Optimal Communication Complexity of Generic Multicast Key Distribution*

Daniele Micciancio[1] and Saurabh Panjwani[1]

University of California, San Diego
9500 Gilman Drive, Mail Code 0114,
La Jolla, CA 92093, USA
{daniele, panjwani}@cs.ucsd.edu

Abstract. We prove a tight lower bound for generic protocols for secure multicast key distribution where the messages sent by the group manager for rekeying the group are obtained by arbitrarily nested application of a symmetric-key encryption scheme, with random or pseudorandom keys. Our lower bound shows that the amortized cost of updating the group key for a secure multicast protocol (measured as the number of messages transmitted per membership change) is $\log_2(n) + o(1)$. This lower bound matches (up to a small additive constant) the upper bound of Canetti, Garay, Itkis, Micciancio, Naor and Pinkas (Infocomm 1999), and is essentially optimal.

Keywords: Multicast, Key Distribution, Lower Bounds.

1 Introduction

Broadcast and multicast are communication primitives of fundamental importance for many emerging internet (or more generally, network) applications, like teleconferencing, pay TV, on-line gaming, electronic news delivery, etc. Roughly speaking, broadcast allows data to be (simultaneously) delivered to all nodes in a network at a much smaller cost (in terms of network resources) than transmitting it individually to each intended recipient, and it is essential for the scalability of the applications to groups of medium and large size. Multicast achieves a similar goal, but with an arbitrary (and, often, dynamically changing) set of recipients that does not necessarily include all the nodes in the network. From a security point of view, broadcast and multicast raise many new and challenging issues that are not directly addressed by conventional (point-to-point) cryptographic techniques. (See [3] for a survey.)

* This material is based upon work supported by the National Science Foundation under Grant CCR-0313241 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.

Security properties. As in point-to-point communication (unicast), the two main security concerns are *secrecy* and *authenticity*. In this paper we concentrate on the secrecy property, i.e., making sure that only group members can receive the transmitted data (See Sect. 3 for a precise definition of our communication and security model). Two distinct models have been considered within the cryptographic community to study secrecy properties in broadcast and multicast scenarios. One, called *broadcast encryption*, is motivated mostly by pay TV and similar applications where an information provider communicates with a large (and highly dynamic) set of low-end receivers (e.g., set-top boxes). The other, usually called *multicast encryption* or *multicast key distribution*, is more closely related to internet applications, where a dynamically changing, but relatively stable, group of users wants to broadcast messages within the group, while keeping the content of the messages hidden from users that do not currently belong to the group. This is the model we study in this paper, and we refer the reader to Sect. 2 for a brief discussion of related work, including broadcast encryption as well as other security properties like authenticity.

In unicast, secrecy is easily achieved by establishing a secret key between the two communicating parties, who, in turn, use the key to encrypt all communication using a conventional (symmetric-key) encryption scheme. A similar approach may be used for multicast as well: once a secret key (common to all group members) is established, secrecy can be achieved by encrypting all communication under the common key. However, in the presence of a dynamically changing group, establishing a common secret key can be quite an onerous task: each time a user leaves the group (voluntarily or not), a new group key needs to be established in order to protect future communication. We consider a setting where a single (physical or logical) entity (called the group center) has authority over deciding group membership and is in charge of group key distribution. The problem is how the group center can securely communicate a new key to all the remaining group members, after one of them leaves the group, in such a way that the evicted user cannot recover the new key. Since the old group key can no longer be used to secure communication, communicating a new key seemingly requires unicasting the new key individually to all group members (e.g., as advocated in [10, 9]), but this is clearly not a scalable solution as it would lose essentially all the potential efficiency benefits of using a multicast channel. The now standard approach to this problem, suggested in [16, 15], is to maintain not only a group key, known to all group members, but also a collection of auxiliary keys known to selected subsets of members that can be used to efficiently communicate to subsets of the group when the group membership changes. The solution described in [16, 15] 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

[1] We measure the communication complexity in basic messages, where each message is a fixed size packet of sufficiently large size to allow for the transmission of a single (possibly encrypted) key.

group[2]. Although this is exponentially more efficient than the trivial solution requiring n (unicast) transmissions, it would be desirable to have even more efficient solutions with smaller communication complexity. In [3] an improved solution is given, where the number of transmissions is reduced by a factor of 2, but remains logarithmic in the number of group members.

