Tackling Adaptive Corruptions in Multicast Encryption Protocols*

Saurabh Panjwani
University of California, San Diego

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

We apply our soundness theorem to the security analysis of multicast encryption protocols and show that a variant of the Logical Key Hierarchy (LKH) protocol is adaptively secure (its security being quasi-polynomially related to the security of the underlying encryption scheme).

Keywords. Adaptive Corruptions, Encryption, Multicast, Selective Decryption

1 Introduction

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

* This material is based upon work supported by the National Science Foundation under the Grant CNS-0430595. A full version of the paper is available from the author's webpage: http://www-cse.ucsd.edu/users/spanjwan

question is—can one be sure that the data for other instants (that is, instants when affected participants were all outside the group) is still secure?

Answering such a question in the affirmative, even for simple security protocols (based on conventional, symmetric-key encryption alone) is often beset with tough challenges. The possibility of user corruptions occurring during protocol execution, and in a manner that is *adaptively* controlled by the attacker, increases the threat to a protocol's security and makes the task of *proving* protocols secure an unnerving task. It is known that, in general, protocols proven secure against non-adaptive attacks may actually turn insecure once an adversary is allowed to corrupt participants adaptively. (See [5] for a simple separation result for protocols based on secret sharing.) The situation is especially annoying for protocols that make use of encryption—adversaries can spy on ciphertexts exchanged between two honest parties, and later, at will, corrupt one of the parties, acquire its internal state, and use such information to "open" all ciphertexts which were previously sent or received by that party. While trying to prove security of such a protocol, one must argue that all "unopened" ciphertexts (those that cannot be decrypted trivially using the compromised keys) leak essentially no information to the adversary (that is, appear as good as encryptions of random bitstrings). The heart of the problem lies in the fact that one does not a priori know *which* ciphertexts are going to be opened by the adversary since these decisions are made only as the protocol proceeds. Besides, every ciphertext is a binding commitment to the plaintext it hides—one cannot hope to "fool" the adversary by sending encryptions of random bitstrings every time and then, when he corrupts a party, somehow convince him that the ciphertexts he saw earlier on (and which he can now open) were, in fact, encryptions of real data.

Previous Approaches. In the past, security analysis of encryption-based multiparty protocols against adaptive adversaries has largely been conducted using three approaches. The first (and the simplest) involves bypassing adaptive security altogether—if you cannot prove a protocol adaptively secure, then so be it. (That is, rest your minds with non-adaptive security.) The second approach attempts to solve the problem, but by studying it in the "erasure" model [3], in which all honest parties are assumed to delete their past state the moment they enter a new state configuration (wherein keys are generated afresh). Proving adaptive security of protocols in such a model is easy because adversaries are trivially disallowed from opening previously-sent ciphertexts—the corresponding decryption keys are assumed to have been erased from the system! However, the model itself is quite unrealistic: an honest party could simply forget to erase its previous states, or else, internally deviate from the rules of the game (that is, store all keys and behave in an "honest-but-curious" manner). Besides, some cryptographic protocols, for the sake of efficiency, *require* users to store keys received in the past and such protocols (an example will be discussed in this paper) would need to be re-designed in order to comply with the model.

The third approach, and perhaps the most compelling one, to adaptive security has been to develop non-standard notions of security of an encryption

scheme. This corresponds to a line of research initiated by Canetti *et al.* [5], who introduced a cryptographic primitive, called *non-committing encryption*, specifically to address the problem of adaptive corruptions in multiparty 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 easily proven to achieve adaptive security—in the security proof, one just simulates the real protocol by transmitting ambiguous ciphertexts and upon corruption of a party, convinces the adversary that the ciphertexts he saw earlier were indeed the encryptions of the revealed data. Non-committing encryption schemes, though interesting in their own right, have their share of limitations—they are typically too inefficient for practical applications, and require bounding (a priori) the number of message bits that can be encrypted using any single key (usually, the number of bits that can be encrypted with a key cannot be more than the size of the key itself, which is highly prohibitive for real applications)[1].

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

Our main contribution is a general computational soundness theorem for encryption protocols which works as follows. Consider an abstract game played between an adversary and a challenger, both being given access to a semantically secure symmetric-key encryption algorithm $\mathcal{E}$. Initially, the challenger generates n independent keys $k_1, k_2, \cdots, k_n$ and keeps them secret from the adversary. During the game, the adversary gradually and adaptively builds a directed graph G

[1] As shown by Nielsen [16], any non-committing encryption scheme that has a non-interactive encryption procedure must use a decryption key that is at least as long as the total number of bits to be decrypted.

over n nodes labeled 1 through n. He arbitrarily introduces edges into the graph and for each such edge $i \rightarrow j$ he asks the challenger to provide an encryption of the key k_j under the key k_i, that is, $\mathcal{E}_{k_i}(k_j)$. (Thus, creation of the edge $i \rightarrow j$ in G depicts the fact that given k_i, the adversary can recover k_j, via the decryption operation corresponding to $\mathcal{E}$.) The adversary can also (again adaptively) decide to "corrupt" some nodes in the graph—from time to time, he instructs the challenger to reveal the key associated with the ith node in G (for any arbitrary i) and the challenger must answer with k_i in such a situation. We refer to G as the *key graph* generated by the adversary and the nodes in G that correspond to the revealed keys are called *corrupt nodes*. Note that any node i' in G that is reachable from a corrupt node i is also effectively corrupt; the adversary can recover the corresponding key using successive decryptions along the path from i to i'. The question is—can we prove that, at the end of the game, keys corresponding to nodes that are *not* reachable from any of the corrupt nodes, are still pseudorandom?

