the first step to protocols that also use nested encryption and secret sharing.

6.3 A First Step Towards the Proof

Consider the class of S-GKD protocols, that is, protocols in which every rekey message transmitted by the center is derived using grammar (4.1). For this class of protocols, we can prove essentially the same result as Theorem 6.2.2.

Theorem 6.3.1 For any $\mathbf{n}$-user S-GKD protocol Π that is secure against single-user attacks (satisfies Definition 3.2.1), and for any n, t such that $n + t \leq \mathbf{n}$, there exists a replace-only sequence $\overrightarrow{\mathcal{S}}^{(t)} = (\mathcal{S}^{(1)}, \mathcal{S}^{(2)}, \ldots, \mathcal{S}^{(t)}) \in (2^{[\mathbf{n}]})^t$ such that $|\mathcal{S}^{(1)}| = n$, and

$$|\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}| \;\geq\; (t-1) \cdot \lceil \log_2(n) \rceil$$

Since security against single-user attacks is equivalent to collusion-resistance for S-GKD protocols (Theorem 4.1.1), the above bound applies to S-GKD protocols that are collusion-resistant as well.

Before we prove Theorem 6.3.1, we develop some terminology that is used in the proofs of both Theorem 6.2.2 and Theorem 6.3.1. Let Π be any $\mathbf{n}$-user GKD protocol that satisfies Definition 3.2.2 (security against collusion attacks) and $\overrightarrow{\mathcal{S}}^{(t)}$ any sequence given as input to Π. The keys used in the protocol when executed on input $\overrightarrow{\mathcal{S}}^{(t)}$ can be partitioned into two classes—those that are recoverable by non-members at time t and those that are not.

Definition 6.3.2 A key K is *useless* at time t if it can be recovered from the keys of non-members at that time, that is, if $K \in \mathbf{Rec}(\{K_i\}_{i \in \overline{\mathcal{S}}^{(t)}} \bigcup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}})$. It is *useful* otherwise.

In the special case of S-GKD protocols, usefulness of a key K can alternatively be defined by requiring that K not be contained in $\mathbf{Rec}(\{K_i\} \bigcup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}})$ for every $i \in \overline{\mathcal{S}}^{(t)}$. Note that for all t, and all sequences $\overrightarrow{\mathcal{S}}^{(t)}$, the group key $K^{(t)}$ and the long-lived keys of members, $\{K_i\}_{i \in \mathcal{S}^{(t)}}$, must be in $\mathcal{U}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ for otherwise the collusion-resistance property of the protocol would be violated.

Usefulness is defined for rekey messages as well, based on the plaintext key they encapsulate. Formally, we say that a message M *encapsulates* a key K if M is obtained by applying a (possibly empty) sequence of encryption and sharing operations to K; in other words, M equals $e_1(e_2(\cdots e_l(K)\cdots))$ for some $l \geq 0$, where each e_i is either equal to $\mathbf{E}_K$ (for some key K) or $\mathbf{S}_j$ (for some $j \in \{1, \cdots, \mathsf{n}\}$). For example, the messages R_3, $\mathbf{E}_{R_1}(R_3)$ and $\mathbf{E}_{R_1}(\mathbf{S}_1(R_3))$ all encapsulate R_3 and the message $\mathbf{E}_{\mathbf{G}_1(R_1)}(\mathbf{S}_1(\mathbf{E}_{R_2}(\mathbf{G}_0(R_3))))$ encapsulates $\mathbf{G}_0(R_3)$.

Definition 6.3.3 A message M is *useful* (resp. *useless*) at time t if M encapsulates a key that is useful (resp. useless) at that time.

For example, if a key K is recoverable by the non-members at time t, then any rekey message of the form $\mathbf{E}_{K'}(K)$ sent up to that time would be useless. Note that a message could be useless even when it is not decipherable by the non-members. For example, if K is recoverable by the non-members at time t, but K' is not, then the message $\mathbf{E}_{K'}(K)$ is useless, even if it cannot be decrypted by the non-members.

Usefulness is a dynamic concept: keys and messages that are useful at one instant may be useless at another instant. Every time a member is removed from the group (or replaced with a non-member), all useful keys known to that member (including the group key) turn useless, and, as a result, all messages encapsulating these keys that were transmitted in the past also become useless.

The intuition underlying the proofs of Theorems 6.2.2 and 6.3.1 is the following: We develop a strategy for replacing members in such a way that after the execution of every replace operation, "sufficiently" many—in particular, logarithmically many—rekey messages become useless. The messages that become useless at any instant t could have been sent at *any* other instant $t' < t$ (and not necessarily at time $t - 1$); all that is relevant for the proof is that there are logarithmically many such messages for every t. In order to cope with so many messages turning useless, the protocol must, on the average, transmit logarithmically many rekey messages at every instant, thus implying that the amortized communication cost it incurs must be logarithmic, too.

Proof of Theorem 6.3.1. The proof exploits the notion of key graphs defined for S-GKD protocols in Chapter 3. For any key K and any key graph $\mathcal{G}$, let $Indegree_{\mathcal{G}}(K)$ denote the in-degree of K in $\mathcal{G}$ and c-$Indegree_{\mathcal{G}}(K)$ the number of c-edges incident upon it in the graph. The following proposition formalizes an important property of key graphs:

Proposition 6.3.4 For any key graph $\mathcal{G}$ and any key K in it, c-$Indegree_{\mathcal{G}}(K) \geq Indegree_{\mathcal{G}}(K) - 1$.

Proof: The semantics of our symbolic security model dictates that every key used in the protocol be obtainable from a unique seed, and using a unique sequence of zero or more PRG computations. In the context of key graphs, this has the implication that every key in a key graph $\mathcal{G}$ must have at most one g-edge incident upon it. (Purely random keys have no g-edge incident upon them while pseudorandom ones have one g-edge per key.) The proposition follows from this. ▌

Let Π be any $\mathbf{n}$-user S-GKD protocol that is secure against single-user attacks and let n, t be integers such that $n + t \leq \mathbf{n}$. For any sequence $\overrightarrow{\mathcal{S}}^{(t)}$, consider the key graph $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ corresponding to the execution of Π on input $\overrightarrow{\mathcal{S}}^{(t)}$. We know that for any user $i \in \overrightarrow{\mathcal{S}}^{(t)}$, the group key $K^{(t)}$ is recoverable from $\{K_i\} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$, and also that $K^{(t)}$ is not recoverable from $\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ alone (for otherwise Π would be insecure). Using Lemma 4.2.1 then, we conclude that for every $i \in \overrightarrow{\mathcal{S}}^{(t)}$, $K^{(t)}$ is reachable from K_i in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$.

For any $i \in \overrightarrow{\mathcal{S}}^{(t)}$, let $P_i^{(t)}$ be a path in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ that goes from K_i to $K^{(t)}$. Let $\tilde{\mathcal{G}}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ be a sub-graph of $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ formed by taking the union of $P_i^{(t)}$ for each $i \in \mathcal{S}^{(t)}$. That is, an edge is in $\tilde{\mathcal{G}}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ if and only if it is contained in $P_i^{(t)}$ for some $i \in \mathcal{S}^{(t)}$. While taking the union of these paths, priority is given to c-edges: if there exists a c-edge from K to K' in $P_i^{(t)}$ (for some $i \in \mathcal{S}^{(t)}$), and a g-edge between the same two keys in $P_j^{(t)}$ (for some $j \in \mathcal{S}^{(t)} \setminus \{i\}$), then $\tilde{\mathcal{G}}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ contains a c-edge from K to K'. We refer to $\tilde{\mathcal{G}}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$

as the *member key graph* for time t.

Claim 6.3.5 For any t, and any sequence $\overrightarrow{\mathcal{S}}^{(t)} = (\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(t)}) \in (2^{[\mathbf{n}]})^t$, every key in the graph $\tilde{\mathcal{G}}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ is useful at time t.

Proof: For every key K in the graph $\tilde{\mathcal{G}}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$, the group key $K^{(t)}$ is reachable from K, and so, $K^{(t)} \in \mathbf{Rec}(\{K\} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}})$. Suppose there exists a key K in $\tilde{\mathcal{G}}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ such that K is useless at time t. Then $K \in \mathbf{Rec}(\{K_i\}_{i \in \overline{\mathcal{S}}^{(t)}} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}})$, which implies that $K^{(t)}$, which is contained in $\mathbf{Rec}(\{K\} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}})$, is also in $\mathbf{Rec}(\{K_i\}_{i \in \overline{\mathcal{S}}^{(t)}} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}})$. But this would mean that $K^{(t)}$ is useless at time t, and this, we know, is not possible. The claim follows. ▌

Claim 6.3.6 For any t, any sequence $\overrightarrow{\mathcal{S}}^{(t)} = (\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(t)}) \in (2^{[\mathbf{n}]})^t$, and any $i \in \mathcal{S}^{(t)}$, $Indegree_{\tilde{\mathcal{G}}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}}(K_i) = 0$.

Proof: Fix t and $\overrightarrow{\mathcal{S}}^{(t)}$ and pick any $i, j \in \mathcal{S}^{(t)}$ such that $i \neq j$. We claim that the key K_j does not lie on the path $P_i^{(t)}$. Suppose this is not the case; that is, suppose there exists $i, j \in \mathcal{S}^{(t)}$ ($i \neq j$) such that K_j lies on the path $P_i^{(t)}$. Then, $K_j \in \mathbf{Rec}(\{K_i\} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t')}})$ for any $t' \geq t$. Let $\mathcal{S}^{(t+1)} = \mathcal{S}^{(t)} \setminus \{i\}$ and consider the execution of Π on input $\overrightarrow{\mathcal{S}}^{(t+1)} = (\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(t)}, \mathcal{S}^{(t+1)})$. From the correctness of the protocol, we know that the group key created for time $t+1$, $K^{(t+1)}$, is contained in $\mathbf{Rec}(\{K_j\} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t+1)}})$, which, in turn, implies that $K^{(t+1)} \in \mathbf{Rec}(\{K_i\} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t+1)}})$. But this means that Π is insecure against single-user attacks, a contradiction!

Thus, there exists no distinct $i, j \in \mathcal{S}^{(t)}$ for which K_j lies on the path $P_i^{(t)}$, which means that for each $i \in \mathcal{S}^{(t)}$, $Indegree_{\tilde{\mathcal{G}}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}}(K_i) = 0$. ▌

The figure shows an example member key graph for three members i_1, i_2, i_3 and their corresponding paths $P_{i_1}^{(t)}$, $P_{i_2}^{(t)}$ and $P_{i_3}^{(t)}$. In the figure, c-edges have been shown with solid lines and g-edges with dashed lines. The paths $P_{i_1}^{(t)}$ and $P_{i_2}^{(t)}$ share a common edge $K' \rightarrow K^{(t)}$.

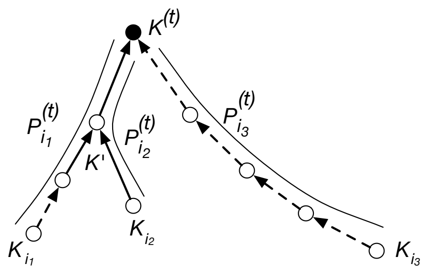

Figure 6.1: A member key graph.

For the proof of Theorem 6.3.1, we need to count the number of c-edges incident upon keys on a path $P_i^{(t)}$ for some member i at time t. To do this, we introduce the concept of the c-edge *weight* of a path.

Definition 6.3.7 Let $t > 0$, $\overrightarrow{\mathcal{S}}^{(t)} = (\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(t)}) \in (2^{[\mathbf{n}]})^t$ and $i \in \mathcal{S}^{(t)}$. The c-edge *weight* of the path $P_i^{(t)}$, denoted $\mathsf{c}\text{-}\mathit{Weight}(P_i^{(t)})$, is defined as:

$$\mathsf{c}\text{-}\mathit{Weight}(P_i^{(t)}) = \sum_{K \in P_i^{(t)}} \mathsf{c}\text{-}\mathit{Indegree}_{\tilde{\mathcal{G}}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}}(K)$$

We are now ready to give the replace-only sequence required to prove Theorem 6.3.1. Define a sequence $\overrightarrow{\mathcal{S}}^{(t)} = (\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(t)})$ as follows. Let $\mathcal{S}^{(1)}$ be any arbitrary subset of $[\mathbf{n}]$ of size n. For each $t' \in \{2, \ldots, t\}$, let $\mathcal{S}^{(t')} = \mathcal{S}^{(t'-1)} \cup \{i_{t'}\} \setminus \{j_{t'}\}$ for any $i_{t'}, j_{t'} \in [\mathbf{n}]$ such that

$$\begin{aligned} i_{t'} &\notin \bigcup_{t''=1}^{t'-1} \mathcal{S}^{(t'')} \\ j_{t'} &= \arg\max_{j \in \mathcal{S}^{(t'-1)}} \left\{ \mathsf{c}\text{-}\mathit{Weight}(P_j^{(t'-1)}) \right\} \end{aligned}$$

In words, at every instant t', we replace an existing member $j_{t'} \in \mathcal{S}^{(t'-1)}$ for which the path $P_{j_{t'}}^{(t'-1)}$ has *maximum* c-edge weight (relative to all paths $P_j^{(t'-1)}$ for $j \in \mathcal{S}^{(t'-1)}$) with a non-member $i_{t'}$ such that $i_{t'}$ was *never* a member before time t'.

We claim that for any such sequence $\overrightarrow{\mathcal{S}}^{(t)}$, the number of rekey messages output by Π when given $\overrightarrow{\mathcal{S}}^{(t)}$ as input is $(t-1) \cdot \lceil \log_2(n) \rceil$. Consider the execution of Π on input $\overrightarrow{\mathcal{S}}^{(t)}$. For each instant $t' > 1$ during the execution, consider the set of keys

$\mathcal{K}_{j_{t'}}$ that lie on the path $P_{j_{t'}}^{(t'-1)}$ where $j_{t'}$ is the member that is removed at time t'. There are two important facts about this set:

- *All keys in $\mathcal{K}_{j_{t'}}$ are useful at time $t'-1$:* This follows from Claim 6.3.5.
- *All keys in $\mathcal{K}_{j_{t'}}$ are useless at time t':* This is because every key $K \in \mathcal{K}_{j_{t'}}$ is reachable from K_j in the key graph at time $t'-1$, that is, $K \in \mathbf{Rec}(\{K_j\} \cup \mathcal{M}^{\Pi}_{\vec{\mathcal{S}}^{(t'-1)}}) \subseteq \mathbf{Rec}(\{K_j\} \cup \mathcal{M}^{\Pi}_{\vec{\mathcal{S}}^{(t')}})$. Since $j_{t'} \in \overline{\mathcal{S}}^{(t')}$, this implies that $K \in \mathbf{Rec}(\{K_i\}_{i \in \overline{\mathcal{S}}^{(t')}} \cup \mathcal{M}^{\Pi}_{\vec{\mathcal{S}}^{(t')}})$ as well.

This means that all rekey messages that encapsulate keys in $\mathcal{K}_{j_{t'}}$ are useful at time $t'-1$ but useless at time t'. Since each rekey message is represented uniquely as a c-edge, the number of such rekey messages is at least equal to the c-edge weight of the path $P_{j_{t'}}^{(t'-1)}$. Furthermore, since $j_{t'}$ is not included in $\mathcal{S}^{(t'')}$ for any $t'' \geq t'$, *each such rekey message remains useless up to time t.*

The number of rekey messages output by Π till time t is at least equal to the number of useful rekey messages output till that time. The latter is, in turn, at least equal to the number of useful rekey messages "that become useless" at any instant from 1 through t. It follows that

$$|\mathcal{M}^{\Pi}_{\vec{\mathcal{S}}^{(t)}}| \geq \sum_{t'=2}^{t} \mathsf{c}\text{-}\mathit{Weight}(P_{j_{t'}}^{(t'-1)}) \tag{6.2}$$

Next, we argue that $\mathsf{c}\text{-}\mathit{Weight}(P_{j_{t'}}^{(t'-1)})$ is at least $\lceil \log_2(n) \rceil$ for each $t' \in \{2, \ldots, t\}$. To make the argument, we introduce a concept similar to that of c-edge weight, but one that is applicable to arbitrary graphs (not necessarily key graphs). From Proposition 6.3.4, we know that for any key graph $\mathcal{G}$ and any key K, $\mathsf{c}\text{-}\mathit{Indegree}_{\mathcal{G}}(K) \geq \mathit{Indegree}_{\mathcal{G}}(K) - 1$. This means that the c-edge weight of any path in a key graph is at least equal to the number of edges that are incident upon keys lying on the path but that are not included in the path. We refer to this number as the *in-degree* of the path.

Definition 6.3.8 Let $\mathcal{G}$ be a directed graph and let P be a path in $\mathcal{G}$ that starts from a node s. The *in-degree* of P in $\mathcal{G}$, denoted $\mathit{Indegree}_{\mathcal{G}}(P)$, is the number of edges in $\mathcal{G}$

that start from a node not in P and are incident upon one that is in P. That is,

$$\mathit{Indegree}_{\mathcal{G}}(P) \quad = \quad \mathit{Indegree}_{\mathcal{G}}(s) + \sum_{v \in P; v \neq s} (\mathit{Indegree}_{\mathcal{G}}(v) - 1)$$

Recall that in any member key graph, the in-degree of any long-lived key in the graph is always zero (Claim 6.3.6). As a result, for any $t' \in \{2, \ldots, t\}$, the c-edge weight of the path $P_{j_{t'}}^{(t'-1)}$ (where $j_{t'}$ is the label of the member removed from the group at time t') is no less than the in-degree of $P_{j_{t'}}^{(t'-1)}$ in $\tilde{\mathcal{G}}^{\Pi}_{\vec{\mathcal{S}}^{(t'-1)}}$. To bound the c-edge weight of $P_{j_{t'}}^{(t'-1)}$ from below, it suffices to bound its in-degree instead.

Lemma 6.3.9 Let $\mathcal{G}$ be an arbitrary directed graph over a set of nodes $\mathcal{V}$ and let $\{v_1, \cdots, v_n, v\}$ be any subset of $\mathcal{V}$ such that for each $i \in \{1, \cdots, n\}$, there exists a path P_i from v_i to v and there exists no j ($j \neq i$) such that v_i occurs in P_j. Then, there exists an $i \in \{1, \cdots, n\}$ such that $\mathit{Indegree}_{\mathcal{G}}(P_i) \geq \lceil \log_2(n) \rceil$

Using the lemma and the fact that long-lived keys always have in-degree zero in member key graphs, we obtain that for every $t' \in \{2, \ldots, t\}$, the in-degree of the path $P_{j_{t'}}^{(t'-1)}$ in the graph $\tilde{\mathcal{G}}^{\Pi}_{\vec{\mathcal{S}}^{(t'-1)}}$ is at least $\lceil \log_2(n) \rceil$. Therefore, the c-edge weight of each such $P_{j_{t'}}^{(t'-1)}$ is at least $\lceil \log_2(n) \rceil$, and invoking equation (6.2), we conclude that the size of $\mathcal{M}^{\Pi}_{\vec{\mathcal{S}}^{(t)}}$ is at least $(t-1) \cdot \lceil \log_2(n) \rceil$.

To complete the proof of Theorem 6.3.1, it suffices to prove Lemma 6.3.9.

6.3.1 Proof of Lemma 6.3.9

The proof of this lemma involves an inductive argument. We perform induction over n, and show that for all n, all subsets $\{v_1, \cdots, v_n, v\}$ of $\mathcal{V}$ and all sets of paths $\{P_1, \cdots, P_n\}$ satisfying the conditions in the lemma, there exists an $i \in \{1, \cdots, n\}$ such that $\mathit{Indegree}_{\mathcal{G}}(P_i) \geq \lceil \log_2(n) \rceil$. We treat paths as sequences of nodes and for any two paths P and Q, $P \cdot Q$ denotes the path (that is, the sequence of nodes) formed by concatenating P and Q.

For $n = 1$, the lemma is trivially true. The path from v_1 to v, P_1, is the only path under consideration and it has in-degree equal to $\lceil \log_2(1) \rceil = 0$.

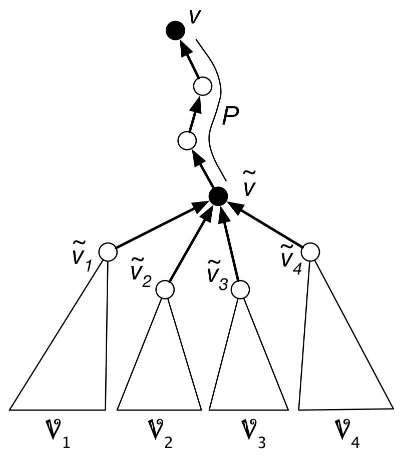

Figure 6.2: Illustration for the proof of Lemma 6.3.9.

We hypothesize that for some $\tilde{n} \geq 2$, the statement of the lemma is true for all values of n less than $\tilde{n}$. That is, for all $n < \tilde{n}$, all sets of nodes $\{v_1, \cdots, v_n\} \subseteq \mathcal{V}$ and all sets of paths $\{P_1, \cdots, P_n\}$ such that for all $i \leq n$ (a) P_i is a path from v_i to v and (b) v_i does not lie on P_j for any $j \neq i$, we can find an $i \in \{1, \cdots, n\}$ such that $\mathit{Indegree}_{\mathcal{G}}(P_i) \geq \lceil \log_2(n) \rceil$.

Consider any subset of $\mathcal{V}$, $\{v_1, \cdots, v_{\tilde{n}}, v\}$, such that there exist paths $\{\tilde{P}_1, \cdots, \tilde{P}_{\tilde{n}}\}$, each path $\tilde{P}_i$ going from v_i to v and no v_i lying on $\tilde{P}_j$ for $j \neq i$. Without loss of generality, we assume that all these paths are loop-free. (The existence of a path with loops implies the existence of one without loops between the same two nodes.) Let P be the largest common suffix of $\tilde{P}_1, \cdots, \tilde{P}_{\tilde{n}}$; that is, P is the longest path such that for each $\tilde{P}_i$, there exists a path $\tilde{Q}_i$ for which $\tilde{P}_i = \tilde{Q}_i \cdot P$. Let $\tilde{v}$ be the first node in P. Since we assumed the $\tilde{P}_i$'s to be loop-free, there is exactly one path $\tilde{Q}_i$ for each $\tilde{P}_i$ such that $\tilde{P}_i = \tilde{Q}_i \cdot P$. We partition the set $\{\tilde{Q}_1, \cdots, \tilde{Q}_{\tilde{n}}\}$ based on the last node on these paths; that is, for each set $\mathbf{Q}_j$ in this partition, the last node on each of the $\tilde{Q}_i$'s in $\mathbf{Q}_j$, is the same, say $\tilde{v}_j$. Let d be the size of this partition, which, given the maximality of the suffix P, must be at least 2. Corresponding to the partition of the $\tilde{Q}_i$'s, the nodes $v_1, \cdots, v_{\tilde{n}}$ can also be partitioned into d sets. Let $\mathcal{V}_1, \cdots, \mathcal{V}_d$ be these sets. The figure shows an example with $d = 4$.

Since the total number of nodes in $\mathcal{V}_1, \cdots, \mathcal{V}_d$ put together is $\tilde{n}$, there must exist some $j \in \{1, \cdots, d\}$ such that $\mathcal{V}_j$ has size at least $\lceil \frac{\tilde{n}}{d} \rceil$. Let $\tilde{\mathcal{G}}$ be the graph formed by taking the union of the $\tilde{Q}_i$'s corresponding to such a $\mathcal{V}_j$. The set of paths $\{\tilde{Q}_i\}_{v_i \in \mathcal{V}_j}$ has the property that (a) each $\tilde{Q}_i$ is a path from v_i to $\tilde{v}_j$ and (b) no v_i lies on a path $\tilde{Q}_j$ for $j \neq i$. From the inductive hypothesis, then, there must exist an index $i = i_{\max}$ ($v_{i_{\max}} \in \mathcal{V}_j$) such that $\mathit{Indegree}_{\tilde{\mathcal{G}}}(\tilde{Q}_{i_{\max}})$ is at least $\lceil \log_2(\lceil \frac{\tilde{n}}{d} \rceil) \rceil$.

Now, the in-degree of path $\tilde{P}_{i_{\max}}$ is at least equal to the sum of the in-degree of $\tilde{Q}_{i_{\max}}$ and $(\mathit{Indegree}_{\mathcal{G}}(\tilde{v}) - 1)$. (This is because $\tilde{v}$ lies on $\tilde{P}_{i_{\max}}$ but not on $\tilde{Q}_{i_{\max}}$.) This sum can be bounded from below as:

$$\begin{aligned} \mathit{Indegree}_{\mathcal{G}}(\tilde{P}'_i) + \mathit{Indegree}_{\tilde{\mathcal{G}}}(\tilde{v}) - 1 &\geq \lceil \log_2(\lceil \frac{\tilde{n}}{d} \rceil) \rceil + d - 1 \\ &\geq \lceil \log_2(\frac{\tilde{n}}{d}) \rceil + d - 1 \\ &= \lceil \log_2(\tilde{n}) - \log_2(d) \rceil + d - 1 \\ &\geq \lceil \log_2(\tilde{n}) \rceil - \log_2(d) + d - 1 \\ &\geq \lceil \log_2(\tilde{n}) \rceil \end{aligned}$$

The last inequality holds because for all integers $d \geq 1$, $d - 1 \geq \log_2(d)$. ▮

6.4 Completing the Proof

For the proof of Theorem 6.2.2, we first suitably extend the notion of key graphs so that it is applicable to arbitrary GKD protocols.

Let Π be any $\mathbf{n}$-user GKD protocol that is secure against collusion attacks and let n, t be integers such that $n + t \leq \mathbf{n}$. For any sequence $\overrightarrow{\mathcal{S}}^{(t)}$ given as input to Π, we define the key graph $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ corresponding to the execution of Π with that input as follows. The nodes in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ are the keys in the protocol that are useful at time t. For any two useful keys K, K' such that $K = \mathbf{G}_b(K')$ for some $b \in \{0, 1\}$, we introduce a g-edge from K' to K just as before. For every useful message $M \in \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ that encapsulates a key K, we introduce *at most* one c-edge in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ using the following rule:

- If there is a sub-expression of M of the form $\mathbf{E}_{K'}(M')$ such that K' is useful, a c-edge corresponding to M *is* introduced in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$. In particular, we locate the sub-expression of M of the form $\mathbf{E}_{K'}(M')$ such that (a) K' is useful, and (b) M' does not contain a sub-expression $\mathbf{E}_{K''}(M'')$ for which K'' is also useful, and introduce the c-edge $K' \rightarrow K$ in the graph. Intuitively, such a K' corresponds to the inner-most useful encryption key in M.

- Otherwise, we introduce no c-edge corresponding to M in the graph.

For example, a rekey message of the form $\mathbf{E}_{R_1}(\mathbf{E}_{R_2}(R_3))$, would get mapped to $R_2 \rightarrow R_3$ if R_2 and R_3 were useful, and to $R_1 \rightarrow R_3$ if R_1 and R_3 were useful but R_2 was not. Similarly, if R_1, R_2 and R_3 were all useful, then the message $\mathbf{E}_{R_1}(\mathbf{S}_1(\mathbf{E}_{R_2}(R_3)))$ and $\mathbf{E}_{R_1}(\mathbf{E}_{R_2}(\mathbf{S}_2(R_3)))$ would both be mapped to the c-edge $R_2 \rightarrow R_3$.

Note that according to this convention, a rekey message in which there are no encryption keys or one in which all encryption keys are useless (for the given instant) is not represented in the key graph; for example, messages of the form $\mathbf{S}_2(R_2)$ and $\mathbf{S}_1(\mathbf{E}_{R_1}(R_2))$ for some useless key R_1 do not have any c-edge associated with them. Such a representation of messages may appear too parsimonious at first glance (for example, protocol messages comprising a key share are not depicted in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ at all, even if the corresponding key is useful at time t) but it suffices for proving the lower bound.

Lemma 6.4.1 For every $t > 0$, for every sequence $\overrightarrow{\mathcal{S}}^{(t)} \in (2^{[\mathbf{n}]})^t$ given as input to Π, and for any two keys K, K' generated during the execution of Π such that both keys are useful at time t, if $K' \in \mathbf{Rec}(\{K\} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}})$, then there exists a path from K to K' in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ such that for every key K'' on this path, $K'' \in \mathbf{Rec}(\{K\} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}})$.

Given this lemma, the proof of Theorem 6.2.2 is essentially the same as that of Theorem 6.3.1. Just as in the proof of that theorem, we define, for every t, a sub-graph $\tilde{\mathcal{G}}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ of $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ formed by taking the unions of all paths from the member long-lived keys to the group key at time t. (One path per member is selected.) This sub-graph is referred to as the member key graph for time t. Claims 6.3.5 and 6.3.6 and Proposition 6.3.4

can now be proven for such member key graphs as well. It can next be shown that the replace-only sequence defined for the proof of Theorem 6.3.1 also applies to prove the $\log_2(n)$ lower bound for the communication complexity of Π.

6.4.1 Proof of Lemma 6.4.1

Let $t > 0$, and let $\overrightarrow{\mathcal{S}}^{(t)}$ be any sequence given as input to Π. Let K and K' be any two keys that are useful at time t and are such that K' is recoverable from the set $\mathcal{M}_0 := \{K\} \cup \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$. Let q be the smallest number of steps in which K' can be recovered from $\mathcal{M}_0$. For example, if $K' \in \mathcal{M}_0$ then $q = 0$; if $K' \notin \mathcal{M}_0$ but there exists a key K'' such that $\{K'', \mathbf{E}_{K''}(K')\} \subseteq \mathcal{M}_0$, then $q = 1$; and so on.

We prove the lemma using induction over q. We show that for all q and for every useful key K' recoverable from $\mathcal{M}_0$ in q steps, there exists a path from K to K' in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ such that every key on this path is recoverable from $\mathcal{M}_0$. The statement is true for $q = 0$ since in this case K' must be equal to K (for if K' was in $\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$, then it would be useless) and there is a trivial path from K to itself.

Suppose that the statement is true for all values of q smaller than a positive integer Q. That is, for all $q < Q$, every useful key recoverable from $\mathcal{M}_0$ in q steps has a path leading to itself from K in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ and all keys along the path are in $\mathbf{Rec}(\mathcal{M}_0)$. Consider any useful key K' recoverable from $\mathcal{M}_0$ in Q steps. Two possibilities arise:

- *There exists a key K'' such that $K' = \mathbf{G}_b(K'')$ for some $b \in \{0, 1\}$ and K'' is recoverable from $\mathcal{M}_0$ in $Q - 1$ steps.* For K' to be useful at time t, the same must be true for K''. From the inductive hypothesis, it follows that there exists a path from K to K'' in $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ and joining this path with the g-edge $K'' \to K'$ gives us the desired path from K to K'.