Previous lower bounds. Our inability to find even better solutions to the multicast key distribution problem has prompted many researchers to explore lower bounds, showing that no such improvement is indeed possible, under reasonable assumptions about the protocol. The first non-trivial communication lower bound for multicast security was proved in [4] for a restricted class of protocols, namely protocols where the group members have a bounded amount of memory, or the key distribution scheme has some special "structure preserving" property. A different, and seemingly optimal, lower bound, for a more general class of protocols without memory or structure restrictions was subsequently proved in [14], where it was shown that any secure multicast key distribution protocol (within a certain class) can be forced to transmit at least $3\log_3 n$ messages for every group update operation (averaged over a long sequence of update operations). [14] also suggested a simple variant of the protocol of [16, 15] (basically, replacing binary trees with ternary ones) meeting their lower bound. This apparently closed the gap between upper and lower bounds for multicast key distribution protocols, putting a final word to our search of an optimal solution. The class of protocols considered in [14] restricts the group center to transmit messages of the form $E_{k_1}(k_2)$ consisting of a key k_2 encrypted with another key k_1. Although not explicitly stated in [14], it is important to note that more general protocols are indeed possible and have also been considered in practice. For example, two relatively standard, and eminently practical, techniques very common in cryptography are the following:

- The use of a pseudorandom generator, say G, to expand a single key k_0 into two or more (seemingly random and independent) keys $(k_1, k_2, \ldots, k_m) = G(k_0)$. In principle, this allows to transmit multiple keys at the price of one, by sending the seed k_0, instead of transmitting the pseudorandom keys individually.
- The use of double (or multiply iterated) encryption, where more encryption functions are applied in a sequence to the same message before transmission. For example, consider a group of four users u_1, u_2, u_3, u_4, where each user u_i knows a private key k_i. Assume two auxiliary keys k and k' are known to groups u_1, u_2, u_3 and u_2, u_3, u_4 respectively. Then a new key k'' can be sent to users u_2 and u_3 by transmitting a single (doubly encrypted) message $E_k(E_{k'}(k''))$. Notice that using single encryption, as in the model considered

[2] For simplicity, we consider groups of fixed size in analyzing multicast key distribution i.e. 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 leave/join operation as a "group update operation". See Sect. 6 for a discussion on variable-sized groups with separate leave and join operations.

in [4, 14], communicating to the same group of users requires the transmission of two messages $E_{k_2}(k'')$ and $E_{k_3}(k'')$.

The inadequacy of the model used by previous lower bounds [4, 14] is clearly demonstrated by known (and practical) protocols that "beat" the lower bound proved in [14]. For example, [3] uses pseudorandom generators to improve the communication complexity of [16, 15] by a factor of 2, resulting in a secure key distribution protocol where all update operations can be performed by transmitting $\log_2(n)$ messages, which is strictly smaller than the $3\log_3(n) \approx 1.89\log_2(n)$ lower bound proved in [14]. This observation opens up again the possibility of further improving the communication complexity of multicast key distribution, or proving more satisfactory lower bounds for more general classes of protocols.

Our contribution. In this paper, we consider generic protocols for multicast key distribution that make arbitrary use of pseudorandom generators and encryption algorithms, where both techniques can be mixed and iteratively applied multiple times in arbitrary ways. In our model, keys can be either freshly generated (i.e. are purely random) or produced by applying a pseudorandom generator (polynomially many times) on freshly generated keys. Messages sent out by the group center for rekeying the group are composed by encrypting keys *iteratively* using different (random or pseudorandom) keys for encrytion at each iteration (See Sect. 3 for a complete description of our model).