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

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

In this paper, we prove security in this game using a significantly different approach, and one that is of much better practical value. We show that if the key graph G generated by the adversary is acyclic[2] and if its *depth* (defined as

[2] Acyclicity of key graphs is an almost-inescapable criterion required in security proofs of protocols based on encryption. We will discuss this issue further in Section 2.

the length of the longest path in G) is upper bounded by a parameter l, then security in our game can be proven via a reduction factor of $O(n \cdot (2n)^l)$. Here, by "security in our game" we mean that keys that (a) cannot be trivially recovered by the adversary (that is, are not reachable from corrupt nodes in G) and (b) are not used to encrypt other keys[3], remain pseudorandom at the end of it. That is, we prove that the security of a semantically secure encryption scheme can degrade in the face of adaptive attacks (as those captured by our game) by a factor of at most $O(n \cdot (2n)^l)$ but not by worse.

An Application. So what is this reduction good for? At first glance, it would appear that it is much worse than the naive solution—l could potentially be of the order of n and $n \cdot (2n)^n$ is obviously no more consoling than 2^n. Well, for arbitrary key graphs, this is indeed the case. However, in practice, key graphs are much smaller (in fact, orders of magnitude smaller) in depth than in total size. For example, the key graphs generated in the execution of most broadcast encryption protocols (those falling under the subset-cover framework introduced by Naor *et al.* [15]) have depth 1 and their depth remains fixed for arbitrarily long runs of the protocol. All encryption-based group key distribution protocols (designed for secure multicast over the Internet, and also called *multicast encryption* protocols) generate key graphs that have depth at most logarithmic in the total number of users in the system (again, the depth remains fixed for arbitrarily long runs of the protocol, once the total space of users has been ascertained). In general, in all encryption protocols, the depth of key graphs created in any execution is likely to be related to the number of decryptions performed by users in order to be able to recover certain keys while their total size to the number of users themselves; it is reasonable to expect that protocol designers, for the purpose of efficiency, would strive to keep the former smaller than the latter.

We exemplify the power of our soundness result by applying it to the security analysis of the Logical Key Hierarchy (LKH) protocol [17]. LKH is a protocol originally developed for secure communication in multicast groups on the Internet (applications of the form we discussed in the first paragraph) and has since then attracted a lot of interest from both cryptographers and researchers in the networking community. Surprisingly, even though the protocol gets mentioned in a lot of papers on cryptography, there has been little effort from within our community towards analyzing its security (adaptive or otherwise) rigorously or to make any claims to the contrary.

The original LKH protocol has a security flaw in it [12]. Although this flaw is quite easy to spot, we are not aware of any work (prior to ours) that rectifies this flaw in a provably secure manner. (In [12], a fix is suggested but not proven secure.) In Section 3 of this paper, we present a variant of LKH which is not only as efficient as the original protocol, but also enjoys strong guarantees of security *against adaptive adversaries*. In particular, we use our soundness theorem to show that the security of the improved protocol is related to the

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

semantic security of the underlying encryption scheme via a reduction factor that is quasi-polynomial in the number of protocol participants. Concretely, our reduction factor is of the order of $\tilde{n}^{\log(n)+2}$, where n is the number of users in the protocol and $\tilde{n} = O(n)$.

This reduction factor, though not strictly polynomial in n, is still quite reasonable from a practical perspective. For example, in a system with 128 users, one is guaranteed that an execution of our protocol provides at least 65 bits of adaptive security when implemented with 128-bit AES in counter mode (for a run with upto 64 key updates)[4]. Our result practically eliminates the need for using expensive techniques like non-committing encryption to build adaptively secure multicast encryption protocols, and it does this while matching the efficiency of existing schemes.

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

Does this paper solve the selective decryption problem? Not really. A crucial ingredient of that problem is the distribution from which the plaintexts $m_1, \cdots, m_n$ are drawn by the challenger. It has been shown [8] that if this distribution is such that each plaintext can be generated independently of the others then the unopened ciphertexts indeed remain secure and the adversary learns essentially no partial information about the plaintexts they hide from his interaction with the challenger. In the GSD game, too, we require all keys, even those which are not used for further encryption, to be generated independently of each

[4] These numbers are computed assuming the protocol is implemented using a binary hierarchy of keys; for non-binary hierarchies, the security guarantee is actually better.

other, and this "independence property" is crucial in our proof[5]. Our soundness theorem essentially builds up on this positive result for selective decryption and extends it to the more general scenario of arbitrarily (and adaptively) generated key graphs. The question of solving selective decryption without the independence assumption on plaintexts still remains open.

