Optimal Communication Complexity of Generic Multicast Key Distribution*

Daniele Micciancio and Saurabh Panjwani

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404
E-mail: {daniele,panjwani}@cs.ucsd.edu
WWW: http://www-cse.ucsd.edu/users/{daniele,spanjwan}

Abstract

We prove a tight lower bound on the communication complexity of secure multicast key distribution protocols in which rekey messages are built using symmetric-key encryption, pseudorandom generators and secret sharing schemes. Our lower bound shows that the amortized cost of updating the group key for each group membership change (as a function of the current group size) is at least $\log_2(n) - o(1)$ basic rekey messages. This lower bound matches, up to a subconstant additive term, the upper bound due to Canetti, Garay, Itkis, Micciancio, Naor and Pinkas (Proc. of Infocomm 1999), who showed that $\log_2(n)$ basic rekey messages (each time a user joins and/or leaves the group) are sufficient. Our lower bound is, thus, optimal up to a small, subconstant additive term.

The result of this paper considerably strengthens previous lower bounds by Canetti, Malkin and Nissim (Proc. of Eurocrypt 1999), and Snoeyink, Suri and Varghese (Computer Networks 47(3):2005), which allowed for neither the use of pseudorandom generators and secret sharing schemes, nor the iterated (nested) application of the encryption function. Our model (which allows for arbitrarily nested combinations of encryption, pseudorandom generators and secret sharing schemes) is much more general, and, in particular, encompasses essentially all known multicast key distribution protocols of practical interest.

Keywords: Multicast, security, key distribution, lower bounds, nested encryption, secret sharing.

1 Introduction

Ensuring privacy of communication over broadcast channels is a security task of considerable practical importance. At a high level, the problem involves a source of information (say, a PayTV broadcast station), which wants to transmit data privately to a privileged subset of users of a broadcast channel (say, the subscribers to the PayTV service), while ensuring that users outside this privileged group cannot decipher what is being sent. Furthermore, membership of the privileged group changes dynamically (in an arbitrary, a-priori unknown manner) and the source desires to keep its data hidden from unprivileged recipients at each point in time. The challenge lies in devising secure solutions for this problem in such a way that the communication and storage overhead is minimized.

Two distinct models have been used in the literature to study this problem: the first, commonly referred to as *broadcast encryption* [13], assumes that all receivers are stateless and so, current data must be decipherable

*To appear in IEEE/ACM Transactions on Networking (2008). A preliminary version of this paper appeared in the Proceedings of Eurocrypt 2004. This material is based upon work supported by the National Science Foundation under Grants CCR-0313241 and CNS-0430595, and a Sloan Research Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

independently of past transmissions. This model has the advantage that it does not require users to be "attentive" to all broadcasts in order to be able to decipher the material sent at a fixed instant. However, the advantage comes at a price: solutions to broadcast encryption are often quite intensive, either in terms of user storage and computation requirements or in terms of communication complexity. The second model, studied under the title of *multicast key distribution* or *multicast encryption* [36, 35], is more general in that it allows users to maintain state and use previously learned keys for decrypting current transmissions. This model is better suited to multicast-based applications on the Internet (like online gaming and Internet-based pay-per-view services) where users are stateful, and, more importantly, conserving bandwidth is an important design goal.

In this paper, our focus is on multicast key distribution protocols, and, specifically, those that are designed using efficient symmetric-key cryptographic techniques. Quite commonly, key distribution in multicast is implemented using a central distribution authority, called the *group center*, responsible for establishing a shared key among all privileged group members, and for "rekeying" the group (that is, changing the shared key) every time a new member joins and/or an existing member leaves the group. The shared key can subsequently be used for various purposes, and, in particular, it can be used to guarantee privacy of group data (by encrypting all such data under the said key using a symmetric-key encryption scheme). We are interested in studying the communication complexity (the cost of transmitting rekey messages) of protocols that fall within this class.

One simple way in which rekey operations can be handled is to have the center share individual long-lived secret keys with every user of the channel and, after any membership change, transmit a fresh group key encrypted under the individual keys of only the new set of members (as advocated in [20, 19]). This, however, is not a scalable solution since the amount of communication required for the rekey operation is linear in the size of the current group. The question is: how much better than the linear-cost solution can we do? The Logical Key Hierarchy (LKH) approach (due to Wong *et al.* [36] and Wallner *et al.* [35]) provides a framework for building protocols in which the average communication cost is reduced to logarithmic in the size of the group. The original LKH protocol [36, 35], which makes use of symmetric-key encryption as the only cryptographic primitive, requires the transmission of $2\log_2(n)$ messages[1] each time a user leaves and another one joins the group, where n is the size of the group[2]. Subsequent work [6, 22] provides improvements over the original protocol, reducing the number of transmissions to exactly $\log_2(n)$ messages per update. Although these solutions are exponentially more efficient than the linear-cost solution, it would be desirable to have protocols with even smaller communication complexity. We're interested in investigating whether this is possible or not using standard symmetric-key cryptography techniques.

1.1 Previous Lower Bounds

The inability to devise more communication-efficient protocols has prompted many researchers to attempt to prove lower bounds for multicast key distribution showing that no such improvement is indeed possible. The first non-trivial communication lower bound for the problem was proved by Canetti, Malkin and Nissim in [7]. This bound applied to a restricted class of protocols, namely protocols where the group members have a bounded amount of memory, or the key distribution scheme has a special "structure-preserving" property. (See [7] for details.) A different, and seemingly optimal, lower bound, for a more general class of protocols without memory or structure restrictions was later proven by Snoeyink, Suri and Varghese [32], who showed that in any secure multicast key distribution 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)[3]. Snoeyink *et al.* [32] also provide a simple variant of the original LKH protocol [36, 35] that meets the established lower bound. (The

[1] By a "message", we refer to a fixed-size packet of sufficiently large size to allow for the transmission of a single (possibly encrypted) key or key share.

[2] For simplicity, we consider groups of fixed size in analyzing protocols, that is, we assume that each time a user leaves, another one is immediately added. So the size of the group is always equal to n. We refer to each simultaneous execution of a leave and a join as a "group update operation" above. Section 5 provides a discussion on groups with separate leave and join operations.

[3] A similar, but quantitatively weaker, result was independently proven by Yang and Lam [38].

variant essentially involves replacing binary trees with ternary ones.) This apparently closed the gap between upper and lower bounds for the problem, putting a final word to the search for an optimal solution.

It turns out, however, that despite the close relationship between existing protocols and the lower bound proven in [32], the class of protocols considered in that work falls short of accommodating all known protocols. In particular, the protocol model used in [32] mandates that the group center transmits rekey messages only of the form $E_{K_1}(K_2)$—a random key K_2 encrypted with another random key K_1. More general protocols are indeed possible (as even mentioned in [32]) and have also been considered in practice. Some standard, and eminently practical, techniques quite commonly used in cryptographic protocols (but not incorporated in the model of [32]) are: the use of (a) *nested* encryption, where more than one encryption operations are applied in a sequence to the same message to create ciphertexts of the form $E_{K_1}(E_{K_2}(\cdots E_{K_m}(K)\cdots))$. Nested encryption has been used in some existing multicast encryption protocols like those of [11, 8, 12]; (b) *pseudorandom generators* or *PRGs*, which can be used to expand a single random key into multiple "seemingly random" ones. In principle, this allows to transmit multiple keys at the price of just one. (In order to transmit a set of pseudorandom keys each derived from a single key K, the center needs to send just K to the recipients.) Various multicast encryption protocols in the literature [6, 29, 12] make use of PRGs; and (c) *secret sharing schemes*, which can be used to "split" a single message into multiple *shares*, such that the original message can be recovered given only specific (pre-specified) subsets of the shares. Such schemes have been used in certain broadcast encryption protocols in the past (e.g., the protocol of [27]).

The inadequacy of the models used by previous lower bounds is clearly demonstrated by known protocols that "beat" the lower bound proved in [32]. For example, [6] uses a combination of symmetric-key encryption and pseudorandom generators to achieve 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 [32].

Another setting which clearly illustrates the limitations of the protocol model used in previous lower bound papers [7, 32], is that of protocols secure only against individual corruption of single receivers. Micciancio and Panjwani [24] have recently proved that when the group center does not make use of nested encryption, security against single receivers is equivalent to security against multiple receivers colluding together. So, although not explicitly stated in [32], the $3\log_3(n)$ lower bound proved in that paper applies to protocols satisfying a weak form of security that does not allow corrupted receivers to collude. This is in constrast to known protocols [12, 8] with *constant* communication cost per update (independent of the group size), which beat the lower bound of [32] (against single corruptions) by making use of nested encryption.