The lower bound we prove in this paper on multicast key distribution protocols matches the upper bound of [3] up to a small *additive* term. We demonstrate that in any protocol where the group center broadcasts arbitrary expressions built according to our formal logic for messages, the center must transmit $\log_2(n) + o(1)$ messages per group update operation in the worst case (here, n is the size of the group and the number of messages per update operation is measured by amortizing over an infinite sequence of such operations). In other words, we demonstrate that the use of pseudorandom generators suggested in [3] is essentially optimal, and that even a combined use of iterated encryption does not substantially help to improve the worst-case communication complexity below $\log_2(n)$ messages per update.

Organization. In Sect. 2 we briefly review related work. In Sect. 3 we give a detailed description of the model used to prove our lower bound. The actual lower bound is proved in Sects. 4 and 5. Section 6 concludes the paper with a discussion on possible extensions to our model.

2 Related Work

Previous work on secure communication in broadcast and multicast scenarios is based on two distinct formulations. The first one, often referred to as *broadcast encryption*, has received much attention from the cryptographic community (e.g., [6, 11].) In this model, as originally introduced by Fiat and Naor [6], receivers are stateless, in the sense that they receive a set of keys at the very beginning of the protocol, and they never update their state during protocol execution. However, broadcast encryption schemes are typically secure only against coalitions of bounded size. An essentially optimal lower bound on the communication complexity of broadcast encryption (as a function of the amount of key storage allowed per user) was given by Luby and Staddon in [11],

In this paper we consider a different scenario more closely related to internet applications, where the users maintain state, the group of recipients changes over time, and all users in the group may broadcast information to the other group members. As discussed in the following section, this problem is equivalent to the key distribution problem, where a common secret key is established among all current group members, and updated over time as the group membership changes. This problem, usually called *multicast encryption* or *multicast key distribution*, is the one studied for example in [16, 15, 4, 14] already discussed in the introduction.

Besides secrecy, other important security issues are *authenticity*, i.e., making sure that only authorized users can transmit messages and these messages cannot be altered during transmission, *independence*, i.e., emulating a synchronous network where all players transmit and receive messages at the same time (e.g., see [7]), and *availability*, e.g., protecting the network against denial of service attacks. These are all different security concerns that can be addressed separately using the appropriate cryptographic techniques. Here we briefly discuss *authenticity*. As discussed in [3], one can distinguish different kinds of authenticity. The simplest kind only ensures that the sender of the information is one of the current group members. This can be achieved using the same techniques studied in this paper (e.g., establishing a common secret key and using it within a message authentication protocol.) Individual authentication is a much harder problem, and it has been shown that it is actually equivalent to using public key digital signatures [2].

3 The Model

We consider a scenario in which an information provider wishes to communicate to a selected (and dynamically changing) set of users over a broadcast channel. At any point in time, all users may receive the information sent over the broadcast channel, and we want to ensure that only current group members can decipher the transmitted information and recover the original messages sent by the information provider. A centralized trusted authority, called the group *center*, governs access to the group[3]. The problem of secure multicast communication is easily seen to be equivalent to the problem of establishing a common secret key, known to all and only the current group members: on the one hand, given a secret key shared among all current group members, the information provider can securely communicate with all group members by encrypting its messages

[3] We remark that such a group center is only a logical abstraction, and does not necessarily correspond to any single physical entity, like the information provider or any of the group members.

using a secure symmetric key encryption scheme. On the other hand, given a secure multicast protocol, the center can immediately establish a common secret key among all current group members by picking a new key at random and securely transmitting it to all group members using the secure multicast protocol. Therefore, in the rest of the paper we identify secure multicast encryption with the group key distribution problem. We remark that a common secret key allows all group members to act as information providers and securely transmit information encrypted under the common secret key. The common secret key can also be used to achieve additional security goals besides secrecy (for eg., message integrity against non-members).