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

Related Work. The notion of computational soundness theorems was introduced by Abadi and Rogaway [1], and has since then found applications in the security analysis of various cryptographic tasks, including key exchange [7, 6], mutual authentication [13, 6], XML security [2] and multicast key distribution [11, 12]. Although most of the literature on computational soundness theorems deals with protocols that make use of encryption as the fundamental primitive, to the best of our knowledge, none of these works prove soundness in the presence of adaptively corrupting adversaries. Recently, Gupta and Shmatikov [10] developed a symbolic logic that allows reasoning about a weak variant of adaptive security for the case of key exchange protocols; however, the protocols they analyze, do not make use of encryption (and instead use Diffie-Hellman exponentiation coupled with signatures).

The soundness result of this paper is of a very different flavor than those in previous works in the area. The protocol model we use is relatively simpler—in the protocols we consider, every message generated during an execution is either a key or an encryption of a key under a key or else, a sequence of values with one of these types[6]. Symbolic analysis of such protocols can be effectively conducted using graph-theoretic terminology: keys can be interpreted as nodes, ciphertexts as edges, and Dolev-Yao attacks on protocols can be expressed in terms of reachability from adversarial nodes (corresponding to corrupted keys). As such, all discussions on symbolic analysis in this paper take place within a graph-theoretic framework (as illustrated by the GSD game). This simplifies our presentation considerably and brings us quickly to the crux of the matter.

Lastly, a few words comparing the result of this paper with our previous work, joint with Micciancio [11, 12], on the computationally sound analysis of encryption protocols are in order. Although both our works address adaptive attacks on encryption protocols, the adversarial model used in the current work is stronger: we not only allow the adversary to adaptively modify the execution flow

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

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

of the protocol (as in our past work) but also to corrupt participants in an adaptive manner. Tackling the latter type of attacks is significantly more non-trivial, and forms the central theme of this paper. Another difference is that our previous soundness results applied only to protocols that satisfied certain syntactic conditions *besides* acyclicity of key graphs. Informally, these conditions require protocols to use every key in two phases—a *distribution* phase in which keys are used as plaintexts, followed by a *deployment* phase in which the distributed keys are used for encrypting other keys or messages. Key distribution is not allowed to succeed key deployment. Our new result, while incorporating adaptive corruptions, also does away with this restriction. The downside, however, is that this result provides security guarantees in a manner that is dependent on the depth of protocol key graphs, and it is not meaningful for protocols that could potentially generate key graphs with arbitrary depth. We believe that improving the result of this paper to overcome this limitation is non-trivial, but a worthy direction for future research; in particular, obtaining an analogous result with a reduction factor smaller than $\Theta(n^l)$ would be quite remarkable, and could lead to even newer techniques to address adaptive corruptions in security protocols.

2 The Main Result

Fix a symmetric-key encryption scheme $\Pi = (\mathcal{E}, \mathcal{D})$[7]. We use the standard notion of indistinguishability against chosen plaintext attacks (Ind-CPA) for encryption schemes as defined by Bellare *et al.* [4]. Specifically, let $\mathcal{O}^{\Pi}_{k,b}$ denote a left-or-right oracle for Π which first generates a key k uniformly at random from $\{0,1\}^\eta$ (η being the security parameter) and subsequently, responds to every query of the form $(m_0, m_1) \in \{0,1\}^* \times \{0,1\}^*$ (such that $|m_0| = |m_1|$) with $\mathcal{E}_k(m_b)$—the encryption of m_b under key k. For any adversary (that is, any arbitrary probabilistic Turing machine) A, let $\mathsf{A}^{\mathcal{O}^{\Pi}_{k,b}}$ denote the random variable corresponding to the output of A when interacting with such an oracle.

Definition 1 Let $t \in \mathbb{N}^+$ and $0 < \epsilon < 1$. An encryption scheme Π is called (t, ϵ)-Ind-CPA secure if for every adversary A running in time t: $|\mathbf{P}[\mathsf{A}^{\mathcal{O}^{\Pi}_{k,b}} = 1 | b = 0] - \mathbf{P}[\mathsf{A}^{\mathcal{O}^{\Pi}_{k,b}} = 1 \mid b = 1]| \leq \epsilon$

The GSD game. Consider the following game, which we call the *generalized selective decryption (GSD)* game, played between an adversary A and a challenger B. Both parties are given blackbox access to the algorithms $\mathcal{E}$ and $\mathcal{D}$. In the beginning, A specifies an integer n, and the challenger generates a set of keys, $k_1, k_2, \cdots, k_n$, each key being sampled independently and uniformly at random

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

from the set $\{0,1\}^\eta$ (where η is the security parameter). B also generates a *challenge* bit b (uniformly at random from $\{0,1\}$), which A is required to guess in the end. It stores the generated values for the rest of the game, and uses them to answer all of A's queries.