These observations highlight the limited applicability of result of [32], and take us back to our initial question: Can we design practical multicast encryption protocols (secure against collusions) with communication efficiency better than $\log_2(n)$?

1.2 Our contribution

We resolve the above question (in the negative) for a large class of symmetric-key multicast encryption protocols, namely, protocols that make arbitrary, blackbox usage of symmetric-key encryption, PRGs and secret sharing schemes. We show that any protocol that uses these three primitives to generate the center's messages and achieves a reasonable level of security must incur a communication cost of at least $\log_2(n)$ (upto an additive sub-constant term) per group update operation in the worst case.

More specifically, we exhibit an adversarial strategy for performing group update operations that forces the center in any secure multicast key distribution 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 protocol model we use to prove this bound is fairly general and captures all known protocols for the problem. Furthermore, our lower bound is extremely tight in the sense that it matches the upper bound of [6] (the most communication-efficient protocol known) *up to a sub-constant, additive term*. This can be interpreted as saying that the gain derived from using pseudorandom generators in [6] is just about the best possible and that even the use of iterated encryption and secret sharing schemes cannot substantially improve the communication complexity of their protocol.

The definition of security that we use in proving this lower bound parallels security notions used in most previous work on multicast encryption. In particular, we require that the group key at every instant be

unrecoverable by the users outside the multicast group, even when these users collude in an attempt to recover it. We remark that our proof makes essential use of the full power of this definition, as illustrated by (insecure) protocols [12, 8] with constant communication complexity that still tolerate individual corruption of single users.

Our lower bound, like those of [7, 32], is a *worst-case* lower bound for multicast encryption protocols, proven with respect to an adversarially chosen sequence of membership update operations. This is perhaps in contrast with the average-case performance analysis more commonly used when evaluating the *performance* of networking protocols. We remark that the use of worst-case analysis is standard in cryptography and security, and it is motivated by the fact that when a system is under attack, the assumptions underlying average-case analysis may not hold. For example, an adversary mounting an attack to a multicast key distribution protocol may let a set of corrupted users join and leave the group in a way that maximizes the success probability of the attack.

One last contribution of this work is the explicit use of a symbolic model of computation in the formulation and proof of our lower bound. (Previous lower bounds relied either on randomized models of ideal cryptography [7], or informal definitions [38, 32].) Symbolic security models were previously used in the design and analysis of secure multicast protocols. (See for example [24] and references therein.) We show how symbolic security models can be used both to precisely describe a large class of multicast encryption protocols and to give simple combinatorial proofs of lower bounds and impossibility results.

1.3 Organization

In Sect. 2 we briefly review related work. Section 3 gives a detailed description of the model used to prove our lower bound. The lower bound is proved in Section 4. Section 5 concludes the paper with a discussion of possible extensions and open problems.

2 Related Work

As already mentioned, previous work on private communication in broadcast scenarios is based on two distinct formulations—*broadcast encryption* and *multicast encryption*. In the broadcast encryption model, introduced by Fiat and Naor [13], a central authority (for example, a PayTV provider) broadcasts encrypted information to a set of low-end receiving devices, such that only a "legitimate" subset of these receivers are able to procure the transmitted information. The receivers that are not part of the legitimate subset are said to be *revoked* and the set of revoked receivers changes with time. An important feature of broadcast encryption is that receivers are assumed to be stateless, in the sense that at any instant, a legitimate receiver can decrypt any transmitted information using only a set of keys given to him at the beginning of the protocol (and without having to record the protocol history). Communication is typically one-way—from the broadcast station to all the receivers (although some recent protocols [40] are "public key" in nature and allow anyone to send information to the legitimate receivers). Also, the set of receivers is typically fixed a priori and in most systems it is not possible to add fresh receivers after protocol initialization.

Broadcast encryption schemes find application in several scenarios, the most important ones being secure PayTV networks and media content protection and have received a lot of attention from the cryptography and digital rights management (DRM) communities [13, 3, 14, 15, 26, 18, 17, 9, 5]. In general, broadcast encryption protocols tend to be less efficient than those for multicast encryption owing to the requirement of statelessness on the part of the receivers. A near-optimal combinatorial lower bound on the communication complexity of symmetric-key broadcast encryption protocols was given by Luby and Staddon [21]; their bound, however, applies to protocols that use only symmetric-key encryption (without allowing for nesting) and is not tight even for this class of protocols [26].

In this paper, we are concerned with *multicast* encryption, which is closely related to broadcast encryption but with some important differences. As in broadcast encryption, this problem relates to communicating secret information to a dynamic group of users (or devices). However, unlike broadcast encryption schemes, users in a multicast encryption protocol must remain online and update their internal states while they are

part of the multicast group. At the same time, communication is not always from a central authority to a set of receivers: the users in the group themselves may broadcast messages directed to the other group members (imagine a private virtual room with dynamic membership). Also, the space of potential users of the channel can be extremely large (for example, all users on the Internet) and is not fixed a priori. This problem has been studied extensively in the computer networks and cryptography communities [36, 35, 25, 7, 6, 30, 38, 12, 22, 32]. Most of the existing protocols for multicast encryption use a centralized architecture, wherein a single central authority is tasked with establishing a common shared key for all group members and users are assumed to share a unique secret key with this authority. (Such a key can be established during protocol setup using, say, public-key techniques.) This paper also deals with the centralized model of key distribution.

As in most previous work on multicast encryption, we use a model in which group key updates are performed by the protocol after every single leave or join operation (or a simultaneous execution of the two) takes place. We point out that some protocols in the literature [37] also allow "batched" key updates; in such protocols, the group key needs to be updated only once for multiple simultaneous leave and join operations. However, the efficiency benefits derived from batched rekeying are typically not very significant; for example, in the protocol of [37], a key update for batches of size upto $\sqrt{n}$ involves transmitting at least $\log_2(n)/2$ rekey messages per user, which is within a factor 2 of the lower bound we prove for single user leaves and joins.

A preliminary version of this paper appeared in [23]. Subsequent to [23], a lower bound of $\log_2(n)$ for a problem similar to the one studied here was proven in [34]. The results in [34] are incomparable to ours. On the one hand, the lower bound proved in [34] applies to individual group update operations, while ours considers the amortized cost of a sequence of group updates. On the other hand, the results of [34] apply only to a very restricted class of multicast protocols (called *key graph protocols* in [34])), even smaller than the class of protocols studied in previous lower bound papers [7, 38, 32]. Needless to say, it is easy to design practical protocols that beat the (non-amortized) lower bound on individual group update operations proved in [34]. In fact, amortizing over a sequence of update operations (as done in this paper) seem necessary to prove robust lower bounds: the communication cost associated to a specific group update operation can always be reduced by transmitting more messages in a previous step.

Other Security Concerns in Multicast. Besides secrecy, other important security issues in broadcast and multicast that have also been studied in the literature are *authenticity* (making sure that only authorized users can transmit messages and these messages are not altered during transmission [6, 4]), *independence* (emulating a synchronous network where all players transmit and receive messages at the same time [16]), *reliability* (making key distribution protocols robust in the presence of packet losses [29, 33]) and *availability* (protecting the network against denial of service attacks). These are all different security concerns that can be addressed separately using appropriate cryptographic techniques. Here, we briefly discuss authenticity. As discussed in [6], one can define two kinds of authenticity over broadcast channels. In some applications, one may want to authenticate whether the sender of some information to the multicast group is a group member or not. This kind of authentication is possible using efficient symmetric-key cryptography techniques and invoking any protocol for multicast key distribution. A more stringent notion is source authentication, that is, authenticating the exact identity of the sender (who could also be from outside the group). Protocols based on symmetric-key techniques (for example, [28]) have also been proposed for source authentication in multicast but such protocols do not guarantee strong security (for example, the protocol of [28] is secure only under some synchronicity assumptions on the channel users). Indeed, the source authentication problem is closely tied to public-key digital signatures and it has been shown that coming up with efficient and secure source authentication protocols for multicast is essentially equivalent to designing efficient digital signature schemes [4].

3 The Model

Our lower bound on the communication complexity of multicast key distribution protocols is proved within a symbolic security model. What we mean by "symbolic" here is that cryptographic objects are treated as

abstract data types in the model, and their constructors and destructors capture the security features offered by them. Such a model was initially proposed in [10] for analyzing security of encryption protocols against active attacks and is often referred to as the Dolev-Yao security model after the authors of [10]. The model we present is essentially a generalization of the Dolev-Yao model that incorporates various cryptographic primitives besides encryption.

3.1 Cryptographic primitives used

We consider security protocols that make use of messages created using *symmetric-key* cryptographic techniques. Our focus is on symmetric-key techniques since these are the most commonly used operations in practice due to their high efficiency in terms of both computation time and memory/bandwith usage[4].