- *There exist a set of rekey messages $\mathcal{M}' \subseteq \mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$, each message encapsulating K', such that the set $\mathcal{K}_e$ of encryption keys in* ***all*** *these messages is recoverable from $\mathcal{M}_0$ in less than Q steps and K' is recoverable from $\mathcal{K}_e \cup \mathcal{M}'$.* In this case, we first observe that at least one of the keys in $\mathcal{K}_e$ must be useful: if all of $\mathcal{K}_e$

was useless, that is, if $\mathcal{K}_e$ was contained in $\mathbf{Rec}(\{K_i\}_{i\in\overline{\mathcal{S}}^{(t)}} \cup \mathcal{M}^{\Pi}_{\vec{\mathcal{S}}^{(t)}})$, then K', which is in $\mathbf{Rec}(\mathcal{K}_e \cup \mathcal{M}') \subseteq \mathbf{Rec}(\mathcal{K}_e \cup \mathcal{M}^{\Pi}_{\vec{\mathcal{S}}^{(t)}})$ would also be contained in $\mathbf{Rec}(\{K_i\}_{i\in\overline{\mathcal{S}}^{(t)}} \cup \mathcal{M}^{\Pi}_{\vec{\mathcal{S}}^{(t)}})$. This would mean K' is useless, which, we know, is not true.

Thus, there must exist at least one message $M \in \mathcal{M}'$ that encapsulates K' and contains a useful encryption key $K'' \in \mathcal{K}_e$, with K'' being the inner-most useful encryption key in M. For such a K'', there must exist a c-edge from K'' to K' in $\mathcal{G}^{\Pi}_{\vec{\mathcal{S}}^{(t)}}$. Furthermore, we know that K'' is recoverable from $\mathcal{M}_0$ in less than Q steps. Again, from our hypothesis, it follows that there exists a path from K to K'' in $\mathcal{G}^{\Pi}_{\vec{\mathcal{S}}^{(t)}}$ and we join this path with the c-edge $K'' \to K'$ to get a path from K to K'.

Note that in the second case above, K' can be recoverable from $\mathcal{M}'$ and $\mathcal{K}_e$ using multiple decryption and share reconstruction operations. All these operations are abstracted into a single edge $K'' \to K'$ in the key graph. ▮

6.5 On Beating the $\log_2(n)$ Barrier

In this chapter, we have shown that designing GKD protocols secure against collusion attacks while achieving communication complexity lower than logarithmic in the number of group members is impossible. This impossibility result holds not only for protocols that use symmetric-key encryption and pseudo-random generators but also those that employ secret sharing schemes in generating the center's rekey messages. Thus, to achieve sub-logarithmic communication efficiency in GKD protocols one must either weaken the security requirements from protocols in some way or else, consider usage of primitives that are not incorporated in our model. We discuss each of these possibilities below.

- **Weaker Security Requirements:** If security against single-user attacks is the only security objective, then GKD protocols with sub-logarithmic communication efficiency can indeed be constructed. In particular, it is possible to design protocols [14, 21] that are secure against single-user attacks (but not collusion-resistant) and that incur only a *constant* communication overhead for every group membership update. Such protocols must rely on the usage of nested encryption in generating rekey messages (as is the case with the protocols of [14, 21]) since, as we showed in Theorem 6.3.1, by using single encryption and PRGs alone, one cannot beat the $\log_2(n)$ bound even for the weaker definition.

We remark that some protocols in the literature [47] achieve communication complexity better than $\log_2(n)$ by performing group key updates in "batches", that is, by executing a single key update for multiple consecutive changes in group membership. Such an approach also compromises security of the protocol since it does not guarantee foolproof privacy of group keys in the interval separating any two key updates. Furthermore, even the efficiency benefit that one derives from batched rekeying is not significant; for example, in the protocol of [47], a key update for batches of size up to $\sqrt{n}$ incurs an average cost of $\log_2(n)/2$ per update, which is within a factor 2 of the lower bound we prove.

- **New Primitives:** It is possible that by using cryptographic primitives not incorporated in our model, one can design protocols that are more efficient than those that are known to exist. One such primitive that comes to mind is a *pseudo-random function (PRF)* [22]—a function that behave like a random function and can produce exponentially-many pseudo-random keys from the same seed key. Our belief is that the use of PRFs is unlikely to lead to GKD protocols with communication complexity better than $\log_2(n)$, though it would be nice to substantiate such a claim with formal arguments[3].

  Some protocols in the literature [8] use bilinear maps to achieve constant communication complexity but they do so at the expense of requiring all parties (the center as well as all the protocol users) to store a public key that is linear in $\mathbf{n}$ (the

[3]We note that some known protocols do make use of PRFs (for example, [38]), but not in a substantial way, meaning that whatever they do can be easily achieved using PRGs instead. In effect, these protocols fit our model of group key distribution.

total number of protocol users). Furthermore, the computation cost incurred by users in such protocols in order to decrypt the group key is significantly more than that incurred in typical protocols based on symmetric-key techniques.

It thus appears that in order to beat the $\log_2(n)$ barrier in the communication complexity of GKD protocols, one must make significant compromises on other aspects of protocol design like security and/or computational and storage requirements of users; surpassing the barrier without such compromises seems highly improbable.

6.6 Acknowledgement

Chapter 6, in part, is a reprint of the material to be published in IEEE/ACM Transactions on Networking, October 2008, Micciancio, Daniele; Panjwani, Saurabh. The dissertation author is the primary investigator and author of this paper. The presentation of the proof of Theorem 6.2.2 is different and more modular in the current work.

Part II

The Computational Model

The main drawback of the symbolic model is that it provides a very coarse abstraction for representing cryptographic primitives and for arguing about security of protocols in general. Treating cryptographic primitives symbolically makes security analysis convenient, but it does not necessarily guarantee security against all practical threats to protocols. Real adversaries can potentially extract more information from protocol messages than is computable using only the symbolic rules of information recovery.

To understand this issue better, let us consider again an example we used in Chapter 2 (Section 2.4). Let $\mathcal{M}$ be a set of symbolic messages defined as follows:

$$\mathcal{M} = \{R_2, \mathbf{E}_{\mathbf{G}_1(R_2)}(\mathbf{E}_{\mathbf{G}_0(R_1)}(R_3)), \mathbf{E}_{R_2}(\mathbf{G}_1(R_1)), \mathbf{E}_{\mathbf{G}_0(\mathbf{G}_1(R_1))}(R_4)\}$$

It is easy to check that the keys R_3 and $\mathbf{G}_0(R_1)$ cannot be recovered from $\mathcal{M}$ using the entailment relation we defined in Section 2.4. However, it is unclear whether these keys would remain completely secret if a protocol were to distribute such messages on a public channel. Could not an adversarial observer learn partial information about R_3 or $\mathbf{G}_0(R_1)$ from $\mathcal{M}$? For one, any adversary can easily distinguish the pair $(R_3, \mathbf{G}_0(R_1))$ from a pair of purely random and independent values—R_3 and $\mathbf{G}_0(R_1)$ are related to each other via the ciphertext $\mathbf{E}_{\mathbf{G}_0(R_1)}(R_3)$ and since the latter is recoverable from $\mathcal{M}$, this leaks sufficient information about the two keys for an attacker to be able to distinguish them from two purely random values. Thus, R_3 and $\mathbf{G}_0(R_1)$ cannot be guaranteed to be secret in an absolute sense.

Besides this, there is another potential problem. Even if the function $\mathbf{E}$ is implemented in a secure manner (such that secrecy of any message encrypted under $\mathbf{E}$ is guaranteed in a strong sense), some non-trivial information about the key $\mathbf{G}_0(R_1)$ could be leaked by the ciphertext $\mathbf{E}_{\mathbf{G}_0(R_1)}(R_3)$: secrecy of the message being encrypted (in this case, R_3) does not imply secrecy of the key used to encrypt it (in this case, $\mathbf{G}_0(R_1)$)[4]. This could, in turn, leak partial information about the seed R_1 used to derive $\mathbf{G}_0(R_1)$, which could potentially reveal information about other keys as well.

[4]Indeed, standard definitions of security for encryption, like the one we use in this thesis (Definition 7.2.1), *do not* require encryption keys to remain completely secret after encryption; it is possible that partial information about keys be leaked by the encryption operation.

The main point here is that the notion of recoverability developed in the symbolic model is insufficient for arguing about secrecy properties of protocols, and it is unclear whether a protocol proven secure using symbolic arguments alone can be guaranteed to be secure in a real implementation of the same. The computational model addresses this limitation of the symbolic model. It provides a rigorous framework for representing cryptographic primitives and protocols, for modeling realistic attacks on protocols and, in particular, for formulating the notion of secrecy in a precise manner.

In the current part of the thesis, we describe the computational model in depth and develop notions of security for group key distribution within this model. Naturally, using a more rigorous model makes analysis of protocols harder, and security proofs more complex. Instead of analyzing every protocol in the new model "from scratch", we develop general techniques that enable us to relate security analyses conducted in the symbolic model with those in the computational model. Specifically, we devise sufficient conditions on encryption protocols such that for any protocol satisfying these conditions, a proof of security of the protocol in the symbolic model also implies security against powerful computational adversaries. These conditions restrict protocols only at a *syntactic* level, and most protocols of practical interest either already satisfy them or can be easily modified (that is, without any loss in efficiency) to do so. Such an approach gives us the best of both worlds—the simplicity of doing security proofs in the symbolic model, as well as the thoroughness and mathematical rigor of computational cryptography.

Chapter 7

Security Definitions

We begin our exposition on the computational model by giving definitions of security for the tasks of pseudo-random generation, encryption and group key distribution within the model.

In the computational model, every piece of data generated and used by a protocol is treated as a bitstring, that is, an element of the set $\{0, 1\}^*$. Keys of protocol users and messages generated and exchanged between them are all viewed as bitstrings. For any bitstring $\mathtt{s} \in \{0, 1\}^*$, we use $|\mathtt{s}|$ to denote the length of $\mathtt{s}$. For any two bitstrings $\mathtt{s}_1, \mathtt{s}_2$, $\mathtt{s}_1 \parallel \mathtt{s}_2$ denotes the string formed by concatenating $\mathtt{s}_1$ and $\mathtt{s}_2$.

By an *adversary*, we refer to any arbitrary randomized algorithm designed to attack a cryptographic primitive or protocol. We use the concrete-security framework [5] to formalize the notions of attacks; that is, we use concrete parameters to bound the running time of adversaries as well as their probability of being able to execute successful attacks. The letter τ is used to denote time complexity of adversaries while ϵ denotes the success probability of the attacks they execute. As such, τ is always a variable that takes positive integral values and ϵ one that takes values in the range $[0, 1]$. The probability of any event E is denoted $\mathbf{P}[E]$.

An important ingredient of the computational model is a security parameter η, an integer that determines the length of all keys used in any protocol, which, in turn, determines the amount of security provided by cryptographic primitives. Let $\mathbb{R}$ be an algorithm that samples a key uniformly at random from the space $\{0,1\}^\eta$ and outputs it. We use $\tau(\mathbb{R})$ to denote the running time of $\mathbb{R}$. By $k \leftarrow \mathbb{R}$, we denote the process of generating a key k via an independent invocation of $\mathbb{R}$.

7.1 Pseudo-random Generators

In computational terminology, a pseudo-random generator (PRG) is a deterministic algorithm $\mathbb{G}$ that takes a bitstring $k \in \{0,1\}^\eta$ as input and outputs another bitstring, denoted $\mathbb{G}(k)$, such that $|\mathbb{G}(k)| = \eta + \gamma$ for some $\gamma > 0$. The parameter γ is referred to as the *stretch* of $\mathbb{G}$. For any PRG $\mathbb{G}$, we denote the running time of $\mathbb{G}$ by $\tau(\mathbb{G})$.

Besides increasing the length of its input, a PRG must also guarantee that the resulting output is "pseudo-random" in the sense that no efficient algorithm can distinguish it from a purely random string of the same length. For any PRG $\mathbb{G}$ with stretch γ, let $X_0^{\mathbb{G}}$ be the random variable corresponding to the output of $\mathbb{G}$ when given, as input, a bitstring k obtained by running $\mathbb{R}$ and let $X_1^{\mathbb{G}}$ be that corresponding to the uniform distribution over $\{0,1\}^{\eta+\gamma}$. For any adversary A and any $b \in \{0,1\}$, let $\mathsf{A}(X_b^{\mathbb{G}})$ denote the random variable corresponding to the output of A when given a sample from $X_b^{\mathbb{G}}$ as input. The *advantage* of A against $\mathbb{G}$ is defined as

$$\Delta_{\text{PRG}}^{\mathbb{G}}(\mathsf{A}) = \left| \mathbf{P}[\mathsf{A}(X_0^{\mathbb{G}}) = 1] - \mathbf{P}[\mathsf{A}(X_1^{\mathbb{G}}) = 1] \right|$$

with the probabilities being taken over the random coins used by A and the randomness involved in generating $X_0^{\mathbb{G}}$ and $X_1^{\mathbb{G}}$.

Definition 7.1.1 (Security of pseudo-random generation) A pseudo-random generator $\mathbb{G}$ is called (τ, ϵ)-secure if for every adversary A running in time at most τ, $\Delta_{\text{PRG}}^{\mathbb{G}}(\mathsf{A}) \leq \epsilon$.

As in the symbolic model, we are interested in PRGs that take a single key as input and output a string that can be parsed as two keys. This essentially corresponds to doubling the length of the input, that is, setting $\gamma = \eta$. Any PRG satisfying this property is referred to as a *length-doubling pseudo-random generator*. For any key k given as input to such a PRG, we denote by $\mathbb{G}_0(k)$ and $\mathbb{G}_1(k)$ the left and right halves of the bitstring $\mathbb{G}(k)$; so, $|\mathbb{G}_0(k)| = |\mathbb{G}_1(k)|$ and $\mathbb{G}(k) = \mathbb{G}_0(k) \,\|\, \mathbb{G}_1(k)$.

7.2 Encryption Schemes

A symmetric-key encryption scheme is a pair of algorithms $\mathbb{P} = (\mathbb{E}, \mathbb{D})$, referred to as the encryption and decryption algorithms respectively, which have the following properties:

- $\mathbb{E}$ is a randomized algorithm that takes, as input, a plaintext $m \in \{0,1\}^*$, and a key $k \in \{0,1\}^\eta$ and outputs a ciphertext $c \in \{0,1\}^*$. The random variable corresponding to the output of $\mathbb{E}$ when given m and k as input is denoted $\mathbb{E}_k(m)$. The running time of $\mathbb{E}$ is denoted $\tau(\mathbb{E})$.
- $\mathbb{D}$, on input a ciphertext $c \in \{0,1\}^*$ and a key $k \in \{0,1\}^\eta$, returns a value in $\{0,1\}^*$. $\mathbb{D}$ must be deterministic and for any $m, c \in \{0,1\}^*$, and any $k \in \{0,1\}^\eta$, the output of $\mathbb{D}$ on input c and k must equal m if and only if c lies in the support of $\mathbb{E}_k(m)$. (The output must be undefined otherwise.)

In this thesis, we use the notion of semantic security against chosen plaintext attacks for encryption schemes, which was first defined by Goldwasser and Micali [23] for *public-key* encryption, and later adapted to the symmetric-key setting by Bellare *et al.* [4]. Informally, an encryption scheme is semantically secure against chosen plaintext attacks, or simply CPA-*secure*, if no efficient adversary A can distinguish between the encryptions of two equal-length plaintexts with noticeable probability, provided the encryptions are created using a randomly chosen key unknown to A. CPA-security is a natural (and reasonably strong) notion of security for encryption schemes and is currently regarded as the "standard" notion of encryption security in the cryptography literature. It implies various useful properties that one might expect from encryption schemes, for example, the property of one-wayness (no efficient adversary can invert a ciphertext to the corresponding plaintext), and of partial plaintext recovery (no efficient adversary can recover from a ciphertext even a single bit of the corresponding plaintext).

For any encryption scheme $\mathbb{P} = (\mathbb{E}, \mathbb{D})$, and any $b \in \{0,1\}$, let $\mathcal{O}_b^{\mathbb{P}}$ denote an oracle procedure that first generates a key k by running $\mathbb{R}$ and subsequently, accepts queries of the form $(m_0, m_1) \in \{0,1\}^* \times \{0,1\}^*$ such that $|m_0| = |m_1|$. Upon receiving such a query, the procedure responds to it with a ciphertext sampled according to $\mathbb{E}_k(m_b)$. (For queries that are not of the said form, the procedure does nothing.) For any adversary A, let $\mathsf{A}^{\mathcal{O}_b^{\mathbb{P}}}$ denote the random variable corresponding to the output of A when given black-box access to $\mathcal{O}_b^{\mathbb{P}}$. The CPA-advantage of A against $\mathbb{P}$ is defined as the following quantity:

$$\Delta_{\text{ENC}}^{\mathbb{P}}(\mathsf{A}) = \left| \mathbf{P}[\mathsf{A}^{\mathcal{O}_0^{\mathbb{P}}} = 1] - \mathbf{P}[\mathsf{A}^{\mathcal{O}_1^{\mathbb{P}}} = 1] \right|$$

that is, $\Delta_{\text{ENC}}^{\mathbb{P}}(\mathsf{A})$ is the difference between the probabilities that A outputs 1 when given black-box access to $\mathcal{O}_0^{\mathbb{P}}$ versus the same when it receives access to $\mathcal{O}_1^{\mathbb{P}}$ instead. Both probabilities are taken over the random coins used by A and by $\mathcal{O}_b^{\mathbb{P}}$ (which includes the randomness used to generate the secret key k used for encryption).

Definition 7.2.1 (Security of encryption) A symmetric-key encryption scheme $\mathbb{P} = (\mathbb{E}, \mathbb{D})$ is called (τ, ϵ)-secure against chosen plaintext attacks if for every adversary A running in time at most τ, $\Delta_{\text{ENC}}^{\mathbb{P}}(\mathsf{A}) \leq \epsilon$.

Pseudo-random generators and symmetric-key encryption schemes are both typically constructed using cryptographic objects called *block ciphers*. The work of Bellare *et al.* [4] provides various block-cipher based constructions of encryption schemes and proves them to be (τ, ϵ)-secure under suitable assumptions with respect to the security of the underlying block cipher.

7.3 Mapping Symbolic Messages to Bitstrings

Every message generated by a protocol in the symbolic model can be mapped to a bitstring in a natural manner by implementing the function $\mathbf{G}$ with a length-doubling

PRG and the function $\mathbf{E}$ with a symmetric-key encryption algorithm. Formally, let $\mathbb{P} = (\mathbb{E}, \mathbb{D})$ be a symmetric-key encryption scheme and $\mathbb{G}$ a length-doubling pseudo-random generator. Let M be any message derived from grammar (2.1) and $\psi\colon \mathcal{R} \to \{0,1\}^\eta$ a map such that $\psi(R_i)$ is defined for every R_i occurring in M. The *evaluation of* M *with respect to* $\psi, \mathbb{E}$ *and* $\mathbb{G}$ is the random variable corresponding to the output of the following recursive procedure:

```
**procedure** $evaluate^{\mathbb{E},\mathbb{G}}(M, \psi)$
    If $M \in \mathcal{R}$, return $\psi(M)$.
    If $M = \mathbf{G}_b(K)$ for some symbol $K$ and some $b \in \{0,1\}$, do the following:
        Let $k \leftarrow evaluate^{\mathbb{E},\mathbb{G}}(K, \psi)$.
        Return $\mathbb{G}_b(k)$.
    If $M = \mathbf{E}_K(M')$ for some symbols $K$ and $M'$, do the following:
        Let $k \leftarrow evaluate^{\mathbb{E},\mathbb{G}}(K, \psi)$.
        Let $m' \leftarrow evaluate^{\mathbb{E},\mathbb{G}}(M', \psi)$.
        Return a sample from $\mathbb{E}_k(m')$.
```

We use $[\![M]\!]_\psi^{\mathbb{E},\mathbb{G}}$ to denote the evaluation of M with respect to $\psi, \mathbb{E}$ and $\mathbb{G}$.

7.4 Group Key Distribution

As in the symbolic model, we consider group key distribution protocols built generically from a length-doubling pseudo-random generator $\mathbb{G}$ and a symmetric-key encryption scheme $\mathbb{P} = (\mathbb{E}, \mathbb{D})$. For any $\mathbf{n}$-user GKD protocol $\Pi = (\mathsf{S}, \mathsf{C})$ in the symbolic model, the computational interpretation of Π is defined as a pair of algorithms $\Pi^{\mathbb{P},\mathbb{G}} = (\mathsf{S}^{\mathbb{P},\mathbb{G}}, \mathsf{C}^{\mathbb{P},\mathbb{G}})$ that work as follows:

- $\mathsf{S}^{\mathbb{P},\mathbb{G}}$ first runs S to obtain the initial state of program C, $\mathcal{Z}^{(0)}$, and the symbolic long-lived keys of all the users, $K_1, \ldots, K_\mathbf{n}$. It then initializes a key map ψ and for each purely random key symbol R_i occurring in $\mathcal{Z}^{(0)} \cup \{K_1, \ldots, K_\mathbf{n}\}$, it samples a bitstring r_i independently and uniformly at random from $\{0,1\}^\eta$ and sets $\psi(R_i)$

to be r_i. Finally, it outputs two things: (a) the initial state $\mathfrak{Z}^{(0)}$ of $\mathsf{C}^{\mathbb{P},\mathbb{G}}$ which is equal to $\{(K, [\![K]\!]_{\psi}^{\mathbb{E},\mathbb{G}}) \mid K \in \mathcal{Z}^{(0)}\}$; and (b) the long-lived keys $k_1, \ldots, k_{\mathbf{n}}$ of all users, where for each $i \in [\mathbf{n}]$, $k_i = [\![K_i]\!]_{\psi}^{\mathbb{E},\mathbb{G}}$.

- $\mathsf{C}^{\mathbb{P},\mathbb{G}}$, when given a set $\mathcal{S}^{(t)} \subseteq [\mathbf{n}]$ and its current state $\mathfrak{Z}^{(t-1)}$, runs C with input $\mathcal{S}^{(t)}$ and $\mathcal{Z}^{(t-1)} := \{K \mid \exists k : (K, k) \in \mathfrak{Z}^{(t-1)}\}$ for which the latter outputs a set of symbolic messages $\mathcal{M}_{\mathcal{S}^{(t)}}^{\Pi}$ and the updated state of C, $\mathcal{Z}^{(t)}$. For each R_i occurring in $\mathcal{M}_{\mathcal{S}^{(t)}}^{\Pi} \cup \mathcal{Z}^{(t)}$ such that $\psi(R_i)$ is undefined, $\mathsf{C}^{\mathbb{P},\mathbb{G}}$ sets $\psi(R_i)$ to be an independent, uniformly random sample from $\{0,1\}^{\eta}$ and then outputs the evaluations of $\mathcal{M}_{\mathcal{S}^{(t)}}^{\Pi}$ and $\mathcal{Z}^{(t)}$; that is, it outputs two sets $\mathfrak{M}_{\mathcal{S}^{(t)}}^{\Pi}$ and $\mathfrak{Z}^{(t)}$ defined as follows:

$$\begin{aligned} \mathfrak{M}_{\mathcal{S}^{(t)}}^{\Pi} &= \{(M, [\![M]\!]_{\psi}^{\mathbb{E},\mathbb{G}}) \mid M \in \mathcal{M}_{\mathcal{S}^{(t)}}^{\Pi}\} \\ \mathfrak{Z}^{(t)} &= \{(K, [\![K]\!]_{\psi}^{\mathbb{E},\mathbb{G}}) \mid K \in \mathcal{Z}^{(t)}\} \end{aligned}$$

We formalize security of $\Pi^{\mathbb{P},\mathbb{G}}$ using an indistinguishability-based definition, akin to the definitions of security for encryption and pseudo-random generation. For any $t > 0$, let $k^{(t)}$ denote the bitstring group key distributed by the protocol at time t; that is, $k^{(t)} = [\![K^{(t)}]\!]_{\psi}^{\mathbb{E},\mathbb{G}}$ where ψ is the key map at time t. Intuitively, the protocol is secure (in the computational model) if for any possible execution, and for any instant t during the execution, the key $k^{(t)}$ appears indistinguishable from random, when viewed by the non-members at that instant. Furthermore, this condition should hold even when changes in membership are made in an adversarial manner and when non-members are corrupted adversarially.

To formalize this notion, we define, for each $b \in \{0,1\}$, an oracle procedure $\mathcal{O}_{\text{GKD},b}^{\Pi,\mathbb{P},\mathbb{G}}$ that emulates $\Pi^{\mathbb{P},\mathbb{G}}$ as follows. First, $\mathcal{O}_{\text{GKD},b}^{\Pi,\mathbb{P},\mathbb{G}}$ runs $\mathsf{S}^{\mathbb{P},\mathbb{G}}$ to obtain the initial state of the protocol—the value $\mathfrak{Z}^{(0)}$ and the long-lived keys $k_1, \ldots, k_{\mathbf{n}}$. It then accepts and responds to three types of queries:

- *Execution queries:* These queries specify membership changes made during the execution of Π and include an argument $\mathcal{S}$ denoting the current set of members. Upon receiving any query of the form $\texttt{execute}(\mathcal{S})$, $\mathcal{O}_{\text{GKD},b}^{\Pi,\mathbb{P},\mathbb{G}}$ first checks if this is the first query of its kind. If so, it initializes a variable t and sets it to be equal to 1; otherwise, it simply increments t. After doing this, it invokes $\mathsf{C}^{\mathbb{P},\mathbb{G}}$ on input $\mathcal{S}^{(t)} := \mathcal{S}$ and $\mathfrak{Z}^{(t-1)}$, stores the updated state $\mathfrak{Z}^{(t)}$ of $\mathsf{C}^{\mathbb{P},\mathbb{G}}$ and returns the set of rekey messages $\mathfrak{M}^{\Pi}_{\mathcal{S}^{(t)}}$ to the querying algorithm.

- *Corruption queries:* These queries model corruption of users in the protocol. When given a query of the form $\texttt{corrupt}(i)$ for some $i \in [\mathbf{n}]$, $\mathcal{O}^{\Pi,\mathbb{P},\mathbb{G}}_{\text{GKD},b}$ returns the long-lived key k_i.

- *Challenge queries:* These queries test the distinguishability of group keys from purely random values. When given a query $\texttt{challenge}(t')$ such that $t' \in \{1,\ldots,t\}$, $\mathcal{O}^{\Pi,\mathbb{P},\mathbb{G}}_{\text{GKD},b}$ replies with $k^{(t')}$ if $b = 0$, and with a key $r^{(t')}$ obtained by running $\mathbb{R}$ if $b = 1$. (If the same query $\texttt{challenge}(t')$ is received multiple times, and if $b = 1$, the reply is $r^{(t')}$ each time.)

Consider any adversary A that is given black-box access to such a procedure and is allowed to make queries to it in an arbitrary, adaptive manner; that is, the choice of any query it makes can depend on the replies it receives for queries made in the past. We would like to be able to show that A cannot guess the bit b with probability noticeably better than half, that is, A's output when given access to $\mathcal{O}^{\Pi,\mathbb{P},\mathbb{G}}_{\text{GKD},0}$ is similarly distributed as its output when given access to $\mathcal{O}^{\Pi,\mathbb{P},\mathbb{G}}_{\text{GKD},1}$.

Clearly, we cannot make such an assertion for every possible adversary: if A makes an execution query $\texttt{execute}(\mathcal{S})$ at time $t = 1$ such that $\mathcal{S}$ contains a corrupt user, and later issues the query $\texttt{challenge}(1)$, it can trivially compute the value of b. Thus, it makes sense to consider only those adversaries that *do not* issue challenge queries for instants when corrupt users are part of the group.

For any adversary A interacting with $\mathcal{O}^{\Pi,\mathbb{P},\mathbb{G}}_{\text{GKD},b}$, let $\mathfrak{c}(\mathsf{A})$ denote the number of corruption queries made by A and $\mathcal{S}^{\mathsf{corr}}(\mathsf{A})$ the set of values i for which A makes the query $\texttt{corrupt}(i)$. Let $\mathcal{T}^{\mathsf{chal}}(\mathsf{A})$ be the set of instants t' for which A issues the query

$\texttt{challenge}(t')$. Let $\mathcal{S}^{\mathsf{chal}}(\mathsf{A})$ be the set defined as

$$\mathcal{S}^{\mathsf{chal}}(\mathsf{A}) := \bigcup_{t' \in \mathcal{T}^{\mathsf{chal}}(\mathsf{A})} \mathcal{S}^{(t')}$$

Note that $\mathfrak{c}(\mathsf{A}), \mathcal{S}^{\mathsf{corr}}(\mathsf{A}), \mathcal{T}^{\mathsf{chal}}(\mathsf{A})$ and $\mathcal{S}^{\mathsf{chal}}(\mathsf{A})$ are all random variables depending on the coins used by A and by the procedure $\mathcal{O}^{\Pi,\mathbb{P},\mathbb{G}}_{\mathrm{GKD},b}$.

Definition 7.4.1 An adversary A is called *legitimate* if for any $b \in \{0,1\}$, in any execution of A involving interaction with $\mathcal{O}^{\Pi,\mathbb{P},\mathbb{G}}_{\mathrm{GKD},b}$, $\mathcal{S}^{\mathsf{corr}}(\mathsf{A}) \cap \mathcal{S}^{\mathsf{chal}}(\mathsf{A}) = \emptyset$.

We distinguish between two types of legitimate adversaries, depending upon the manner in which the adversary issues its corruption queries.

Definition 7.4.2 A legitimate adversary A is called *non-adaptive* if for any $b \in \{0,1\}$, in any execution of A involving interaction with $\mathcal{O}^{\Pi,\mathbb{P},\mathbb{G}}_{\mathrm{GKD},b}$, every corruption query of A is made *before* an execution query. A is called *adaptive* otherwise.

Non-adaptive adversaries correspond to a scenario in which the set of corrupt users in the protocol is decided at the outset, that is, before the execution of the protocol begins. Security analysis of protocols against such adversaries is relatively easier and is addressed first in the forthcoming chapters. Note that we allow non-adaptive adversaries to issue execution queries and challenge queries in an interleaving and adaptive manner, as long as these queries are made after making all corruption queries. Furthermore, the corruption queries themselves can be such that the choice of the ith user to corrupt depends on the keys of the first $i-1$ corrupt users.

The GKD-advantage of any legitimate adversary A is defined as the following quantity:

$$\Delta^{\Pi^{\mathbb{P},\mathbb{G}}}_{\mathrm{GKD}}(\mathsf{A}) = \left| \mathbf{P}[\mathsf{A}^{\mathcal{O}^{\Pi,\mathbb{P},\mathbb{G}}_{\mathrm{GKD},0}} = 1] - \mathbf{P}[\mathsf{A}^{\mathcal{O}^{\Pi,\mathbb{P},\mathbb{G}}_{\mathrm{GKD},1}} = 1] \right|$$

As usual, both probabilities are taken over the random choices made by A as well as those made by $\mathcal{O}^{\Pi,\mathbb{P},\mathbb{G}}_{\mathrm{GKD},b}$.

Definition 7.4.3 (Security of group key distribution) Let $\mathbb{P}$ be a symmetric-key encryption scheme and $\mathbb{G}$ a length-doubling pseudo-random generator. Let `adv-type` $\in \{\textit{adaptive}, \textit{non-adaptive}\}$. An $\mathbf{n}$-user GKD protocol $\Pi^{\mathbb{P},\mathbb{G}}$ is called (τ, t, ϵ)-`adv-type`*ly secure against single-user attacks* in the computational model, if for every `adv-type` adversary A that runs in time at most τ, makes at most t execution queries and for which $\mathfrak{c}(\mathsf{A})$ is always at most 1, $\Delta^{\Pi^{\mathbb{P},\mathbb{G}}}_{\text{GKD}}(\mathsf{A}) \leq \epsilon$. The protocol is called (τ, t, ϵ)-`adv-type`*ly secure against collusion attacks* in the computational model if for every `adv-type` adversary A that runs in time at most τ and makes at most t execution queries, $\Delta^{\Pi^{\mathbb{P},\mathbb{G}}}_{\text{GKD}}(\mathsf{A}) \leq \epsilon$.