A can make three types of queries to B:

1. `encrypt`: At any point, A can make a query of the form $\texttt{encrypt}(i,j)$, in response to which B creates a ciphertext $c \leftarrow \mathcal{E}_{k_i}(k_j)$ (using fresh coins for the encryption operation each time) and returns c to A.
2. `corrupt`: A can also ask for the value of any key initially generated by B; it does this by issuing a query of the form $\texttt{corrupt}(i)$, in response to which it receives k_i.
3. `challenge`: Finally, A can issue a query of the form $\texttt{challenge}(i)$. The response for such a query is decided based on the bit b: if $b = 0$, B returns the key k_i to A, whereas if $b = 1$, it generates a value r_i uniformly at random from $\{0,1\}^\eta$, and sends r_i to A[8].

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

We think of the queries of A as creating a directed graph over n nodes (labeled $1, 2, \cdots, n$), edge by edge, and in an adaptive fashion. Each query $\texttt{encrypt}(i,j)$ corresponds to creating an edge from i to j, denoted $i \to j$, in this graph. For any adversary A, the graph created by its queries in this manner is called the *key graph* generated by A and is denoted $G(\mathsf{A})$. A node i in $G(\mathsf{A})$ for which A issues a query $\texttt{corrupt}(i)$ 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 $V^{\mathsf{corr}}(\mathsf{A})$ and that of all challenge nodes is denoted $V^{\mathsf{chal}}(\mathsf{A})$. Note that $G(\mathsf{A}), V^{\mathsf{corr}}(\mathsf{A})$ and $V^{\mathsf{chal}}(\mathsf{A})$ are all random variables depending on the coins used by both A and B.

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

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

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

Definition 2 An adversary A is called *legitimate* if in any execution of A in the GSD game, the values of $G(\mathsf{A}), V^{\mathsf{corr}}(\mathsf{A})$ and $V^{\mathsf{chal}}(\mathsf{A})$ are such that:

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

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

Definition 3 Let $t, n, e, l \in \mathbb{N}^+$ and $0 < \epsilon < 1$. An encryption scheme Π is called (t, ϵ, n, e, l)-GSD secure if for every legitimate (n, e, l)-adversary A running in time t: $|\mathbf{P}[\mathsf{A}^{\mathsf{B}^{\Pi}_b} = 1 \mid b = 0] - \mathbf{P}[\mathsf{A}^{\mathsf{B}^{\Pi}_b} = 1 \mid b = 1]| \leq \epsilon$

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

Theorem 4 Let $t, n, e, l \in \mathbb{N}^+$ and $0 < \epsilon < 1$. If an encryption scheme Π is (t, ϵ)-Ind-CPA secure, then it is (t', ϵ', n, e, l)-GSD secure for quantities t' and ϵ' defined as:

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

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

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

Overview of the Proof. The starting point of the proof of our theorem is the positive result on the selective decryption problem (more precisely, the selective *decommitment* problem) due to Dwork *et al.* [8]. Consider first the GSD game for the case $l = 1$. The graph $G(\mathsf{A})$ in this case is a directed bipartite graph mapping a set of sources to a set of sinks. (In the problem studied in [8], the map from sources to sinks is one-to-one. In our case, it could be many-to-many; plus, it could be adaptively generated based on previous corruptions.) How can we argue about security in this case? Intuitively, an attacker's ability to differentiate between real and random values for *all* nodes in $V^{\mathsf{chal}}(\mathsf{A})$ translates into its ability to differentiate between the two values for *some* node (say the jth one) in $V^{\mathsf{chal}}(\mathsf{A})$; that is, such an adversary can effectively differentiate between two worlds, one in which the reply to each of the first $j-1$ queries of the form $\texttt{challenge}(i)$ is r_i (and for the rest, it is k_i), and the other in which the reply to each of the first j queries of this form is r_i (and that for the rest is k_i).