The most common cryptographic operation used to achieve secrecy goals is *encryption*. An encryption function $\mathbf{E}$ takes as input a key K and a message M, and outputs a *ciphertext* $C = \mathbf{E}_K(M)$, called the encryption of M under K. Informally, an encryption scheme is secure if $\mathbf{E}_K(M)$ hides the content of the message M. The message can be recovered from the ciphertext (if and only if the encryption key K is known) using the corresponding decryption operation $\mathbf{D}_K$:

$$\mathbf{D}_K(\mathbf{E}_K(M)) = M. \tag{1}$$

Another operation frequently used in conjunction with keyed cryptographic primitives (like encryption) is *pseudorandom generation*. A pseudorandom generator is a function $\mathbf{G}$ that allows to expand a single key K (called the *seed*) into a longer sequence of seemingly random and independent keys:

$$\mathbf{G}(K) = (\mathbf{G}_0(K), \mathbf{G}_1(K), \ldots, \mathbf{G}_{\ell-1}(K)). \tag{2}$$

We remark that the keys $\mathbf{G}_0(K), \ldots, \mathbf{G}_{\ell-1}(K)$ can be efficiently computed from K, so any party that knows K, will also know $\mathbf{G}_0(K), \ldots, \mathbf{G}_{\ell-1}(K)$. Pseudorandom generators can be (and, in fact, have been) used to improve the communication efficiency of multicast encryption protocols as they allow to transmit multiple keys $\mathbf{G}_0(K), \ldots, \mathbf{G}_{\ell-1}(K)$ at the price of transmitting only the seed K. From a security point of view, it is assumed that pseudorandom keys are as good as (formally, indistinguishable from) real ones. In particular, encryption schemes remain secure when used with pseudorandom keys instead of truly random ones.

One last cryptographic primitive we consider is *secret sharing*. A secret sharing scheme [31] allows to "split" a message M into components (called shares) $\mathbf{S}_1(M), \ldots, \mathbf{S}_{\mathsf{n}}(M)$, for some fixed integer n, such that the message can be recovered only from certain subsets of the shares. These special subsets are defined using an *access structure* $\Gamma \subseteq 2^{\{1,\ldots,\mathsf{n}\}}$. For every subset $I \in \Gamma$, if we are given the set of shares $\mathbf{S}_I(M) = \{\mathbf{S}_i(M)\}_{i \in I}$ of some message M, then we can efficiently recover M using a reconstruction function $\mathbf{R}$:

$$\mathbf{R}(I, \mathbf{S}_I(M)) = M. \tag{3}$$

A typical example is the "k out of n" secret sharing scheme where the access structure is the set of all subsets of $\{1, \ldots, \mathsf{n}\}$ of size at least k, that is, the secret can be reconstructed if and only if at least k 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.) In this paper, we consider secret sharing schemes with arbitrary access structure Γ.

3.2 Symbolic security model

We consider security protocols where messages are created using arbitrary nested combination of encryption, pseudorandom generators and secret sharing schemes. Formally, messages are described by expressions

[4]We remark, though, that our result also extends to protocols that use public-key encryption schemes instead of symmetric-key ones.

derived from the following grammar:

$$M \rightarrow K \mid \mathbf{E}_K(M) \mid \mathbf{S}_1(M) \mid \ldots \mid \mathbf{S}_n(M)$$
$$K \rightarrow R \mid \mathbf{G}_0(K) \mid \mathbf{G}_1(K)$$

where the variables M and K represent messages and keys, and R is a variable that ranges over a (potentially infinite) set of truly random keys $\{r_1, r_2, \ldots\}$. Examples of messages generated by our grammar are:

- $\mathbf{E}_{r_1}(\mathbf{G}_0(\mathbf{G}_1(r_3)))$—the encryption of pseudorandom key $\mathbf{G}_0(\mathbf{G}_1(r_3))$ under key r_1;
- $\mathbf{E}_{\mathbf{G}_0(r_1)}(\mathbf{E}_{\mathbf{G}_1(r_2)}(r_5))$—the double encryption of random key r_5 under pseudorandom keys $\mathbf{G}_1(r_2)$ and $\mathbf{G}_0(r_1)$;
- $\mathbf{E}_{r_2}(\mathbf{E}_{\mathbf{G}_0(r_3)}(\mathbf{S}_2(r_5)))$—the double encryption of the key share $\mathbf{S}_2(r_5)$ under $\mathbf{G}_0(r_3)$ and r_2.

A few remarks regarding our model are in order:

- For simplicity, we have assumed that the stretch factor of the pseudorandom generator equals $\ell = 2$. (Such pseudorandom generators are called "length-doubling".) This is without loss of generality because pseudorandom generators with larger stretch factors can be easily built from length-doubling ones using standard chaining techniques [39, 2].
- Although our grammar allows ciphertexts to be created using arbitrary, nested applications of the encryption function, it does not permit the use of ciphertexts either as encryption keys or as seeds to the pseudorandom generator. Using ciphertexts to play the role of keys is injudicious cryptographic practice since, in general, they do not have the pseudorandomness properties that keys are required to possess[5]. No security protocol that we are aware of uses ciphertexts in this manner.
- Our grammar enables shares to be created by applying the sharing function iteratively on both keys and ciphertexts. Shares can themselves be encrypted under multiple keys but cannot 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.[6]
- All functions $\mathbf{E}(\cdot)$, $\mathbf{G}_b(\cdot)$ and $\mathbf{S}_i(\cdot)$ in our grammar have the property that (when implemented) they produce outputs of length at least as big as (but usually comparable to) their input values. In particular, we can regard all messages M as having approximately the same length, and measure the communication complexity of a protocol as the number of messages M transmitted, regardless of the structure of the messages.

Messages as defined by the above grammar are purely syntactic objects. The intended meaning of the expressions and the syntactic operations employed in their construction is described by an entailment relation $\mathsf{M} \vdash m$ specifying what information m can be recovered given a set of messages M. The entailment relation

[5] For example, given any (provably secure) encryption function, $\mathbf{E}$, it is possible to design another (provably secure) encryption function, $\mathbf{E}'$, such that one can easily recover a message, M, from a corresponding ciphertext $\mathbf{E}'_{\mathbf{E}'_{K_1}(M_1)}(M)$ without knowing either K_1 or M_1. As a trivial example, consider this: Let $K_1[1..n]$ denote the first n bits of key K_1 and define $\mathbf{E}'$ as $\mathbf{E}'_{K_1}(M_1) = \mathbf{0} \| \mathbf{E}_{K_1}(M_1)$ ($|\mathbf{0}| = n$) if $K_1[1..n] \neq \mathbf{0}$ and $\mathbf{E}'_{K_1}(M_1) = M_1$ if $K_1[1..n] = \mathbf{0}$. If we choose a large enough value for n (e.g., polynomial in the security parameter) then provable security of $\mathbf{E}$ (under the standard notion of indistinguishability against chosen plaintext attacks) implies the same for $\mathbf{E}'$. However, $\mathbf{E}'_{\mathbf{E}'_{K_1}(M_1)}(M)$ is almost always equal to M.

[6] For a protocol that uses shares for encryption or pseudorandom generation to be provably secure, the sharing scheme must be implemented in a manner that guarantees pseudorandomness of all shares. (This need not be true for every implementation of the scheme.)

is defined inductively by the following rules:

$$\begin{aligned} m \in \mathsf{M} &\Rightarrow \mathsf{M} \vdash m \\ \mathsf{M} \vdash k &\Rightarrow \mathsf{M} \vdash \mathbf{G}_0(k), \mathbf{G}_1(k) \\ \mathsf{M} \vdash \mathbf{E}_k(m), k &\Rightarrow \mathsf{M} \vdash m \\ \exists I \in \Gamma. \forall i \in I. \mathsf{M} \vdash \mathbf{S}_i(m) &\Rightarrow \mathsf{M} \vdash m \end{aligned}$$