This definition corresponds with the notions of "strong" security against single-user and collusion attacks (definitions 3.2.3 and 3.2.4) used in the symbolic model, and as in the symbolic model, security against collusion attacks implies security against single-user attacks here as well. We do not consider computational counterparts of the weaker notions (definitions 3.2.1 and 3.2.2) since our interest is in analyzing protocols (and proving them secure) in the strong sense only.

Our security definition for GKD protocols in the computational model is similar to the definition of security for broadcast encryption protocols typically found in the literature [34], but for three differences. First, we consider the security of a key distribution protocol, rather than of an encryption protocol built using it (which is the situation considered in broadcast encryption protocols). This is not a substantial difference as the two protocol classes are equivalent in the sense that a protocol from one can be easily converted into a protocol from the other. Second, we allow the adversary to issue multiple challenge queries of its choice; this models the requirement that group keys for different instants be "jointly" pseudo-random (in particular, that they be independent of each other). In contrast, definitions of broadcast encryption used in the literature consider adversaries that are challenged only for a single instant in the protocol. In this respect, our definition is seemingly stronger.[1] Third, our definition captures the notion of CPA-security, which is weaker than the notion of CCA-security (security against cho-

[1] For the case of broadcast encryption, this restriction does not matter, that is, allowing the adversary the power to issue multiple challenge queries does not make the definition any stronger. For group key distribution, however, it does make a difference: there exist protocols that are secure against adversaries that make single challenge queries but insecure against those that are allowed to make multiple such queries.

sen ciphertext attacks [40]) used in [34]; that is, we do not consider adversaries that can, besides issuing execution queries, ask for decryptions of arbitrary rekey messages of their choice. Extending the results of this thesis to incorporate CCA-security is possible provided we define the encryption scheme $\mathbb{P}$ to be secure in the CCA sense as well. We focus on CPA-based definitions in order to keep the exposition simpler and the results easier to comprehend.

Chapter 8

Computational Security against Non-Adaptive Adversaries

In the symbolic world, all security definitions are formulated using the notion of recoverability, and in order to prove secrecy of a key, one argues that the key is not recoverable, in its entirety, by the malicious users in the protocol. On the other hand, secrecy properties in the computational model are expressed in terms of distinguishability: to prove secrecy of a key, one shows that it is hard to distinguish the *bitstring* value of that key from a purely random (and independently chosen) value.

Before we begin to analyze GKD protocols in the computational model, we ask ourselves a more general question: *how do the notions of recoverability and distinguishability relate to each other?* More precisely, when can we say that unrecoverability of a key also implies indistinguishability of the same from a random value? Or, under what conditions is symbolic analysis of protocols "computationally sound" in the sense that a protocol that is proven symbolically secure (using the notion of key recoverability) is guaranteed to be computationally secure (in terms of key indistinguishability) as well? Recall that in the computational model, adversaries can be arbitrary randomized algorithms and can potentially control the execution of protocols, modify their execution flow in an adaptive manner, and, worst still, corrupt protocol users adaptively. Can we hope to prove soundness of symbolic analysis with respect to such entities?

At an abstract level, we are faced with the following problem. Consider an adversary A that creates an arbitrary sequence of symbolic messages $M_1, M_2, \ldots, M_q$ (generated according to grammar (2.1)), and receives, in return, the evaluations of each of these messages with respect to some key map ψ that is kept secret from it. These symbolic messages can be either ciphertexts (for example, rekey messages in a GKD protocol) or simply keys (for example, keys of corrupt protocol users), and can be generated in an adaptive manner by the adversary. Let K be any symbolic key that is *not* recoverable from the set $\{M_1, M_2, \ldots, M_q\}$; that is, $K \notin \mathbf{Rec}(\{M_1, M_2, \ldots, M_q\})$. Then, can we assert that A cannot distinguish between the evaluation of K and a uniformly random bitstring? Can we find reasonable restrictions on the choice of the messages $M_1, \ldots, M_q$ such that the assertion holds for *any* adversary satisfying those restrictions?

In the current and the succeeding chapter, we consider the above problem and present two different positive solutions to it. Our first solution, presented in the current chapter, involves imposing certain restrictions on the *order* in which the messages $M_1, \ldots, M_q$ are issued by the adversary. These restrictions are fairly mild and enable us to relate symbolic and computational definitions of security for a large variety of GKD protocols. However, under these restrictions, symbolic security can be proven to imply computational security, only as long as the latter is considered with respect to non-adaptive adversaries (that is, adversaries who corrupt protocol users in a non-adaptive manner). In the next chapter, we take a different approach to address the problem, and show how computational security against adaptive adversaries is also possible to achieve.

8.1 A Computational Game

Let $\mathbb{P} = (\mathbb{E}, \mathbb{D})$ be any symmetric-key encryption scheme and $\mathbb{G}$ a length-doubling pseudo-random generator. For any bit $b \in \{0, 1\}$, we define an oracle procedure $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$ that first initializes a key map ψ and subsequently, accepts two types of queries: *evaluation* queries and *challenge* queries. Every evaluation query $\texttt{eval}(M)$ in-

```
procedure $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$
    Initialize a key map $\psi$ to be empty.

    Upon receiving a query of the form $\texttt{eval}(M)$, do the following:
        For every fresh key $R_i$ occurring in $M$,
            If $\psi(R_i)$ is undefined, set $\psi(R_i) \leftarrow \mathbb{R}$.
        Reply with $[\![M_i]\!]^{\mathbb{E},\mathbb{G}}_{\psi}$.

    Upon receiving a query of the form $\texttt{challenge}(K)$, do the following:
        If $\psi(\mathsf{root}(K))$ is undefined, set $\psi(\mathsf{root}(K)) \leftarrow \mathbb{R}$.
        If $b = 0$, reply with $[\![K]\!]^{\mathbb{E},\mathbb{G}}_{\psi}$.
        Else, reply with an independent, random sample from $\{0,1\}^{\eta}$.
```

Figure 8.1: Procedure $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$ used in our computational game.

cludes a symbolic message M (derived from grammar (2.1)) as an argument, and $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$ responds to the query with the bitstring evaluation of M, $[\![M]\!]^{\mathbb{E},\mathbb{G}}_{\psi}$. (If, for some fresh key R_i occurring in M, $\psi(R_i)$ is undefined, the procedure first defines it appropriately.) Every challenge query is of the form $\texttt{challenge}(K)$—K being a symbolic key—and, depending upon the value of b, the procedure responds with either the evaluation of K or a random, independent bitstring key. Figure Figure 8.1 shows the details of how the responses are created.

Consider any adversary A that is given black-box access to $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$ for some bit b (that is kept secret from A). A can make both evaluation and challenge queries to $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$ of its choice, and can do so in an adaptive manner; that is, at any instant, it can decide its next query to $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$ based on the replies it received for the previously-made queries. Its objective is to guess the value of b correctly.

Let $\mathcal{M}(\mathsf{A})$ be the set of all symbolic messages M such that A makes a query $\texttt{eval}(M)$ and $\mathcal{K}(\mathsf{A})$ the set of all K such that it makes a query $\texttt{challenge}(K)$. Note that both $\mathcal{M}(\mathsf{A})$ and $\mathcal{K}(\mathsf{A})$ are random variables depending on the coins used by A as well as by $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$.

Definition 8.1.1 An adversary A is called *valid* relative to $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$ if in any execution of A involving interaction with $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$, the following is true for every $K \in \mathcal{K}(\mathsf{A})$:

- For every $K' \in \mathbf{Rec}(\mathcal{M}(\mathsf{A}))$, $K' \notin \mathbf{Rec}(K)$;
- For every $\mathbf{E}_{K'}(M) \in \mathbf{Rec}(\mathcal{M}(\mathsf{A}))$, $K' \notin \mathbf{Rec}(K)$; and
- For every $K' \in \mathcal{K}(\mathsf{A}) \setminus \{K\}$, $K' \notin \mathbf{Rec}(K)$.

The three conditions that characterize validity are necessary if our goal is to prove pseudo-randomness of all keys in $\mathcal{K}(\mathsf{A})$ (more precisely, of the evaluations of all symbols in $\mathcal{K}(\mathsf{A})$). This is because for any key K, a key $K' \in \mathbf{Rec}(K)$ (or a ciphertext $\mathbf{E}_{K'}(M)$ for which $K' \in \mathbf{Rec}(K)$) leaks sufficient information about K for it to be distinguishable from a purely random value. Such a K' should not be revealed to the adversary for any $K \in \mathcal{K}(\mathsf{A})$.

We would like to be able to show that for every valid adversary A, the keys in $\mathcal{K}(\mathsf{A})$ are indeed pseudo-random; in other words, no valid adversary can distinguish between the behavior of the procedure $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$ when $b = 0$ from its behavior when $b = 1$ with noticeable probability. However, proving this in general, assuming only the standard notions of security of encryption and pseudo-random generation (definitions 7.2.1 and 7.1.1), is rather difficult and, in fact, it is not even possible without restricting A's queries in some way. We next define a set of syntactic restrictions on A's queries which enable us to prove such a claim.

8.2 Syntactic Restrictions

Before we define our syntactic restrictions, we need some notations. For any symbolic message M derived from grammar (2.1), let $\mathit{msgkey}(M)$ denote the key that occurs as a plaintext in M and let $\mathit{enckeys}(M)$ denote the set of keys used to perform encryption in M. For example, if $M = \mathbf{G}_0(\mathbf{G}_1(R_1))$, $\mathit{msgkey}(M) = \mathbf{G}_0(\mathbf{G}_1(R_1))$ and $\mathit{enckeys}(M) = \emptyset$, and if $M = \mathbf{E}_{\mathbf{G}_1(R_2)}(\mathbf{E}_{\mathbf{G}_0(R_1)}(R_3))$, $\mathit{msgkey}(M) = R_3$ and $\mathit{enckeys}(M) = \{\mathbf{G}_1(R_2), \mathbf{G}_0(R_1)\}$. Let $\mathit{keys}(M)$ denote the union of $\mathit{enckeys}(M)$ and

$\{msgkey(M)\}$. For any set of symbolic messages $\mathcal{M}$, let $enckeys(\mathcal{M}), msgkeys(\mathcal{M})$ and $keys(\mathcal{M})$ denote the sets formed by taking the union of $enckeys(M), msgkeys(M)$ and $keys(M)$ (respectively) over all messages $M \in \mathcal{M}$.

Definition 8.2.1 A set of symbolic messages $\mathcal{M}$ is called *safe* if,

(a) for every $K \in enckeys(\mathcal{M})$ and for every $K' \in keys(\mathcal{M}) \setminus \{K\}$, $K' \notin \mathbf{Rec}(K)$.

(b) for every $M \in \mathcal{M}$, and for every $K \in enckeys(M)$, $K \notin \mathbf{Rec}(msgkey(M))$.

Both these conditions correspond to commonly accepted norms in computational cryptography. For example, it is known that using a key as input to more than one cryptographic primitive could completely compromise it, *even for some secure implementations of the primitives*. Condition (a) above prohibits this from happening in our setting: it says that no key K can be used both as an encryption key and as a seed to the pseudo-random generator. In a similar vein, encrypting a key with itself or with a key related to it is considered bad cryptographic practice; condition (b) disallows such usage of keys. Note that we exclude the possibility $msgkey(M) \notin \mathbf{Rec}(K)$ in this condition since this is already prohibited by condition (a).

Besides the safety conditions, we impose one more restriction on the queries of valid adversaries. This restriction in on the order in which queries are made by the adversary and it enables computational proof techniques to work in our setting.

Definition 8.2.2 A sequence of messages $(M_1, \ldots, M_q)$ is called *well-ordered* if for every $i, j \in \{1, \ldots, q\}$ such that $i < j$, there exists no $K \in enckeys(M_i)$ for which $K \in \mathbf{Rec}(msgkey(M_j))$.

For example, $(\mathbf{E}_{R_1}(R_2), \mathbf{E}_{R_2}(R_3))$ and $(\mathbf{E}_{R_1}(R_2), \mathbf{E}_{\mathbf{G}_0(R_2)}(R_3))$ are well-ordered sequences but neither $(\mathbf{E}_{R_2}(R_3), \mathbf{E}_{R_1}(R_2))$ nor $(\mathbf{E}_{R_1}(R_2), \mathbf{E}_{R_2}(R_1))$ is so. Encryption protocols usually distribute keys while satisfying the well-ordered constraint: keys are typically distributed before they are used for encrypting other keys (and, sometimes, simultaneously as such encryption is performed). All protocols we consider in this thesis are of this nature. At the same time, Definition 8.2.2 is overly restrictive in that it disallows the evaluation of a key K to be revealed after a ciphertext of the form $\mathbf{E}_K(M)$ has been evaluated. As such, by imposing such a restriction, we are able to analyze security of protocols only against adversaries that corrupt protocol users non-adaptively. Proving computational security of protocols against adaptive corruptions requires very different techniques, and forms the topic of discussion of the next chapter.

For any adversary A given black-box access to the procedure $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$, let $\hat{\mathcal{M}}(\mathsf{A})$ denote the sequence of messages M for which A issues the query $\texttt{eval}(M)$; each message is included in $\hat{\mathcal{M}}(\mathsf{A})$ in the order of its occurrence as an argument of an evaluation query. Like $\mathcal{M}(\mathsf{A})$, $\hat{\mathcal{M}}(\mathsf{A})$ is also a random variable depending upon the coins of A and $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$.

Definition 8.2.3 An adversary A is called *compliant* if in any execution involving interaction with $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$,

- $\mathcal{M}(\mathsf{A}) \cup \mathcal{K}(\mathsf{A})$ is safe; and
- $\hat{\mathcal{M}}(\mathsf{A})$ is well-ordered.

8.3 The Soundness Theorem

For any adversary A, let $\mathsf{A}^{\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}}$ denote the random variable corresponding to the output of A when given black-box access to $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$. The adaptive advantage of A against $\mathbb{P}$ and $\mathbb{G}$ is defined as the following quantity:

$$\Delta^{\mathbb{P},\mathbb{G}}_{\text{ADPT}}(\mathsf{A}) = \left| \mathbf{P}[\mathsf{A}^{\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},0}} = 1] = \mathbf{P}[\mathsf{A}^{\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},1}} = 1] \right|$$

As usual, both probabilities are taken over the coins used by A as well as by $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$.

Let $n, q, d \in I\!N$. We refer to A as an (n, q, d)-adversary if in every execution, the number of purely random symbolic keys used in A's queries (that is, the size of the set $\{R_i \mid \exists K \in \mathit{keys}(\mathcal{M}(\mathsf{A})) \cup \mathcal{K}(\mathsf{A}) \text{ s.t. } \mathsf{root}(K) = R_i\}$) is at most n, the size of $\mathcal{M}(\mathsf{A})$ is at most q, and every pseudo-random key in $\mathit{enckeys}(\mathcal{M}(\mathsf{A})) \cup \mathcal{K}(\mathsf{A})$ has depth at most d.

Definition 8.3.1 Let $\mathbb{P} = (\mathbb{E}, \mathbb{D})$ be a symmetric-key encryption scheme and $\mathbb{G}$ a length-doubling pseudo-random generator. $\mathbb{P}$ and $\mathbb{G}$ are called $(\tau, n, q, d, \epsilon)$-*secure against partial adaptive attacks* if for every valid and compliant (n, q, d)-adversary A that runs in time at most τ, $\Delta^{\mathbb{P},\mathbb{G}}_{\text{ADPT}}(\mathsf{A}) \leq \epsilon$.

By "partial" adaptive attacks, we refer to the fact that we consider security only against adversaries that satisfy the well-ordered constraint in the above definition; in particular, adversaries that can corrupt keys adaptively are not considered. The following is the main result of this chapter:

Theorem 8.3.2 Let $\mathbb{G}$ be a length-doubling pseudo-random generator and $\mathbb{P} = (\mathbb{E}, \mathbb{D})$ a symmetric-key encryption scheme. If $\mathbb{G}$ is (τ_1, ϵ_1)-secure and $\mathbb{P}$ is (τ_2, ϵ_2)-secure against chosen plaintext attacks, then they are together $(\tau, n, q, d, \epsilon)$-secure against partial adaptive attacks for any parameters τ, n, q, d, ϵ satisfying

$$\begin{aligned} \tau &= \min\{\tau_1, \tau_2\} - n \cdot (\tau(\mathbb{R}) + 2^d \tau(\mathbb{G}) + q\tau(\mathbb{E})) - O(1) \\ \epsilon &= 2n2^d \cdot (5d\epsilon_1 + 2\epsilon_2) \end{aligned}$$

Notice that the reduction factor in the above theorem (the quantity that relates ϵ to ϵ_1 and ϵ_2) is linear in the number of purely random keys involved, n, but exponential in the maximum depth d of all pseudo-random keys. Since in most security protocols (and particularly, in the ones we consider in this thesis), the depth of pseudo-random keys generated during any protocol execution is logarithmic in the number of protocol users, application of the theorem to such protocols still gives a polynomial reduction factor in the security analysis. The question of whether one can prove a result analogous to Theorem 8.3.2 with a reduction factor that is sub-exponential in the depth of pseudo-random keys is left open by this thesis.

8.4 Application to GKD protocols

Before we give the proof of Theorem 8.3.2, we illustrate how it is useful in the context of analyzing GKD protocols against non-adaptive adversaries.

For any GKD protocol Π, and for any set of rekey messages $\mathcal{M}^{\Pi}_{\mathcal{S}^{(t)}}$ output by it, an *ordering* of $\mathcal{M}^{\Pi}_{\mathcal{S}^{(t)}}$ is any sequence that contains all and only the elements of $\mathcal{M}^{\Pi}_{\mathcal{S}^{(t)}}$. We say that a sequence of rekey message-sets, $(\mathcal{M}^{\Pi}_{\mathcal{S}^{(1)}}, \mathcal{M}^{\Pi}_{\mathcal{S}^{(2)}}, \ldots, \mathcal{M}^{\Pi}_{\mathcal{S}^{(t)}})$ output by the protocol is well-ordered if for every $t' \in [t]$, there exists an ordering $\hat{\mathcal{M}}^{\Pi}_{\mathcal{S}^{(t')}}$ of $\mathcal{M}^{\Pi}_{\mathcal{S}^{(t')}}$ such that the concatenation of the sequences $\hat{\mathcal{M}}^{\Pi}_{\mathcal{S}^{(1)}}, \hat{\mathcal{M}}^{\Pi}_{\mathcal{S}^{(2)}}, \ldots, \hat{\mathcal{M}}^{\Pi}_{\mathcal{S}^{(t)}}$ is well-ordered (satisfies Definition 8.2.2).

Definition 8.4.1 An $\mathbf{n}$-user GKD protocol Π is called *compliant* if for all $t > 0$, for all sequences $\overrightarrow{\mathcal{S}}^{(t)} = (\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(t)}) \in (2^{[\mathbf{n}]})^t$, the following is true:

(a) For every $t' \in [t]$, the group key $K^{(t')}$ is used neither as an encryption key nor as an input to $\mathbf{G}$ in any message in $\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}} \cup \{K_1, \ldots, K_{\mathbf{n}}\}$.

(b) $\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}} \cup \{K_1, \ldots, K_{\mathbf{n}}\}$ is safe and $(\mathcal{M}^{\Pi}_{\mathcal{S}^{(1)}}, \ldots, \mathcal{M}^{\Pi}_{\mathcal{S}^{(t)}})$ is well-ordered.

Let $\hat{n}, \hat{q}, \hat{d} : I\!N \to I\!N$. We say that a protocol Π for $\mathbf{n}$ users is an $(\hat{n}, \hat{q}, \hat{d})$-GKD protocol if for all $t > 0$, for all sequences $\overrightarrow{\mathcal{S}}^{(t)} \in (2^{[\mathbf{n}]})^t$, the number of purely random keys in $\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ is at most $\hat{n}(t)$, the size of $\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ is at most $\hat{q}(t)$, and the depth of any key in this set (or in $\{K_1, \ldots, K_{\mathbf{n}}\}$) is at most $\hat{d}(t)$.

Theorem 8.4.2 Let $\mathbb{G}$ be a length-doubling pseudo-random generator that is (τ_1, ϵ_1)-secure and $\mathbb{P} = (\mathbb{E}, \mathbb{D})$ a symmetric-key encryption scheme that is (τ_2, ϵ_2) secure (against chosen plaintext attacks). Let Π be a compliant $(\hat{n}, \hat{q}, \hat{d})$-GKD protocol for $\mathbf{n}$ users. If Π is strongly secure against single-user (resp. collusion) attacks in the symbolic model, then $\Pi^{\mathbb{P},\mathbb{G}}$ is (τ, t, ϵ)-non-adaptively secure against single-user (resp. collusion) attacks in the computational model, for any τ, t and ϵ satisfying:

$$\begin{aligned}\tau &= \min\{\tau_1, \tau_2\} - \hat{n}(t) \cdot (\tau(\mathbb{R}) + 2^{\hat{d}(t)}\tau(\mathbb{G}) + \hat{q}(t)\tau(\mathbb{E})) - \mathbf{n} \cdot (\tau(\mathbb{R}) + 2^{\hat{d}(t)}\tau(\mathbb{G})) - O(1) \\ \epsilon &= 2\hat{n}(t)2^{\hat{d}(t)} \cdot (5\hat{d}(t)\epsilon_1 + 2\epsilon_2)\end{aligned}$$

Proof: We prove the theorem only for the case of security against collusion attacks; the proof for the other case is very similar and is, thus, omitted.

```
ADVERSARY A'

Run the setup program S of $\Pi$ and store its output $\mathcal{Z}^{(0)} \cup \{K_1, \ldots, K_{\mathbf{n}}\}$.
Initialize a counter $t_1 \leftarrow 0$.
When A issues a corruption query $\texttt{corrupt}(i)$,
        Send $\texttt{eval}(K_i)$ to $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$. Let $k_i$ be the response.
        Return $k_i$ to A.
When A issues an execution query $\texttt{execute}(\mathcal{S})$,
        $t_1 \leftarrow t_1 + 1$.
        Run the key distribution program C of $\Pi$ on input $\mathcal{S}^{(t_1)} := \mathcal{S}$ and $\mathcal{Z}^{(t_1-1)}$.
        Order its output $\mathcal{M}^{\Pi}_{\mathcal{S}^{(t_1)}}$ into a well-ordered sequence $\hat{\mathcal{M}}^{\Pi}_{\mathcal{S}^{(t_1)}}$.
        For each $M_i \in \hat{\mathcal{M}}^{\Pi}_{\mathcal{S}^{(t_1)}}$ in order, send $\texttt{eval}(M_i)$ to $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$; let $m_i$ be
                the response.
        Return the set $\mathfrak{M}^{\Pi}_{\mathcal{S}^{(t_1)}} = \{(M_i, m_i) \mid M_i \in \mathcal{M}^{\Pi}_{\mathcal{S}^{(t_1)}}\}$ to A.
        Store the updated state of C, $\mathcal{Z}^{(t_1)}$.
When A issues a challenge query $\texttt{challenge}(t')$,
        Send $\texttt{challenge}(K^{(t')})$ to $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$. Let $k_{t'}$ be the response.
        Send $k_{t'}$ to A.
Output whatever A outputs.
```

Figure 8.2: The adversary constructed for the proof of Theorem 8.4.2.

Let Π be any compliant $(\hat{n}, \hat{q}, \hat{d})$-GKD protocol for $\mathbf{n}$ users, $\mathbb{G}$ a length-doubling (τ_1, ϵ_1)-secure PRG and $\mathbb{P}$ a (τ_2, ϵ_2)-secure symmetric-key encryption scheme. Suppose that Π is secure against collusion attacks in the symbolic model but $\Pi^{\mathbb{P},\mathbb{G}}$ does not satisfy the definition of (τ, t, ϵ)-security in the computational model for some set of parameters τ, t and ϵ defined in the theorem. That is, there exists some non-adaptive adversary A that runs in time τ, makes at most t execution queries, but for which $\Delta^{\Pi^{\mathbb{P},\mathbb{G}}}_{\text{GKD}}(\mathsf{A}) > \epsilon$. We claim that any such adversary A can be used to construct a valid and compliant $(\hat{n}(t), \hat{q}(t), \hat{d}(t))$-adversary A′ in the computational game defined in Section 8.1 such that A′ runs in time $\tau + \mathbf{n} \cdot (\tau(\mathbb{R}) + 2^{\hat{d}(t)}\tau(\mathbb{G})) + O(1)$, and still $\Delta^{\mathbb{P},\mathbb{G}}_{\text{ADPT}}(\mathsf{A}') > \epsilon$. This falsifies Theorem 8.3.2, thus implying that Theorem 8.4.2 must be correct.

At a high level, the adversary A′ works as follows: It emulates the execution of the protocol Π, invokes A in a black-box manner and for each query that A makes, it executes

Π in accordance with the query. However, A expects to receive bitstrings in reply, and so, A' uses its oracle $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$ to evaluate all the symbolic messages and keys generated by Π before providing them as replies to A's queries. In the end, it outputs whatever A outputs. The detailed construction of A' is given in figure Figure 8.2.

Since A is non-adaptive (and thus issues all its corruption queries before issuing queries of other types), and since the rekey messages output by Π satisfy the well-ordered property, the message-sequence $\hat{\mathcal{M}}(\mathsf{A}')$ is well-ordered. Furthermore, since the rekey messages output by Π in any execution are safe, this implies that the set $\mathcal{M}(\mathsf{A}) \cup \mathcal{K}(\mathsf{A})$ is also safe. Thus, A' is a compliant adversary.

The validity of A' follows from the fact that group keys in Π are never used as encryption keys or as inputs to $\mathbf{G}$ and the fact that Π is secure in the symbolic model. That A' is a $(\hat{n}(t), \hat{q}(t), \hat{d}(t))$-adversary follows from the fact that Π is a $(\hat{n}, \hat{q}, \hat{d})$-GKD protocol and that A issues at most t execution queries.

Lastly, it is easy to check that A' runs in time $\tau + \mathbf{n} \cdot (\tau(\mathbb{R}) + 2^{\hat{d}(t)}\tau(\mathbb{G})) + O(1)$ and that $\Delta^{\mathbb{P},\mathbb{G}}_{\text{ADPT}}(\mathsf{A}')$ is the same as $\Delta^{\Pi^{\mathbb{P},\mathbb{G}}}_{\text{GKD}}(\mathsf{A})$, which is greater than ϵ. A contradiction to Theorem 8.3.2 has been reached. ▌

8.4.1 Analysis of Protocols

Next, we use Theorem 8.4.2 to analyze the protocols we presented in the first part of the thesis. Consider first the LKH protocols. Recall that the protocols $\mathsf{plain\text{-}LKH}^+$ and $\mathsf{improved\text{-}LKH}^+$ are strongly secure against collusion attacks in the symbolic model (Theorem 5.1.1). Using this fact, and Theorem 8.4.2, we can directly establish collusion-resistance of these protocols in the computational model. For simplicity, we consider collusion-resistance of the protocols against adversaries whose execution queries form simple sequences: for every $t > 0$, the tth execution query, $\texttt{execute}(\mathcal{S}^{(t)})$, issued by the adversary is such that either $\mathcal{S}^{(t)} = \mathcal{S}^{(t-1)} \cup \{i\}$ for some $i \notin \mathcal{S}^{(t-1)}$ or $\mathcal{S}^{(t)} = \mathcal{S}^{(t-1)} \setminus \{i\}$ for some $i \in \mathcal{S}^{(t-1)}$. ($\mathcal{S}^{(0)}$ is assumed to be the empty set.) In the theorems below, the term "security against collusion attacks" refers to security with respect to such adversaries only.

Theorem 8.4.3 Let $\mathbf{n} \geq 2$ and $d \in \{2, 3, \ldots, \mathbf{n}\}$. The d-ary instance of the plain-LKH$^+$ protocol, when implemented for $\mathbf{n}$ users using a (τ_1, ϵ_1)-secure encryption scheme $\mathbb{P} = (\mathbb{E}, \mathbb{D})$, is $(\tau_{\mathsf{p}}, t, \epsilon_{\mathsf{p}})$-non-adaptively secure against collusion attacks in the computational model for any $\tau_{\mathsf{p}}, t, \epsilon_{\mathsf{p}}$ satisfying:

$$\begin{aligned} \tau_{\mathsf{p}} &= \tau_1 - (t(\lceil \log_d(\mathbf{n}) \rceil - 1) + \mathbf{n}) \cdot (\tau(\mathbb{R}) + dt\lceil \log_d(\mathbf{n}) \rceil \tau(\mathbb{E})) - \mathbf{n}\tau(\mathbb{R}) - O(1) \\ \epsilon_{\mathsf{p}} &= 4(t(\lceil \log_d(\mathbf{n}) \rceil - 1) + \mathbf{n}) \cdot \epsilon_1 \end{aligned}$$

Theorem 8.4.4 Let $\mathbf{n} \geq 2$ and $d \in \{2, 3, \ldots, \mathbf{n}\}$. The d-ary instance of the improved-LKH$^+$ protocol, when implemented for $\mathbf{n}$ users using a (τ_1, ϵ_1)-secure pseudo-random generator $\mathbb{G}$ and a (τ_2, ϵ_2)-secure encryption scheme $\mathbb{P} = (\mathbb{E}, \mathbb{D})$, is $(\tau_{\mathsf{i}}, t, \epsilon_{\mathsf{i}})$-non-adaptively secure against collusion attacks in the computational model for any $\tau_{\mathsf{i}}, t, \epsilon_{\mathsf{i}}$ satisfying:

$$\begin{aligned} \tau_{\mathsf{i}} &= \min\{\tau_1, \tau_2\} - (t + \mathbf{n}) \cdot (\tau(\mathbb{R}) + 2^{\lceil \log_d(\mathbf{n}) \rceil} \tau(\mathbb{G}) + t(d-1)\lceil \log_d(\mathbf{n}) \rceil \tau(\mathbb{E})) \\ &\qquad - \mathbf{n} \cdot (\tau(\mathbb{R}) + 2^{\lceil \log_d(\mathbf{n}) \rceil} \tau(\mathbb{G})) - O(1) \\ \epsilon_{\mathsf{i}} &= 2^{1 + \lceil \log_d(\mathbf{n}) \rceil} (t + \mathbf{n}) \cdot (5\lceil \log_d(\mathbf{n}) \rceil \epsilon_1 + 2\epsilon_2) \end{aligned}$$

Both these theorems follow immediately from Theorem 8.4.2, given the fact that plain-LKH$^+$ and improved-LKH$^+$ are collusion-resistant in the symbolic model (Theorem 5.1.1), and that both protocols are compliant in the sense of Definition 8.4.1. The latter follows from the facts that group keys in plain-LKH$^+$ and improved-LKH$^+$ are used neither for encryption nor for pseudo-random generation, that both protocols are safe, and that rekey messages generated by both protocols in any execution satisfy the well-ordered property. The choice of the parameters $\tau_{\mathsf{p}}, \epsilon_{\mathsf{p}}, \tau_{\mathsf{i}}$ and ϵ_{i} is based on the following observation: for any t-time execution of plain-LKH$^+$ (resp. improved-LKH$^+$), the number of purely random keys generated by the protocol is at most $t \cdot (\lceil \log_d(\mathbf{n}) \rceil - 1) + \mathbf{n}$ (resp. $\mathbf{n} + t$), the number of ciphertexts output as rekey messages is at most $td \cdot \lceil \log_d(\mathbf{n}) \rceil$ (resp. $t(d-1) \cdot \lceil \log_d(\mathbf{n}) \rceil$), and the maximum depth of any pseudo-random key is 0 (resp. $\lceil \log_d(\mathbf{n}) \rceil$). Notice that for both the protocols, the security reduction factors we obtain are polynomial in $\mathbf{n}$ and t.