3.1 Protocol Initialization

We assume users come from a fixed, but potentially infinite, set $\mathcal{U}$ and that they communicate with the group center using a reliable and authenticated broadcast channel. At every time instant t a finite set of users, $\mathcal{M}_t \subset \mathcal{U}$, referred to as members, holds a shared secret key which is supposed to be known only to the users in this set. All users and the center have black-box access to three functions, E, D and G, where the functions (E, D) model an encryption/decryption pair and G models a pseudorandom generator. We think of these three functions as abstract operations satisfying the following conditions :

- E takes as input two expressions, K (a key) and γ (a message), and outputs another expression, β (a ciphertext). D takes two expressions, K' and β', as input and outputs a third expression γ'. These operations satisfy the obvious correctness condition : $D(K, E(K, \gamma)) = \gamma$. We write $E_K(\gamma)$ for $E(K, \gamma)$.
- G takes as input a key K and outputs two keys, denoted $G_0(K)$ and $G_1(K)$. In other words, the function G models a length-doubling pseudorandom generator. We remark that our choice of using a length-doubling generator (and not a more general one) is only for the purpose of simplifying the analysis and it does not impact our lower bound in any way[4].

Every user $u_i \in \mathcal{U}$ also has a secret key K_i (referred to as the *unique* key of that user) that is known only to him and the group center C from the beginning of protocol execution (Such a key may be established using different techniques in a setup phase using, say, unicast and public key cryptography).

3.2 Rekey Messages

Changes in the group membership (i.e. the set $\mathcal{M}_t$) over time are modeled using an adversary who adaptively chooses to add and delete members from the group.

[4] Indeed, our lower bound can be shown to hold even if we replace G with a function that takes as input a single key and outputs arbitrarily many pseudorandom keys. An intuitive reason for this is that any pseudorandom generator with arbitrary expansion-factor can be easily built using only a length-doubling generator. The proof of Lemma 2 makes this clearer.

At every point in time t, our adversary examines the history and current state of the protocol and issues one of the following three commands:

- $JOIN(u_i)$: set $\mathcal{M}_{t+1} = \mathcal{M}_t \cup \{u_i\}$,
- $LEAVE(u_i)$: set $\mathcal{M}_{t+1} = \mathcal{M}_t \setminus \{u_i\}$,
- $REPLACE(u_i, u_j)$: set $\mathcal{M}_{t+1} = \mathcal{M}_t \setminus \{u_i\} \cup \{u_j\}$,

In response to a membership change request, the group center transmits a set of messages $S_t = (\gamma_1, \ldots, \gamma_{|S_t|})$, known as *rekey messages*, over the broadcast channel where each rekey message, γ_i, is a symbolic expression derived using the following grammar :

$$\begin{aligned} \mathbf{M} &\rightarrow E_{\mathbf{K}}(\mathbf{M}) \mid \mathbf{K} \\ \mathbf{K} &\rightarrow K \mid G_0(\mathbf{K}) \mid G_1(\mathbf{K}) \end{aligned} \tag{1}$$

Here, the symbol $\mathbf{M}$ represents *messages* while the symbol $\mathbf{K}$ represents *keys*. The expression K models any basic (i.e. freshly generated) key, including unique keys of users. Messages can be built from keys by iterated application of the encryption function, E, with basic keys or derived keys (obtained using the pseudorandom generator)[5].

Communication Complexity. The communication complexity of a group key distribution protocol is defined in terms of the *number* of rekey messages transmitted by the center per update operation performed on the group. The cost of transmitting a set of messages S_t equals the number of basic messages in the set (i.e. $|S_t|$). The *amortized cost* of a group key distribution protocol in the course of a sequence of such adversarial operations is the ratio of the total number of messages transmitted by the center in that period to the total number of operations performed. This is expressed in terms of the *size* of the group, which is the maximum number of members in the group at any stage in that sequence of operations (As we will see, in our lower bound analysis, the number of members is kept constant across time). The *amortized communication complexity* of the protocol is the maximum amortized cost it has to incur in the course of *any* sequence of adversarial operations. We are interested in a lower bound on the amortized communication complexity for any group key distribution protocol satisfying certain constraints. We next describe what these constraints are.