The first rule is obvious: every message in M can be recovered from M. The other three rules correspond to the correctness conditions (1), (2) and (3) associated to the three cryptographic primitives. For example, if $\mathsf{M} = \{r_1, \mathbf{E}_{r_1}(\mathbf{S}_1(r_2)), \mathbf{E}_{G_0(r_1)}(\mathbf{S}_2(r_2))\}$ and $\{1,2\} \in \Gamma$, then we can recover r_2 from M using the following sequence of operations:

$$\begin{aligned} r_1 \in \mathsf{M} &\Rightarrow \mathsf{M} \vdash r_1 \\ \mathbf{E}_{r_1}(\mathbf{S}_1(r_2)) \in \mathsf{M} &\Rightarrow \mathsf{M} \vdash \mathbf{E}_{r_1}(\mathbf{S}_1(r_2)) \\ \mathsf{M} \vdash r_1, \mathbf{E}_{r_1}(\mathbf{S}_1(r_2)) &\Rightarrow \mathsf{M} \vdash \mathbf{S}_1(r_2) \\ \mathsf{M} \vdash r_1 &\Rightarrow \mathsf{M} \vdash G_0(r_1) \\ \mathbf{E}_{G_0(r_1)}(\mathbf{S}_2(r_2)) \in \mathsf{M} &\Rightarrow \mathsf{M} \vdash \mathbf{E}_{G_0(r_1)}(\mathbf{S}_2(r_2)) \\ \mathsf{M} \vdash G_0(r_1), \mathbf{E}_{G_0(r_1)}(\mathbf{S}_2(r_2)) &\Rightarrow \mathsf{M} \vdash \mathbf{S}_2(r_2) \\ \mathsf{M} \vdash \mathbf{S}_1(r_2), \mathbf{S}_2(r_2) &\Rightarrow \mathsf{M} \vdash r_2 \end{aligned}$$

The last step in this derivation process holds because we assumed $\{1,2\}$ to be a part of the access structure Γ. For any M, the set of messages that can be recovered from M using $\vdash$ is denoted $Rec(\mathsf{M})$.

3.3 Secure multicast key distribution

Let $[N] := \{1, \cdots, N\}$ denote a set of users having access to a reliable and authenticated broadcast channel, with N being a large (potentially infinite) value. A multicast key distribution protocol Π for N users has two components: a setup program $\mathcal{I}$, and a key distribution program $\mathcal{C}$. The setup program assigns long-lived keys to all users and decides the initial state of the program $\mathcal{C}$. For all i, the long-lived key of user i, denoted k^i, is an arbitrary random or pseudorandom key (that is, an arbitrary expression derived from K in grammar (4)), subject to the requirement that it is not derivable from any other long-lived key. In other words, there must exist no two keys k^i, k^j such that $k^i = \mathbf{G}_{b_1}(\mathbf{G}_{b_2}(\cdots \mathbf{G}_{b_l}(k^j) \cdots))$ for some bits $b_1, \cdots, b_l$ and $l \geq 0$. This condition guarantees that long-lived keys are computationally indistinguishable from independent truly random keys, and is essential for the security of the protocol.

The program $\mathcal{C}$ models the behavior of the group key distribution authority. For every instant t, it takes a set $\mathcal{M}_t \subseteq [N]$ (the set of *members* at time t) and outputs a set of messages (while also updating its state internally). The output messages, called *rekey* messages, are used to establish a shared secret key $k(t)$ among all members at time t. Each message is derived from the variable M in our grammar.

The set of members and non-members changes with time and these changes are determined by an adversarial entity, $\mathcal{A}$. At every instant t, $\mathcal{A}$ issues one of three types of commands:

- $LEAVE(i)$: set $\mathcal{M}_t = \mathcal{M}_{t-1} \setminus \{i\}$;
- $JOIN(i)$: set $\mathcal{M}_t = \mathcal{M}_{t-1} \cup \{i\}$;
- $REPLACE(i,j)$: set $\mathcal{M}_t = \mathcal{M}_{t-1} \setminus \{i\} \cup \{j\}$

$\mathcal{A}$ also decides the initial membership of the group, $\mathcal{M}_0$. For any instant t, the set of non-members at that instant (that is, the set $[N] \setminus \mathcal{M}_t$) is denoted $\mathcal{N}_t$.

For any sequence of sets, $\widetilde{\mathcal{M}}_t := (\mathcal{M}_0, \mathcal{M}_1, \cdots, \mathcal{M}_t)$ determined by such an adversary, let $\mathsf{Msgs}(\widetilde{\mathcal{M}}_t)$ denote the union of all rekey messages output by $\mathcal{C}$ when given $\mathcal{M}_1, \cdots, \mathcal{M}_t$ (in that order) as input. We measure the communication complexity of the protocol by amortizing over such sequences of membership

updates. Namely, the communication cost, $c(\widetilde{\mathcal{M}}_t)$, incurred by the protocol when run on input $\widetilde{\mathcal{M}}_t$, is equal to $\frac{|\mathsf{Msgs}(\widetilde{\mathcal{M}}_t)|}{t+1}$. This is expressed in terms of the maximum number of members at any point during this execution. (In fact, as we will see, in our lower bound analysis, the adversary always keeps the number of members equal to the initial size of the group $|\mathcal{M}_0|$.)

Definition 1 (Security) *A multicast key distribution protocol $\Pi = (\mathcal{I}, \mathcal{C})$ is secure if for all t, and all sequences $\widetilde{\mathcal{M}}_t = (\mathcal{M}_0, \mathcal{M}_1, \cdots, \mathcal{M}_t)$ determined by the adversary, there exists a key $k(t)$ (called the* **group key** *at time t) such that*

- *Each member at time t can recover $k(t)$; that is, $\forall i \in \mathcal{M}_t : k(t) \in Rec(\{k^i\} \bigcup \mathsf{Msgs}(\widetilde{\mathcal{M}}_t))$*
- *Non-members at time t cannot recover $k(t)$ even by colluding; that is, $k(t) \notin Rec(\{k^i\}_{i \in \mathcal{N}_t} \bigcup \mathsf{Msgs}(\widetilde{\mathcal{M}}_t))$*

The above definition of security says that non-members should not be able to obtain the group key at any instant based only on the information obtained *at or before* that instant. This kind of security is often referred to as *forward secrecy* [32]. To strengthen the security definition, one can also require *backward secrecy*—the group key at any instant should not be computable by a non-member even *after* that instant (which implies, in particular, that new entrants to the group cannot compute past group keys). However, using a weak definition of security only makes our results stronger: Our goal here is to prove a lower bound and proving a lower bound on protocols that satisfy the above (weaker) definition immediately implies that the same bound holds for more secure protocols.

Another important remark is the following. Since most networking protocols do not provide any form of security, it is a good practice to assume that an adversary attacking the network has access to all transmitted data, which needs to be properly protected using appropriate cryptographic techniques. Moreover, this allows for the development of security solutions that are independent of the underlying networking technology. In the above definition, the criterion $k(t) \notin Rec(\{k^i\}_{i \in \mathcal{N}_t} \cup \mathsf{Msgs}(\widetilde{\mathcal{M}}_t))$ models the fact that the adversary has complete knowledge of all past communication. The assumption of infinite memory is less reasonable in the case of group members in our correctness criterion ($\forall i \in \mathcal{M}_t$: $k(t) \in Rec(\{k^i\} \cup \mathsf{Msgs}(\widetilde{\mathcal{M}}_t))$), because we would like legitimate group members to be able to determine the group key given only the information transmitted since the moment they joined the group. However, giving all past rekey messages to every user makes our security definition less stringent, which, as already discussed, only strengthens our lower bound.

4 The Lower Bound

Let $\Pi = (\mathcal{I}, \mathcal{C})$ be any secure multicast key distribution protocol and consider an execution of the protocol upto time t, given an arbitrary sequence of member sets, $\widetilde{\mathcal{M}}_t$, as input. For any such execution, the keys used in the protocol can be partitioned into two classes—those that are recoverable by non-members at time t and those that are not.

Definition 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 $k \in Rec(\mathsf{Msgs}(\widetilde{\mathcal{M}}_t) \bigcup \{k^i\}_{i \in \mathcal{N}_t})$. It is* useful *otherwise.*

The set of useful keys at time t is denoted $\mathsf{UsefulKeys}_\Pi(\widetilde{\mathcal{M}}_t)$. For all t, and all sequences $\widetilde{\mathcal{M}}_t$, the group key $k(t)$ and the long-lived keys of members, $\{k^i\}_{i \in \mathcal{M}_t}$, must be in $\mathsf{UsefulKeys}_\Pi(\widetilde{\mathcal{M}}_t)$ (for otherwise the security definition, defn. 1, would be violated by the protocol).

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 $\mathbf{E}_{k_i}$ (for some key k_i) 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 $\mathbf{G}_0(r_3)$. For any message m, the key that it encapsulates is denoted $\kappa(m)$.