Security statements of the above nature cannot be made for plain-LKH and improved-LKH [46, 45, 10] since neither of these protocols satisfy the compliance property required by Theorem 8.4.2. Indeed, since group keys are used to encrypt other keys in both these protocols, their pseudo-randomness (in the sense of Definition 7.4.3) is not even possible to prove. In principle, using group keys for rekeying operations (as done in plain-LKH and improved-LKH) as well as for performing other cryptographic applications (like group message encryption or group message authentication) could lead to complete compromise of these keys *even by a passive eavesdropper on the channel*.

Now consider the subset cover protocols. Recall that both the CS protocol and the SD protocol are strongly secure against collusion attacks in the symbolic model (Theorems 5.2.1 and 5.2.2). By observing that both these protocols are compliant, and by invoking Theorem 8.4.2 again, we conclude the following.

Theorem 8.4.5 The CS protocol, when implemented for $\mathbf{n}$ users using a (τ_1, ϵ_1)-secure encryption scheme $\mathbb{P} = (\mathbb{E}, \mathbb{D})$, is $(\tau_{\mathsf{c}}, t, \epsilon_{\mathsf{c}})$-non-adaptively secure against collusion attacks in the computational model, for any $\tau_{\mathsf{c}}, t, \epsilon_{\mathsf{c}}$ satisfying:

$$\begin{aligned}
\tau_{\mathsf{c}} &= \tau_1 - (t + 2\mathbf{n}) \cdot (\tau(\mathbb{R}) + \mathbf{n}t\lceil \log_2(\mathbf{n}) \rceil \tau(\mathbb{E})) - \mathbf{n}\tau(\mathbb{R}) - O(1) \\
\epsilon_{\mathsf{c}} &= 4(t + 2\mathbf{n}) \cdot \epsilon_1
\end{aligned}$$

Theorem 8.4.6 The SD protocol, when implemented for $\mathbf{n}$ users using a (τ_1, ϵ_1)-secure pseudo-random generator $\mathbb{G}$ and a (τ_2, ϵ_2)-secure encryption scheme $\mathbb{P} = (\mathbb{E}, \mathbb{D})$, is $(\tau_{\mathsf{s}}, t, \epsilon_{\mathsf{s}})$-non-adaptively secure against collusion attacks in the computational model, for any $\tau_{\mathsf{s}}, t, \epsilon_{\mathsf{s}}$ satisfying:

$$\begin{aligned}
\tau_{\mathsf{s}} &= \min\{\tau_1, \tau_2\} - (t + 2\mathbf{n}) \cdot (\tau(\mathbb{R}) + 4\mathbf{n}^2\tau(\mathbb{G}) + t\mathbf{n}\tau(\mathbb{E})) \\
&\qquad - \mathbf{n} \cdot (\tau(\mathbb{R}) + 4\mathbf{n}^2\tau(\mathbb{G})) - O(1) \\
\epsilon_{\mathsf{s}} &= 8\mathbf{n}^2(t + 2\mathbf{n}) \cdot (10\lceil \log_2(\mathbf{n}) \rceil \epsilon_1 + 2\epsilon_2)
\end{aligned}$$

In comparison with the security results in [34], the proofs of the above theorems are simpler, and both theorems follow almost immediately from the symbolic security of the protocols under consideration. The choice of the parameters is based on the observation that in any t-time execution of the CS (resp. SD) protocol, the number of purely random keys generated by the protocol is at most $2\mathbf{n}+t$, the number of ciphertexts output as rekey messages is at most $\mathbf{n}t \cdot \lceil \log_2(\mathbf{n}) \rceil$ (resp. $\mathbf{n}t$), and the maximum depth of any pseudo-random key used in the protocol is 0 (resp. $2\lceil \log_2(\mathbf{n}) \rceil$). We remark that the above theorems establish security of these protocols against non-adaptive adversaries only, whereas in [34], security against adaptive adversaries is also proven. The issue of proving security against adaptive adversaries is addressed separately in Chapter 9.

8.5 Proof of Theorem 8.3.2

We conclude the chapter with a proof of the main result, Theorem 8.3.2. Given any (n, q, d)-adversary A that is valid and compliant, and that runs in time τ, we construct a set of adversaries, which fall into three categories:

- adversaries of $\mathsf{type\text{-}1}$, denoted $\mathsf{A}^1_{d',b_e,b_c}$, one for each $d' \in \{0, 1, \ldots, d\}$ and $b_e, b_c \in \{0, 1\}$ and each running in time at most τ_1;
- adversaries of $\mathsf{type\text{-}2}$, denoted $\mathsf{A}^2_{b_c}$, one for each $b_c \in \{0, 1\}$ and each running in time at most τ_2; and
- adversaries of $\mathsf{type\text{-}3}$, denoted $\mathsf{A}^3_{d'}$, one for each $d' \in \{0, 1, \ldots, d\}$ and each running in time at most τ_1.

We construct these adversaries in a manner such that if $\Delta^{\mathbb{P},\mathbb{G}}_{\mathrm{ADPT}}(\mathsf{A}) > \epsilon$, then for some tuple $(d', b_e, b_c) \in \{0, 1, \ldots, d\} \times \{0, 1\} \times \{0, 1\}$, either $\Delta^{\mathbb{G}}_{\mathrm{PRG}}(\mathsf{A}^1_{d',b_e,b_c}) > \epsilon_1$, or $\Delta^{\mathbb{G}}_{\mathrm{PRG}}(\mathsf{A}^3_{d'}) > \epsilon_1$ or $\Delta^{\mathbb{P}}_{\mathrm{ENC}}(\mathsf{A}^2_{b_c}) > \epsilon_2$. Such a condition contradicts the security assumptions we made for $\mathbb{P}$ and $\mathbb{G}$, thus implying that $\Delta^{\mathbb{P},\mathbb{G}}_{\mathrm{ADPT}}(\mathsf{A})$ is bounded from above by ϵ for any A that is valid and compliant and that runs in time at most τ.

8.5.1 Notations

Let Σ^d be the set of all binary sequences of length at most d; this includes the empty sequence, which we denote by ε. For any $\tilde{d} \in [d]$, any sequence $\sigma = (b_1, \ldots, b_{\tilde{d}}) \in \Sigma^d$ and any $R_i \in \mathcal{R}$, let $\mathbf{G}_\sigma(R_i)$ denote the symbolic key $\mathbf{G}_{b_{\tilde{d}}}(\mathbf{G}_{b_{\tilde{d}-1}}(\cdots \mathbf{G}_{b_1}(R_i)\cdots))$, and let $\mathbf{G}_\varepsilon(R_i)$ denote the key R_i. Since the number of purely random symbolic keys used in A's queries is at most n and since all keys are of depth at most d, we can, without loss of generality, assume that all keys used in A's queries are of the form $\mathbf{G}_\sigma(R_i)$ for some $i \in [n]$ and some $\sigma \in \Sigma^d$.

Let $\tilde{d} \in [d]$ and $\sigma = (b_1, \ldots, b_{\tilde{d}}) \in \Sigma^d$. For any $d' \in \{0, 1, \ldots, d\}$, let $\mathsf{prefix}(\sigma, d')$ denote the d'-long prefix of σ if $d' \le \tilde{d}$; let it denote σ otherwise. That is,

$$\mathsf{prefix}(\sigma, d') = \begin{cases} \varepsilon & \text{if } d' = 0 \\ (b_1, \ldots, b_{d'}) & \text{if } 1 \le d' \le \tilde{d} \\ \sigma & \text{otherwise} \end{cases}$$

For any $d' \in \{1, \ldots, d\}$, let $\mathsf{near\_prefix}(\sigma, d')$ be defined as follows:

$$\mathsf{near\_prefix}(\sigma, d') = \begin{cases} (b_1, \ldots, b_{d'-1}, 1 - b_{d'}) & \text{if } 1 \le d' \le \tilde{d} \\ (b_1, \ldots, b_{\tilde{d}-1}, 1 - b_{\tilde{d}}) & \text{otherwise} \end{cases}$$

We extend the notion of prefixes and near-prefixes to symbolic keys in the natural manner. For any $\sigma \in \Sigma^d$, any key $K = \mathbf{G}_\sigma(R_i)$ and any $d' \in \{0, 1, \ldots, d\}$, let $\mathsf{prefix}(K, d') = \mathbf{G}_{\mathsf{prefix}(\sigma, d')}(R_i)$. If $K = R_i$, $\mathsf{prefix}(K, d')$ denotes R_i for any $d' \ge 0$. Let $\mathsf{near\_prefixes}(K)$ denote the set $\{\mathbf{G}_{\mathsf{near\_prefix}(\sigma, d')}(R_i)\}_{d'=1,\ldots,d}$; this set is empty if $K = R_i$.

8.5.2 The Reduction

Each of our adversaries (of type-1, type-2 as well as type-3) works in two phases: a *setup* phase and an *execution* phase. The setup phase is similar for all types of adversaries and is illustrated in figure Figure 8.3. In this phase, the adversaries essentially initialize some variables (to be used in the execution phase) and generate a

```
PHASE I: SETUP
001.  Initialize two maps $\psi$ and $\psi'$, a set $\mathcal{M}$, and a list $\mathcal{L}$, all to be empty
002.  Initialize a boolean value good_event $\leftarrow 0$
003.  Sample $i$ and $\sigma$ uniformly at random from $[n]$ and $\Sigma^d$ respectively
004.  Let $K_{\mathsf{crit}} = \mathbf{G}_\sigma(R_i)$.
      ($K_{\mathsf{crit}}$ stands for the "critical" (symbolic) key. Adversaries of type-2 asso-
      ciate $K_{\mathsf{crit}}$ with the key used by their oracle $\mathcal{O}_b^{\mathbb{P}}$ while those of type-1 and
      type-3 associate it with a key derived from their input.)
```

Figure 8.3: The setup phase for each of our adversaries for Theorem 8.3.2.

symbolic key K_{crit} (called the *critical key*), uniformly at random from the space of all symbolic keys of depth d that can be generated from the set $\{R_1, \ldots, R_n\}$.

The execution phase of each of the adversary types is shown in three different figures—figures Figure 8.4, Figure 8.6 and Figure 8.8. Adversaries of each type execute A in a black-box manner and use two sub-procedures to evaluate the keys and the messages generated by A in the execution. The procedure used to evaluate messages is denoted $eval^{(i)}$ for type-i adversaries and that to evaluate keys is denoted $key\_eval^{(i)}$. Each of the type-1 and type-3 adversaries receive, as input, a bitstring $\mathtt{s} = \mathtt{s}_0 \| \mathtt{s}_1$ (such that $\mathtt{s}_0, \mathtt{s}_1 \in \{0,1\}^\eta$), which is sampled from either $X_0^{\mathbb{G}}$ or $X_1^{\mathbb{G}}$, and its objective is to deduce which distribution $\mathtt{s}$ is sampled from. Adversaries of both these types use $\mathtt{s}$ to evaluate keys created by A via a common procedure $key\_eval^{(1)}$ shown in figure Figure 8.5. Each of the type-2 adversaries is given black-box access to a procedure $\mathcal{O}_b^{\mathbb{P}}$, and its objective is to guess the bit b. Each such adversary uses $\mathcal{O}_b^{\mathbb{P}}$ to evaluate ciphertexts queried by A, as shown in figure Figure 8.6. It is easy to check that the running time of each of the adversaries is at least $\tau + n \cdot (\tau(\mathbb{R}) + 2^d\tau(\mathbb{G}) + q\tau(\mathbb{E})) + O(1)$, which is bounded from above by both τ_1 and τ_2, as desired.

```
ADVERSARY $\mathsf{A}^1_{d',b_e,b_c}$, PHASE II: EXECUTION
000. Let $K_{\mathsf{target}} = \mathsf{prefix}(K_{\mathsf{crit}}, d')$.
     (While evaluating symbolic keys, $\mathsf{A}^1_{d',b_e,b_c}$ maps $\mathbf{G}(K_{\mathsf{target}})$ to its input
     $\mathtt{s}_0 \,\|\, \mathtt{s}_1$.)
010. Run A.
020. When A makes a query of the form eval$(M)$,
021.     Let $\mathcal{M} \leftarrow \mathcal{M} \cup \{M\}$.
022.     Let $K_{\mathsf{first}}$ be the key s.t. $K_{\mathsf{first}}$ occurs in $M$, $K_{\mathsf{first}} \notin \mathbf{Rec}(\mathcal{M})$
             and there exists $M'$ for which $\mathbf{E}_{K_{\mathsf{first}}}(M') \in \mathbf{Rec}(\mathcal{M})$.
             (If there is no such key, $K_{\mathsf{first}} = \perp$)
023.     Reply with $eval^{(1)}(M, K_{\mathsf{first}}, 0)$
030. When A makes a query of the form challenge$(K)$,
031.     If $b_c = 0$, reply with $key\_eval^{(1)}(K, 0)$.
032.     Else, reply with a uniformly random sample from $\{0,1\}^\eta$.
040. When A halts, output whatever A outputs.

procedure $eval^{(1)}(M, K_{\mathsf{first}}, \mathsf{flag})$
100. If $M = K$ for some symbolic key $K$,
101.     Return $key\_eval^{(1)}(K, \mathsf{flag})$.
110. If $M = \mathbf{E}_K(M')$ for some symbolic key $K$ and symbolic message $M'$,
120.     If $\mathsf{flag} = 0$ and $K \neq K_{\mathsf{first}}$,
121.         Return $\mathbb{E}_{key\_eval^{(1)}(K,0)}(eval^{(1)}(M', K_{\mathsf{first}}, 0))$.
130.     If $\mathsf{flag} = 0$ and $K = K_{\mathsf{first}}$ and $K \neq K_{\mathsf{crit}}$,
131.         If $K \notin \mathcal{L}$ and $\mathsf{good\_event} = 0$, append $K$ to $\mathcal{L}$.
132.         If $K \in \mathcal{L}$, return $\mathbb{E}_{key\_eval^{(1)}(K,0)}(eval^{(1)}(M', K_{\mathsf{first}}, 1))$.
133.         Else, return $\mathbb{E}_{key\_eval^{(1)}(K,0)}(eval^{(1)}(M', K_{\mathsf{first}}, 0))$.
140.     If $\mathsf{flag} = 0$ and $K = K_{\mathsf{first}} = K_{\mathsf{crit}}$,
141.         $\mathsf{good\_event} \leftarrow 1$.
142.         Return $\mathbb{E}_{key\_eval^{(1)}(K,0)}(eval^{(1)}(M', K_{\mathsf{first}}, b_e))$
150.     If $\mathsf{flag} = 1$, return $\mathbb{E}_{key\_eval^{(1)}(K,1)}(eval^{(1)}(M', K_{\mathsf{first}}, 1))$.
```

Figure 8.4: The execution phase for type-1 adversaries.

8.5.3 The Analysis

For any execution of adversary A, let $outkeys(\mathsf{A})$ denote the random variable corresponding to the subset of $enckeys(\mathcal{M}(\mathsf{A}))$ such that for every $K \in outkeys(\mathsf{A})$, the following is true:

```
procedure $key\_eval^{(1)}(K, \mathsf{flag})$
200.  If $\mathsf{flag} = 0$, do the following:
201.      If $K_{\mathsf{target}} \in \mathbf{Rec}(K)$,
202.          If $K = K_{\mathsf{target}} = K_{\mathsf{crit}}$ and $\mathsf{good\_event} = 1$, do the following:
203.              If $\psi(K)$ is undefined, set $\psi(K) \leftarrow \mathbb{R}$. Return $\psi(K)$.
204.          Else, output a random bit. $\mathbf{Halt}$!
205.      If $K = \mathbf{G}_b(K_{\mathsf{target}})$ for some $b \in \{0,1\}$,
206.          If $\mathsf{good\_event} = 1$, return $\mathtt{s}_b$.
207.          Else, output a random bit. $\mathbf{Halt}$!
208.      If $K \in \mathsf{near\_prefixes}(K_{\mathsf{target}})$,
209.          If $\psi'(K)$ is undefined, set $\psi'(K) \leftarrow \mathbb{R}$. Return $\psi'(K)$.
210.      If $K \in \mathcal{R}$,
211.          If $\psi(K)$ is undefined, set $\psi(K) \leftarrow \mathbb{R}$. Return $\psi(K)$.
212.      If $K = \mathbf{G}_{b'}(K')$ for some $b' \in \{0,1\}$ and $K' \neq K_{\mathsf{target}}$,
          Return $\mathbb{G}_{b'}(key\_eval^{(1)}(K', 0))$.
220.  If $\mathsf{flag} = 1$, return a uniformly random sample from $\{0,1\}^\eta$
```

Figure 8.5: Procedure $key\_eval^{(1)}$ used by type-1 and type-3 adversaries.

- $K \notin \mathbf{Rec}(\mathcal{M}(\mathsf{A}))$; and
- There exists some M' such that $\mathbf{E}_K(M') \in \mathbf{Rec}(\mathcal{M}(\mathsf{A}))$.

Intuitively, $outkeys(\mathsf{A})$ is the set of all unrecoverable encryption keys occurring in A's queries such that each such key occurs as the "outermost" unrecoverable key in some query of A.

Before proceeding to analyze the three types of adversaries we have constructed, we make one important remark about them. Observe line 022 for each of these adversaries. The key K_{first} defined in this line is the outermost unrecoverable key in the message M queried by A. We claim that K_{first} is, in fact, contained in $outkeys(\mathsf{A})$. This follows from the fact that A is a compliant adversary: Since A is compliant, its evaluation queries satisfy the well-ordered property (Definition 8.2.2) and so K_{first} (or any pseudo-random inverse of it) *cannot* be used as a message key in any other message that is evaluated *after* M. This, in turn, implies that K_{first} cannot be recoverable in any

```
ADVERSARY $\mathsf{A}^2_{b_c}$, PHASE II: EXECUTION
010. Run $\mathsf{A}$.
020. When $\mathsf{A}$ makes a query of the form eval$(M)$,
021.     Let $\mathcal{M} \leftarrow \mathcal{M} \cup \{M\}$.
022.     Let $K_{\mathsf{first}}$ be the key s.t. $K_{\mathsf{first}}$ occurs in $M$, $K_{\mathsf{first}} \notin \mathbf{Rec}(\mathcal{M})$
             and there exists $M'$ for which $\mathbf{E}_{K_{\mathsf{first}}}(M') \in \mathbf{Rec}(\mathcal{M})$.
             (If there is no such key, $K_{\mathsf{first}} = \perp$)
023.     Reply with $eval^{(2)}(M, K_{\mathsf{first}}, 0)$
030. When $\mathsf{A}$ makes a query of the form challenge$(K)$,
031.     If $b_c = 0$, reply with $key\_eval^{(2)}(K, 0)$.
032.     Else, reply with a uniformly random sample from $\{0,1\}^\eta$.
040. When $\mathsf{A}$ halts, output whatever $\mathsf{A}$ outputs.

procedure $eval^{(2)}(M, K_{\mathsf{first}}, \mathsf{flag})$
100. If $M = K$ for some symbolic key $K$,
101.     Return $key\_eval^{(2)}(K, \mathsf{flag})$.
110. If $M = \mathbf{E}_K(M')$ for some symbolic key $K$ and symbolic message $M'$,
120.     If $\mathsf{flag} = 0$ and $K \neq K_{\mathsf{first}}$,
121.         Return $\mathbb{E}_{key\_eval^{(2)}(K,0)}(eval^{(2)}(M', K_{\mathsf{first}}, 0))$.
130.     If $\mathsf{flag} = 0$ and $K = K_{\mathsf{first}}$ and $K \neq K_{\mathsf{crit}}$,
131.         If $K \notin \mathcal{L}$ and $\mathsf{good\_event} = 0$, append $K$ to $\mathcal{L}$.
132.         If $K \in \mathcal{L}$, return $\mathbb{E}_{key\_eval^{(2)}(K,0)}(eval^{(2)}(M', K_{\mathsf{first}}, 1))$.
133.         Else, return $\mathbb{E}_{key\_eval^{(2)}(K,0)}(eval^{(2)}(M', K_{\mathsf{first}}, 0))$.
140.     If $\mathsf{flag} = 0$ and $K = K_{\mathsf{first}} = K_{\mathsf{crit}}$,
141.         $\mathsf{good\_event} \leftarrow 1$.
142.         $m'_0 \leftarrow eval^{(2)}(M', K_{\mathsf{first}}, 0)$
143.         $m'_1 \leftarrow eval^{(2)}(M', K_{\mathsf{first}}, 1)$
144.         Return $\mathcal{O}^{\mathbb{P}}_b(m'_0, m'_1)$
150.     If $\mathsf{flag} = 1$, return $\mathbb{E}_{key\_eval^{(2)}(K,1)}(eval^{(2)}(M', K_{\mathsf{first}}, 1))$.
```

Figure 8.6: The execution phase for type-2 adversaries.

evaluation query made after eval(M), which means $K_{\mathsf{first}} \notin \mathbf{Rec}(\mathcal{M}(\mathsf{A}))$. This, and the the fact that $\mathbf{E}_{K_{\mathsf{first}}}(M') \in \mathbf{Rec}(\mathcal{M}) \subseteq \mathbf{Rec}(\mathcal{M}(\mathsf{A}))$ (M' as defined in line 022), implies that K_{first} is contained in $outkeys(\mathsf{A})$.

We first consider the adversaries of type-1. Let b denote the bit corresponding to the input provided to any adversary of this type: if $b = 0$, the input is sampled

```
procedure $key\_eval^{(2)}(K, \mathsf{flag})$
200.  If $\mathsf{flag} = 0$, do the following:
201.      If $K_{\mathsf{crit}} \in \mathbf{Rec}(K)$, output a random bit. $\mathbf{Halt}$!
202.      If $K \in \mathsf{near\_prefixes}(K_{\mathsf{crit}})$,
203.          If $\psi'(K)$ is undefined, set $\psi'(K) \leftarrow \mathbb{R}$.
204.          Return $\psi'(K)$.
205.      If $K \in \mathcal{R}$,
206.          If $\psi(K)$ is undefined, set $\psi(K) \leftarrow \mathbb{R}$.
207.          Return $\psi(K)$.
208.      If $K = \mathbf{G}_{b'}(K')$ for some key $K'$ and $b' \in \{0,1\}$,
              Return $\mathbb{G}_{b'}(key\_eval^{(2)}(K', 0))$.
210.  If $\mathsf{flag} = 1$, return a uniformly random sample from $\{0,1\}^{\eta}$
```

Figure 8.7: Procedure $key\_eval^{(2)}$ used by type-2 adversaries.

according to $X_0^{\mathbb{G}}$, and if $b = 1$, the input is sampled according to $X_1^{\mathbb{G}}$. Let Bad be the event that the critical key K_{crit} is *not* in $outkeys(\mathsf{A})$ for the execution of A as defined in any adversary of type-1. It is straightforward to check that when event Bad occurs, the output of any such adversary is independent of its input. Furthermore, the occurrence of Bad itself is computationally independent of the input provided to the adversary, as formalized in the following proposition.

Proposition 8.5.1 $|\mathbf{P}[\mathsf{Bad} \mid b = 0] - \mathbf{P}[\mathsf{Bad} \mid b = 1]| \leq \epsilon_1$

Proof: The proof is by contradiction. Any adversary of type-1 for which the condition in the proposition is defied can be transformed into one that outputs 1 if and only if Bad occurs. (This can be done with essentially no loss in running time complexity.) The transformed adversary would violate the (τ_1, ϵ_1)-security of $\mathbb{G}$. ▮

For any $(d', b_e, b_c) \in \{0, 1, \ldots, d\} \times \{0,1\} \times \{0,1\}$, consider the type-1 adversary $\mathsf{A}^1_{d',b_e,b_c}$. Let Θ^1_{d',b_e,b_c} denote the event that the output of $\mathsf{A}^1_{d',b_e,b_c}$ equals 1. So,

$$\Delta^{\mathbb{G}}_{\mathrm{PRG}}(\mathsf{A}^1_{d',b_e,b_c}) = |\mathbf{P}[\Theta^1_{d',b_e,b_c} \mid b = 0] - \mathbf{P}[\Theta^1_{d',b_e,b_c} \mid b = 1]|$$

We expand this quantity in terms of 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.

$$\begin{aligned}
\Delta^{\mathbb{G}}_{\text{PRG}}(\mathsf{A}^1_{d',b_e,b_c}) &= |\mathbf{P}[\Theta^1_{d',b_e,b_c};\ \overline{\mathsf{Bad}} \mid b=0] - \mathbf{P}[\Theta^1_{d',b_e,b_c};\ \overline{\mathsf{Bad}} \mid b=1] \\
&\quad + \mathbf{P}[\Theta^1_{d',b_e,b_c};\ \mathsf{Bad} \mid b=0] - \mathbf{P}[\Theta^1_{d',b_e,b_c};\ \mathsf{Bad} \mid b=1]| \\
&\geq |\mathbf{P}[\Theta^1_{d',b_e,b_c};\ \overline{\mathsf{Bad}} \mid b=0] - \mathbf{P}[\Theta^1_{d',b_e,b_c};\ \overline{\mathsf{Bad}} \mid b=1]| \\
&\quad - |\mathbf{P}[\Theta^1_{d',b_e,b_c};\ \mathsf{Bad} \mid b=0] - \mathbf{P}[\Theta^1_{d',b_e,b_c};\ \mathsf{Bad} \mid b=1]| \\
&= |\mathbf{P}[\Theta^1_{d',b_e,b_c};\ \overline{\mathsf{Bad}} \mid b=0] - \mathbf{P}[\Theta^1_{d',b_e,b_c};\ \overline{\mathsf{Bad}} \mid b=1]| \\
&\quad - \mathbf{P}[\Theta^1_{d',b_e,b_c} \mid \mathsf{Bad}] \cdot |\mathbf{P}[\mathsf{Bad} \mid b=0] - \mathbf{P}[\mathsf{Bad} \mid b=1]| \quad (8.1) \\
&\geq |\mathbf{P}[\Theta^1_{d',b_e,b_c};\ \overline{\mathsf{Bad}} \mid b=0] - \mathbf{P}[\Theta^1_{d',b_e,b_c};\ \overline{\mathsf{Bad}} \mid b=1]| \\
&\quad - \epsilon_1 \quad (8.2)
\end{aligned}$$

Here, equation (8.1) follows from the fact that given the occurrence of Bad, the output of $\mathsf{A}^1_{d',b_e,b_c}$ is independent of the bit b, and equation (8.2) follows from the proposition 8.5.1.

Let Δ^1_{d',b_e,b_c} denote the quantity $\mathbf{P}[\Theta^1_{d',b_e,b_c};\ \overline{\mathsf{Bad}} \mid b=0] - \mathbf{P}[\Theta^1_{d',b_e,b_c};\ \overline{\mathsf{Bad}} \mid b=1]$. Since $\Delta^{\mathbb{G}}_{\text{PRG}}(\mathsf{A}^1_{d',b_e,b_c})$ must be bounded from above by ϵ_1 for any $d' \in \{0,1,\ldots,d\}$ and any $b_e, b_c \in \{0,1\}$, we conclude that for any such values of d', b_e and b_c,

$$\begin{aligned}
|\Delta^1_{d',b_e,b_c}| &\leq \Delta^{\mathbb{G}}_{\text{PRG}}(\mathsf{A}^1_{d',b_e,b_c}) + \epsilon_1 \\
&\leq 2\epsilon_1 \quad (8.3)
\end{aligned}$$

In a similar vein as above, we define events Bad and $\Theta^2_{b_c}$ for adversaries of type-2: Bad is the event that K_{crit} does not occur in $\mathit{outkeys}(\mathsf{A})$ when A is executed by $\mathsf{A}^2_{b_c}$ (as shown in figure Figure 8.6) and $\Theta^2_{b_c}$ is the event that $\mathsf{A}^2_{b_c}$ outputs 1. Let $\Delta^2_{b_c}$ denote the quantity $\mathbf{P}[\Theta^2_{b_c};\ \overline{\mathsf{Bad}} \mid b=0] - \mathbf{P}[\Theta^2_{b_c};\ \overline{\mathsf{Bad}} \mid b=1]$ with b being the bit chosen by the oracle $\mathcal{O}^{\mathbb{P}}_b$ given to $\mathsf{A}^2_{b_c}$. As above, we can prove that for any $b_c \in \{0,1\}$:

$$|\Delta^2_{b_c}| \leq 2\epsilon_2 \quad (8.4)$$

For any type-3 adversary $\mathsf{A}^3_{d'}$, let Bad be the event that the critical key K_{crit} is not contained in $\mathcal{K}(\mathsf{A})$ when A is executed by $\mathsf{A}^3_{d'}$ (as shown in figure Figure 8.8). Let $\Theta^3_{d'}$

```
ADVERSARY $\mathsf{A}^3_{d'}$, PHASE II: EXECUTION
000. Let $K_{\mathsf{target}} = \mathsf{prefix}(K_{\mathsf{crit}}, d')$.
     (While evaluating symbolic keys, $\mathsf{A}^1_{d'}$ maps $\mathbf{G}(K_{\mathsf{target}})$ to its input $\mathtt{s}_0 \parallel \mathtt{s}_1$.)
010. Run A.
020. When A makes a query of the form $\mathtt{eval}(M)$,
021.     Let $\mathcal{M} \leftarrow \mathcal{M} \cup \{M\}$.
022.     Let $K_{\mathsf{first}}$ be the key s.t. $K_{\mathsf{first}}$ occurs in $M$, $K_{\mathsf{first}} \notin \mathbf{Rec}(\mathcal{M})$
             and there exists $M'$ for which $\mathbf{E}_{K_{\mathsf{first}}}(M') \in \mathbf{Rec}(\mathcal{M})$.
             (If there is no such key, $K_{\mathsf{first}} = \perp$)
023.     Reply with $eval^{(3)}(M, K_{\mathsf{first}}, 0)$
030. When A makes a query of the form $\mathtt{challenge}(K)$,
031.     If $K = K_{\mathsf{crit}}$,
032.         $\mathsf{good\_event} \leftarrow 1$. Return $key\_eval^{(1)}(K, 0)$.
033.     Else,
034.         If $\mathsf{good\_event} = 0$, return $key\_eval^{(1)}(K, 1)$.
035.         Else, return $key\_eval^{(1)}(K, 0)$.
040. When A halts, output whatever A outputs.

procedure $eval^{(3)}(M, K_{\mathsf{first}}, \mathsf{flag})$
100. If $M = K$ for some symbolic key $K$,
101.     Return $key\_eval^{(1)}(K, \mathsf{flag})$.
110. If $M = \mathbf{E}_K(M')$ for some symbolic key $K$ and symbolic message $M'$,
120.     If $\mathsf{flag} = 0$ and $K \neq K_{\mathsf{first}}$,
121.         Return $\mathbb{E}_{key\_eval^{(1)}(K,0)}(eval^{(3)}(M', K_{\mathsf{first}}, 0))$.
130.     If $\mathsf{flag} = 0$ and $K = K_{\mathsf{first}}$,
131.         Return $\mathbb{E}_{key\_eval^{(1)}(K,0)}(eval^{(3)}(M', K_{\mathsf{first}}, 1))$.
140.     If $\mathsf{flag} = 1$,
141.         Return $\mathbb{E}_{key\_eval^{(1)}(K,1)}(eval^{(3)}(M', K_{\mathsf{first}}, 1))$.
```

Figure 8.8: The execution phase for type-3 adversaries.

be the event that $\mathsf{A}^3_{d'}$ outputs 1 and $\Delta^3_{d'}$ the quantity $\mathbf{P}[\Theta^3_{d'};\ \overline{\mathsf{Bad}} \mid b = 0] - \mathbf{P}[\Theta^3_{d'};\ \overline{\mathsf{Bad}} \mid b = 1]$. (Here, b denotes the bit corresponding to the input given to $\mathsf{A}^3_{d'}$.) As in the case of type-1 and type-2 adversaries, we can prove that for any $d' \in \{0, 1, \ldots, d\}$:

$$|\Delta^3_{d'}| \leq 2\epsilon_1 \tag{8.5}$$

RELATING $\Delta^{\mathbb{P},\mathbb{G}}_{\text{ADPT}}(\mathsf{A})$ TO $\Delta^1_{d',b_e,b_c}, \Delta^2_{b_c}, \Delta^3_{d'}$. Next, we express $\Delta^{\mathbb{P},\mathbb{G}}_{\text{ADPT}}(\mathsf{A})$ in terms of the quantities $\Delta^1_{d',b_e,b_c}, \Delta^2_{b_c}$ and $\Delta^3_{d'}$ defined above. Towards this, we first define a sequence of events for each type of adversary we have constructed, and then use these events to relate $\Delta^1_{d',b_e,b_c}, \Delta^2_{b_c}$ and $\Delta^3_{d'}$, first, to each other, and then, to $\Delta^{\mathbb{P},\mathbb{G}}_{\text{ADPT}}(\mathsf{A})$.