Let us call these worlds $\mathsf{World}_j(0)$ and $\mathsf{World}_j(1)$ respectively. Let us assume that the argument specified in A's jth `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 $G(\mathsf{A})$. Now consider this modified version of the game: While generating keys in the beginning of the game, B also generates a random key $\tilde{k}_{i_j}$, independently of all other keys. It replies to the adversary's queries in one of two worlds again, but now the worlds are defined as follows. Each query of the form $\texttt{encrypt}(i_s, i_j)$ is replied to with the real ciphertext $\mathcal{E}_{k_{i_s}}(k_{i_j})$ in the first world, $\mathsf{World}'_j(0)$, but with a *fake* one, namely $\mathcal{E}_{k_{i_s}}(\tilde{k}_{i_j})$, in the other one, $\mathsf{World}'_j(1)$. All other `encrypt` queries are replied to with real ciphertexts in both worlds. For the `challenge` queries the replies always have the same distribution—r_i for the first $j-1$ `challenge` queries and k_i for the rest. (In particular, the reply for $\texttt{challenge}(i_j)$ is always k_{i_j}.) It is easy to see that the distribution on the challenger's replies in $\mathsf{World}'_j(0)$ is exactly the same as in $\mathsf{World}_j(0)$. The key observation to make here is that the distribution on the replies in $\mathsf{World}'_j(1)$ is also the same as that in $\mathsf{World}_j(1)$! This is true because the keys $k_{i_j}, \tilde{k}_{i_j}$ and r_{i_j} are generated by the challenger independently of each other, and so, replying to $\texttt{encrypt}(i_s, i_j)$ with $\mathcal{E}_{k_{i_s}}(k_{i_j})$ and $\texttt{challenge}(i_j)$ with r_{i_j} (as done in $\mathsf{World}_j(1)$) produces the same distribution as replying to the former with $\mathcal{E}_{k_{i_s}}(\tilde{k}_{i_j})$ and the latter with k_{i_j} (as done in $\mathsf{World}'_j(1)$). Thus, our adversary can differentiate between $\mathsf{World}_j(0)$ and $\mathsf{World}_j(1)$ with the same probability as it can differentiate between $\mathsf{World}'_j(0)$ and $\mathsf{World}'_j(1)$.

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

Going beyond $l = 1$. In the general setting, a node i_s, pointing at any node $i_j \in V^{\mathsf{chal}}(\mathsf{A})$ need not be a source—there could be other edges incident upon

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

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

(a) The distributions differ in the reply corresponding to a *single* edge $i_s \rightarrow i_t$; 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 $i_r \rightarrow i_s$ is faked.
(c) There exists a path from i_t to i_j in the key graph and in both distributions, "some" of the edges incident upon this path are faked, the faked edges being the same in both distributions.
(d) No other edge in the key graph is faked in either of the distributions.

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

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 (using a left-or-right oracle) and subsequently, *proving* that the simulation works correctly is the most challenging part of the proof. After developing an appropriate simulation strategy, we prove its correctness using an inductive argument—assuming that, for some $l' \leq l$, the simulation behaves correctly whenever i_s is at depth smaller than l' in the key graph, we show that the simulation is correct also when i_s is at depth smaller than $l' + 1$; this simplifies our analysis considerably. Details of the entire proof are given in the full version of the paper.

Other Variants. A natural variant of the GSD game would be one in which the adversary is allowed to acquire encryptions of messages of its choice (besides receiving encryptions of keys, as in the original game). Consider the following modified version of the game: A issues `encrypt` and `corrupt` queries, as before, but instead of making `challenge` queries, it makes queries of the form $\texttt{encrypt\_msg}(i, m_0, m_1)$ (such that $m_0, m_1 \in \{0,1\}^*$ and $|m_0| = |m_1|$). In return for each such query, the challenger sends it the ciphertext $\mathcal{E}_{k_i}(m_b)$. A legitimate adversary in this modified game would be one whose key graph is always a DAG and for whom every query $\texttt{encrypt\_msg}(i, m_0, m_1)$ is such that i is unreachable from the corrupt nodes in the DAG. We remark that a result analogous to Theorem 4 can also be proven for this modified game, and with only a slight modification to the proof of that theorem. Specifically, we can show that if Π is (t, ϵ)-Ind-CPA secure, then for any t'-time (n, e, l) adversary A (t' as defined in Theorem 4),

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

A different variant of our game would be one in which A is provided encryptions of messages, but these messages are sampled by the *challenger* using some fixed distribution known to A. In this variant, the messages themselves can be thought of as nodes (more specifically, sinks) in the key graph, whose values are hidden from A but whose probability distribution is defined differently from that of the keys. The goal now would be to argue that from A's perspective, all "unopened" messages (that is, messages that are not reachable from corrupt nodes in the key graph) appear as good as fresh samples from the same message space. If we assume that messages are sampled independently of each

other, then security in this variant can also be proven, and with almost the same reduction factor as in Theorem 4. (Specifically, the reduction factor would be $(3/2) \cdot M(n+1)(2n+1)^l$, where M is an upper bound on the total number of messages that are encrypted.) However, in the absence of this assumption, it becomes considerably more challenging to prove the same claim. The techniques developed in this paper do not allow us to argue about security in such a setting, not even in the case where the key graph has depth 0 (only messages, and not keys, are used as plaintexts)[9].