Definition 3 *A message m is* useful *(resp.* useless*) at time t if m encapsulates a key that is useful (resp. useless) at that time, that is, if $\kappa(m) \in \mathsf{UsefulKeys}_\Pi(\widetilde{\mathcal{M}}_t)$ (resp. $\kappa(m) \notin \mathsf{UsefulKeys}_\Pi(\widetilde{\mathcal{M}}_t)$).*

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 upto 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 multicast group (that is, is made a non-member), all useful keys known to that member (in particular, the group key) turn useless, and, as a result, all messages encapsulating these keys that were transmitted in the past also become useless. Our goal in this paper is to design an adversarial strategy for performing group updates in such a way that for every update involving the removal of a member, "sufficiently" many protocol transmissions become useless. Specifically, we show how the adversary can execute a sequence of replace operations such that, for *every* operation in this sequence, at least logarithmically many useful rekey messages (sent upto that point of time) become useless. Such messages may have been sent at any time in the past (and not necessarily at the moment when the replaced member joined the group) but the protocol must incur a communication cost of one for each such message. Thus, in order to maintain the security constraint, the protocol must, *on the average*, incur a logarithmic communication cost for every membership update.

4.1 A combinatorial interpretation

For the proof, we interpret keys and messages using graph-theoretic terminology. With every execution of protocol Π for a sequence of sets $\widetilde{\mathcal{M}}_t$ we associate a directed graph $\mathcal{G}_\Pi(\widetilde{\mathcal{M}}_t)$ called the *key graph* at that time. Every vertex in this graph models a useful key at time t and paths between vertices abstract the process of key recovery (defined by the entailment relation $\vdash$) starting from a single useful key. We introduce edges in the graph in a manner that ensures that for every two (useful) keys k_1, k_2, if k_2 is recoverable from k_1 at time t (that is, $k_2 \in \mathit{Rec}(\{k_1\} \cup \mathsf{Msgs}(\widetilde{\mathcal{M}}_t))$), then there exists a path from k_1 to k_2 in $\mathcal{G}_\Pi(t)$. For example, for any two keys k_1 and k_2 such that $k_2 = G_0(k_1)$, we introduce an edge $k_1 \to k_2$ in $\mathcal{G}_\Pi(\widetilde{\mathcal{M}}_t)$. Similarly, if the message $\mathbf{E}_{k_1}(k_2)$ was sent by the protocol at or before time t (that is, if $\mathbf{E}_{k_1}(k_2) \in \mathsf{Msgs}(\widetilde{\mathcal{M}}_t)$), then the graph contains the edge $k_1 \to k_2$. More generally, if the messages $\mathbf{E}_{k_1}(k_2), \mathbf{E}_{k_2}(k_3), \cdots, \mathbf{E}_{k_{n-1}}(k_n)$ were sent at or before time t ($k_1, k_2, \cdots, k_n$ all being useful), then the graph contains the path $k_1 \to k_2 \to \cdots \to k_{n-1} \to k_n$.

How do we extend this idea to the case of arbitrarily nested messages? Recall that in our model, every key can be encrypted more than once, using different encryption keys, in the *same* rekey message and the sharing operation can also be applied to various intermediate ciphertexts in an arbitrarily nested manner. Messages can be mapped to edges in several different ways, and the proof of our lower bound relies on a careful choice of this mapping. For every message m encapsulating a key k, we pick a *single* useful key k_i (if there is such a key) under which k is encrypted in m, and create an edge from k_i to k. If k is encrypted in m under multiple useful keys, then we select k_i as the key of the inner-most encryption operation. For example, a rekey message of the form $\mathbf{E}_{k_1'}(\mathbf{E}_{k_1}(k_2))$, where k_1, k_1', k_2 are all useful, gets mapped to the edge $k_1 \to k_2$ (and not $k_1' \to k_2$) since k_1 is the inner-most useful encryption key in it. The same is the case for the messages, $\mathbf{E}_{k_1''}(\mathbf{E}_{k_1'}(\mathbf{E}_{k_1}(k_2)))$ and $\mathbf{E}_{k_1'}(\mathbf{E}_{k_1}(\mathbf{S}_1(k_2)))$, and if k_0 and k_0' are useless keys (for time t), then also for $\mathbf{E}_{k_1}(\mathbf{E}_{k_0}(\mathbf{S}_1(\mathbf{E}_{k_0'}(k_2))))$. 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 $\mathbf{S}_2(k_2)$ and $\mathbf{S}_1(\mathbf{E}_{k_0}(k_2))$, with k_0 being useless at time t, do not map to any edge in $\mathcal{G}_\Pi(\widetilde{\mathcal{M}}_t)$. Such a representation of messages may appear too parsimonious at first glance (for example, protocol messages comprising a key share are not depicted in $\mathcal{G}_\Pi(\widetilde{\mathcal{M}}_t)$ at all, even if the corresponding key is useful at time t) but, as we will see, it suffices for proving our lower bound. A formal definition of $\mathcal{G}_\Pi(\widetilde{\mathcal{M}}_t)$ is given below.

Definition 4 *Let $\Pi = (\mathcal{I}, \mathcal{C})$ be a secure multicast key distribution protocol executed with a sequence of membership update operations $\widetilde{\mathcal{M}}_t$. The* key graph *for the protocol at time t is a directed graph $\mathcal{G}_\Pi(\widetilde{\mathcal{M}}_t) =$*

$(\mathcal{V}, \mathcal{E})$ *where* $\mathcal{V} = \mathsf{UsefulKeys}_\Pi(\widetilde{\mathcal{M}}_t)$ *and* $\mathcal{E}$ *is the set of all ordered pairs* $(k_1, k_2) \in \mathcal{V} \times \mathcal{V}$ *such that at least one of the following is true:*

1. *There exists* $b \in \{0, 1\}$ *such that* $k_2 = G_b(k_1)$*;*
2. *There exists* $m \in \mathsf{Msgs}(\widetilde{\mathcal{M}}_t)$ *such that* $m = e_1(\mathbf{E}_{k_1}(e_2(k_2)))$ *(for arbitrary sequences of encryption and sharing operations* e_1, e_2*) and for all* $k \in \mathsf{UsefulKeys}_\Pi(\widetilde{\mathcal{M}}_t)$*,* $\mathbf{E}_k$ *is not contained in* e_2*.*

The edges in $\mathcal{G}_\Pi(\widetilde{\mathcal{M}}_t)$ that satisfy condition 2 above are called *communication edges*, or c-edges, since they correspond to at least one rekey message transmitted by the protocol upto time t. The map from useful rekey messages to c-edges is a partial function—every useful message sent at or before time t corresponds to zero or one (but not more than one) c-edges in $\mathcal{G}_\Pi(\widetilde{\mathcal{M}}_t)$. Edges other than communication edges are referred to as *given edges*, or g-edges, since they correspond to PRG computations that can be performed without incurring any communication cost.

For any key graph $\mathcal{G}$ and a key k in it, let $in_{\mathcal{G}}(k)$ denote the indegree of key k in $\mathcal{G}$. An important property of key graphs is given by the following proposition:

Proposition 1 *The number of communication edges (c-edges) incident upon any node* k *in a key graph* $\mathcal{G}$ *is at least* $in_{\mathcal{G}}(k) - 1$*.*

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. ∎

The following lemma characterizes how reachability in key graphs corresponds to our notion of key recovery.

Lemma 1 *For any secure multicast key distribution protocol* Π*, for every* $t \geq 0$*, for every sequence of membership updates* $\widetilde{\mathcal{M}}_t$ *performed by the adversary and for every pair of keys* $k_i, k_j \in \mathsf{UsefulKeys}_\Pi(\widetilde{\mathcal{M}}_t)$*, if* $k_j \in Rec(\{k_i\} \bigcup \mathsf{Msgs}(\widetilde{\mathcal{M}}_t))$*, then there exists a path from* k_i *to* k_j *in* $\mathcal{G}_\Pi(\widetilde{\mathcal{M}}_t)$ *such that for every key* k *on this path,* $k \in Rec(\{k_i\} \bigcup \mathsf{Msgs}(\widetilde{\mathcal{M}}_t))$*.*

We defer the proof of this lemma to Section 4.3 and first show how we use it in proving the lower bound.

Since the group key $k(t)$ and the long-lived keys of members $\{k^i\}_{i \in \mathcal{M}_t}$ are useful at time t, they must all be vertices in the key graph for that time. Furthermore, $k(t)$ must be recoverable from the long-lived key k^i of each member $i \in \mathcal{M}_t$ and the set $\mathsf{Msgs}(\widetilde{\mathcal{M}}_t)$. From the lemma, then, it follows that for every t, and every $i \in \mathcal{M}_t$, there exists a path from k^i to $k(t)$ in $\mathcal{G}_\Pi(\widetilde{\mathcal{M}}_t)$ and each key k on this path is known to member i (that is, $k \in Rec(\{k^i\} \bigcup \mathsf{Msgs}(\widetilde{\mathcal{M}}_t))$). For every such t and i, we pick an arbitrary path satisfying this criterion and denote it as $P_i(t)$. We will henceforth focus on the sub-graph of every key graph consisting only of (the union of) these paths, that is, paths from member long-lived keys to the group key.