Consider again the adversaries of type-1. For any such adversary $\mathsf{A}^1_{d',b_e,b_c}$ and any $i \in [q]$, let Φ_i be the event that the size of $\mathit{outkeys}(\mathsf{A})$, as defined for the execution of A by $\mathsf{A}^1_{d',b_e,b_c}$, is equal to i. (We assume, without loss of generality, that $\mathit{outkeys}(\mathsf{A})$ is non-empty for any execution of A.) For any $j \in [q]$, let Ψ_j be the event that the critical key K_{crit} is the jth among all keys in $\mathit{outkeys}(\mathsf{A})$ (where the latter are considered in the order of their occurrence as outermost unrecoverable keys in A's queries). Clearly, for any $i_1 \neq i_2$, Φ_{i_1} and Φ_{i_2} are mutually exclusive and for any $j_1 \neq j_2$, Ψ_{j_1} and Ψ_{j_2} are mutually exclusive. Furthermore, the event $\overline{\mathsf{Bad}}$ (as defined for type-1 adversaries) can be expressed in terms of these events as follows:

$$\begin{aligned}\overline{\mathsf{Bad}} &= \bigvee_{i=1}^{q} \left(\Phi_i \wedge \left[\bigvee_{j=1}^{i} \Psi_j \right] \right) \\ &= \bigvee_{i=1}^{q} \bigvee_{j=1}^{i} \left(\Phi_i \wedge \Psi_j \right) \qquad (8.6)\end{aligned}$$

Events of this kind can be defined for type-2 adversaries as well. We abuse notation and use Φ_i to denote also the event that $\mathit{outkeys}(\mathsf{A})$, as defined for the execution of A by some type-2 adversary, has size i and Ψ_j the event that K_{crit} is the jth element in the set $\mathit{outkeys}(\mathsf{A})$ so defined. Clearly, equation (8.6) holds even in the setting of type-2 adversaries as well.

In the context of type-3 adversaries, we use Φ_i to denote the event that $\mathcal{K}(\mathsf{A})$ (for the adversary's execution of A) has size i and Ψ_j the event that K_{crit} is the jth element in $\mathcal{K}(\mathsf{A})$. Let q_c be an upper bound on the number of challenge queries made by A in any execution. Equation (8.6) holds even in the setting of type-3 adversaries, with the slight difference that q gets substituted with q_c here.

For any $(d', b_e, b_c) \in \{0, 1, \ldots, d\} \times \{0,1\} \times \{0,1\}$, let $\Delta^1_{d',b_e,b_c,i,j}, \Delta^2_{b_c,i,j}$ and $\Delta^3_{d',i,j}$ be probability differences defined as follows:

$$\begin{aligned}
\Delta^1_{d',b_e,b_c,i,j} &= \mathbf{P}[\Theta^1_{d',b_e,b_c};\ \Phi_i;\ \Psi_j \mid b=0] - \mathbf{P}[\Theta^1_{d',b_e,b_c};\ \Phi_i;\ \Psi_j \mid b=1] \\
\Delta^2_{b_c,i,j} &= \mathbf{P}[\Theta^2_{b_c};\ \Phi_i;\ \Psi_j \mid b=0] - \mathbf{P}[\Theta^2_{b_c};\ \Phi_i;\ \Psi_j \mid b=1] \\
\Delta^3_{d',i,j} &= \mathbf{P}[\Theta^3_{d'};\ \Phi_i;\ \Psi_j \mid b=0] - \mathbf{P}[\Theta^3_{d'};\ \Phi_i;\ \Psi_j \mid b=1]
\end{aligned}$$

Using equation (8.6) and the fact that the events $\Psi_1, \ldots, \Psi_q$ and $\Phi_1, \ldots, \Phi_q$ are mutually exclusive, we obtain the following:

$$\Delta^1_{d',b_e,b_c} = \sum_{i=1}^{q} \sum_{j=1}^{i} \Delta^1_{d',b_e,b_c,i,j} \tag{8.7}$$

$$\Delta^2_{b_c} = \sum_{i=1}^{q} \sum_{j=1}^{i} \Delta^2_{b_c,i,j} \tag{8.8}$$

$$\Delta^3_{d'} = \sum_{i=1}^{q_c} \sum_{j=1}^{i} \Delta^3_{d',i,j} \tag{8.9}$$

We bound the quantity $\Delta^{\mathbb{P},\mathbb{G}}_{\text{ADPT}}(\mathsf{A})$ (from above) in terms of quantities of the form $\Delta^1_{d',b_e,b_c,i,j}, \Delta^2_{b_c,i,j}$ and $\Delta^3_{d',i,j}$ and then use equations (8.7), (8.8) and (8.9) (and (8.3), (9.4), (8.5)) to relate it to ϵ_1 and ϵ_2 as desired. First, we re-write $\Delta^{\mathbb{P},\mathbb{G}}_{\text{ADPT}}(\mathsf{A})$ in terms of probabilities of the form $\mathbf{P}[\Theta^1_*;\ \Phi_i;\ \Psi_j | b = *]$. The following lemma does precisely this.

Lemma 8.5.2

$$\Delta^{\mathbb{P},\mathbb{G}}_{\text{ADPT}}(\mathsf{A}) = n \cdot 2^d \cdot \left| \sum_{i=1}^{q} \left[\mathbf{P}[\Theta^1_{0,0,0};\ \Phi_i;\ \Psi_1 \mid b=0] - \mathbf{P}[\Theta^1_{0,0,1};\ \Phi_i;\ \Psi_1 \mid b=0] \right] \right|$$

In the next lemma, we relate probabilities of the form $\mathbf{P}[\Theta^1_*;\ \Phi_i;\ \Psi_j | b = *]$ to each other and to those of the form $\mathbf{P}[\Theta^2_*;\ \Phi_i;\ \Psi_j | b = *]$ and $\mathbf{P}[\Theta^3_*;\ \Phi_i;\ \Psi_j | b = *]$.

Lemma 8.5.3 (Hybrid Cancellation Lemma - I)

1. $\forall d' \in \{0, 1, \ldots, d-1\}, \forall (b_e, b_c) \in \{0,1\}^2, \forall i \in [q], \forall j \in [i]$:

$$\mathbf{P}[\Theta^1_{d',b_e,b_c};\ \Phi_i;\ \Psi_j \mid b=1] = \mathbf{P}[\Theta^1_{d'+1,b_e,b_c};\ \Phi_i;\ \Psi_j \mid b=0]$$

2. $\forall b_c \in \{0,1\}, \forall i \in [q], \forall j \in [i]$:

$$\begin{aligned} \mathbf{P}[\Theta^1_{d,0,b_c};\ \Phi_i;\ \Psi_j \mid b=1] &= \mathbf{P}[\Theta^2_{b_c};\ \Phi_i;\ \Psi_j \mid b=0] \\ \text{and,}\qquad \mathbf{P}[\Theta^1_{d,1,b_c};\ \Phi_i;\ \Psi_j \mid b=1] &= \mathbf{P}[\Theta^2_{b_c};\ \Phi_i;\ \Psi_j \mid b=1] \end{aligned}$$

3. $\forall b_c \in \{0,1\}, \forall i \in [q], \forall j \in \{0,1,\ldots,i-1\}$:

$$\mathbf{P}[\Theta^1_{0,1,b_c};\ \Phi_i;\ \Psi_j \mid b=0] = \mathbf{P}[\Theta^1_{0,0,b_c};\ \Phi_i;\ \Psi_{j+1} \mid b=0]$$

4. $\forall d' \in \{0,1,\ldots,d-1\}, \forall i \in [q_c], \forall j \in [i]$:

$$\mathbf{P}[\Theta^3_{d'};\ \Phi_i;\ \Psi_j \mid b=1] = \mathbf{P}[\Theta^3_{d'+1};\ \Phi_i;\ \Psi_j \mid b=0]$$

5. $\forall i \in [q_c], \forall j \in \{0,1,\ldots,i-1\}$:

$$\mathbf{P}[\Theta^3_{d};\ \Phi_i;\ \Psi_j \mid b=1] = \mathbf{P}[\Theta^3_{0};\ \Phi_i;\ \Psi_{j+1} \mid b=0]$$

6. And finally,

$$\begin{aligned} \sum_{i=1}^{q} \mathbf{P}[\Theta^1_{0,1,0};\ \Phi_i;\ \Psi_i \mid b=0] &= \sum_{i=1}^{q_c} \mathbf{P}[\Theta^3_{0};\ \Phi_i;\ \Psi_1 \mid b=0] \\ \text{and,}\qquad \sum_{i=1}^{q} \mathbf{P}[\Theta^1_{0,1,1};\ \Phi_i;\ \Psi_i \mid b=0] &= \sum_{i=1}^{q_c} \mathbf{P}[\Theta^3_{d};\ \Phi_i;\ \Psi_i \mid b=1] \end{aligned}$$

Before presenting the proofs of these two lemmas, we illustrate how they suffice to prove the theorem. Let Δ be defined as follows:

$$\Delta = \sum_{i=1}^{q} \left[\mathbf{P}[\Theta^1_{0,0,0};\ \Phi_i;\ \Psi_j \mid b=0] - \mathbf{P}[\Theta^1_{0,0,1};\ \Phi_i;\ \Psi_j \mid b=0] \right] \tag{8.10}$$

From Lemma 8.5.2, we know that $\Delta^{\mathbb{P},\mathbb{G}}_{\text{ADPT}}(\mathsf{A}) = n \cdot 2^d \cdot |\Delta|$. By invoking Lemma 8.5.3 multiple times, we express Δ as a sequence of summations as follows:

$$\Delta = \left\{ \begin{array}{c} \sum_{i=1}^{q} \left[\begin{array}{c} \sum_{j=1}^{i} \left(\begin{array}{c} \sum_{d'=0}^{d} \left(\begin{array}{c} \mathbf{P}[\Theta^1_{d',0,0};\ \Phi_i;\ \Psi_j \mid b=0] \\ -\mathbf{P}[\Theta^1_{d',0,0};\ \Phi_i;\ \Psi_j \mid b=1] \end{array} \right) \\ \\ + \left(\begin{array}{c} \mathbf{P}[\Theta^2_{0};\ \Phi_i;\ \Psi_j \mid b=0] \\ -\mathbf{P}[\Theta^2_{0};\ \Phi_i;\ \Psi_j \mid b=1] \end{array} \right) \\ \\ - \sum_{d'=0}^{d} \left(\begin{array}{c} \mathbf{P}[\Theta^1_{d',1,0};\ \Phi_i;\ \Psi_j \mid b=0] \\ -\mathbf{P}[\Theta^1_{d',1,0};\ \Phi_i;\ \Psi_j \mid b=1] \end{array} \right) \end{array} \right) \\ \\ - \sum_{j=1}^{i} \left(\begin{array}{c} \sum_{d'=0}^{d} \left(\begin{array}{c} \mathbf{P}[\Theta^1_{d',0,1};\ \Phi_i;\ \Psi_j \mid b=0] \\ -\mathbf{P}[\Theta^1_{d',0,1};\ \Phi_i;\ \Psi_j \mid b=1] \end{array} \right) \\ \\ + \left(\begin{array}{c} \mathbf{P}[\Theta^2_{1};\ \Phi_i;\ \Psi_j \mid b=0] \\ -\mathbf{P}[\Theta^2_{1};\ \Phi_i;\ \Psi_j \mid b=1] \end{array} \right) \\ \\ - \sum_{d'=0}^{d} \left(\begin{array}{c} \mathbf{P}[\Theta^1_{d',1,1};\ \Phi_i;\ \Psi_j \mid b=0] \\ -\mathbf{P}[\Theta^1_{d',1,1};\ \Phi_i;\ \Psi_j \mid b=1] \end{array} \right) \end{array} \right) \end{array} \right] \\ \\ + \sum_{i=1}^{q_c} \left[\sum_{j=1}^{i} \left(\sum_{d'=0}^{d} \left(\begin{array}{c} \mathbf{P}[\Theta^3_{d'};\ \Phi_i;\ \Psi_j \mid b=0] \\ -\mathbf{P}[\Theta^3_{d'};\ \Phi_i;\ \Psi_j \mid b=1] \end{array} \right) \right) \right] \end{array} \right\}$$

which can be written, in short, as:

$$\Delta \;=\; \left\{ \begin{array}{c} \sum_{i=1}^{q} \left[\begin{array}{c} \sum_{j=1}^{i} \left(\sum_{d'=0}^{d} \Delta^1_{d',0,0,i,j} + \Delta^2_{0,i,j} - \sum_{d'=1}^{d} \Delta^3_{d',1,0,i,j} \right) \\ \\ - \sum_{j=1}^{i} \left(\sum_{d'=0}^{d} \Delta^1_{d',0,1,i,j} + \Delta^2_{1,i,j} - \sum_{d'=1}^{d} \Delta^3_{d',1,1,i,j} \right) \end{array} \right] \\ \\ + \sum_{i=1}^{q_c} \sum_{j=1}^{q_c} \sum_{d'=0}^{d} \Delta^3_{d',i,j} \end{array} \right\}$$

Now, applying equations (8.7), (8.8) and (8.9), we get:

$$\Delta \;=\; \left\{ \begin{array}{c} \sum_{d'=0}^{d} \Delta^1_{d',0,0} + \Delta^2_0 - \sum_{d'=1}^{d} \Delta^3_{d',1,0} \\ \\ - \sum_{d'=0}^{d} \Delta^1_{d',0,1} + \Delta^2_1 - \sum_{d'=1}^{d} \Delta^3_{d',1,1} + \sum_{d'=0}^{d} \Delta^3_{d'} \end{array} \right\}$$

Finally, using equations (8.3), (9.4) and (8.5), we obtain our desired goal:

$$\begin{aligned} \Delta^{\mathbb{P},\mathbb{G}}_{\text{ADPT}}(\mathsf{A}) &= n \cdot 2^d \cdot |\Delta| \\ &\le n \cdot 2^d \cdot \left\{ \begin{array}{c} \sum_{d'=0}^{d} |\Delta^1_{d',0,0}| + |\Delta^2_0| + \sum_{d'=1}^{d} |\Delta^3_{d',1,0}| \\ \\ + \sum_{d'=0}^{d} |\Delta^1_{d',0,1}| + |\Delta^2_1| + \sum_{d'=1}^{d} |\Delta^3_{d',1,1}| + \sum_{d'=0}^{d} |\Delta^3_{d'}| \end{array} \right\} \\ &= 2n2^d \cdot \{4d\epsilon_1 + 2\epsilon_2 + d\epsilon_1\} \\ &= 2n2^d \cdot \{5d\epsilon_1 + 2\epsilon_2\} \end{aligned}$$

8.5.4 Proof of Lemma 8.5.2

Let Δ be the term defined in equation (8.10). Our goal is to show that $|\Delta| = \frac{1}{n \cdot 2^d} \cdot \Delta^{\mathbb{P},\mathbb{G}}_{\text{ADPT}}(\mathsf{A})$. Towards this, we first re-write Δ as follows:

$$\Delta = \sum_{i=1}^{q} \left[\begin{array}{c} \mathbf{P}[\Theta^1_{0,0,0};\ \Phi_i \mid \Psi_1;\ b = 0] \cdot \mathbf{P}[\Psi_1;\ b = 0] \\ -\ \mathbf{P}[\Theta^1_{0,0,1};\ \Phi_i \mid \Psi_1;\ b = 0] \cdot \mathbf{P}[\Psi_1;\ b = 0] \end{array} \right]$$

We claim that the probability that the event Ψ_1 occurs in either $\mathsf{A}^1_{0,0,0}$ or $\mathsf{A}^1_{0,0,1}$, conditioned on bit b being equal to 0, is exactly $\frac{1}{n \cdot 2^d}$. The proof of this claim follows from two observations:

- Let $\mathcal{K}_{n,d}$ be the set of all keys of depth at most d that can be derived from the keys $R_1, \ldots, R_n$; that is, $\mathcal{K}_{n,d} = \{K \mid \exists i \in [n] \text{ s.t. } K \in \mathbf{Rec}(R_i) \wedge \mathsf{depth}(K) \leq d\}$. Our first observation is that for any $K \in \mathcal{K}_{n,d}$, the probability that K_{crit} (as define in $\mathsf{A}^1_{0,0,0}$ or $\mathsf{A}^1_{0,0,1}$) equals K is equal to $\frac{1}{n \cdot 2^d}$.

- Second, we note that for any $K \in \mathcal{K}_{n,d}$, the probability that the *first* key in $outkeys(\mathsf{A})$, say K_{f}, as defined for A's execution in $\mathsf{A}^1_{0,0,0}$ (resp. $\mathsf{A}^1_{0,0,1}$), equals K *conditioned on the events* $K_{\mathsf{crit}} = K$ *and* $b = 0$, is equal to the probability that K_{f} equals K when A executes with black-box access to $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},0}$ (resp. $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},1}$). This second observation is a bit non-trivial and requires a careful understanding of the execution phases of $\mathsf{A}^1_{0,0,0}$ and $\mathsf{A}^1_{0,0,1}$ and comparison of the same with the procedure $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$; details are omitted.

Thus, Δ can further be re-written as:

$$\Delta = \frac{1}{n \cdot 2^d} \cdot \sum_{i=1}^{q} \left[\mathbf{P}[\Theta^1_{0,0,0};\ \Phi_i \mid \Psi_1;\ b = 0] - \mathbf{P}[\Theta^1_{0,0,1};\ \Phi_i \mid \Psi_1;\ b = 0] \right]$$

Finally, we observe that the probability of $\mathsf{A}^1_{0,0,0}$ (resp. $\mathsf{A}^1_{0,0,1}$) outputting 1 and of event Φ_i taking place, conditioned on events Ψ_1 and $b = 0$ is the same as the probability of A outputting 1 and Φ_i taking place, when the former is given black-box access to $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},0}$ (resp. $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},1}$). This follows from the fact that A is a valid and compliant adversary and that conditioned on events Ψ_1 and $b = 0$ taking place, all replies given by $\mathsf{A}^1_{0,0,0}$ (resp. $\mathsf{A}^1_{0,0,1}$) to A's queries are determined just as in the procedure $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},0}$ (resp. $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},1}$). Thus,

$$\Delta \;=\; \frac{1}{n \cdot 2^d} \cdot \sum_{i=1}^{q} \left[\mathbf{P}[\mathsf{A}^{\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},0}} = 1] - \mathbf{P}[\mathsf{A}^{\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},1}} = 1] \right]$$

From this, the lemma follows.

8.6 Related Work

The question of relating symbolic security notions with notions of security in the computational model has been considered in several contexts prior to our work. The pioneering efforts in this direction were made by Abadi and Rogaway [1] who proved a computational soundness theorem for protocols based on symmetric-key encryption only. (No other primitives were considered in [1].) Although the result of [1] is important as a first step towards bridging the symbolic-computational divide, it is applicable only in the context of passive adversaries (that is, adversaries who eavesdrop on the protocol communication but do nothing else). In contrast, our result (Theorem 8.3.2) enables security of protocols to be analyzed against adversaries who can adaptively influence the protocol execution. Security analysis with respect to such adversaries is necessary for multi-user protocols like the ones we consider here. (Later on, in Chapter 9, we further strengthen the adversarial model to incorporate adaptive corruptions as well.) [2] extends the Abadi-Rogaway result to a wider class of protocols, namely, to protocols that make use of encryption as well as secret sharing [42]; however, the adversarial model used in [2] is the same as that of [1] and, in particular, it does not incorporate adaptive attacks on protocols.

Other extensions of the Abadi-Rogaway result also exist in the literature but they are largely orthogonal to the problem considered in this thesis. One notable extension is that due to Micciancio and Warinschi [33], who devise new techniques for proving soundness of symbolic security proofs in the face of *active* computational adversaries, that is, adversaries who can intercept and modify all communication between protocol users. In order to prove soundness against such adversaries, a stronger computational security notion for encryption schemes (namely, CCA-security [40]) is used in [33]. (In contrast, our soundness result only relies on CPA-security.) We remark that the result of [33] imposes several syntactic restrictions on the syntax of protocols which are more stringent than the ordering constraint we have introduced; for example, in the protocol class considered in [33], protocol users are not allowed to “forward” any ciphertext received from one user to another user, and the use of nested encryption is also prohibited. Several extensions and generalizations of [33] have appeared in the literature (see, for example, [12, 16]); all these works primarily focus on security of two-party cryptographic protocols (in particular, protocols for key exchange and mutual authentication) in the presence of active adversaries. Our model, on the other hand, focusses on multi-party protocols and considers their security in the face of passive but *adaptive* adversaries. We do not incorporate active attacks in our attack model primarily for simplicity, although we do believe that such an extension is possible and we leave it open for future work.

8.7 Acknowledgement

Chapter 8, in part, is a reprint of the material as it appears in 33rd Internation Colloquium on Automata, Languages and Programming (ICALP), July 2006, Micciancio, Daniele; Panjwani, Saurabh. The dissertation author was the primary investigator and author of this paper. The presentation of the material has been significantly improved from the time of the original publication and some errors have also been eliminated.

Chapter 9

Computational Security against Adaptive Adversaries

The techniques developed in the previous chapter can be used for analyzing security of GKD protocols in the computational model but only as long as security against *non-adaptive* adversaries (that is, adversaries who corrupt protocol users non-adaptively) is the desired objective. In this chapter, we present a different, and more powerful, approach for conducting security analysis of these protocols, which is applicable even when user corruptions are performed by the adversary in an adaptive manner.

9.1 The Challenge Ahead

Arguing about security of multi-party protocols in the face of adaptive corruptions is a hard problem in cryptography. The possibility of user corruptions occurring during protocol execution, and in a manner that is arbitrarily controlled by the attacker, increases the threat to a protocol's security and makes the task of *proving* protocols secure an unnerving task. Indeed, there exist protocols that are provably secure against non-adaptive attacks but which can be completely broken once the adversary is allowed to corrupt participants adaptively. (See [9] for an example based on secret sharing schemes.) 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 destined to be opened by the adversary and which are not, 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.

In the past, security analysis of encryption-based multi-party protocols against adaptive adversaries has largely been conducted using three approaches. The first (and the simplest) one involves reducing the problem to the problem of proving security against non-adaptive adversaries: in the security proof, one tries to guess the set of parties that the adversary corrupts "beforehand" (that is, prior to protocol execution) and then, if the guess is correct, does the rest of the proof just as in the non-adaptive case. However, since the probability of guessing the corrupt set correctly is extremely small (in particular, it is inversely exponential in the number of parties), such a proof can be of very little use in practice. Reducing the problem of adaptive security to that of non-adaptive security does not address the real challenge; it just bypasses it.

The second approach to adaptive security has been to study it in restricted models where strict rules are imposed on the behavior of honest participants. The most common imposition is that of *erasure* [3]—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 (several GKD protocols are of this nature) and such protocols 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.* [9], 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 adaptively secure quite easily: in the security proof, one 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.

Although interesting in their own right, non-committing encryption schemes have their share of limitations as well: they are typically too inefficient for practical applications, and allow only a restricted number of encryptions to be performed under a single key. In fact, it has been shown [36] that 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 decrypted using it*[1]. Such a restriction is too prohibitive for real applications.

9.2 Overview of Our Result

We show that it is possible to argue about adaptive security of a large class of encryption protocols—and of GKD protocols, in particular—without requiring erasures and without resorting to primitives like non-committing encryption, while still achieving efficiency that meets practical requirements. For simplicity, we focus on protocols that use symmetric-key encryption only (that is, no pseudo-random generators are used) and those in which every ciphertext is created by encrypting a key with a *single* other key (no nesting of the encryption operation)[2]. We show that in any such protocol, the quality of adaptive security that can be provably achieved by a protocol is closely related to the *depth* of the key graph[3] generated in any protocol execution. More precisely, we prove that in any encryption protocol for which execution key graphs (as defined in Chapter 4) have size at most n and depth at most ℓ, security in the symbolic model implies security *against adaptively-corrupting adversaries* via a reduction factor that is $O(n \cdot (2n)^{\ell})$. That is, the smaller the depth of the key graph generated in any execution, the greater is its strength against adaptive adversaries.

What makes our result of practical benefit is the fact that key graphs generated by most known encryption protocols have depth much smaller (in fact, orders of magnitude smaller) than their total size. In the particular case of GKD protocols, key graphs typically have depth at most logarithmic in the number of users, and thus, even in the

[1]Some non-committing encryption schemes [11] circumvent this impossibility result by studying the problem in a restricted model where bounds on the frequency of communication between parties are placed.

[2]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 GKD protocols don't use nesting).

[3]The depth of a graph refers to the length of the longest path in it.

size of graph. (See the examples in Chapter 5.) Furthermore, in most GKD protocols, the depth of key graphs remains fixed *even for arbitrarily long runs of the protocol*, assuming the space of users has been ascertained beforehand. In general, the depth of key graphs in encryption protocols is related to the number of decryptions performed by users to recover certain keys or messages, while their size to the total number of users; it is reasonable to expect that protocol designers, for the purpose of efficiency, would strive to keep the former quantity smaller than the latter.

We apply our result to the security analysis of the plain-LKH$^+$ protocol presented in Chapter 5 and show that the protocol's security against adaptive corruptions 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. (Specifically, the reduction factor is $O(\mathbf{n}^{\log_2(\mathbf{n})+1})$ with $\mathbf{n}$ being the total number of protocol users.) This reduction factor, though not strictly polynomial, is still quite reasonable from a practical perspective; for example, in a system with 128 users, one is guaranteed that a 64-round execution of plain-LKH$^+$ provides at least 64 bits of adaptive security (that is, an efficient adversary has probability at most 2^{-64} in subverting the protocol) when implemented with commonly-used symmetric-key encryption schemes like counter-mode AES [4][4]. Our result practically eliminates the need for using expensive techniques like non-committing encryption to build adaptively secure group key distribution protocols, and it does this while matching the efficiency of existing schemes.

9.3 The Result

Let $\mathbb{P} = (\mathbb{E}, \mathbb{D})$ be a symmetric-key encryption scheme. We consider a simplified version of procedure $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$ (defined in Chapter 8) which only implements the encryption scheme $\mathbb{P}$, and denote it by $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$. This procedure receives a parameter n as input and works as follows: it generates a set of keys $k_1, \ldots, k_n$, each key being sampled using an independent invocation of $\mathbb{R}$, and then accepts and responds to queries of *three*

[4]This calculation is based on the assumption that the block cipher AES used in the encryption scheme is a pseudorandom permutation with 128 bits of security.

types:

- *Encryption queries:* Upon receiving a query of the form $\texttt{encrypt}(i, j)$ for some $i, j \in [n]$, $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$ responds with a sample from $\mathbb{E}_{k_i}(k_j)$.
- *Corruption queries:* Upon receiving a query of the form $\texttt{corrupt}(i)$ for some $i \in [n]$, $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$ returns the key k_i.
- *Challenge queries:* Finally, $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$ receives and responds to challenge queries just like $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$. A valid challenge query is of the form $\texttt{challenge}(i)$ for some $i \in [n]$, and for each such query, $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$ responds with the key k_i if $b = 0$, and, if $b = 1$, it responds with a key r_i, sampled independently and uniformly at random from $\{0, 1\}^{\eta}$. (If $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$ receives the same query $\texttt{challenge}(i)$ multiple times, and if $b = 1$, the reply equals r_i each time.)

Encryption queries and corruption queries together model the evaluation queries defined in the context of $\mathcal{O}^{\mathbb{P},\mathbb{G}}_{\text{ADPT},b}$, with the difference that the use of nested encryption and pseudo-random generators is not permitted.

Consider any adversary A given black-box access to $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$. A can decide the input n given to the procedure and can make multiple queries to it, interleavingly and adaptively. We think of the queries of A as creating a directed graph over n nodes (labeled $1, 2, \ldots, n$), edge by edge, and in an adaptive fashion. Each query $\texttt{encrypt}(i, j)$ made by A corresponds to the creation of an edge from i to j, denoted $i \to j$, in this graph; such an edge models the fact that given k_i and the ciphertext $\mathbb{E}_{k_i}(k_j)$, A can easily recover the key k_j. (So, knowledge of k_i "leads to" the knowledge of k_j.) For any adversary A, the graph created by its queries in this manner is called the *key graph* generated by A and is denoted $\mathcal{G}(\mathsf{A})$. A node i in $\mathcal{G}(\mathsf{A})$ for which A issues a query $\texttt{corrupt}(i)$ is called a *corrupt node* while one for which A issues a query $\texttt{challenge}(i)$ is referred to as a *challenge node*. The set of all corrupt nodes is denoted $\mathcal{V}^{\mathsf{corr}}(\mathsf{A})$ and that of all challenge nodes is denoted $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$. Note that $\mathcal{G}(\mathsf{A}), \mathcal{V}^{\mathsf{corr}}(\mathsf{A})$ and $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ are all random variables depending on the coins used by both A and $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$.

We assume that $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ is always non-empty and in any execution of A, every node $i \in \mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ 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 $\texttt{encrypt}(x, i)$ during its entire execution. This is without any loss of generality since an adversary for which these conditions are not satisfied can be easily transformed (that is, without any significant difference in attack advantage or time complexity) into one for which they are.