3.3 Security Definition

We analyze the security of key distribution protocols with respect to the abstract cryptographic operations E, D and G. This approach is similar to that taken in

[5] Note that we do not allow the use of expressions of the form $E_{\mathbf{K}}(\mathbf{M})$ (i.e. ciphertexts) either as keys or as inputs to the pseudorandom generator because ciphertexts do not necessarily have the (pseudo)randomness properties necessary to prove that such an application would be secure. For example, given any (provably secure) encryption function, E, it is possible to build another (provably secure) encryption function, E', such that one can easily recover a message, γ, from a corresponding ciphertext $E'_{E'_{K_0}(K_1)}(\gamma)$ even without knowing any of the keys K_0 and K_1.

previous lower bounds for this problem[4, 14], except that we also allow for the use of pseudorandomness and arbitrarily nested encryption as dictated by our grammar.

Definition 1. *For any set, S, of messages obtained using grammar 1, we define the set of keys that can be derived from S as the smallest set, $\mathbf{Keys}(S)$, which satisfies the following three conditions :*

- *If $K_0 \in S$, then $K_0 \in \mathbf{Keys}(S)$.*
- *If $K_0 \in \mathbf{Keys}(S)$, then $G_0(K_0) \in \mathbf{Keys}(S)$ and $G_1(K_0) \in \mathbf{Keys}(S)$.*
- *If $E_{K_1}(E_{K_2}(\cdots(E_{K_l}(K_0)))) \in S$ and $K_1, \ldots, K_l \in \mathbf{Keys}(S)$, then $K_0 \in \mathbf{Keys}(S)$.*

This definition corresponds to the intuitive idea that given $E_K(M)$ one can compute M if and only if K is known, and given K everybody can compute $G_0(K)$ and $G_1(K)$ applying the pseudorandom generator to K. However, since pseudorandom generators are one-way, given $G_0(K)$ or $G_1(K)$ (or both) one cannot recover K, or even tell if $G_0(K), G_1(K)$ is in the range of the pseudorandom generator. This is essentially a straightforward generalization of the Dolev-Yao [5] model of encryption, extended with pseudorandom generation. Analyzing security of protocols with respect to this formal cryptographic model is motivated by the fact that we would like the protocols to be secure independently of the specific instantiation of the underlying cryptographic building blocks. The formal analysis can be made precise and interpreted in standard complexity-theoretic terms, by extending known soundness and completeness results of [1, 12].

Definition 2. *We say that a group key distribution protocol is* **secure** *if for any sequence of adverserial operations, and for every time instant t, there exists a key, K, such that*

- *$K \in \mathbf{Keys}(S_1 \cup \cdots \cup S_t \cup \{K_i\})$ for all $u_i \in \mathcal{M}_t$, i.e., key K can be computed by all current group members at time t;*
- *$K \notin \mathbf{Keys}(S_1 \cup \cdots \cup S_t \cup \{K_i \colon u_i \notin \mathcal{M}_t\})$, i.e., the users that do not belong to the group at time t cannot compute K even if they collude and pool together all the information available to them.*