Definition 5 *For any secure multicast key distribution protocol* Π*, executed with a sequence of membership updates* $\widetilde{\mathcal{M}}_t$*, the* member key graph *at time* t*, denoted* $\hat{\mathcal{G}}_\Pi(\widetilde{\mathcal{M}}_t)$*, is the sub-graph of* $\mathcal{G}_\Pi(\widetilde{\mathcal{M}}_t)$ *such that an edge is in* $\hat{\mathcal{G}}_\Pi(\widetilde{\mathcal{M}}_t)$ *if and only if it is contained in the path* $P_i(t)$ *(from* k^i *to the group key* $k(t)$*) for some* $i \in \mathcal{M}_t$*.*

Note that for any $i, j \in \mathcal{M}_t$ such that $i \neq j$, the key k^j must not lie on the path $P_i(t)$ since otherwise it would be recoverable by member i, and, as a result, the protocol would be insecure. (For example, if k^j was on the path $P_i(t)$, then the adversary could remove i from the group at time $t+1$ and then, any group key known to j, in particular $k(t+1)$, would be known to i as well.) This means that for all $i \in \mathcal{M}_t$, the node k^i has no edges incident upon it in the member key graph $\hat{\mathcal{G}}_\Pi(\widetilde{\mathcal{M}}_t)$; that is, $in_{\hat{\mathcal{G}}_\Pi(\widetilde{\mathcal{M}}_t)}(k^i) = 0$. Figure 1 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 $P_{i_3}(t)$.

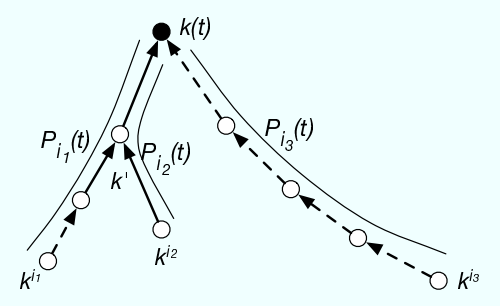

Figure 1: A member key graph corresponding to $\mathcal{M}_t = \{i_1, i_2, i_3\}$. c-edges have been shown with solid lines and g-edges with broken lines. The paths $P_{i_1}(t)$ and $P_{i_2}(t)$ share a common edge $k' \rightarrow k(t)$.

4.2 The adversarial strategy

We are now ready to describe the adversarial strategy that we use to prove our lower bound. We define an adversary that issues only $REPLACE$ commands to the protocol and at each instant t, he replaces a current member i_t with a non-member j_t in a way such that the maximum number of useful rekey messages at time t become useless.

A first attempt at defining such a strategy would be to have the adversary consider the member key graph at time $t' := t - 1$, $\hat{\mathcal{G}}_\Pi(\widetilde{\mathcal{M}}_{t'})$, and to have him replace the member i for whom the group key $k(t')$ is *furthest* from the key k^i in this graph; that is, for whom the length of the path $P_i(t')$ is maximum. Since the removal of i causes all keys known to i (in particular, those that lie on $P_i(t')$), to become useless, one would hope that such a strategy guarantees the above requirement. This, however, is not always the case. For example, consider the member key graph shown in figure 1. Here, $P_{i_3}(t)$ is the longest amongst all paths between a member long-lived key and the group key but by removing i_3, the adversary would make only one useful message, namely the one corresponding to the edge $k' \rightarrow k(t)$, useless. On the other hand, removing member i_1 is more beneficial for him since it leads to 3 such messages becoming useless.

This example illustrates that one needs to be a bit more careful in picking the member to replace at every step. A good adversarial strategy should involve considering, for every current member i, the total number of *c-edges* that are incident upon keys in $P_i(t')$ (rather than just the length of $P_i(t')$) and replacing the member for which *this* number is the largest. According to proposition 1, the number of c-edges incident upon a single node in a key graph is no less than the in-degree of the node minus one. Thus, the total number of c-edges incident upon a path $P_i(t')$ can be no less than the sum of the in-degrees of all nodes on the path, minus the number of nodes in $P_i(t')$. We formalize this idea using the following definition:

Definition 6 *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ 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 $in_{\mathcal{G}}(P)$, is the number of edges in $\mathcal{E}$ starting from a node not in P and incident upon one that is in P. That is,*

$$\begin{aligned} in_{\mathcal{G}}(P) &= \Big|\{(u,v) \in \mathcal{E} \ : \ u \notin P, v \in P\}\Big| \\ &= in_{\mathcal{G}}(s) + \sum_{v \in P; v \neq s} (in_{\mathcal{G}}(v) - 1) \end{aligned}$$

In the special case that $\mathcal{G}$ is a member key graph $\hat{\mathcal{G}}_\Pi(\widetilde{\mathcal{M}}_{t'})$ and $P = P_i(t')$ for some $i \in \mathcal{M}_{t'}$, the node s has in-degree zero and so, the in-degree of P simply equals $\sum_{v \in P; v \neq s}(in_{\mathcal{G}}(v) - 1)$. From proposition 1, then, it follows that the in-degree of $P_i(t')$ bounds from below the total number of c-edges incident upon nodes in it.

Our adversarial strategy can now be formulated in terms of in-degrees of paths. The adversary starts off by setting $\mathcal{M}_0$ to be an arbitrary subset of $[N]$ of size n. At each instant t, he inspects the member

key graph $\hat{\mathcal{G}} := \hat{\mathcal{G}}_\Pi(\widetilde{\mathcal{M}}_{t-1})$, sets i_t to be the member label i for which $in_{\hat{\mathcal{G}}}(P_i(t-1))$ is maximum, sets j_t to be any non-member label such that j_t was not a member before time t, and executes the command $REPLACE(i_t, j_t)$. The procedure is repeated for all instants $t \geq 1$.

The strategy involves only $REPLACE$ operations, and so the size of the group remains fixed at $n = |\mathcal{M}_0|$ throughout the execution of the protocol. We now argue that replacing members in this manner guarantees that at each instant in the protocol, the number of useful messages that become useless is at least $\log_2(n)$. This is proven using the following lemma on the in-degree of paths in graphs. (The proof of the lemma is postponed to Section 4.4.)

Lemma 2 *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an arbitrary graph and let $\{v_1, \cdots, v_n, v\}$ be any set of nodes in the graph 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*

$$in_{\mathcal{G}}(P_i) \geq \lceil \log_2(n) \rceil$$

In the setting of member key graphs, the v_i's in the above lemma correspond to the long-lived keys of the current members, $\{k^i\}_{i \in \mathcal{M}_t}$ and each path $P_i = P_i(t)$ goes from the key k^i to the current group key $k(t)$. Recall that member key graphs have the property that no member long-lived key k^i lies on the path $P_j(t)$ for $j \neq i$. Thus, from the above lemma it follows that for every $t \geq 1$, the member i_t replaced by our adversary at time t is such that the path $P_{i_t}(t-1)$ in the member key graph at time $t-1$ has in-degree at least $\lceil \log_2(n) \rceil$.

As already discussed, the in-degree of a path in a member key graph bounds, from below, the number of c-edges incident upon keys on that path. So, at every instant t, the adversary effectively removes a member i_t for whom the number of c-edges incident upon nodes in $P_{i_t}(t-1)$ is at least $\lceil \log_2(n) \rceil$. Once member i_t is removed, all useful keys known to i_t become useless and thus, all messages encapsulating such keys become useless, too. This, in particular, includes the messages corresponding to the c-edges incident upon nodes in $P_{i_t}(t-1)$, which, by definition, were useful till time $t-1$. Note that the number of such useful messages must at least be equal to the number of c-edges incident upon nodes in $P_{i_t}(t-1)$ itself. Therefore, under the above adversarial strategy, at least $\lceil \log_2(n) \rceil$ useful messages must turn useless at time t. Also, since i_t is never added back to the group by the adversary, these messages must remain useless for all instants after t.

The total number of rekey messages communicated by the protocol upto time t is at least equal to the sum of useful messages sent upto that time, which, we know from above, is at least $t \cdot \lceil \log_2(n) \rceil$. Thus, the communication cost incurred by the protocol upto time t, $\frac{|\mathsf{Msgs}(\widetilde{\mathcal{M}}_t)|}{t+1}$, must be at least $\frac{t}{t+1} \cdot \lceil \log_2(n) \rceil$, which is asymptotically the same as $\lceil \log_2(n) \rceil$. We summarize our result in the following theorem:

Theorem 1 *There exists an adversary strategy consisting only of $REPLACE$ commands such that for any secure multicast key distribution protocol Π, the amortized communication cost incurred by the protocol when executed against that strategy, is at least $(1 - \frac{1}{t}) \cdot \lceil \log_2(n) \rceil = (1 - o(1)) \cdot \lceil \log_2(n) \rceil$. Thus, the asymptotic communication cost of every protocol against this strategy is at least $\lceil \log_2(n) \rceil$.*

To complete the proof of the theorem, we now present the proofs of Lemma 1 and Lemma 2.