3 The Application

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

MULTICAST ENCRYPTION. A group of n users, labeled $U_1, \cdots, U_n$, share a broadcast channel and wish to use it for secure communication with each other. At any point in time t, only a subset of users, labeled S_{t}, are "logged in" to the network, that is, are authorized to receive information sent on the channel. We would like to ensure that for all t, only the users in S_{t} (called *group members*) be able to decipher the broadcasts. We assume the existence of a central group manager C who shares a unique long-lived key k_{U_i} with each user U_i[10] and runs a key distribution program, KD, in order to accomplish the said task. The manager (or, equivalently, the program KD) receives user `login` and `logout` requests and for the request at time t, sends out a set of *rekey messages*, $\mathcal{M}_{\mathsf{t}}$, on the channel. These rekey messages carry information about a key $k[\mathsf{t}]$ (the *group key* for t), and are such that only the group members can decipher them (and, subsequently, recover $k[\mathsf{t}]$). The key $k[\mathsf{t}]$ can then be used to carry out all group-specific security tasks until the next `login`/`logout` request arrives, which, we assume, happens at time $\mathsf{t}+1$. For example, it can be used for ensuring privacy of all data sent between time t and $\mathsf{t}+1$ and/or guaranteeing "group authenticity" of data (that is, enabling members to verify that the sender of the data is a group member at time t, and not an outsider). To ensure security of any such task, it is important to guarantee that $k[\mathsf{t}]$ appears pseudorandom to users *not in* S_{t} (the non-members) for all instants t, even when such users can collude with each other and share all their information. The problem is to design the program KD in a manner such that this guarantee is achieved.

[9] Here, by "argue about security" we mean the following: Consider an adversary A who makes only `encrypt` and `corrupt` queries in the above variant of the GSD game. At the end of the game, provide A with one out of two sets of values: in one world, reveal the real values of all unopened messages; in the other, provide an equal number of messages, sampled from the same probability space *conditioned* on the values of the opened messages. Now show that A cannot tell the two worlds apart. This problem is essentially the same as the selective decryption problem where plaintexts are allowed to be mutually dependent. We don't know of a solution to this problem yet.

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

Fiat and Naor [9] were the first to define this problem formally and they introduced it under the title of *broadcast* encryption—a formulation in which all users are assumed to be stateless and group members are required to be able to recover $k[\mathsf{t}]$, given only $\mathcal{M}_\mathsf{t}$ and their long-lived keys. Subsequent work (for example, [14, 17]) lifted the problem to the more general setting of stateful users, and studied it in the context of ensuring privacy in multicast groups on the Internet (hence the name *multicast* encryption). LKH is a protocol that relies on the statefulness assumption.

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

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

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

Suppose that initially ($\mathsf{t} = 0$), the set of group members $S_0 = \{U_1, U_2, U_3, U_6, U_7\}$ as shown in Figure 1(a). The center's key distribution program KD generates the initial group key $k[0] = k_\epsilon$ (the root node) and all auxiliary keys (internal nodes) which are supposed to be given to users in S_0. For example, since k_{00} and k_0 lie on the path from k_{U_1} to $k[0]$, these keys must be generated afresh and sent securely to U_1. KD transmits the keys to the designated users by sending the ciphertexts shown by dark edges in the figure. So, for example, user U_1 can obtain all the keys it is supposed to know $(k_{00}, k_0, k_\epsilon)$ by decrypting, in order, the ciphertexts $\mathcal{E}_{k_{U_1}}(k_{00}), \mathcal{E}_{k_{00}}(k_0)$ and $\mathcal{E}_{k_0}(k_\epsilon)$.

Now suppose that at time $\mathsf{t} = 1$, user U_1 logs out of the group. That is, $S_1 = \{U_2, U_3, U_6, U_7\}$. The program KD should re-generate the group key k_ϵ, and the auxiliary keys which were known to U_1 at $\mathsf{t} = 0$ (k_{00} and k_0) and distribute the new values in a manner such that U_1 cannot recover them but other

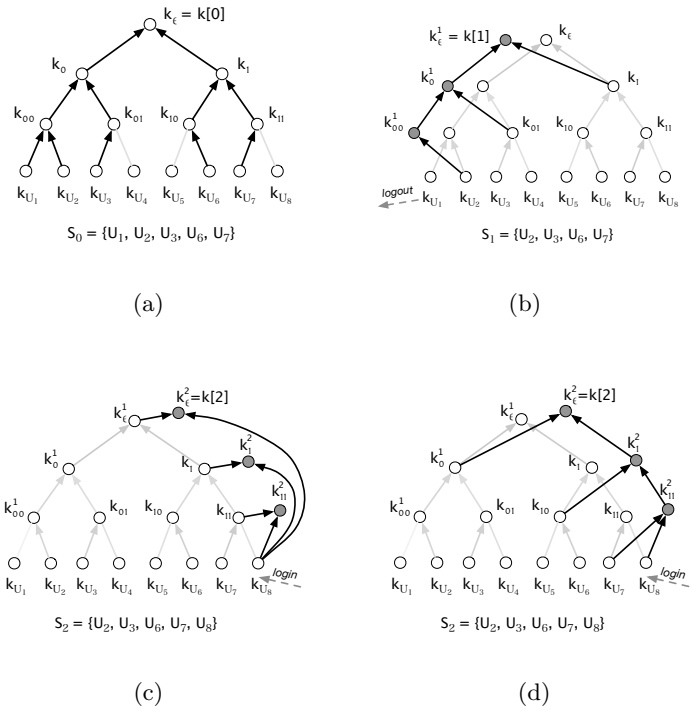

Fig. 1. LKH and rLKH: Figure 1(a) shows how key distribution to the initial set of users S_0 is performed while figure 1(b) demonstrates the rekeying process for user `logout` (both these procedures are the same in LKH and rLKH). Figure 1(c) shows how rekeying for user `login` works in LKH and fig. 1(d)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