The first clause in the definition is a correctness criterion while the second clause is the main security condition. Note that our definition of security is a bit restrictive in that it requires non-members not to be able to obtain the shared secret key at any instant of time based only on the information obtained *at or before* that instant. Intuitively, this captures the idea that if a user leaves the group (i.e. becomes a non-member) at some point, then he should not be able to decrypt any future communication (even if he colludes with other non-members to do so), unless, of course, he is added back to the group. This kind of security is often referred to as *forward secrecy*. However, one could also require that the shared secret key at any instant be such that the non-members (at that instant) not to be able to compute it even *later on* i.e. even if some of them become members in the future. Such a security requirement is more stringent and it captures the notion that any new entrant to the group should not be able to compute the shared key for any past instant when he was not a member (a requirement often referred to as *backward secrecy*). In order for a protocol to satisfy both forward and backward secrecy, we must strengthen the security condition above so that $K \notin \mathbf{Keys}(S_1 \cup \cdots \cup S_{t'} \cup \{k_i : u_i \notin \mathcal{M}_t\})$ *for all* $t' \geq t$. We remark that backward secrecy is usually considered a less important property than forward secrecy, as in many multicast applications (e.g., stock quotes) information looses value over time. The lower bound proved in this paper only requires forward secrecy and is, thus, applicable to protocols satisfying the more stringent definition, too.

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 security criterion 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, but giving all past broadcast messages to all users makes our security definition less stringent, and consequently it only makes our lower bound stronger. We refer the interested reader to Sect. 6 for a discussion on possible extensions to our model.

4 The Multicast Game

For the actual lower bound analysis, it is useful to view every secure group key distribution protocol as an abstract game, which we call the *multicast game*, played between the group center, C, and the adversary, A. In this game, keys are modelled as nodes in an infinite hypergraph. Each node corresponds to either a basic key (recall that basic keys include unique keys of users as well) or a derived key obtained by applying the pseudorandom generator to some other key. Messages transmitted by the group center are modeled as directed hyperedges, so that the cost incurred by the center equals the number of hyperedges in the graph. For any user, the set of keys known to him at any time is defined as the set of nodes that can be "reached" from the node representing his unique key following the hyperedges. Details follow.

4.1 Game Configurations

The playing board for the multicast game is an infinite collection of rooted binary trees, $T = \{T_1, T_2, \cdots\}$ each containing an infinite number of nodes. The entire set of nodes in these trees is denoted $\mathcal{V}$. The edges are directed edges and every tree in T has one root node which has zero in-degree while all other nodes have in-degree equal to 1. The out-degree of all nodes, including the root, is equal to 2. Every node in this playing board represents a key K. The roots of the trees are associated to the basic keys, while the internal nodes are pseudorandom keys.

The two children of a node represent keys $G_0(K)$ and $G_1(K)$, the keys that can be obtained by applying the pseudorandom generator to the key, K, of the parent node.

The root nodes of some (but not all) trees in T correspond to the unique keys, K_i, of all users. We refer to these special trees as *user trees* and denote the entire set of user trees by U. At any given point in time during the game, the root of every tree in U has one of two labels associated with it – *member* or *non-member*. We refer to the edges in the trees in T as *tree-edges* or simply t-edges and the entire set of t-edges in all the trees is denoted $\mathcal{T}$. A t-edge from a node v_1 to a node v_2 is denoted $v_1 \stackrel{t}{\rightarrow} v_2$.

Rekey messages sent by the group center are modeled as hyperedges as follows. A *directed hyper-edge*, or simply an *h-edge*, over nodes in $\mathcal{V}$ is a pair $\{V, v\}$, denoted $V \stackrel{h}{\rightarrow} v$, where V is a finite subset of $\mathcal{V}$ and v is a single node. The h-edge $V \stackrel{h}{\rightarrow} v$ is said to be incident on v. The hyperedge, $\{K_1, \cdots, K_d\} \stackrel{h}{\rightarrow} K$ models a rekey message of the form $E_{K_1}(E_{K_2}(\cdots E_{K_d}(K)\cdots))$. Here, $K_1, \ldots, K_d, K$ can be either basic or derived keys (i.e., keys associated to either root or internal nodes), and the encryptions can be performed in any order.