Similarly to the definition of valid adversaries in Chapter 8, we define validity in the context of $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$ as follows.

Definition 9.3.1 An adversary A is called *valid* relative to $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$ if in any execution involving interaction with $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$, the values of $\mathcal{G}(\mathsf{A}), \mathcal{V}^{\mathsf{corr}}(\mathsf{A})$ and $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ are such that:

1. For any $i \in \mathcal{V}^{\mathsf{corr}}(\mathsf{A})$ and any $j \in \mathcal{V}^{\mathsf{chal}}(\mathsf{A})$, j is not reachable from i in $\mathcal{G}(\mathsf{A})$.

2. Every node in $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ has zero out-degree in $\mathcal{G}(\mathsf{A})$.

The first condition is equivalent to requiring that keys corresponding to challenge nodes not be symbolically recoverable by A. The second condition restricts these keys from being used for encrypting other keys, which, as already discussed in Chapter 8, is necessary for guaranteeing their pseudo-randomness.

For any adversary A, let $\mathsf{A}^{\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}}$ denote the random variable corresponding to the output of A when given black-box access to $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$. The adaptive advantage of A against $\mathbb{P}$ is defined as the following quantity:

$$\Delta^{\mathbb{P}}_{\text{ADPT}}(\mathsf{A}) = \left| \mathbf{P}[\mathsf{A}^{\mathcal{O}^{\mathbb{P}}_{\text{ADPT},0}} = 1] - \mathbf{P}[\mathsf{A}^{\mathcal{O}^{\mathbb{P}}_{\text{ADPT},1}} = 1] \right|$$

As usual, both probabilities are taken over the coins used by A as well as by $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$.

Let $n, q, \ell \in I\!N$ such that $\ell < n$. We say that A is an (n, q, 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 q respectively and the *depth* of the graph (that is, the length of the longest path in it) is at most ℓ. Note that the last condition also implies that the key graph generated by A is acyclic (since in any graph with cycles, there exist paths with infinite length).

Definition 9.3.2 An encryption scheme $\mathbb{P}$ is called $(\tau, n, q, \ell, \epsilon)$*-secure against adaptive attacks* if for every valid (n, q, ℓ)-adversary A running in time at most τ, $\Delta^{\mathbb{P}}_{\text{ADPT}}(\mathsf{A}) \leq \epsilon$.

The following is the main result of this chapter:

Theorem 9.3.3 Let $\mathbb{P} = (\mathbb{E}, \mathbb{D})$ a symmetric-key encryption scheme. If $\mathbb{P}$ is (τ_1, ϵ_1)-secure against chosen plaintext attacks, then it is $(\tau, n, q, \ell, \epsilon)$-secure against adaptive attacks for any parameters $\tau, n, q, \ell, \epsilon$ satisfying

$$\begin{aligned} \tau &= \tau_1 - (2n\tau(\mathbb{R}) + q\tau(\mathbb{E}) + O(1)) \\ \epsilon &= \epsilon_1 \cdot \frac{3n^2}{2} \cdot (2n+1)^{\ell-1} \end{aligned}$$

We emphasize that in contrast to the soundness result of the previous chapter (Theorem 8.3.2), the above result guarantees security *without any restrictions on the order of queries made by adversaries*. Thus, for protocols which generate key graphs of small depth (like the ones we study in this thesis), this gives a more powerful tool to analyze security—security can be proven against adaptively-corrupting adversaries, and without requiring protocol messages to satisfy any ordering constraints. The downside, however, is that the "amount" of security that can be guaranteed is intricately related to the depth of protocol key graphs, which makes the result of limited applicability in the context of protocols that generate arbitrary-depth key graphs. We believe that improving the result to overcome this limitation is non-trivial, but a worthy direction for future research; in particular, obtaining an analogous result but with a reduction factor smaller than $\Theta(n^\ell)$ would be quite remarkable, and could lead to even newer techniques to address adaptive corruptions in encryption protocols.

9.3.1 Relation with the Selective Decryption Problem

Our abstract game (involving the adversary A and the oracle procedure $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$) is closely related to a well-studied problem in the cryptography literature called the *selective decryption problem*. This problem often arises in the context of proving adaptive security of multi-party protocols (see, for example, [9, 20]) but till date, no satisfactory solution to the problem has been found. The problem involves an adversary who interacts with an oracle in the following manner: the oracle 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; furthermore, they may be related to each other in an arbitrary manner.) The adversary then requests the encryptions of all the plaintexts, $\{\mathbb{E}_{k_i}(m_i)\}_{i=1}^{n}$ (where $\mathbb{E}$ is a semantically secure encryption function), and later "opens" some of these encryptions adaptively; that is, it queries an arbitrary set $I \subseteq [n]$ and the oracle 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 can learn no more information about them than what it could anyway learn from the revealed plaintexts. In our result, we are essentially generalizing the selective decryption problem by allowing the adversary to request not only *single* ciphertexts but *chains* of ciphertexts of the form $\mathbb{E}_{k_1}(k_2), \mathbb{E}_{k_2}(k_3), \mathbb{E}_{k_3}(k_4), \cdots$ and also to open such chains adaptively. Besides, we allow the adversary to interleave its "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 the proof of Theorem 9.3.3.

While the problem we are considering appears more general than the selective decryption problem in several ways, there is one important caveat. In our problem, the plaintexts on which encryption is performed are always keys, and the latter are assumed to be generated in a mutually independent manner. It has been shown [20] that when the selective decryption problem is considered with mutually independent plaintexts (as in our problem), a solution to the problem does exist, and, in particular, it is possible to show that the adversary learns no more information about the hidden plaintexts than what is easily computable from the opened ones. Our soundness theorem (Theorem 9.3.3) 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, though, still remains open.

We remark here that independence of all keys is not just a simplifying assumption in our theorem; it is a *requirement* for the security of the protocols we are interested in analyzing: a group key distribution protocol that uses related keys across key updates cannot guarantee good security at all. Furthermore, an adaptively-secure group key distribution protocol suffices to build adaptively-secure protocols for private group communication since, as noted in Chapter 7, the two protocol problems are equivalent to each other (even in the face of adaptive corruptions).

9.4 Analysis of Protocols

Before proving Theorem 9.3.3, we first illustrate how it is applicable to the security analysis of GKD protocols against adaptive adversaries.

Let Π be any S-GKD protocol for $\mathbf{n}$ users and let $\hat{n}, \hat{q}, \hat{\ell}$ be functions defined over the set of natural numbers. We say that Π is an $(\hat{n}, \hat{q}, \hat{\ell})$-S-GKD protocol if for all $t > 0$, for all sequences $\overrightarrow{\mathcal{S}}^{(t)} \in (2^{[\mathbf{n}]})^t$, the number of purely random keys used in $\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ is at most $\hat{n}(t)$, the number of pseudo-random keys used in it is zero, the size of $\mathcal{M}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ is at most $\hat{q}(t)$ and the depth of the key graph $\mathcal{G}^{\Pi}_{\overrightarrow{\mathcal{S}}^{(t)}}$ (corresponding to the execution of Π when given $\overrightarrow{\mathcal{S}}^{(t)}$ as input) is at most $\hat{\ell}(t)$. We say that Π is *group-key-compliant* if no group key generated by the protocol is ever used to encrypt any other key (that is, Π satisfies condition (a) of Definition 8.4.1 from Chapter 8). For any symmetric-key encryption scheme $\mathbb{P} = (\mathbb{E}, \mathbb{D})$, let $\Pi^{\mathbb{P}}$ denote the computational interpretation of Π when its encryption function $\mathbf{E}$ is implemented using $\mathbb{E}$.

Theorem 9.4.1 Let $\mathbb{P}$ be a symmetric-key encryption scheme and let Π be a $(\hat{n}, \hat{q}, \hat{\ell})$-S-GKD protocol that is group-key-compliant. If Π is secure against single-user (resp. collusion) attacks in the symbolic model, and if $\mathbb{P}$ is (τ_1, ϵ_1)-secure against chosen plaintext attacks, then $\Pi^{\mathbb{P}}$ is (τ, t, ϵ)-adaptively secure against single-user (resp. collusion) attacks in the computational model for any τ, t, ϵ satisfying

$$\begin{aligned}\tau &= \tau_1 - (2\hat{n}(t)\tau(\mathbb{R}) + \hat{q}(t)\tau(\mathbb{E}) + O(1)) \\ \epsilon &= \epsilon_1 \cdot \frac{3\hat{n}(t)^2}{2} \cdot (2\hat{n}(t)+1)^{\hat{\ell}(t)-1}\end{aligned}$$

Proof: The proof of this theorem is very similar to that of Theorem 8.4.2 and we only sketch it here. As in the proof of Theorem 8.4.2, we consider only the case of security against collusion attacks. (The other case is very similar and thus omitted.)

Let Π be any $(\hat{n}, \hat{q}, \hat{\ell})$-S-GKD protocol that is group-key-compliant and $\mathbb{P}$ a (τ_1, ϵ_1)-secure encryption scheme. Suppose that Π is secure against collusion attacks in the symbolic model and still $\Pi^{\mathbb{P}}$ does not satisfy the definition of (τ, t, ϵ)-security against adaptive adversaries for some parameters τ, t, ϵ satisfying the conditions given in the theorem. That is, there exists some adaptive adversary A that runs in time τ, makes at most t execution queries, but for which $\Delta^{\Pi^{\mathbb{P}}}_{\mathrm{GKD}}(\mathsf{A}) > \epsilon$. We claim that any such adversary A can be used to construct a valid $(\hat{n}(t), \hat{q}(t), \hat{\ell}(t))$-adversary A' such that A' runs in time τ, and still $\Delta^{\mathbb{P}}_{\mathrm{ADPT}}(\mathsf{A}') > \epsilon$. This falsifies Theorem 9.3.3, thus implying that Theorem 9.4.1 must hold.

The adversary A' simply emulates the execution of Π, invokes A in a black-box manner and for each query that A makes, it executes Π in accordance with the query. Since A expects to receive bitstrings in reply for each query, A' uses its oracle $\mathcal{O}^{\mathbb{P}}_{\mathrm{ADPT},b}$ to evaluate all the symbolic messages and keys generated by Π before providing them as replies to A's queries. In the end, it outputs whatever A outputs.

Since Π is secure in the symbolic model and since it is group-key-compliant, the adversary A' is valid with respect to $\mathcal{O}^{\mathbb{P}}_{\mathrm{ADPT},b}$. Since Π is a $(\hat{n}, \hat{q}, \hat{d})$-S-GKD protocol and since A issues at most t queries, A' is a $(\hat{n}(t), \hat{q}(t), \hat{\ell}(t))$-adversary. The running time of A' is the same as that of A (modulo some constant overhead) and it is easy to check that its advantage against the procedure $\mathcal{O}^{\mathbb{P}}_{\mathrm{ADPT},b}$ is also the same as that of A against $\Pi^{\mathbb{P}}$. The theorem follows. ▮

We now use Theorem 9.4.1 to establish adaptive security of the $\mathsf{plain\text{-}LKH}^+$

protocol and the CS protocol. First consider the $\mathsf{plain\text{-}LKH}^+$ protocol. Recall that the key graph generated in any execution of this protocol has depth exactly $\lceil \log_d(\mathbf{n}) \rceil$ where d is the arity of the key hierarchy and $\mathbf{n}$ the total number of users. Recall also that $\mathsf{plain\text{-}LKH}^+$ is secure against collusion attacks in the symbolic model (Theorem 5.1.1) and is also group-key-compliant. Using these facts one can prove security of $\mathsf{plain\text{-}LKH}^+$ against adaptive adversaries in the computational model as follows. (As in the case of non-adaptive security, we consider security with respect to adversaries whose execution queries form simple sequences.)

Theorem 9.4.2 Let $\mathbf{n} \geq 2$ and $d \in \{2, 3, \ldots, \mathbf{n}\}$. The d-ary instance of the $\mathsf{plain\text{-}LKH}^+$ protocol, when implemented for $\mathbf{n}$ users using a (τ_1, ϵ_1)-secure encryption scheme $\mathbb{P} = (\mathbb{E}, \mathbb{D})$, is $(\tau_{\mathsf{p}}, t, \epsilon_{\mathsf{p}})$-adaptively secure against collusion attacks in the computational model for any $\tau_{\mathsf{p}}, t, \epsilon_{\mathsf{p}}$ satisfying:

$$\begin{aligned} \tau_{\mathsf{p}} &= \tau_1 - (2\tilde{\mathbf{n}}\tau(\mathbb{R}) + td\lceil \log_d(\mathbf{n}) \rceil \tau(\mathbb{E}) + O(1)) \\ \epsilon_{\mathsf{p}} &= \epsilon_1 \cdot \frac{3\tilde{\mathbf{n}}^2}{2} \cdot (2\tilde{\mathbf{n}} + 1)^{\lceil \log_d(\mathbf{n}) \rceil - 1} \end{aligned}$$

where $\tilde{\mathbf{n}} = \mathbf{n} + t \cdot \lceil \log_d(\mathbf{n}) \rceil$.

Notice that the reduction factor in the above theorem is exponential in $\log_d(\mathbf{n})$ which is independent of the number of rounds t that $\mathsf{plain\text{-}LKH}^+$ is executed for. (So, the adaptive security of $\mathsf{plain\text{-}LKH}^+$ degrades polynomially with the number of rounds of execution.) Changing the hierarchy structure in the protocol involves a natural trade-off between efficiency and security: if we increase the arity d of the hierarchy, the communication efficiency of the protocol suffers but we get a better guarantee on its adaptive security. An extreme case is the $\mathbf{n}$-ary hierarchy for which the protocol incurs a linear communication cost but is guaranteed to be adaptively secure via a reduction factor of $O(\mathbf{n}^2)$. (Note that this is exactly the trivial approach to group key distribution described in the introduction.) Whether or not one can further improve this trade-off between efficiency and security across different instances of $\mathsf{plain\text{-}LKH}^+$, and, in particular, prove its adaptive security via a reduction factor smaller than the one given in Theorem 9.4.2, assuming only the semantic security of $\mathbb{P}$, is a question left open by this work.

The case of the CS protocol is much simpler. Key graphs generated in that protocol (and in any subset cover protocol, in general) always have depth one. Recall that this protocol too is collusion-resistant in the symbolic model (Theorem 5.2.1) and group-key-compliant. We conclude:

Theorem 9.4.3 The CS protocol, when implemented for $\mathbf{n}$ users using a (τ_1, ϵ_1)-secure encryption scheme $\mathbb{P} = (\mathbb{E}, \mathbb{D})$, is $(\tau_{\mathsf{c}}, t, \epsilon_{\mathsf{c}})$-adaptively secure against collusion attacks in the computational model, for any $\tau_{\mathsf{c}}, t, \epsilon_{\mathsf{c}}$ satisfying:

$$\begin{aligned}\tau_{\mathsf{c}} &= \tau_1 - (2(\mathbf{n}+t)\tau(\mathbb{R}) + t\mathbf{n}\tau(\mathbb{E}) + O(1)) \\ \epsilon_{\mathsf{c}} &= \epsilon_1 \cdot \frac{3(\mathbf{n}+t)^2}{2}\end{aligned}$$

Proving security theorems of the form of Theorem 8.4.3 and 9.4.3 for the $\mathsf{improved\text{-}LKH}^+$ and SD protocols requires suitably extending Theorem 9.3.3 to incorporate pseudo-random generators, and is postponed to future work.

9.5 Proof of Theorem 9.3.3

The proof of Theorem 9.3.3 is much more involved than that of the soundness theorem of Chapter 8 (Theorem 8.3.2). We begin with providing some intuition behind the proof.

9.5.1 The Intuition

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.* [20]. Consider a situation in which the adversary is restricted to generate key graphs of depth exactly 1; that is, its key graph is a directed bipartite graph mapping a set of sources to a set of sinks. (In the selective decryption problem, 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 $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ translates into its ability to differentiate between the two values for *some* node (say the jth one) in $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$; 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 `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 `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 $i_s \rightarrow i_j$ in $\mathcal{G}(\mathsf{A})$. Now consider this modified version of the original game: while generating keys in the beginning, the procedure $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$ 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 `encrypt`(i_s, i_j) is replied to with the real ciphertext $\mathbb{E}_{k_{i_s}}(k_{i_j})$ in the first world, $\mathsf{World}'_j(0)$, but with a *fake* one, namely $\mathbb{E}_{k_{i_s}}(\tilde{k}_{i_j})$, in the other one, $\mathsf{World}'_j(1)$. All other `encrypt` queries are replied to with real ciphertexts in both worlds. For the `challenge` queries *the replies have the same distribution*—r_i for the first $j-1$ `challenge` queries and k_i for the rest. (In particular, the reply for `challenge`(i_j) is always k_{i_j}.)

It is easy to see that the distribution on the replies of $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$ in $\mathsf{World}'_j(0)$ is exactly the same as in $\mathsf{World}_j(0)$. (The replies to all `encrypt`, `corrupt` and `challenge` queries are decided in the same manner.) 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 independently of each other, and so, replying to `encrypt`(i_s, i_j) with $\mathbb{E}_{k_{i_s}}(k_{i_j})$ and `challenge`(i_j) with r_{i_j} (as done in $\mathsf{World}_j(1)$) produces the same distribution as replying to the former with $\mathbb{E}_{k_{i_s}}(\tilde{k}_{i_j})$ 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 semantically secure. If the adversary can distinguish between two sets of ciphertexts $\{\mathbb{E}_{k_{i_s}}(k_{i_j})\}_{i_s \in I(i_j)}$ (the real ones) and $\{\mathbb{E}_{k_{i_s}}(\tilde{k}_{i_j})\}_{i_s \in I(i_j)}$ (the fake ones) then it must be able to tell the difference between $\mathbb{E}_{k_{i_s}}(k_{i_j})$ and $\mathbb{E}_{k_{i_s}}(\tilde{k}_{i_j})$ for *some* node $i_s \in I(i_j)$. (A standard hybrid argument applies here[5].) This goes against the semantic security of $\mathbb{P}$.

GOING BEYOND $\ell = 1$. In the general setting, a node i_s, pointing at any node $i_j \in \mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ 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 $\mathbb{E}_{k_{i_s}}(k_{i_j})$ is indistinguishable from $\mathbb{E}_{k_{i_s}}(\tilde{k}_{i_j})$"*, one must first argue that every ciphertext of the form $\mathbb{E}_{k_{i'_s}}(k_{i_s})$ (where $i'_s \rightarrow i_s$ is an edge in $\mathcal{G}(\mathsf{A})$) looks the same as one of the form $\mathbb{E}_{k_{i'_s}}(\tilde{k}_{i_s})$ (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 $\mathcal{G}(\mathsf{A})$ 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 $\mathcal{G}(\mathsf{A})$ 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 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 $\ell = 1$: in one extreme, the replies corresponding to all edges are real,

[5]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 $\ell = 1$.

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 encryption oracle $\mathcal{O}_b^{\mathbb{P}}$ (and so, A's capability to distinguish between them would imply that the encryption scheme is not 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 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 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 $\ell' \leq \ell$, 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 $\ell' + 1$. We now proceed to give all the details of the proof.

9.5.2 The Reduction

Let A be any (n, q, ℓ)-adversary that is valid relative to $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$. Suppose that A runs in time τ and is such that $\Delta^{\mathbb{P}}_{\text{ADPT}}(\mathsf{A}) > \epsilon$ (τ and ϵ as defined in Theorem 8.3.2); that is,

$$\left| \mathbf{P}[\mathsf{A}^{\mathcal{O}^{\mathbb{P}}_{\text{ADPT},0}} = 1] - \mathbf{P}[\mathsf{A}^{\mathcal{O}^{\mathbb{P}}_{\text{ADPT},1}} = 1] \right| \; > \; \epsilon \tag{9.1}$$

Given any such adversary, we construct another adversary A' that runs in time τ_1 and is such that $\Delta^{\mathbb{P}}_{\text{ENC}}(\mathsf{A}') > \epsilon_1$. This contradicts the security assumption we made for $\mathbb{P}$, hence implying that our assumption about A was incorrect.

Before we give the construction, we need one small technical definition. Any path in the graph $\mathcal{G}(\mathsf{A})$ generated by A can be represented using a sequence of length $\ell + 1$ 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 $\ell + 1$ nodes), *prepend* this sequence with a 0 as many times as is required to make its length equal $\ell + 1$. For example, a path $i_1 \to i_2 \to i_3$ (with 2 edges only) would be represented under this convention as:

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

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

Our construction of adversary A' is organized in two parts. In the first part, called the *setup* phase (Figure Figure 9.1), 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 Figure 9.1 give some intuition about the semantics of these random values.) The second part of A'—the *execution* phase—which is shown in Figure Figure 9.2, is the one in which A' runs A (in a black-box 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 procedure $\mathcal{O}_b^{\mathbb{P}}$.

It is easy to check that the running time of A' is bounded from above by $\tau + 2n\tau(\mathbb{R}) + q\tau(\mathbb{E})$, which is the same as the quantity τ_1 in Theorem 9.3.3.

9.5.3 The Analysis

For any execution of A, we define the *transcript* of that execution as 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 $\mathcal{G}(\mathsf{A}), \mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ and $\mathcal{V}^{\mathsf{corr}}(\mathsf{A})$ 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 Figure 9.1 and Figure 9.2. So, unless otherwise specified, $\mathcal{G}(\mathsf{A}), \mathcal{V}^{\mathsf{chal}}(\mathsf{A}), \mathcal{V}^{\mathsf{corr}}(\mathsf{A})$ should be treated as random variables defined for this particular view.

Let Bad be the event that A''s simulation of replies to A's queries is unsuccess-

<u>PHASE I: SETUP</u>
(Generating keys and preparing to reply to A*'s queries)*

001. Sample $\ell - 1$ numbers $u_0, u_1, \cdots, u_{\ell-2}$ independently from the set $\{0, 1, 2, \cdots, n\}$ such that for each $j \in \{0, 1, \cdots, \ell - 2\}$, 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 $u_{\ell-1}$ and u_ℓ independently and uniformly at random from $[n]$.
   (The sequence $(u_0, u_1, \cdots, u_\ell)$ is A''s "guess" for a path; a successful execution of A' will be one in which this sequence is a valid path in $\mathcal{G}(\mathsf{A})$. 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 $(u_0, u_1, \cdots, u_{\ell-1}, u_\ell)$.
   (u_s is the **s***tart node of the path guessed by A'. While replying to A's queries, A' will associate the key used by its encryption oracle $\mathcal{O}_b^{\mathbb{P}}$, with u_s. Note that $s \leq \ell - 1$ always.)*

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

004. Generate keys $k_1, \cdots, k_{u_s-1}, k_{u_s+1}, \cdots, k_n$ and $r_{u_s}, r_{u_{s+1}}, \cdots, r_{u_\ell}$ using independent invocations of $\mathbb{R}$.
   (The r_i's are used in creating "fake" ciphertexts in the execution phase.)

Figure 9.1: The first phase of the adversary constructed for the proof of Theorem 9.3.3.

ful, that is, the values $u_0, \cdots, u_\ell$ that it selects are such that:

(a) $u_\ell \notin \mathcal{V}^{\mathsf{chal}}(\mathsf{A})$; OR

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

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

100.  Initialize an array of booleans $seen[s], seen[s+1], \dots, seen[\ell-1]$.
          Set each of these to be $false$.
200.  Run A. When A issues a query encrypt$(x, y)$, do the following—
210.      If $x = u_s \wedge y = u_{s+1}$,
              $seen[s] \leftarrow true$.
              If $b_s = 0$, reply with $\mathcal{O}_b^{\mathbb{P}}(k_{u_{s+1}}, r_{u_{s+1}})$.
              If $b_s = 1$, reply with $\mathcal{O}_b^{\mathbb{P}}(r_{u_{s+1}}, k_{u_{s+1}})$.
220.      If $x = u_s \wedge y \notin \{u_{s+1}, \cdots, u_{\ell-1}, u_\ell\}$,
              Reply with $\mathcal{O}_b^{\mathbb{P}}(k_y, k_y)$.
230.      If $y = u_s$,
              Reply with $\mathbb{E}_{k_x}(r_{u_s})$. (Since A' does not know the value of
              $k$—the key associated with $u_s$—its reply to every edge
              $x \rightarrow u_s$ created by A is a fake ciphertext.)
240.      For every $j \in \{s, \cdots, \ell-1\}$, 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 $\mathbb{E}_{k_x}(r_{u_{j+1}})$
                  If $x = u_s$, reply with $\mathcal{O}_b^{\mathbb{P}}(r_{u_{j+1}}, r_{u_{j+1}})$
242.          If $x \neq u_j \wedge y = u_{j+1} \wedge seen[j]$,
                  If $x \neq u_s$, reply with $\mathbb{E}_{k_x}(k_{u_{j+1}})$
                  If $x = u_s$, reply with $\mathcal{O}_b^{\mathbb{P}}(k_{u_{j+1}}, k_{u_{j+1}})$
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)
                  $seen[j] \leftarrow true$.
                  If $b_j = b_{j-1}$, reply with $\mathbb{E}_{k_{u_j}}(k_{u_{j+1}})$.
                  If $b_j \neq b_{j-1}$, reply with $\mathbb{E}_{k_{u_j}}(r_{u_{j+1}})$.
250.      If none of the above conditions are satisfied, reply with $\mathbb{E}_{k_x}(k_y)$.

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!
```

Figure 9.2: The second phase of the adversary constructed for the proof of Theorem 9.3.3.

Notice that when event Bad occurs, the output of A$'$ is a uniformly random bit. (See line 500 in Figure Figure 9.3.) Intuitively, the occurrence of Bad is computationally independent of the choice of the bit b (used in the procedure $\mathcal{O}_b^{\mathbb{P}}$), for otherwise we

```
PHASE II: EXECUTION
400.  When $\mathsf{A}$ issues a query $\texttt{challenge}(x)$, do the following—
          Reply with $k_x$ if either of the following is true:
              (a) $x = u_\ell$.
              (b) $x \neq u_\ell$ but the query $\texttt{challenge}(u_\ell)$ has been
                  made already.
          Otherwise, reply with a fresh key, sampled by running $\mathbb{R}$.
          (We stress that the reply for the query $\texttt{challenge}(u_\ell)$ is $k_{u_\ell}$.
          Also, until the query $\texttt{challenge}(u_\ell)$ is made, the reply to every
          query of the from $\texttt{challenge}(x)$ is a random value, sampled
          independently of $k_x$.)