users who are required to do so (according to the protocol invariant) still can. Specifically, it generates new keys k^1_{00}, k^1_0 and $k^1_\epsilon =: k[1]$ (independently and uniformly at random) and sends out the ciphertexts shown in figure 1(b). Thus, every rekey operation for a user `logout` requires sending logarithmically many (specifically, $2\log_2(n) - 1$) ciphertexts; in our example, this number is 5.

The Flaw and the Fix. The flaw in the original LKH protocol lies in the way it implements rekeying for user `login` operations. Suppose U_8 sends a `login` request at time $\mathtt{t} = 2$. The center must now re-generate keys k_{11}, k_1, k_ϵ and send them securely to all the designated users (including U_8). The protocol does this by transmitting the ciphertexts shown in figure 1(c). ($k^2_{11}, k^2_1, k^2_\epsilon$ denote the newly generated keys.)

Note that the group key at $\mathtt{t} = 1$, $k[1] = k^1_\epsilon$, is used to encrypt the group key at $\mathtt{t} = 2$, k^2_ϵ. This is a problem since our initial goal was to guarantee pseudo-randomness of all group keys but deploying $k[1]$ in this manner clearly fails that purpose. In principle, if $k[1]$ is used in keying other applications (for example, in a message authentication scheme) at $\mathtt{t} = 1$, and is also used for rekeying in the

manner shown, then the protocol could be completely subverted (both $k[1]$ and $k[2]$ fully recovered) *even by a passive eavesdropper on the channel*. Of course, this does not mean that the protocol is broken for *any* secure implementation of the encryption scheme; but for *some*, it is.

We propose to fix the LKH protocol by changing the rekeying procedure for user `login`s as shown in Figure 1(d). (We remark that this fix is different from the one suggested in [12].) Notice that the communication cost incurred is the same as in the original protocol ($2\log_2(n)$ ciphertexts for a user space of size n). Notice also that the structure of the rekey messages is now similar to that of the messages sent upon a user `logout` request (figure 1(b)). We refer to this modified version of LKH as "rLKH" (the r stands for "repaired"). The protocol can be easily generalized to work with arbitrary hierarchies; in particular, when the key hierarchy is a d-ary tree (so its height equals $\lceil \log_d(n) \rceil$), the communication complexity (number of ciphertexts transmitted) of rekeying would be $d\lceil \log_d(n) \rceil$ for user `login`s and $d\lceil \log_d(n) \rceil - 1$ for user `logout`s. An implementation of rLKH with n users and a d-ary hierarchy is referred to as the (n, d)-*instance* of the protocol.

One could conceive other ways of fixing the user `login` process of LKH (possibly as secure and as efficient as the one we propose). We prefer this fix for two reasons: (a) the key hierarchy in rLKH has the nice property that at all instants, every auxiliary key (and even the group key) is transmitted to the legitimate recipients by encrypting it under its two children only (and no other keys). This property could potentially simplify implementation of the protocol in practice; (b) more importantly, our fix ensures that the depth of the key graph generated in any execution of the protocol is independent of the number of protocol rounds; this property is useful in arguing about the protocol's adaptive security.

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

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

All queries can be issued interleavingly and adaptively. Let $U^{\mathsf{corr}}(\mathsf{A})$ be the set of all users corrupted by A during the game. Let $T^{\mathsf{chal}}(\mathsf{A})$ be the set of instants $\mathtt{t}$ at which A issues a `challenge` query. We say that A is *legitimate* if in every execution of A in the MKD game, for all $\mathtt{t} \in T^{\mathsf{chal}}(\mathsf{A})$, $S_{\mathtt{t}} \cap U^{\mathsf{corr}}(\mathsf{A}) = \emptyset$. Let $\mathsf{A}^{\mathsf{B}^{\mathsf{KD}}_b}$ denote the random variable corresponding to the output of A in the game, conditioned on the event that B selects b as the challenge bit.