A *configuration*, $\mathcal{C}$, of the multicast game is defined as a triple $\mathcal{C} = (\mathcal{M}, \mathcal{N}, \mathcal{H})$, where $\mathcal{M}$ is the set of all member nodes, $\mathcal{N}$ is the set of all non-member nodes and $\mathcal{H}$ is a (finite) set of h-edges over nodes in $\mathcal{V}$. The union $\mathcal{M} \cup \mathcal{N}$ is always equal to the set of roots of the user trees in U. A configuration of the game at time t corresponds to the state of the group key distribution protocol at time t with $\mathcal{M}$ representing the set of members, $\mathcal{N}$ the set of non-members and $\mathcal{H}$ the set $S_0 \cup S_1 \cup S_2 \cup \cdots \cup S_t$ of rekey messages transmitted by C in response to the first t group update operations (plus an optional set S_0 corresponding to the initial configuration of the game).

4.2 Defining Moves of Players

Each move by player C in our game involves adding zero or more h-edges and each move by player A involves changing the label on a node labelled *member* to *non-member* or vice versa, or swapping a member with a non-member. Formally, if the game is in a configuration $\mathcal{C} = (\mathcal{M}, \mathcal{N}, \mathcal{H})$, then

- a move by player C changes the configuration of the game to $\mathcal{C}' = (\mathcal{M}, \mathcal{N}, \mathcal{H}')$ where $\mathcal{H}' = \mathcal{H} \bigcup \mathcal{H}_a$ and $\mathcal{H}_a$ is a finite (possibly empty) set of h-edges over nodes in $\mathcal{V}$;
- a move by player A changes the configuration to $\mathcal{C}' = (\mathcal{M}', \mathcal{N}', \mathcal{H})$ where either
  - $\mathcal{M}' = \mathcal{M} \setminus \{v_m\}$ and $\mathcal{N}' = \mathcal{N} \bigcup \{v_m\}$ for some $v_m \in \mathcal{M}$ (we call this a `delete` move and we say that the node v_m gets deleted from $\mathcal{M}$); or
  - $\mathcal{M}' = \mathcal{M} \bigcup \{v_n\}$ and $\mathcal{N}' = \mathcal{N} \setminus \{v_n\}$ for some $v_n \in \mathcal{N}$ (we call this an `add` move and we say that the node v_n gets added to $\mathcal{M}$).
  - $\mathcal{M}' = \mathcal{M} \bigcup \{v_n\} \setminus \{v_m\}$ and $\mathcal{N}' = \mathcal{N} \setminus \{v_n\} \cup \{v_m\}$ for some $v_n \in \mathcal{N}$ and $v_m \in \mathcal{M}$ (we call this a `replace` move and we say that the node

v_m gets replaced by v_n). This corresponds to a simultaneous execution of an `add` move and a `delete` move, and it leaves the size, $|\mathcal{M}|$, of the group unchanged.

At any time instant t, a pair of moves is played, the first move being played by A, followed by a response by C. Associated with each player's move is a cost function. The cost of a move by player C is the number of h-edges added by him i.e. if a move by player C takes the game from $\mathcal{C} = (\mathcal{M}, \mathcal{N}, \mathcal{H})$ to $\mathcal{C}' = (\mathcal{M}, \mathcal{N}, \mathcal{H}')$, then the cost of the move is $|\mathcal{H}'| - |\mathcal{H}|$. The cost of any move by player A is 1. For simplicity, we concentrate on `replace` operations that leave the size of the group unchanged (since we are interested in proving a lower bound, considering only `replace` operations only makes our result stronger).

4.3 Defining Goals of Players

The security notion described in Sect. 3 is easily modeled in terms of reachability between nodes in the hypergraph corresponding to the current configuration.