4.3 Proof of Lemma 1

Let k_i and k_j be any two keys satisfying the conditions in the statement of the lemma. That is, both the keys are useful (at time t) and k_j can be recovered from the set of expressions $S := \{k_i\} \bigcup \mathsf{Msgs}(\widetilde{\mathcal{M}}_t)$. Let q be the smallest number of applications of the entailment operation $\vdash$ required to derive k_j from S. For example, if $k_j \in S$ then $q = 1$; if $k_j \notin S$ but there exists a key k'_j such that $k'_j, \mathbf{E}_{k'_j}(k_j) \in S$, then $q = 3$; and so on. In the sequel, we refer to every application of $\vdash$ towards the derivation of k_j as a "step" in the derivation process.

We will prove the lemma using induction over q. Namely, we will show, inductively, that for all q and for every useful key k_j recoverable from S in q steps, there exists a path from k_i to k_j in $\mathcal{G} := \mathcal{G}_\Pi(\widetilde{\mathcal{M}}_t)$. The

statement is true for $q = 1$ since in this case k_j must be equal to k_i (for if $k_j \in \mathsf{Msgs}(\widetilde{\mathcal{M}}_t)$, then it would be useless) and so there is a trivial path from k_i to k_j. Suppose that the statement is true for all values of q smaller than a number $Q > 1$. That is, for all $q < Q$, every useful key recoverable from S in q steps has a path leading to itself from k_i in $\mathcal{G}$ and all keys along the path are in $Rec(S)$. Consider any useful key k_j recoverable from S in Q steps. Two possibilities arise:

- *There exists a key k'_j such that $k_j = G_b(k'_j)$ for some $b \in \{0, 1\}$ and k'_j is recoverable from S in $Q - 1$ steps.* For k_j to be useful at time t, the same must be true for k'_j. From the inductive hypothesis, it follows that there exists a path from k_i to k'_j in $\mathcal{G}$ and joining this path with the g-edge $k'_j \rightarrow k_j$ gives us the desired path from k_i to k_j.

- *There exist a set of rekey messages $\mathsf{M} \subseteq \mathsf{Msgs}(\widetilde{\mathcal{M}}_t)$, each message encapsulating k_j, such that the set S_e of encryption keys in* all *these messages is recoverable from S in less than Q steps and $k_j \in Rec(S_e \bigcup \mathsf{M})$.* In this case, we first observe that at least one of the keys in S_e must be useful. (If all of S_e was useless, that is if S_e was contained in $Rec(\{k_i\}_{i \in \mathcal{N}_t} \bigcup \mathsf{Msgs}(\widetilde{\mathcal{M}}_t))$, then k_j, which is in $Rec(S_e \bigcup \mathsf{M}) \subseteq Rec(S_e \bigcup \mathsf{Msgs}(\widetilde{\mathcal{M}}_t))$ would also be contained in $Rec(\{k_i\}_{i \in \mathcal{N}_t} \bigcup \mathsf{Msgs}(\widetilde{\mathcal{M}}_t))$. This would mean k_j is useless—a contradiction!) Thus, there must exist at least one message $m \in \mathsf{M}$ that encapsulates k_j and contains a useful encryption key k'_j, with k'_j being (a) the inner-most useful encryption key in m and (b) recoverable from S in less than Q steps. Again, from our hypothesis, it follows that there exists a path from k_i to k'_j in $\mathcal{G}$ and we join this path with the c-edge $k'_j \rightarrow k_j$ to get a path from k_i to k_j.

Note that in the second case above, k_j can be recoverable from M and S_e using multiple decryption and share reconstruction operations. All these operations are abstracted into a single edge in the key graph $k'_j \rightarrow k_j$.

4.4 Proof of Lemma 2

Our proof of this lemma, again, 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 $in_{\mathcal{G}}(P_i) \geq \lceil \log_2(n) \rceil$. 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 $\lceil \log_2(1) \rceil = 0$.

We hypothesize that for all values of n less than some $\tilde{n} \geq 2$, the statement of the lemma is true. 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 $in_{\mathcal{G}}(P_i) \geq \lceil \log_2(n) \rceil$.

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. (Note that by the maximality of the suffix P, d 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. Figure 2 shows an example with $d = 4$.

Since the total number of vertices 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)

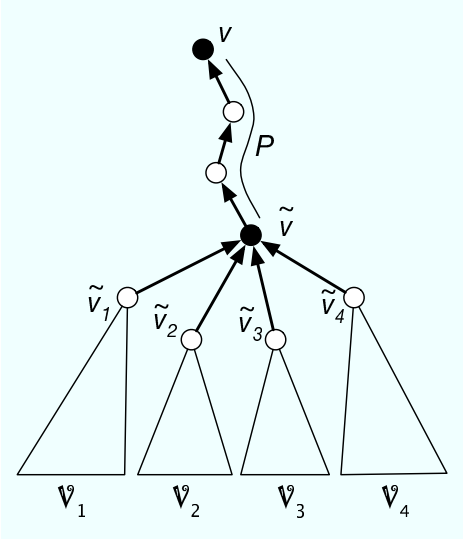

Figure 2: Figure illustrating the proof of Lemma 2.

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 $in_{\tilde{\mathcal{G}}}(\tilde{Q}_{i_{\max}})$ 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 $(in_{\mathcal{G}}(\tilde{v}) - 1)$. (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:

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

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

5 Conclusion

In this paper, we have proven a tight lower bound for the problem of multicast encryption, that applies to a fairly large class of protocols, encompassing, in particular, all protocols currently known in the literature. Our bound shows that even by exploiting sophisticated cryptographic techniques like nested encryption, pseudorandom keys and secret sharing, it is not possible to go below the $\log_2 n$ barrier in the communication complexity of these protocols. Our lower bound is essentially optimal, matching known upper bounds up to small subconstant additive terms.

The lower bound is proved within a generic model of computation where cryptographic operations are treated symbolically. While symbolic models like the one used in this paper have been widely used for the analysis of cryptographic protocols, to the best of our knowledge they had never been explicitly used before to formulate and prove negative results, like lower bounds on the efficiency of secure protocols. We consider our use of symbolic models for the proof of lower bounds an important contribution of our work, and believe that similar symbolic models can be useful to prove lower bounds or impossibility results for other classes of protocols as well.

We conclude the paper discussing some potential variations of and extensions to our model:

Message Concatenation. Symbolic security models as the one used in this paper, typically include a rule $\mathbf{M} \rightarrow (\mathbf{M}, \mathbf{M})$ (within the grammar defining the message space) that allows to concatenate messages together. Our lower bound immediately extends to such more general grammars, provided the communication complexity is appropriately defined, that is, instead of counting the *number* of messages sent by the group center, we consider the total size of the messages.

Groups without Simultaneous Leave and Join. The proof of our lower bound involves defining a sequence of adversarially-chosen $REPLACE$ operations (simultaneous execution of a $LEAVE$ and $JOIN$) and showing that every protocol must incur an average communication cost of $\log_2 n$ for such a sequence. Having $REPLACE$ operations as part of the model considerably simplifies the entire analysis by helping us keep the group size constant over time. However, in a situation where such an operation is not allowed, the bound that we get using our technique is $\log_2(n)/2$ (which is tight upto a "multiplicative" factor of 2). We remark that it is possible to construct (using ideas from [6]) a protocol secure according to definition 1, in which every individual $LEAVE$ can be performed for a cost of $\log_2(n)$ multicast messages and every individual $JOIN$ for $\log_2(n)$ *unicast* messages. This implies an average cost of $\log_2(n)/2$ multicast and $\log_2(n)/2$ unicast per update operation (which meets our lower bound if we ignore the unicast overhead). An interesting question is whether our bound can be extended so that it is tight even when the $\log_2(n)/2$ unicast cost is included in computing communication complexity or whether one can come up with better protocols that involve no unicast at all.

Incorporating Pseudorandom Functions. It would be interesting to extend our model even further to incorporate the use of other cryptographic primitives, like pseudorandom functions (PRFs), in generating rekey messages, and to study multicast key distribution in such a model. Our belief is that an extension with PRFs will not affect our lower bound (that is, the communication complexity cannot be improved beyond $\log_2 n$) but it is still worthwhile trying to investigate the possibility of better and more efficient protocols using other primitives[7].