Definition 5 Let $t, r \in \mathbb{N}^+$ and $0 < \epsilon < 1$. A multicast key distribution program KD is (t, r, ϵ)-secure against adaptive adversaries if for every legitimate adversary A that runs in time t, and makes r `rekey` queries: $|\mathbf{P}[\mathsf{A}^{\mathsf{B}^{\mathsf{KD}}_b} = 1 \mid b = 0] - \mathbf{P}[\mathsf{A}^{\mathsf{B}^{\mathsf{KD}}_b} = 1 \mid b = 1]| \leq \epsilon$

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

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

Theorem 6 Let $n, d, t, r' \in \mathbb{N}^+$ such that $1 < d \leq n$. Let $0 < \epsilon < 1$. The (n, d)-instance of rLKH, when implemented using a (t, ϵ)-Ind-CPA secure encryption scheme Π, is (t', r', ϵ')-secure against adaptive adversaries for

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

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

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

The proof of this theorem follows almost immediately from our soundness result of Section 2, given that (a) the key graph generated by any execution of rLKH is acyclic; (b) all group keys correspond to sinks in the protocol key graph; (c)

the depth of the graph remains $\lceil \log_d(n) \rceil$ throughout; and (d) for any r'-round execution of the protocol, and for all $\mathsf{t} \leq r'$, the group key $k[\mathsf{t}]$ can be reached from a long-lived key k_{U_i} if and only if $U_i \in S_{\mathsf{t}}$. (The last part can be proven using a straightforward inductive argument, with the induction being performed on r'.) The reduction factor given in the theorem is slightly better than what one gets using a direct invocation of Theorem 4: this is achieved using the fact that in any r'-round execution of the rLKH protocol, (a) a key at depth i in the key graph (that is, at distance i from some source) is encrypted only by keys at depth $i-1$ and (b) there are at most $d^{\lceil \log_d(n) \rceil - 1} + r'$ keys at any depth in the graph (and at most n sources in it). Note that our reduction factor is exponential in $\lceil \log_d(n) \rceil$ which is independent of the number of rounds the protocol is executed for. That is, the adaptive security of rLKH degrades polynomially (and not exponentially) with the number of rounds in the protocol execution.

Changing the hierarchy structure in rLKH involves a natural trade-off between efficiency and security: If we increase the arity d of the hierarchy (and correspondingly, reduce the height), the communication efficiency of the protocol suffers, but we get a better guarantee on its adaptive security. The extreme case is the n-ary hierarchy that has a linear rekeying communication complexity but provides adaptive security via a reduction factor of only $O(\tilde{n}^2)$. (Note that this is exactly the trivial approach to key distribution we discussed earlier on.) Whether or not one can further improve this trade-off between efficiency and security across different instances of rLKH, and, in particular, prove its adaptive security via a reduction factor smaller than the one given in Theorem 6, assuming only the semantic security of Π, is a question left open by this work.

Acknowledgements

Thanks to Daniele Micciancio, Thomas Ristenpart and Scott Yilek, for commenting on an earlier draft of the paper. Thanks also to the anonymous referees.

References

1. Martin Abadi and Philip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.
2. Martin Abadi and Bogdan Warinschi. Security analysis of cryptographically controlled access to xml documents. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems (PODS)*, pages 108–117, Baltimore, Maryland, June 2005. ACM.
3. Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In Rainer A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT'92*, volume 658 of *Lecture Notes in Computer Science*, pages 307–323. Springer-Verlag, May 1992.
4. Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Society Press, October 1997.

5. Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multiparty computation. In *28th Annual ACM Symposium on Theory of Computing*, pages 639–648. ACM Press, May 1996.
6. Ran Canetti and Jonathan Herzog. Universally composable symbolic analysis of mutual authentication and key exchange protocols. In Shai Halevi and Tal Rabin, editors, *TCC '06: Third Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 380–403. Springer-Verlag, 2006.
7. Anupam Datta, Ante Derek, John Mitchell, and Bogdan Warinschi. Computationally sound compositional logic for key exchange protocols. In *19th IEEE Computer Security Foundations Workshop (CSFW '06)*, pages 321–334. IEEE Computer Society, 2006.
8. Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions. *Journal of the ACM*, 50(6):852–921, 2003.
9. Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer-Verlag, August 1993.
10. Prateek Gupta and Vitaly Shmatikov. Key confirmation and adaptive corruptions in the protocol security logic. In *FCS-ARSPA 2006 (Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis)*, 2006.
11. Daniele Micciancio and Saurabh Panjwani. Adaptive security of symbolic encryption. In J. Kilian, editor, *Theory of Cryptography Conference, TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 169–187, Cambridge, MA, USA, February 2005. Springer-Verlag, Berlin, Germany.
12. Daniele Micciancio and Saurabh Panjwani. Corrupting one vs. corrupting many: The case of broadcast and multicast encryption. In *Automata, Languages, and Programming: 33rd International Colloquium, ICALP 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*. Springer-Verlag, January 2006.
13. Daniele Micciancio and Bogdan Warinschi. Soundness of formal encryption in the presence of active adversaries. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer-Verlag, February 2004.
14. Suvo Mittra. Iolus: A framework for scalable secure multicasting. In *Proceedings of ACM SIGCOMM*, pages 277–288, Cannes, France, September 14–18, 1997.
15. Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer-Verlag, August 2001.
16. Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126. Springer-Verlag, August 2002.
17. Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–30, February 2000.