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

600.  In the end, output whatever $\mathsf{A}$ outputs.
```

Figure 9.3: The second phase of the adversary constructed for the proof of Theorem 9.3.3 (continued).

would be contradicting the security of $\mathbb{P}$. This intuition is formalized in the following proposition:

Proposition 9.5.1 $\Delta_{\mathsf{Bad}} := |\mathbf{P}[\mathsf{Bad} \mid b = 0] - \mathbf{P}[\mathsf{Bad} \mid b = 1]| \leq \epsilon_1$

Proof Sketch: The proof uses a straightforward reduction argument. Suppose that the statement is false, that is, $\Delta_{\mathsf{Bad}} > \epsilon_1$. 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 (τ_1, ϵ_1)-security of $\mathbb{P}$. ▮

Let $\Theta_{\mathsf{A}'}$ denote the event that A' outputs 1 when given black-box access to $\mathcal{O}_b^{\mathbb{P}}$. Our goal is to bound the CPA-advantage of A', which can now be re-written as

$$\Delta^{\mathbb{P}}_{\text{ENC}}(\mathsf{A}') = |\mathbf{P}[\Theta_{\mathsf{A}'} \mid b = 0] - \mathbf{P}[\Theta_{\mathsf{A}'} \mid b = 1]|$$

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.

$$\begin{aligned}
\Delta^{\mathbb{P}}_{\text{ENC}}(\mathsf{A}') &= \left| \left(\mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}} \mid b=0] - \mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}} \mid b=1]\right) \right.\\
&\qquad \left. +\ \left(\mathbf{P}[\Theta_{\mathsf{A}'};\ \mathsf{Bad} \mid b=0] - \mathbf{P}[\Theta_{\mathsf{A}'};\ \mathsf{Bad} \mid b=1]\right) \right| \\
&\geq |\mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}} \mid b=0] - \mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}} \mid b=1]| \\
&\qquad - |\mathbf{P}[\Theta_{\mathsf{A}'};\ \mathsf{Bad} \mid b=0] - \mathbf{P}[\Theta_{\mathsf{A}'};\ \mathsf{Bad} \mid b=1]| \\
&= |\mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}} \mid b=0] - \mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}} \mid b=1]| \\
&\qquad - \left| \underbrace{\mathbf{P}[\Theta_{\mathsf{A}'} \mid b=0;\ \mathsf{Bad}]}_{=\frac{1}{2}} \cdot \mathbf{P}[\mathsf{Bad} \mid b=0] \right. \\
&\qquad\quad \left. - \underbrace{\mathbf{P}[\Theta_{\mathsf{A}'} \mid b=1;\ \mathsf{Bad}]}_{=\frac{1}{2}} \cdot \mathbf{P}[\mathsf{Bad} \mid b=1] \right| \\
&= |\mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}} \mid b=0] - \mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}} \mid b=1]| - \frac{1}{2} \cdot \Delta_{\mathsf{Bad}} \\
&\geq |\mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}} \mid b=0] - \mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}} \mid b=1]| - \frac{\epsilon_1}{2} \qquad (9.2)
\end{aligned}$$

(Follows from Prop. 9.5.1)

Let $\Delta := \mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}};\ b=0] - \mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}};\ b=1]$. Inequality (9.2) can be re-written in terms of Δ as follows:

$$\begin{aligned}
\Delta^{\mathbb{P}}_{\text{ENC}}(\mathsf{A}') &\geq |\mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}} \mid b=0] - \mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}} \mid b=1]| - \frac{\epsilon_1}{2} \\
&= \left| \frac{\mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}};\ b=0]}{\mathbf{P}[b=0]} - \frac{\mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}};\ b=1]}{\mathbf{P}[b=1]} \right| - \frac{\epsilon_1}{2} \\
&= \left| \frac{\mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}};\ b=0]}{1/2} - \frac{\mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}};\ b=1]}{1/2} \right| - \frac{\epsilon_1}{2} \\
&= 2 \cdot \left|\mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}};\ b=0] - \mathbf{P}[\Theta_{\mathsf{A}'};\ \overline{\mathsf{Bad}};\ b=1]\right| - \frac{\epsilon_1}{2} \\
&= 2 \cdot |\Delta| - \frac{\epsilon_1}{2}
\end{aligned}$$

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

$$\alpha = \frac{1}{n^2 \cdot (2n+1)^{\ell-1}} \qquad (9.3)$$

Our goal is to show that:

Lemma 9.5.2 $|\Delta| > \frac{\alpha\epsilon}{2}$

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

$$\begin{aligned}\Delta^{\mathbb{P}}_{\text{ENC}}(\mathsf{A}') &\geq 2 \cdot |\Delta| - \epsilon_1/2 \\ &> 2 \cdot \frac{\alpha\epsilon}{2} - \frac{\epsilon_1}{2} \qquad \text{(Plugging in Lemma 9.5.2)} \\ &= \alpha\epsilon - \frac{1}{2} \cdot \frac{2\alpha\epsilon}{3} = \alpha\epsilon - \frac{\alpha\epsilon}{3} = \frac{2\alpha\epsilon}{3} = \epsilon_1\end{aligned}$$

which is what our initial goal—inequality (9.1)—was.

9.5.4 Proof of Lemma 9.5.2

Let Θ_{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 Θ_{A} and $\Theta_{\mathsf{A}'}$ are exactly the same; that is, $\mathbf{P}[\Theta_{\mathsf{A}'} \mid \overline{\mathsf{Bad}}] = \mathbf{P}[\Theta_{\mathsf{A}} \mid \overline{\mathsf{Bad}}]$. Using this observation, we can re-write Δ as

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

We break up the event $\overline{\mathsf{Bad}}$ into ℓ mutually exclusive events $\Lambda_0, \Lambda_1, \cdots, \Lambda_{\ell-1}$ as follows: for each $j \in \{0, \cdots, \ell - 1\}$, define Λ_j as the event that the values $u_0, u_1, \cdots, u_\ell$ selected by A' satisfy the following three conditions:

(a) $u_\ell \in \mathcal{V}^{\mathsf{chal}}(\mathsf{A})$;

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

(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 $j' \in \{j, j + 1, \cdots, \ell\}$, $u_{j'} \neq 0$. Clearly, for any distinct j and j' (in $\{0, \cdots, \ell - 1\}$), Λ_j and $\Lambda_{j'}$ are mutually exclusive and $\overline{\mathsf{Bad}} = \bigvee_{j=0}^{\ell-1} \Lambda_j$. For each $j \in \{0, \cdots, \ell-1\}$, we define a quantity Δ_j as follows:

$$\Delta_j := \mathbf{P}[\Theta_{\mathsf{A}};\ \Lambda_j;\ b=0] - \mathbf{P}[\Theta_{\mathsf{A}};\ \Lambda_j;\ b=1]$$

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

$$\begin{aligned}
\Delta &= \mathbf{P}[\Theta_{\mathsf{A}};\ \overline{\mathsf{Bad}};\ b=0] - \mathbf{P}[\Theta_{\mathsf{A}};\ \overline{\mathsf{Bad}};\ b=1] \\
&= \mathbf{P}[\Theta_{\mathsf{A}};\ \bigvee_{j=0}^{\ell-1} \Lambda_j;\ b=0] - \mathbf{P}[\Theta_{\mathsf{A}};\ \bigvee_{j=0}^{\ell-1} \Lambda_j;\ b=1] \\
&= \sum_{j=0}^{\ell-1} (\mathbf{P}[\Theta_{\mathsf{A}};\ \Lambda_j;\ b=0] - \mathbf{P}[\Theta_{\mathsf{A}};\ \Lambda_j;\ b=1]) \\
&= \sum_{j=0}^{\ell-1} \Delta_j \qquad (9.4)
\end{aligned}$$

We now work towards breaking down the events $\Lambda_0, \cdots, \Lambda_{\ell-1}$ 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 $\mathcal{G}(\mathsf{A})$ created during this interaction by $\mathit{Indegree}(u_j)$. (If $u_j = 0$, we define $\mathit{Indegree}(u_j)$ to be 0.) We think of nodes in $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ 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 $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$", we imply that `challenge`$(u_\ell)$ 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 $j \in [\ell]$, any $d, d_\ell, d_{\ell-1}, \cdots, d_{j+1}, d_j \in [n]$, and any $i, i_\ell, i_{\ell-1}, \cdots, i_{j+1}, i_j \in [n]$ such that $i \leq d, i_\ell \leq d_\ell, \cdots, i_j \leq d_j$, let $\Psi_{(i, i_\ell, \cdots, i_j)}^{(d, d_\ell, \cdots, d_j)}$ denote the event that

(a) Λ_{j-1} occurs; and

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

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

The events $\Lambda_0, \Lambda_1, \cdots, \Lambda_{\ell-1}$ can quite easily be expressed in terms of events of the above type. For any $j \in \{0, \cdots, \ell-1\}$, Λ_j occurs if and only if the size of $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ equals d for *some* $d \in [n]$, and u_ℓ is the ith node in $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ for *some* $i \in [d]$ and has non-zero in-degree d_ℓ for *some* d_ℓ in $[n]$, and $u_{\ell-1}$ is the i_ℓth node pointing at u_ℓ in $\mathcal{G}(\mathsf{A})$ for *some* $i_\ell \in [d_\ell]$, and so on, all the way upto u_j. Put succinctly, for any $j \in \{0, \cdots, \ell-1\}$:

$$\Lambda_j = \bigvee_{d=1}^{n} \bigvee_{i=1}^{d} \bigvee_{d_\ell=1}^{n} \bigvee_{i_\ell=1}^{d_\ell} \cdots \bigvee_{d_{j+1}=1}^{n} \bigvee_{i_{j+1}=1}^{d_{j+1}} \Psi_{(i,i_\ell,\cdots,i_{j+1})}^{(d,d_\ell,\cdots,d_{j+1})}$$

Clearly, for any distinct pairs of vectors $(d, d_\ell, \cdots, d_j), (i, i_\ell, \cdots, i_j)$ and $(d', d'_\ell, \cdots, d'_{j'}), (i', i'_\ell, \cdots, i'_{j'})$, the events $\Psi_{(i,i_\ell,\cdots,i_j)}^{(d,d_\ell,\cdots,d_j)}$ and $\Psi_{(i',i'_\ell,\cdots,i'_{j'})}^{(d',d'_\ell,\cdots,d'_{j'})}$ are mutually exclusive, and so

$$\begin{aligned}\Delta_j &= \mathbf{P}[\Theta_{\mathsf{A}};\ \Lambda_j;\ b=0] - \mathbf{P}[\Theta_{\mathsf{A}};\ \Lambda_j;\ b=1] \\ &= \sum_{\substack{d\in[n],\\ i\in[d]}} \sum_{\substack{d_\ell\in[n],\\ i_\ell\in[d_\ell]}} \cdots \sum_{\substack{d_{j+1}\in[n],\\ i_{j+1}\in[d_{j+1}]}} \left[\begin{array}{r} \mathbf{P}[\Theta_{\mathsf{A}};\ \Psi_{(i,i_\ell,\cdots,i_{j+1})}^{(d,d_\ell,\cdots,d_{j+1})};\ b=0] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}};\ \Psi_{(i,i_\ell,\cdots,i_{j+1})}^{(d,d_\ell,\cdots,d_{j+1})};\ b=1] \end{array} \right]\end{aligned} \quad (9.5)$$

We will now show how to sum up this quantity with j ranging from 0 through $\ell-1$ and to express the sum (which is the same as Δ) in terms of $\Delta_{\mathrm{ADPT}}^{\mathbb{P}}(\mathsf{A})$. Towards this, we first define another type of event, similar to events of the other type.

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

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

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

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

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

(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 $c \leftarrow \mathbb{E}_{k_{u_{j'-1}}}(k_{u_{j'}})$.)

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:

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

In words, Θ_{first} (resp. Θ_{last}) is the event that for some $\tilde{j} \in \{0, \cdots, \ell-1\}$, $\Lambda_{\tilde{j}}$ occurs and $\mathit{Indegree}(u_{\tilde{j}}) = 0$, that u_ℓ is the *first* (resp. *last*) node in $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$, that $u_{\ell-1}$ is the *first* (resp. *last*) node pointing at u_ℓ in $\mathcal{G}(\mathsf{A})$, that the reply A receives for the query $\mathtt{encrypt}(u_{\ell-1}, u_\ell)$ 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 its view when interacting with the procedure $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$. This is formalized in the following claim.

Claim 9.5.3 $|\mathbf{P}[\Theta_{\mathsf{A}} \mid \Theta_{\text{first}}] - \mathbf{P}[\Theta_{\mathsf{A}} \mid \Theta_{\text{last}}]| = \Delta^{\mathbb{P}}_{\text{ADPT}}(\mathsf{A})$

In the next claim, we relate the probabilities of the events Θ_{first} and Θ_{last} to the quantity α defined in equation 9.3.

Claim 9.5.4 $\mathbf{P}[\Theta_{\text{first}}] = \mathbf{P}[\Theta_{\text{last}}] = \alpha/2$

The proof of these claims are postponed to Sections 9.5.5 and 9.5.6 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 9.5), express the sum in terms of Θ_{first} and Θ_{last}, and thus relate it to $\Delta^{\mathbb{P}}_{\text{ADPT}}(\mathsf{A})$.

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:

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

(If $j = \ell - 1$, $\mathsf{XOR}(\nu, \overrightarrow{\nu}_j)$ equals $(\nu \oplus \nu_j)$.) Let $\Theta^{(d,d_\ell,\cdots,d_j,0)}_{(i,i_\ell,\cdots,i_j,0)}(1 \cdot \overrightarrow{\nu}_{j-1})$ denote the event that $\Theta^{(d,d_\ell,\cdots,d_j)}_{(i,i_\ell,\cdots,i_j)}(\overrightarrow{\nu}_{j-1})$ 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 $\mathbf{P}[\Theta_{\mathsf{A}};\ E_1] = \mathbf{P}[\Theta_{\mathsf{A}};\ E_2]$.

Lemma 9.5.5 (Hybrid Cancellation Lemma - II)

1. For all $d \in \{2, \cdots, n\}$ and $i \in \{1, \cdots, d-1\}$,
$$\bigvee_{d_\ell=1}^{n} \Theta^{(d,d_\ell)}_{(i,d_\ell)}((1)) \equiv \bigvee_{d_\ell=1}^{n} \Theta^{(d,d_\ell)}_{(i+1,1)}((0))$$

2. For all $j \in \{1, \cdots, \ell\}$, for all $d, d_\ell, \cdots, d_j \in [n]$ and $i, i_\ell, \cdots, i_j$ such that $1 \leq i \leq d, 1 \leq i_\ell \leq d_\ell, \cdots, 1 \leq i_j \leq d_j$, and for any bitvector $\overrightarrow{\nu}_j = (\nu_j, \cdots, \nu_{\ell-1}) \in \{0, 1\}^{\ell-j}$:
$$\Theta^{(d,d_\ell,\cdots,d_j)}_{(i,i_\ell,\cdots,i_j)}(1 \cdot \overrightarrow{\nu}_j) \equiv \Theta^{(d,d_\ell,\cdots,d_{j+1},d_j)}_{(i,i_\ell,\cdots,i_{j+1},i_j+1)}(0 \cdot \overrightarrow{\nu}_j)$$

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

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

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

   (a) If $j = 1$, then

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

   (b) If $j > 1$, then

$$\begin{aligned}&\Psi^{(d,d_\ell,\cdots,d_j)}_{(i,i_\ell,\cdots,i_j)} \wedge (b = \nu) \equiv \\ &\quad \bigvee\nolimits_{d_{j-1}=0}^{n} \left(\bigvee\nolimits_{\nu_{j-1},\nu_j,\cdots,\nu_{\ell-2} \in \{0,1\}} \Theta^{(d,d_\ell,\cdots,d_j,d_{j-1})}_{(i,i_\ell,\cdots,i_j,d_{j-1})}(1 \cdot \mathsf{XOR}(\nu, \overrightarrow{\nu}_{j-1}))\right)\end{aligned}$$

The proof of this lemma appears in Section 9.5.7. 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 $j \in \{0, \cdots, \ell-1\}$, let

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

Lemma 9.5.6 (Telescoping Sums Lemma) For all $j \in \{0, 1, \cdots, \ell-1\}$, $\overline{\Delta}_j$ equals

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

The proof of this lemma is given in Section 9.5.8. Given these two lemmas and Claims 9.5.3 and 9.5.4, the final result (Lemma 9.5.2) is quite easy to prove. Using Lemma 9.5.6, equation (9.4) can be re-written as:

$$\begin{aligned}
\Delta &= \overline{\Delta}_{\ell-1} \\
&= \sum_{\substack{d\in[n],\\ i\in[d]}} \sum_{\substack{d_\ell\in[n],\\ i_\ell\in[d_\ell]}} \sum_{\nu_{\ell-1}=0} \left(\begin{array}{l} \mathbf{P}[\Theta_{\mathsf{A}};\ \Theta^{(d,d_\ell)}_{(i,i_\ell)}(\mathsf{XOR}(0, \overrightarrow{\nu}_{\ell-1}))] \\ \qquad - \mathbf{P}[\Theta_{\mathsf{A}};\ \Theta^{(d,d_\ell)}_{(i,i_\ell)}(\mathsf{XOR}(1, \overrightarrow{\nu}_{\ell-1}))] \end{array} \right) \\
&= \sum_{\substack{d\in[n],\\ i\in[d]}} \sum_{\substack{d_\ell\in[n],\\ i_\ell\in[d_\ell]}} \Big(\mathbf{P}[\Theta_{\mathsf{A}};\ \Theta^{(d,d_\ell)}_{(i,i_\ell)}((0))] - \mathbf{P}[\Theta_{\mathsf{A}};\ \Theta^{(d,d_\ell)}_{(i,i_\ell)}((1))]\Big) \\
&= \sum_{\substack{d\in[n],\\ i\in[d]}} \sum_{d_\ell\in[n]} \Big(\mathbf{P}[\Theta_{\mathsf{A}};\ \Theta^{(d,d_\ell)}_{(i,1)}((0))] - \mathbf{P}[\Theta_{\mathsf{A}};\ \Theta^{(d,d_\ell)}_{(i,d_\ell)}((1))]\Big) \\
&\quad \text{(Follows from the hybrid cancellation lemma, Lemma 9.5.5, part 2)} \\
&= \sum_{d=1}^{n} \sum_{d_\ell=1}^{n} \left[\sum_{i=1}^{d} \Big(\mathbf{P}[\Theta_{\mathsf{A}};\ \Theta^{(d,d_\ell)}_{(i,1)}((0))] - \mathbf{P}[\Theta_{\mathsf{A}};\ \Theta^{(d,d_\ell)}_{(i,d_\ell)}((1))]\Big) \right] \\
&= \sum_{d=1}^{n} \sum_{d_\ell=1}^{n} \Big(\mathbf{P}[\Theta_{\mathsf{A}};\ \Theta^{(d,d_\ell)}_{(1,1)}((0))] - \mathbf{P}[\Theta_{\mathsf{A}};\ \Theta^{(d,d_\ell)}_{(d,d_\ell)}((1))]\Big) \\
&\quad \text{(Follows from the hybrid cancellation lemma, part 1)} \\
&= \mathbf{P}[\Theta_{\mathsf{A}};\ \Theta_{\text{first}}] - \mathbf{P}[\Theta_{\mathsf{A}};\ \Theta_{\text{last}}] \\
&= \mathbf{P}[\Theta_{\mathsf{A}} \mid \Theta_{\text{first}}] \cdot \mathbf{P}[\Theta_{\text{first}}] - \mathbf{P}[\Theta_{\mathsf{A}} \mid \Theta_{\text{last}}] \cdot \mathbf{P}[\Theta_{\text{last}}]
\end{aligned}$$

Now invoking Claims 9.5.3 and 9.5.4, we get

$$\begin{aligned}
|\Delta| &= |\mathbf{P}[\Theta_{\mathsf{A}} \mid \Theta_{\text{first}}] \cdot \mathbf{P}[\Theta_{\text{first}}] - \mathbf{P}[\Theta_{\mathsf{A}} \mid \Theta_{\text{last}}] \cdot \mathbf{P}[\Theta_{\text{last}}]| \\
&= \frac{\alpha}{2} \cdot |\mathbf{P}[\Theta_{\mathsf{A}} \mid \Theta_{\text{first}}] - \mathbf{P}[\Theta_{\mathsf{A}} \mid \Theta_{\text{last}}]| \\
&= \frac{\alpha}{2} \cdot \Delta^{\mathbb{P}}_{\text{ADPT}}(\mathsf{A}) \\
&> \frac{\alpha\epsilon'}{2} \quad \text{(Follows from our initial assumption, inequality (9.1))}
\end{aligned}$$

We next give the proofs of all the lemmas and claims used to prove Lemma 9.5.2.

9.5.5 Proof of Claim 9.5.3

We 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 when interacting with $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},0}$ (resp. $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},1}$). This suffices to prove the claim.

When Θ_{first} occurs, *every* query of A of the form $\texttt{encrypt}(x, y)$ is replied to with a real ciphertext ($\mathbb{E}_{k_x}(k_y)$) and that of the form $\texttt{challenge}(x)$ is replied to with k_x—exactly the manner in which replies are created by the procedure $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},0}$. Why is this? To see why the former is true, note that under the occurrence of Θ_{first}, the only queries that could be replied to with fake ciphertexts are those of the form $\texttt{encrypt}(x, u_\ell)$ and either the same as or preceding the query $\texttt{encrypt}(u_{\ell-1}, u_\ell)$. But, given Θ_{first} occurs, $u_{\ell-1}$ is the first node pointing at u_ℓ and the reply to $\texttt{encrypt}(u_{\ell-1}, u_\ell)$ is real, thus implying that the replies for all 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 $\texttt{challenge}(u_\ell)$ is always k_x (line 400 of A′'s code), and when Θ_{first} occurs, $\texttt{challenge}(u_\ell)$ is the first 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 $\texttt{encrypt}(x, u_\ell)$, either including or preceding $\texttt{encrypt}(u_{\ell-1}, u_\ell)$. However, under the occurrence of Θ_{last}, $\texttt{encrypt}(u_{\ell-1}, u_\ell)$ is the *last* query of this form, and even the reply for $\texttt{encrypt}(u_{\ell-1}, u_\ell)$ is fake, thus implying that every query of the form $\texttt{encrypt}(x, u_\ell)$ is replied to in a fake manner (specifically, with $\mathbb{E}_{k_x}(r_{u_\ell})$). Note also that given Θ_{last}, $\texttt{challenge}(u_\ell)$ is the last challenge query, and thus, the reply to all $\texttt{challenge}(x)$ queries *except* $\texttt{challenge}(u_\ell)$ is a random key, sampled independently of k_x. For $\texttt{challenge}(u_\ell)$, the reply is k_{u_ℓ}. Now comes the crucial part: Replying to the query $\texttt{challenge}(u_\ell)$ with k_{u_ℓ} and to every query of the form $\texttt{encrypt}(x, u_\ell)$ with $\mathbb{E}_{k_x}(r_{u_\ell})$—where r_{u_ℓ} is independent of (but identically distributed as) k_{u_ℓ}—produces the same distribution on the replies as replying to $\texttt{challenge}(u_\ell)$ with r_{u_ℓ} and to every query of the form $\texttt{encrypt}(x, u_\ell)$ with $\mathbb{E}_{k_x}(k_{u_\ell})$. Furthermore, we observe that neither k_{u_ℓ} nor r_{u_ℓ} is used in creating replies for A other than those of the form $\texttt{encrypt}(x, u_\ell)$ or

$\texttt{challenge}(u_\ell)$. In effect, the replies that A' provides to A conditioned on event Θ_{last}, are distributed exactly as the procedure $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},1}$'s replies to A—real ciphertexts for all $\texttt{encrypt}$ queries and a random key (independent of k_x) for every query of the form $\texttt{challenge}(x)$. ▌

9.5.6 Proof of Claim 9.5.4

We 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.

First, some notations. Recall that in any execution of A, the graph $\mathcal{G}(\mathsf{A})$ and the set $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ are random variables depending 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 $\mathcal{G}(\mathsf{A})$ and $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ and the manner in which these are created in any execution.

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

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

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 $\mathcal{G}(\mathsf{A}), \mathcal{V}^{\mathsf{chal}}(\mathsf{A}), u_0, \cdots, u_\ell, s, b_s, \cdots, b_{\ell-2}$ and the bit b (chosen by $\mathcal{O}^{\mathbb{P}}_b$) are such that:

1. $\mathit{fchal} = u_\ell$;
2. $\mathit{fnode}(u_\ell) = u_{\ell-1}$;

3. $s = \ell - \mathit{flen}(u_\ell) = \ell - \mathit{flen}(u_{\ell-1}) - 1$ and $\mathit{fpath}(u_{\ell-1}) = (u_s, u_{s+1}, \cdots, u_{\ell-1})$[6]

4. For all $j \in \{0, 1, \cdots, s-1\}$, $u_j = 0$.

5. $b = 0$ and $b_s = b_{s+1} = \cdots = b_{\ell-2} = 0$.[7]

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 $b_{\ell-1} = 0$ 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:

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

The above expression of $\mathbf{P}[\Theta_{\text{first}}]$ gives us some intuition about why Claim 9.5.4 is correct: for any value of $\ell' \in \{0, \ldots, \ell-1\}$ and $w_\ell, \ldots, w_{\ell-\ell'-1} \in [n]$, the probability that $(u_\ell, \ldots, u_{\ell-\ell'-1}) = (w_\ell, \ldots, w_{\ell-\ell'-1})$, $u_{\ell-\ell'-2} = \cdots = u_0 = 0$ and $b = b_{\ell-\ell'-1} = \cdots = b_{\ell-2} = 0$ is exactly equal to $\alpha/2$. (This probability is computed

[6]Throughout the proof of Claim 9.5.4 and Lemma 9.5.5, we denote paths in $\mathcal{G}(\mathsf{A})$ as sequences of nodes, but without any zeroes prepended to the first node.

[7]While expressing Θ_{last} in terms of $\mathit{fchal}, \mathit{fnode}(\cdot), \mathit{fpath}(\cdot)$ etc., the first two conditions and the last condition are suitably modified as follows—u_ℓ is the *last* node in $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$, $u_{\ell-1}$ is the *last* node pointing at u_ℓ and $b, b_s, \cdots, b_{\ell-2}$ are all equal to 1. The other conditions remain the same as above.

using the independence property of the u_i's and the b_i's; details appear below.) Thus, $\mathbf{P}[\Theta_{\text{first}}]$ equals $\alpha/2$ multiplied by the probability of the event $fchal = w_\ell;\ fnode(w_\ell) = w_{\ell-1};\ flen(w_{\ell-1}) = \ell';\ fpath(w_{\ell-1}) = (w_{\ell-\ell'-1}, \cdots, w_{\ell-2}, w_{\ell-1})$ *summed up* over all possible values of ℓ' and the w_i's. Since the latter sum equals 1, the value of $\mathbf{P}[\Theta_{\text{first}}]$ is exactly $\alpha/2$.

Translating this intuition into proof is, however, not straightforward since we need to account for the dependencies between the u_i's and the random functions $fnode(\cdot), fpath(\cdot)$ and $flen(\cdot)$. The details of this process are somewhat cumbersome and could be skipped for faster perusal of the rest of the proof of Theorem 9.3.3.

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

$$E_1^{(\ell',\overrightarrow{w})} = \begin{pmatrix} fchal = w_\ell \ \wedge\ fnode(w_\ell) = w_{\ell-1} \ \wedge\ flen(w_{\ell-1}) = \ell' \\ \wedge\ fpath(w_{\ell-1}) = (w_{\ell-\ell'-1}, \cdots, w_{\ell-1}) \end{pmatrix}$$

$$E_2^{(\ell',\overrightarrow{w})} = \begin{pmatrix} u_l = w_\ell \ \wedge\ \cdots \ \wedge\ u_{\ell-\ell'-1} = w_{\ell-\ell'-1} \ \wedge\ u_{\ell-\ell'-2} = \cdots = u_0 = 0 \\ \wedge\ b = 0 \ \wedge\ b_{\ell-\ell'-1} = \cdots = b_{\ell-2} = 0 \end{pmatrix}$$

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

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

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

$$\begin{aligned}
\mathbf{P}[E_2^{(\ell',\vec{w})}] &= \mathbf{P}\left[\begin{array}{c} u_\ell = w_\ell;\ \ u_{\ell-1} = w_{\ell-1};\ \ \cdots;\ u_{\ell-\ell'-1} = w_{\ell-\ell'-1}; \\ u_{\ell-\ell'-2} = \cdots = u_0 = 0; \\ b = 0;\ \ b_{\ell-\ell'-1} = \cdots = b_{\ell-2} = 0 \end{array}\right] \\
&= \mathbf{P}[u_\ell = w_\ell] \cdot \mathbf{P}[u_{\ell-1} = w_{\ell-1}] \cdots \mathbf{P}[u_{\ell-\ell'-1} = w_{\ell-\ell'-1}] \\
&\qquad\qquad \times\ \mathbf{P}[u_{\ell-\ell'-2} = 0] \cdot \mathbf{P}[u_{\ell-\ell'-3} = 0] \cdots \mathbf{P}[u_0 = 0] \\
&\qquad\qquad\qquad \times\ \mathbf{P}[b = 0] \cdot \mathbf{P}[b_{\ell-\ell'-1} = 0] \cdots \mathbf{P}[b_{\ell-2} = 0] \\
&= \left\{\frac{1}{n} \cdot \frac{1}{n} \cdot \left(\frac{2}{2n+1}\right)^{\ell'}\right\} \times \left\{\left(\frac{1}{2n+1}\right)^{\ell-\ell'-1}\right\} \times \left\{\frac{1}{2} \cdot \left(\frac{1}{2}\right)^{\ell'}\right\}
\end{aligned}$$

(To understand this step, observe the probability distribution associated with the u_j's in line 001 of figure Figure 9.1)

$$= \frac{1}{2n^2} \cdot \left(\frac{1}{2n+1}\right)^{\ell'} \cdot \left(\frac{1}{2n+1}\right)^{\ell-\ell'-1} \quad = \quad \frac{1}{2n^2(2n+1)^{\ell-1}} \quad = \quad \frac{\alpha}{2}$$

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

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

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

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

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

PROVING $S_p = 1$: Fix ℓ' and $\vec{w}$. A transcript of A's execution is called a *good* transcript if the event $E_1^{(\ell',\vec{w})}$ 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)* $\mathit{fchal} = w_\ell$; *(ii)* $\mathit{fnode}(w_\ell) = w_{\ell-1}$; *(iii)* $\mathit{flen}(w_{\ell-1}) = \ell'$; and *(iv)* $\mathit{fpath}(w_{\ell-1}) = (w_{\ell-\ell'-1}, \cdots, w_{\ell-1})$. This essentially boils down to verifying that:

(a) $\texttt{challenge}(w_\ell)$ is the first `challenge` query made by A;

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

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

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

- $\mathsf{q}_i = \texttt{challenge}(x)$ for some $x \neq w_\ell$ and the query $\texttt{challenge}(w_\ell)$ does not occur before it in $\mathfrak{t}$; or
- $\mathsf{q}_i = \texttt{encrypt}(x, w_j)$ (for some $j \in \{\ell-\ell', \cdots, \ell-1\}$ and $x \neq w_{j-1}$) and the query $\texttt{encrypt}(w_{j-1}, w_j)$ does not occur before it in $\mathfrak{t}$; or
- q_i is of the form $\texttt{encrypt}(x, w_{\ell-\ell'-1})$.[8]

Clearly, every bad transcript contains at least one bad query. The *longest good prefix* of a bad transcript $\mathfrak{t} = (\mathsf{q}_1, \mathsf{r}_1, \cdots, \mathsf{q}_f, \mathsf{r}_f)$ 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 $\mathfrak{t}$, the *relevant* transcript corresponding to $\mathfrak{t}$, $\mathit{rel}(\mathfrak{t})$, is defined as follows: if $\mathfrak{t}$ is good, $\mathit{rel}(\mathfrak{t}) = \mathfrak{t}$, else $\mathit{rel}(\mathfrak{t})$ is the longest good prefix of $\mathfrak{t}$. Note that relevant transcripts contain only good queries. For any multiset of transcripts, $\mathcal{T}$, let $\mathit{rel}(\mathcal{T})$ denote the multiset of relevant transcripts corresponding to $\mathcal{T}$; that is, $\mathit{rel}(\mathcal{T}) = \{\mathit{rel}(\mathfrak{t}) \mid \mathfrak{t} \in \mathcal{T}\}$. 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.)

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

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

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

(a) First, notice that, for each transcript in $rel(\mathcal{T}_{\mathsf{A} \mid E_2^{(\ell',\overrightarrow{w})}})$, none of the replies given to the adversary involve the random variables $r_{u_s}, r_{u_s+1}, \cdots, r_{u_\ell}$ at all. Why is this? Given that $E_2^{(\ell',\overrightarrow{w})}$ occurs, these variables could be used in replying to queries either of the form $\texttt{encrypt}(x, w_s)$, or else of the form $\texttt{encrypt}(x, w_j)$ ($j \in \{s+1, \cdots, \ell\}$) *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 $rel(\mathcal{T}_{\mathsf{A} \mid E_2^{(\ell',\overrightarrow{w})}})$ do not contain such queries!

Thus, every relevant execution of A in its interaction with A' given $E_2^{(\ell',\overrightarrow{w})}$ 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 $r_{\mathbb{E}}$ and r_{A}. Let $\hat{\mathcal{T}}_{\mathsf{A} \mid E_2^{(\ell',\overrightarrow{w})}}$ denote the multiset of transcripts containing one transcript for each such execution. (*Note:* $\hat{\mathcal{T}}_{\mathsf{A} \mid E_2^{(\ell',\overrightarrow{w})}}$ contains the same transcripts as in $rel(\mathcal{T}_{\mathsf{A} \mid E_2^{(\ell',\overrightarrow{w})}})$ 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 $\hat{\mathcal{T}}_{\mathsf{A} \mid E_2^{(\ell',\overrightarrow{w})}}$, the replies given for a query of the form $\texttt{encrypt}(x, y)$ is a real ciphertext (that is, $\mathbb{E}_{k_x}(k_y)$) and that of the form $\texttt{challenge}(x)$ is a real key (that is, k_x). This is exactly the manner in which replies to queries are created by the procedure $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},0}$. Thus, if we were to consider only relevant executions of A, the view of A when interacting with $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},0}$ has the same distribution as its view in its interaction with A$'$ given $E_2^{(\ell',\overrightarrow{w})}$ occurs.

It follows that the quantity p is equal to the probability that event $E_1^{(\ell',\overrightarrow{w})}$ takes place when A interacts with $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},0}$. S_p simply sums this probability over all possible values of ℓ' and $\overrightarrow{w}$ and must thus be equal to 1. ∎

9.5.7 Proof of the Hybrid Cancellation Lemma II (Lemma 9.5.5)

Let us first set up some new notations specific to the proof of 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 $u_0, \cdots, u_\ell, b_s, \cdots, b_{\ell-2}$ (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:

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

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 is done by $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$. The responses to all `corrupt` queries are also just as they are given by $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$. 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 $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ 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 $\Theta_{(i+1,d_\ell)}^{(d,d_\ell)}((0)))$. 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 encrypt(x, u_ℓ), the reply is $\mathbb{E}_{k_x}(r_{u_\ell})$. (This is because, given $E_{d,i}^{(0)}$ occurs, the reply for the last query of the form encrypt(x, u_ℓ)—which is the same as encrypt$(u_{\ell-1}, u_\ell)$—is a fake ciphertext $\mathbb{E}_{k_{u_{\ell-1}}}(r_{u_\ell})$. This means that the reply for *every* query of that form must be fake, too.)

(d) for every other encrypt (and corrupt) query, the reply is just as given by $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$.

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 encrypt(x, u_ℓ) or challenge(u_ℓ). So, if we modify conditions (b) and (c) as below

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

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

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 given by $\mathcal{O}^{\mathbb{P}}_{\text{ADPT},b}$.

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 $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ 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 9.5.4; 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 9.5.6. Let $\mathit{lnode}(w)$ be the random variable corresponding to the *last* node pointing at w in $\mathcal{G}(\mathsf{A})$. Let $\Phi^{(d)}$ denote the event that the size of $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ 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 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:

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

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

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

$$E_1^{(\ell',\overrightarrow{w})} = \begin{pmatrix} \Phi_{(i)}^{(d)}(w_\ell) \ \wedge \ \mathit{Indegree}(w_\ell) = d_\ell \ \wedge \ \mathit{lnode}(w_\ell) = w_{\ell-1} \wedge \\ \mathit{flen}(w_{\ell-1}) = \ell' \ \wedge \ \mathit{fpath}(w_{\ell-1}) = (w_{\ell-\ell'-1}, \cdots, w_{\ell-1}) \end{pmatrix}$$

$$E_2^{(\ell',\overrightarrow{w})} = \begin{pmatrix} u_\ell = w_\ell \ \wedge \ \cdots \ \wedge \ u_{\ell-\ell'-1} = w_{\ell-\ell'-1} \ \wedge \ u_{\ell-\ell'-2} = \cdots = u_0 = 0 \wedge \\ b = 1 \ \wedge \ b_{\ell-\ell'-1} = \cdots = b_{\ell-2} = 1 \end{pmatrix}$$

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

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

As in the proof of Claim 9.5.4, we can show that $\mathbf{P}[E_2^{(\ell',\overrightarrow{w})}]$ is equal to $\alpha/2$ for every choice of ℓ' and $\overrightarrow{w}$, and so

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

We now claim that the conditional probability $\mathbf{P}[E_1^{(\ell',\overrightarrow{w})} \mid E_2^{(\ell',\overrightarrow{w})}]$ is equal to the probability of occurrence of $E_1^{(\ell',\overrightarrow{w})}$ in $\mathbf{Game}_i$. The proof of this claim uses the same ideas as used in the proof of Claim 9.5.4 (specifically, one needs to carefully establish a one-to-one correspondence between transcripts—complete ones as well as partial ones—that do not violate $E_1^{(\ell',\overrightarrow{w})}$ 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

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

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 $\tilde{E}_1^{(\ell',\overrightarrow{w})}$ and $\tilde{E}_2^{(\ell',\overrightarrow{w})}$ as follows:

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

$$\tilde{E}_2^{(\ell',\overrightarrow{w})} = \left(\begin{array}{c} u_\ell = w_\ell \ \wedge\ \cdots \ \wedge\ u_{\ell-\ell'-1} = w_{\ell-\ell'-1} \ \wedge\ u_{\ell-\ell'-2} = \cdots = u_0 = 0 \\ \wedge\ b = 0 \ \wedge\ b_{\ell-\ell'-1} = \cdots = b_{\ell-2} = 0 \end{array} \right)$$

(Notice how $\tilde{E}_1^{(\ell',\overrightarrow{w})}$ differs from $E_1^{(\ell',\overrightarrow{w})}$: We require w_ℓ to be the $(i+1)$th node in $\mathcal{V}^{\mathsf{chal}}(\mathsf{A})$ and $w_{\ell-1}$ to be the *first* node pointing at w_ℓ. Also notice that in $\tilde{E}_2^{(\ell',\overrightarrow{w})}$, 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:

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

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

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

Proof of Part 2. Fix $j, d, d_\ell, \cdots, d_j$ and $i, i_\ell, \cdots, i_j$ such that $i \in [d], i_\ell \in [d_\ell], \cdots, i_j \in [d_j]$ and fix a bitvector $\vec{\nu}_j \in \{0,1\}^{\ell-j}$. 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 $u_j, u_{j+1}, \cdots, u_\ell$ and $b_j, \cdots, b_{\ell-2}, b_{\ell-1}$ and it does so exactly as in the original version: u_ℓ and $u_{\ell-1}$ are sampled uniformly at random from $[n]$; for each $j' \in \{j, \cdots, \ell-2\}$, $\mathbf{P}[u_{j'} = 0] = 1/(2n+1)$ and $\mathbf{P}[u_{j'} = x]$ (for any $x \in [n]$) equals $2/(2n+1)$; $b_{\ell-1} = 0$; and, $b_j, \cdots, b_{\ell-2}$ are all sampled uniformly at random from $\{0,1\}$. A' then sets $b_{j-1} = \nu_j \oplus \nu_{j+1} \oplus \cdots \oplus \nu_{\ell-1}$. Finally, A' generates keys $k_1, \cdots, k_n$ (*Note again: All* keys are generated!) and also some other random values $r_{u_j}, \cdots, r_{u_\ell}$ (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 $j' \in \{j, \cdots, \ell-1\}$, the reply is real, that is $\mathbb{E}_{k_{u_{j'}}}(k_{u_{j'+1}})$, if and only if $b_{j'} = b_{j'-1}$ (and is fake, that is $\mathbb{E}_{k_{u_{j'}}}(r_{u_{j'+1}})$ otherwise); for every query of the form $\texttt{encrypt}(x, u_{j'})$ made before (resp. after) $\texttt{encrypt}(u_{j'-1}, u_{j'})$, the reply is fake (resp. real). For all other encrypt queries, except those of the form $\texttt{encrypt}(x, u_\ell)$, 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 ($\mathbb{E}_{k_x}(r_{u_j})$)* while the rest with real ones ($\mathbb{E}_{k_x}(k_{u_j})$).

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 $\Phi^{(d,d_\ell,\cdots,d_j)}_{(i,i_\ell,\cdots,i_j)}(\overrightarrow{\nu}_j, w)$. This event occurs if and only if

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

Let $\Phi^{(d,d_\ell,\cdots,d_{j+1},d_j)}_{(i,i_\ell,\cdots,i_{j+1})}(\overrightarrow{\nu}_j)$ denote the event that $\Phi^{(d,d_\ell,\cdots,d_j)}_{(i,i_\ell,\cdots,i_j)}(\overrightarrow{\nu}_j, w)$ occurs for some $w \in [n]$; that is, $\Phi^{(d,d_\ell,\cdots,d_{j+1},d_j)}_{(i,i_\ell,\cdots,i_{j+1})}(\overrightarrow{\nu}_j) = \bigvee_{w\in[n]} \Phi^{(d,d_\ell,\cdots,d_j)}_{(i,i_\ell,\cdots,i_j)}(\overrightarrow{\nu}_j, w)$. 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:

$$\begin{aligned} E^{(0)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j) &= \Theta^{(d,d_\ell,\cdots,d_j)}_{(i,i_\ell,\cdots,i_j)}(1 \cdot \overrightarrow{\nu}_j) \\ E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j) &= \Theta^{(d,d_\ell,\cdots,d_{j+1},d_j)}_{(i,i_\ell,\cdots,i_{j+1},i_j+1)}(0 \cdot \overrightarrow{\nu}_j) \end{aligned}$$

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 $\Phi^{(d,d_\ell,\cdots,d_{j+1},d_j)}_{(i,i_\ell,\cdots,i_{j+1})}(\overrightarrow{\nu}_j)$ 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 $\Phi^{(d,d_\ell,\cdots,d_{j+1},d_j)}_{(i,i_\ell,\cdots,i_{j+1})}(\overrightarrow{\nu}_j)$. (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 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 $\mathbf{P}[E_1 \mid E_2]$ is equated to an expression of the form $\mathbf{P}_{\mathbf{Game}_{j,i_j}}[E_1]$, and this is then 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} is the i_jth node pointing at u_j and the reply to the query $\texttt{encrypt}(u_{j-1}, u_j)$ is 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 is 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 the procedure $\mathcal{O}^{\mathbb{P}}_b$) 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 assignment is as follows:

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

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

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

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

$$E^{(\ell',\overrightarrow{w})}_{j,1} = \left(\begin{array}{c} \Phi^{(d,d_\ell,\cdots,d_j)}_{(i,i_\ell,\cdots,i_j)}(\overrightarrow{\nu}_j, w_{j-1}) \ \wedge\ \mathit{flen}(w_{j-1}) = \ell' \\ \wedge\ \mathit{fpath}(w_{j-1}) = (w_{j-\ell'-1}, \cdots, w_{j-1}) \end{array}\right)$$

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

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

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

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

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

Thus,

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

We claim that the probability $\mathbf{P}[E_{j,1}^{(\ell',\overrightarrow{w})} \mid E_{j,2}^{(\ell',\overrightarrow{w})}]$ is equal to the probability that $E_{j,1}^{(\ell',\overrightarrow{w})}$ occurs in $\mathbf{Game}_{j,i_j}$, that is, $\mathbf{P}[E_{j,1}^{(\ell',\overrightarrow{w})} \mid E_{j,2}^{(\ell',\overrightarrow{w})}] = \mathbf{P}_{\mathbf{Game}_{j,i_j}}[E_{j,1}^{(\ell',\overrightarrow{w})}]$. The intuition behind this is the following—given that $E_{j,2}^{(\ell',\overrightarrow{w})}$ occurs in $\mathbf{Game}_0$, none of the random values $r_{j-\ell'-1}, \cdots, r_{j-1}$ are used in any execution of A unless and until one or more conditions under $E_{j,1}^{(\ell',\overrightarrow{w})}$ are violated. Thus, if we consider transcripts of this interaction—given $E_{j,2}^{(\ell',\overrightarrow{w})}$ occurs—only upto the point where a violation of $E_{j,1}^{(\ell',\overrightarrow{w})}$ is found (that is, truncate the "bad" transcripts at the point where they start violating $E_{j,1}^{(\ell',\overrightarrow{w})}$), 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 correspondence between "non-violating" transcripts in these two interactions. The probability $\mathbf{P}[E_{j,1}^{(\ell',\overrightarrow{w})} \mid E_{j,2}^{(\ell',\overrightarrow{w})}]$ is the ratio of the *good* (that is, untruncated) 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 $\mathbf{P}_{\mathbf{Game}_{j,i_j}}[E_{j,1}^{(\ell',\overrightarrow{w})}]$. The details of the proof are similar to those in the proof of Claim 9.5.4 and are omitted.

Using the above claim, we write:

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

And now using essentially the same approach one can also equate $\mathbf{P}[E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j)]$ to $\alpha' \cdot \mathbf{P}_{\mathbf{Game}_{j,i_j}}[\Phi^{(d,d_\ell,\cdots,d_{j+1},d_j)}_{(i,i_\ell,\cdots,i_{j+1})}(\overrightarrow{\nu}_j)]$. 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 $E^{(1)}_{j,\mathbf{d},\mathbf{i}}(\overrightarrow{\nu}_j) = \Theta^{(d,d_\ell,\cdots,d_{j+1},d_j)}_{(i,i_\ell,\cdots,i_{j+1},i_j+1)}(0 \cdot \overrightarrow{\nu}_j)$ 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 $\texttt{encrypt}(u_{j-1}, u_j)$ is real. The replies to all queries $\texttt{encrypt}(u_{j'-1}, u_{j'})$ 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:

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

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

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

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

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

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

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

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

<u>**Proof of Part 3**</u>. The proof of this part follows from the definition of $\Theta_{(i,i_\ell,\cdots,i_j)}^{(d,d_\ell,\cdots,d_j)}(\overrightarrow{\nu}_{j-1})$. When $\Theta_{(i,i_\ell,\cdots,i_j)}^{(d,d_\ell,\cdots,d_j)}(\overrightarrow{\nu}_{j-1})$ occurs, the in-degree of u_{j-1} in $\mathcal{G}(\mathsf{A})$ is either zero or in the range of 1 through n. If the in-degree is zero, this is the same as the occurrence of $\Theta_{(i,i_\ell,\cdots,i_j,0)}^{(d,d_\ell,\cdots,d_j,0)}(1\cdot\overrightarrow{\nu}_{j-1})$. 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 $\mathcal{G}(\mathsf{A})$ and the event $\Theta_{(i,i_\ell,\cdots,i_j,1)}^{(d,d_\ell,\cdots,d_j,d_{j-1})}(0\cdot\overrightarrow{\nu}_{j-1})$ must occur. This gives us the desired expression for $\Theta_{(i,i_\ell,\cdots,i_j)}^{(d,d_\ell,\cdots,d_j)}(\overrightarrow{\nu}_{j-1})$. $\blacksquare$

<u>**Proof of Part 4**</u>. The proof of part (a) is relatively straightforward and follows immediately from the definition of $\Psi_{(i,i_\ell,\cdots,i_j)}^{(d,d_\ell,\cdots,d_j)}$ and by inspection of the code of A'. No-

tice that when $\Psi^{(d,d_\ell,\cdots,d_1)}_{(i,i_\ell,\cdots,i_1)}$ occurs, the event Λ_0 must occur (that is, $u_0, u_1, \cdots, u_\ell$ must all be non-zero and must form a valid path in $\mathcal{G}(\mathsf{A})$), and since the depth of $\mathcal{G}(\mathsf{A})$ is at most ℓ, $\mathit{Indegree}(u_0)$ must equal 0. Also, for each $j' \in \{0, \cdots, \ell-2\}$, $b_{j'}$ 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 $\Psi^{(d,d_\ell,\cdots,d_1)}_{(i,i_\ell,\cdots,i_1)}$ is equivalent to the occurrence of $\Theta^{(d,d_\ell,\cdots,d_1)}_{(i,i_\ell,\cdots,i_1)}(\mathsf{XOR}(b, \overrightarrow{\nu}_0))$ for "some" choice of $\nu_0, \nu_1, \cdots, \nu_{\ell-2} \in \{0,1\}$. From this, the desired result follows.

The proof of part 4(b) is similar to the proof of part 2. 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 $\nu_j \oplus \cdots \oplus \nu_{\ell-1}$ 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 $\Phi^{(d,d_\ell,\cdots,d_{j+1},d_j)}_{(i,i_\ell,\cdots,i_{j+1})}(\overrightarrow{\nu}_j)$ is well-defined for the game $\mathbf{Game}_{j,\text{all}}$.

$$\text{Let} \quad E^{(0)}_{j,\mathbf{d},\mathbf{i},\nu} = \Psi^{(d,d_\ell,\cdots,d_j)}_{(i,i_\ell,\cdots,i_j)} \wedge (b = \nu) \ ; \qquad \text{and}$$

$$E^{(1)}_{j,\mathbf{d},\mathbf{i},\nu} = \bigvee_{d_{j-1}=1}^{n} \left(\bigvee_{\nu_{j-1},\cdots,\nu_{\ell-2}\in\{0,1\}} \Theta^{(d,d_\ell,\cdots,d_j,d_{j-1})}_{(i,i_\ell,\cdots,i_j,d_{j-1})}(1 \cdot \mathsf{XOR}(\nu, \overrightarrow{\nu}_{j-1})) \right)$$

Let $\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}} := \Phi^{(d,d_\ell,\cdots,d_j,d_{j-1})}_{(i,i_\ell,\cdots,i_j)}(\mathsf{XOR}(\nu, \overrightarrow{\nu}_{j-1}))$. It is easy to check that the view of A in $\mathbf{Game}_0$ given $\Psi^{(d,d_\ell,\cdots,d_j)}_{(i,i_\ell,\cdots,i_j)} \wedge (b = \nu)$ 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 $\overrightarrow{\nu}_{j-1} \in \{0,1\}^{\ell-j}$ (such that $\nu_{\ell-1} = 0$). (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 $\Theta^{(d,d_\ell,\cdots,d_j,d_{j-1})}_{(i,i_\ell,\cdots,i_j,d_{j-1})}(1 \cdot \mathsf{XOR}(\nu, \overrightarrow{\nu}_{j-1}))$ 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 $\overrightarrow{\nu}_{j-1} \in \{0,1\}^{\ell-j}$ ($\nu_{\ell-1} = 0$). 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 $\nu_{\ell-1} = 0$. For any $\nu_{j-1}, \cdots, \nu_{\ell-2}$, we use $\sum_{\nu_{j-1,\ldots,\ell-2}}$ as shorthand for the summation $\sum_{(\nu_{j-1},\cdots,\nu_{\ell-2}) \in \{0,1\}^{\ell-j}}$.

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:

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

The probability that event $\Phi^{(d_{j-1})}_{\overrightarrow{\nu}_{j-1}} = \Phi^{(d,d_\ell,\cdots,d_j,d_{j-1})}_{(i,i_\ell,\cdots,i_j)}(\mathsf{XOR}(\nu, \overrightarrow{\nu}_{j-1}))$ 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,

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

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 $\mathcal{G}(\mathsf{A})$ being zero or non-zero.

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

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

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

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

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

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

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

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

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

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

Thus,

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

9.5.8 Proof of the Telescoping Sums Lemma (Lemma 9.5.6)

We will prove the lemma using induction over j. Recall the expression for Δ_j (equation 9.5):

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

When $j=0$, this becomes:

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

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

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

From this, and the fact that $\overline{\Delta}_0 = \Delta_0$, it follows that Lemma 9.5.6 is true for $j=0$.

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

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

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

$$\sum_{\substack{d\in[n],\\ i\in[d]}} \sum_{\substack{d_\ell\in[n],\\ i_\ell\in[d_\ell]}} \cdots \sum_{\substack{d_j\in[n],\\ i_j\in[d_j]}}$$

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

$$\overline{\Delta}_{\hat{j}-1} = \sum_{d,i,d_\ell,\cdots,d_{\hat{j}},i_{\hat{j}}} \sum_{\nu^+_{\hat{j}-1}} \begin{bmatrix} \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(\mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}-1}))] \\ \\ -\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(\mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}-1}))] \end{bmatrix}$$

$$= \sum_{d,i,d_\ell,\cdots,d_{\hat{j}},i_{\hat{j}}} \sum_{\nu^+_{\hat{j}-1}} \begin{bmatrix} \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}((\nu_{\hat{j}-1}, \nu_{\hat{j}-1}\oplus\nu_{\hat{j}}, \cdots, \nu_{\ell-2}\oplus\nu_{\ell-1}))] \\ \\ -\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}((\overline{\nu_{\hat{j}-1}}, \nu_{\hat{j}-1}\oplus\nu_{\hat{j}}, \cdots, \nu_{\ell-2}\oplus\nu_{\ell-1}))] \end{bmatrix}$$

Let us now expand the innermost sequence of summations based on the value assigned to $\nu_{\hat{j}-1}$. We get that $\overline{\Delta}_{\hat{j}-1}$ equals

$$\sum_{d,i,d_\ell,\cdots,d_{\hat{j}},i_{\hat{j}}} \sum_{\nu^+_{\hat{j}}} \begin{bmatrix} \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}((0, \nu_{\hat{j}}, \nu_{\hat{j}}\oplus\nu_{\hat{j}+1}, \cdots, \nu_{\ell-2}\oplus\nu_{\ell-1}))] \\ +\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}((1, \overline{\nu_{\hat{j}}}, \nu_{\hat{j}}\oplus\nu_{\hat{j}+1}, \cdots, \nu_{\ell-2}\oplus\nu_{\ell-1}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}((1, \nu_{\hat{j}}, \nu_{\hat{j}}\oplus\nu_{\hat{j}+1}, \cdots, \nu_{\ell-2}\oplus\nu_{\ell-1}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}((0, \overline{\nu_{\hat{j}}}, \nu_{\hat{j}}\oplus\nu_{\hat{j}+1}, \cdots, \nu_{\ell-2}\oplus\nu_{\ell-1}))] \end{bmatrix}$$

$$= \sum_{d,i,d_\ell,\cdots,d_{\hat{j}},i_{\hat{j}}} \sum_{\nu^+_{\hat{j}}} \begin{bmatrix} \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(0\cdot\mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ +\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(1\cdot\mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(1\cdot\mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(0\cdot\mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{bmatrix}$$

$$= \sum_{d,i,d_\ell,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{d_{\hat{j}}\in[n]} \sum_{i_{\hat{j}}\in[d_{\hat{j}}]} \sum_{\nu^+_{\hat{j}}} \begin{bmatrix} \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(0\cdot\mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(1\cdot\mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ +\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(1\cdot\mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(0\cdot\mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{bmatrix}$$

and this can be written as:

$$\sum_{d,i,d_\ell,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{d_{\hat{j}}\in[n]} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \sum_{i_{\hat{j}}\in[d_{\hat{j}}]} \left(\begin{array}{l} \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(0\cdot\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(1\cdot\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}}))] \end{array} \right) \\ \\ +\ \sum_{i_{\hat{j}}\in[d_{\hat{j}}]} \left(\begin{array}{l} \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(1\cdot\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}})}(0\cdot\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}}))] \end{array} \right) \end{array} \right]$$

And now let us apply the hybrid cancellation lemma (part 2) to the terms in the innermost summations. We get that $\overline{\Delta}_{\hat{j}-1}$ equals

$$\sum_{d,i,d_\ell,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{d_{\hat{j}}\in[n]} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \left(\begin{array}{l} \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}+1},1)}(0\cdot\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1\cdot\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}}))] \end{array} \right) \\ \\ +\ \left(\begin{array}{l} \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1\cdot\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}+1},1)}(0\cdot\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}}))] \end{array} \right) \end{array} \right] \tag{9.6}$$

Now, let us recall the expression for $\Delta_{\hat{j}}$ (equation (9.5)), and let us re-write it in terms of the Θ's by invoking part 4 of the hybrid cancellation lemma

$$\begin{aligned} \Delta_{\hat{j}} = & \\ & \sum_{d,i,d_\ell,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{0\le d_{\hat{j}}\le n} \sum_{\substack{\nu_{\hat{j}},\cdots,\nu_{\ell-2}\in\{0,1\},\\ \nu_{\ell-1}=0}} \left[\begin{array}{l} \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1\cdot\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1\cdot\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}}))] \end{array} \right] \\ = & \sum_{d,i,d_\ell,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{d_{\hat{j}}\in[n]} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1\cdot\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},d_{\hat{j}})}_{(i,i_\ell,\cdots,i_{\hat{j}+1},d_{\hat{j}})}(1\cdot\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}}))] \end{array} \right] \\ & +\ \sum_{d,i,d_\ell,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},0)}_{(i,i_\ell,\cdots,i_{\hat{j}+1},0)}(1\cdot\mathsf{XOR}(0,\overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}};\Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},0)}_{(i,i_\ell,\cdots,i_{\hat{j}+1},0)}(1\cdot\mathsf{XOR}(1,\overrightarrow{\nu}_{\hat{j}}))] \end{array} \right] \end{aligned} \tag{9.7}$$

Let us now use the above equation and equation (9.6) to express $\overline{\Delta}_{\hat{j}}$ in terms of the Θ's:

$$\overline{\Delta}_{\hat{j}} = \overline{\Delta}_{\hat{j}-1} + \Delta_{\hat{j}}$$

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

$$+ \sum_{d,i,d_\ell,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},0)}_{(i,i_\ell,\cdots,i_{\hat{j}+1},0)}(1 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},0)}_{(i,i_\ell,\cdots,i_{\hat{j}+1},0)}(1 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right]$$

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:

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

$$+ \sum_{d,i,d_\ell,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{\nu^+_{\hat{j}}} \left[\begin{array}{l} \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},0)}_{(i,i_\ell,\cdots,i_{\hat{j}+1},0)}(1 \cdot \mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta^{(d,d_\ell,\cdots,d_{\hat{j}+1},0)}_{(i,i_\ell,\cdots,i_{\hat{j}+1},0)}(1 \cdot \mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right]$$

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

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

$$\overline{\Delta}_{\hat{j}} = \sum_{d,i,d_\ell,\cdots,d_{\hat{j}+1},i_{\hat{j}+1}} \sum_{\nu_{\hat{j}}^+} \left[\begin{array}{l} \mathbf{P}[\Theta_{\mathsf{A}}; \Theta_{(i,i_\ell,\cdots,i_{\hat{j}+1})}^{(d,d_\ell,\cdots,d_{\hat{j}+1})}(\mathsf{XOR}(0, \overrightarrow{\nu}_{\hat{j}}))] \\ -\ \mathbf{P}[\Theta_{\mathsf{A}}; \Theta_{(i,i_\ell,\cdots,i_{\hat{j}+1})}^{(d,d_\ell,\cdots,d_{\hat{j}+1})}(\mathsf{XOR}(1, \overrightarrow{\nu}_{\hat{j}}))] \end{array} \right]$$

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

The main result of this chapter, Theorem 9.3.3, was previously published in [37]. An analysis of the plain-LKH$^+$ protocol was also published in that work but the presentation is slightly different in the current work.

Bibliography

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

[2] M. Abadi and B. 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.

[3] D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In R. A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT'92*, volume 658 of *Lecture Notes in Computer Science*, pages 307–323, Balatonfred, Hungary, May 24–28, 1992. Springer-Verlag, Berlin, Germany.

[4] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403, Miami Beach, Florida, Oct. 19–22, 1997. IEEE Computer Society Press.

[5] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. In Y. Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 341–358, Santa Barbara, CA, USA, Aug. 21–25, 1994. Springer-Verlag, Berlin, Germany.

[6] M. Bellare and B. Yee. Forward security in private key cryptography. In M. Joye, editor, *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 1–18, San Francisco, CA, USA, Apr. 13–17, 2003. Springer-Verlag, Berlin, Germany.

[7] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[8] D. Boneh, C. Gentry, and B. 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.

[9] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multiparty computation. In *28th Annual ACM Symposium on Theory of Computing*, pages 639–648, Philadephia, Pennsylvania, USA, May 22–24, 1996. ACM Press.

[10] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Proceedings of INFO-COM 1999*, volume 2, pages 708–716, New York, NY, USA, March 1999. IEEE.

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

[12] R. Canetti and J. Herzog. Universally composable symbolic analysis of mutual authentication and key exchange protocols. In S. Halevi and T. Rabin, editors, *TCC '06: Third Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 380–403. Springer-Verlag, 2006.

[13] R. Canetti, T. Malkin, and K. Nissim. Efficient communication-storage tradeoffs for multicast encryption. In J. Stern, editor, *Advances in Cryptology - Eurocrypt 1999*, volume 1592 of *Lecture Notes in Computer Science*, Prague, Czech Reppublic, May 1999. Springer-Verlag.

[14] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha. Key management for secure internet multicast using boolean function minimization techniques. In *Proceedings of IEEE Infocomm '99*, volume 2, pages 689–698, New York, NY, USA, March 1999. IEEE Computer and Communication Societies.

[15] J. H. Cheon, N. su Jho, M.-H. Kim, and E. S. Yoo. Skipping, cascade, and combined chain schemes for broadcast encryption. Cryptology ePrint Archive, Report 2005/136. Preliminary version in Eurocrypt 2005, 2005.

[16] A. Datta, A. Derek, J. Mitchell, and B. 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.

[17] S. E. Deering. Multicast Routing in Internetworks and Extended LANs. In *Proceedings of ACM SIGCOMM '88*, pages 55–64. ACM Press, August 1988.

[18] D. Dolev and A. C. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983.

[19] L. R. Dondeti, S. Mukherjee, and A. Samal. Scalable secure one-to-many group communication using dual encryption. *Computer Communication*, 23(17):1681–1701, November 1999.

[20] C. Dwork, M. Naor, O. Reingold, and L. Stockmeyer. Magic functions. *Journal of the ACM*, 50(6):852–921, 2003.

[21] J. Fan, P. Judge, and M. H. Ammar. Hysor: Group key management with collusion-scalability tradeoffs using a hybrid structuring of receivers. In *Proceedings of the IEEE International Conference on Computer Communications Networks*, 2002.

[22] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. In *25th Annual Symposium on Foundations of Computer Science*, pages 464–479, Singer Island, Florida, Oct. 24–26, 1984. IEEE Computer Society Press.

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

[24] M. T. Goodrich, J. Z. Sun, and R. Tamassia. Efficient tree-based revocation in groups of low-state devices. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 511–527, Santa Barbara, CA, USA, Aug. 15–19, 2004. Springer-Verlag, Berlin, Germany.

[25] D. Halevy and A. Shamir. The lsd broadcast encryption scheme. In M. Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 47–60, Santa Barbara, CA, USA, August 2002. Springer-Verlag, Berlin, Germany.

[26] J. Y. Hwang, D. H. Lee, and J. Lim. Generic transformation for scalable broadcast encryption schemes. In V. Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture notes in Computer Science*, pages 276–292, Santa Barbara, CA, USA, August 2005. Springer Verlag, Berlin, Germany.

[27] E. Kiltz, A. Mityagin, S. Panjwani, and B. Raghavan. Append-only signatures. In A. de Santis, editor, *Automata, Languages and Programming, 32nd International Colloquium, ICALP Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 434–445, Lisboa, Portugal, July 2005. Springer-Verlag, Berlin, Germany.

[28] R. Merkle and M. Hellman. On the security of multiple encryption. *Communications of the ACM*, 24(7):465–467, July 1981.

[29] D. Micciancio and S. Panjwani. Optimal communication complexity of generic multicast key distribution. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 153–170, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.

[30] D. Micciancio and S. 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.

[31] D. Micciancio and S. 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, July 2006.

[32] D. Micciancio and S. Panjwani. Optimal communication complexity of generic multicast key distribution. *IEEE/ACM Transactions in Networking*, 2008. To appear.

[33] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In M. Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151, Cambridge, MA, USA, Feb. 19–21, 2004. Springer-Verlag, Berlin, Germany.

[34] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62, Santa Barbara, CA, USA, Aug. 19–23, 2001. Springer-Verlag, Berlin, Germany.

[35] M. Naor and B. Pinkas. Efficient Trace and Revoke Schemes. In *FC '00: Proceedings of the 4th International Conference on Financial Cryptography*, pages 1–20. Springer-Verlag, London, UK, 2000.

[36] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126, Santa Barbara, CA, USA, Aug. 18–22, 2002. Springer-Verlag, Berlin, Germany.

[37] S. Panjwani. Tackling adaptive corruptions in multicast encryption protocols. In S. Vadhan, editor, *Theory of Cryptography Conference, TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 21–40. Springer-Verlag, Berlin, Germany, February 2007.

[38] A. Perrig, D. Song, and D. 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.

[39] B. Pinkas. Efficient state updates for key management. In T. Sander, editor, *Security and Privacy in Digital Rights Management: ACM CCS-8 Workshop DRM 2001, Philadelphia, PA, USA*. Springer-Verlag, Berlin, Germany, November 2001.

[40] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 1991. Springer-Verlag.

[41] B. Raghavan, S. Panjwani, and A. Mityagin. Analysis of the spv secure routing protocol: weaknesses and lessons. *ACM SIGCOMM Computer Communication Review*, 37(2):29–38, 2007.

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

[43] J. Snoeyink, S. Suri, and G. Varghese. A lower bound for multicast key distribution. *Computer Networks*, 47(3):429–441, 2005).

[44] R. Tamassia and N. Triandopoulos. Computational bounds on hierarchical data processing with applications to information security. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 153–165. Springer-Verlag, Berlin, Germany, July 2005.

[45] D. M. Wallner, E. J. Harder, and R. C. Agee. Key management for multicast: Issues and architectures. Internet Draft, Sept. 1998.

[46] C. K. Wong, M. Gouda, and S. S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–30, Feb. 2000.

[47] R. Yang, X. Li, X. Zhang, and S. S. Lam. Reliable group rekeying: A performance analysis. In *Proceedings of ACM SIGCOMM '01*, pages 27–38. ACM Press, August 2001.

[48] Y. R. Yang and S. S. Lam. A secure group key management protocol communication lower bound. Technical Report TR-00-24, 2000.

[49] A. C. Yao. Theory and applications of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, Nov. 3–5, 1982. IEEE Computer Society Press.