Analyzing Upper Bounds. The model for multicast key distribution we study in this paper can also be used to analyze upper bounds but in doing so, one must take care of some issues which we ignore in our framework. For example, in our model, the group members can compute the shared secret key at any instant by looking at the rekey messages sent out in the entire history of the protocol. Practical protocols should require that members be able to get the key using just the rekey messages sent since they joined the group (or, even better, using messages sent in the "recent" past). Also, we do not address the issue of storage limitations of the users or the group center: in practice, key updates should be made possible not only with minimal communication overhead but also with minimal state information stored by each user. Finally, for real protocols, it is important to be able to prove security in a strong computational sense (rather than using a symbolic adversarial model as in our definition, definition 1). In [24], we prove a general soundness theorem that provides sufficient conditions under which symbolic definitions of security for multicast encryption (like definition 1) imply security against arbitrary computationally-bounded attackers. The conditions laid down by this soundness theorem must be verified before one can assert that a given protocol provides computational security guarantees[8].

[7] We note that some known protocols do make use of pseudorandom functions (for example, [29]), but not in a substantial way, meaning that whatever they do can be easily achieved using pseudorandom generators instead. In effect, these protocols fit the model of multicast key distribution used in this paper.

[8] The soundness result of [24] applies to protocols that use nested encryption and PRGs only; a soundness result for protocols that use secret sharing as well is proved in [1], although the latter result uses a weaker adversarial model than that used in [24].

References

[1] 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.

[2] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo random bits. *SIAM Journal on Computing*, 13(4):850–864, Nov. 1984. Preliminary version in FOCS 1982.

[3] C. Blundo, L. A. F. Mattos, and D. R. Stinson. Trade-offs between communication and storage in unconditionally secure schemes for broadcast encryption and interactive key distribution. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO '96, Proceedings of the 16th annual international Cryptology conference*, volume 1109 of *Lecture Notes in Computer Science*, pages 387–400, Santa Barbara, CA, USA, Aug. 1996. Springer.

[4] D. Boneh, G. Durfee, and M. Franklin. Lower bounds for multicast message authentication. In B. Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, volume 2045 of *Lecture Notes in Computer Science*, pages 437–452, Innsbruck, Austria, May 2001. Springer.

[5] 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, proceedings of the 25th annual international cryptology conference*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275, Santa Barbara, CA, USA, Aug. 2005. Springer.

[6] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOM 1999. Proceedings of the Eighteenth Annual Joint conference of the IEEE computer and communications societies*, volume 2, pages 708–716, New York, NY, Mar. 1999. IEEE.

[7] R. Canetti, T. Malkin, and K. Nissim. Efficient communication-storage tradeoffs for multicast encryption. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT '99, Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, volume 1592 of *Lecture Notes in Computer Science*, Prague, Czech Republic, May 1999. Springer.

[8] 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.

[9] J. H. Cheon, N. su Jho, M.-H. Kim, and E. S. Yoo. Skipping, cascade, and combined chain schemes for broadcast encryption. In *Cryptology ePrint Archive, Report 2005/136*, 2005.

[10] D. Dolev and A. C. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983.

[11] 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.

[12] 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.

[13] A. Fiat and M. Naor. Broadcast encryption. In D. R. Stinson, editor, *Advances in Cryptology - Crypto '93*, volume 773 of *Lecture Notes in Computer Science*, Santa Barbara, California, USA, August 1993. Springer-Verlag.

[14] E. Gafni, J. Staddon, and Y. L. Yin. Efficient methods for integrating traceability and broadcast encryption. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, Proceedings of the 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 372–387, Santa Barbara, CA, USA, Aug. 1999. Springer.

[15] J. Garay, J. Staddon, and A. Wool. Long-lived broadcast encryption. In M. Bellare, editor, *Advances in Cryptology - CRYPTO 2000, Proceedings of the 20th annual international Cryptology conference*, volume 1880 of *Lecture Notes in Computer Science*, pages 333–352, Santa Barbara, CA, USA, Aug. 2000. Springer.

[16] R. Gennaro. A protocol to achieve independence in constant rounds. *IEEE Transactions on Parallel And Distributed Systems*, 11(7):636–647, 2000.

[17] M. Goodrich, J. Sun, and R. Tamassia. Efficient tree-based revocation in groups of low-state devices. In M. Franklin, editor, *Advances in cryptology - CRYPTO 2004, proceedings of the 24th annual international cryptology conference*, volume 3152 of *Lecture Notes in Computer Science*, pages 511–527, Santa Barbara, CA, USA, Aug. 2004. Springer.

[18] D. Halevy and A. Shamir. The lsd broadcast encryption scheme. In M. Yung, editor, *Advances in cryptology - CRYPTO 2002, proceedings of the 22nd annual international cryptology conference*, volume 2442 of *Lecture Notes in Computer Science*, pages 47–60, Santa Barbara, CA, USA, Aug. 2002. Springer.

[19] H. Harney and C. Muckenhirn. Group key management protocol (gkmp) architecture. Request for Comments 2094, Internet Engineering Task Force, July 1997.

[20] H. Harney and C. Muckenhirn. Group key management protocol (gkmp) specification. Request for Comments 2093, Internet Engineering Task Force, July 1997.

[21] M. Luby and J. Staddon. Combinatorial bounds for broadcast encryption. In K. Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, volume 1403 of *Lecture Notes in Computer Science*, pages 512–526, Espoo, Finland, May 1998. Springer.

[22] D. A. McGrew and A. T. Sherman. Key establishment in large dynamic groups using one-way function trees. *IEEE Transactions on Software Engineering*, 29(5):444–458, May 2003.

[23] 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, proceedings of the international conference on the theory and application of cryptographic techniques*, volume 3027 of *Lecture Notes in Computer Science*, pages 153–170, Interlaken, Switzerland, May 2004. Springer.

[24] D. Micciancio and S. Panjwani. Corrupting one vs. corrupting many: the case of broadcast and multicast encryption. In I. Wegener, V. Sassone, and B. Preneel, editors, *Proceedings of the 33rd international colloquium on automata, languages and programming - ICALP 2006*, volume 4052 of *Lecture Notes in Computer Science*, pages 70–82, Venice, Italy, July 2006. Springer.

[25] S. Mittra. Iolus: a framework for scalable secure multicasting. *ACM Computer Communication Review*, 27(4):277–288, Oct. 1997. Presented at ACM SIGCOMM '97.

[26] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001, Proceedings of the 21st annual international Cryptology conference*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62, Santa Barbara, CA, USA, Aug. 2001. Springer.

[27] M. Naor and B. Pinkas. Efficient trace and revoke schemes. In Y. Frankel, editor, *Financial Cryptography*, volume 1962 of *Lecture Notes in Computer Science*, pages 1–20, Anguilla, British West Indies, Feb. 2000. Springer.

[28] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient and secure source authentication for multicast. In *Network and Distributed Systems Security Symposium (NDSS) '01*, pages 35–46, February 2001.

[29] 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.

[30] R. Safavi-Naini and H. Wang. New constructions for multicast rekeying schemes using perfect hash families. In *Proceedings of the 7th ACM conference on computer and communications security - CCS 2000*, pages 228–234, Athens, Greece, Nov. 2000. ACM.

[31] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.

[32] J. Snoeyink, S. Suri, and G. Varghese. A lower bound for multicast key distribution. *Computer Networks*, 47(3):429–441, 2005. Preliminary version in INFOCOMM 2001.

[33] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, and D. Dean. Self-healing key distribution with revocation. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 241, Washington DC, USA, 2002. IEEE Computer Society.

[34] R. Tamassia and N. Triandopoulos. Computational bounds on hierarchical data processing with applications to information security. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proceedings of the 32nd international colloquium on automata, languages and programming - ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 153–165, Lisbon, Portugal, 2005. Springer.

[35] D. M. Wallner, E. G. Harder, and R. C. Agee. Key management for multicast: issues and architecture. Request for Comments 2627, Internet Engineering Task Force, June 1999.

[36] 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. Preliminary versions in ACM Computer Communication Review, SIGCOMM '98.

[37] R. Yang, X. Li, X. Zhang, and S. S. Lam. Reliable group rekeying: a performance analysis. In *SIGCOMM '01*, pages 27–38. ACM, 2001.

[38] Y. R. Yang and S. S. Lam. A secure group key management protocol communication lower bound. Technical Report TR-00-24, 2000.

[39] A. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd annual symposium on foundations of computer science - FOCS '82*, pages 80–91, Chicago, Illinois, USA, Nov. 1982. IEEE.

[40] D. Yevgeniy and N. Fazio. Public-key broadcast encryption for stateless receivers. In *Digital Rights Management – DRM '02*, volume 2696 of *Lecture Notes in Computer Science*, pages 61–80. Springer, 2002.