Definition 3. *A node, $v \in \mathcal{V}$, is called h-reachable from a set of nodes, $V \subseteq \mathcal{V}$, under a configuration $\mathcal{C} = (\mathcal{M}, \mathcal{N}, \mathcal{H})$ if any of the following conditions hold:*

- *$v \in V$.*
- *There exists a t-edge from some node v' to v and v' is h-reachable from V.*
- *For some $m > 0$ and a set of nodes, $V = \{v_1, v_2 \cdots, v_m\} \subseteq \mathcal{V}$, there exists an h-edge $V \xrightarrow{\text{h}} v$ in $\mathcal{H}$ and each of the nodes, $v_1, \cdots, v_m$ is h-reachable from V.*

We write $V \Rightarrow_{\mathcal{C}} v$ to denote that v is h-reachable from V under $\mathcal{C}$ and $V \not\Rightarrow_{\mathcal{C}} v$ to denote the converse. We say that v is h-reachable from a *node* v' under $\mathcal{C}$ if $\{v'\} \Rightarrow_{\mathcal{C}} v$ holds; this is denoted simply by $v' \Rightarrow_{\mathcal{C}} v$ (similarly, $v' \not\Rightarrow_{\mathcal{C}} v$ denotes that v is not h-reachable from v' under $\mathcal{C}$). If $\mathcal{S}$ is the set of rekey messages represented by $\mathcal{H}$, the set of h-edges in $\mathcal{C}$, and $\mathcal{K}$ the set of keys represented by V, then the set of nodes h-reachable from V under $\mathcal{C}$ corresponds exactly to the set of keys $\mathbf{Keys}(\mathcal{K} \cup \mathcal{S})$ that can be computed from $\mathcal{K}$ and $\mathcal{S}$ according to the Dolev-Yao model of abstract encryption described in Sect. 3.

A configuration which satisfies the security constraint for group key distribution is called a secure configuration:

Definition 4. (Secure Configuration) *A configuration $\mathcal{C} = \{\mathcal{M}, \mathcal{N}, \mathcal{H}\}$ is called a secure configuration if there exists a node, $v_s \in \mathcal{V}$, such that*

- *v_s is h-reachable from every node in $\mathcal{M}$ under $\mathcal{C}$ i.e. $\forall v \in \mathcal{M}, v \Rightarrow_{\mathcal{C}} v_s$*
- *v_s is not h-reachable from $\mathcal{N}$ under $\mathcal{C}$ i.e. $\mathcal{N} \not\Rightarrow_{\mathcal{C}} v_s$*

A node, v_s, which satisfies this property is called a secret node for the corresponding secure configuration.

Clearly, the shared secret key at any instant of time, t, in the protocol, must be (represented by) one of the secret nodes for the game configuration corresponding to time t.

Goals of the players can now be defined in terms of secure configurations. The goal of player C is that at the end of *each* of his moves, the game be in a secure configuration. The goal of player A is the converse of this i.e. at the end of *at least one* of player C's moves, the configuration of the game is not secure. Our aim here is to determine the minimum cost that *every* player C needs to pay, relative to the cost paid by player A, in order to be able to attain his goal in the game against *any* player A.

5 The Lower Bound Proof

In this section we present our main technical result on multicast games which directly implies the lower bound for secure group key distribution protocols.

5.1 Usefulness of h-edges and Canonical Graphs

Let us fix a configuration, $\mathcal{C} = (\mathcal{M}, \mathcal{N}, \mathcal{H})$ in the multicast game for this entire subsection. An h-edge, $V \overset{\mathrm{h}}{\to} v$, in $\mathcal{H}$ is said to be *useless* under $\mathcal{C}$ if $\mathcal{N} \Rightarrow_{\mathcal{C}} v$. An h-edge which is not useless under $\mathcal{C}$ is called *useful* under it. By the definition of h-reachability, for every useful h-edge, $V \overset{\mathrm{h}}{\to} v$, in $\mathcal{H}$, there must exist at least one node in V which is not h-reachable from $\mathcal{N}$ under $\mathcal{C}$. We assume an arbitrary total order on the set $\mathcal{V}$ of all nodes. For any useful h-edge $V \overset{\mathrm{h}}{\to} v$, the first node (according to the total ordering) in V which is not h-reachable from $\mathcal{N}$ (under $\mathcal{C}$) is referred to as the *canonical node* of that h-edge. Canonical nodes are defined only for useful h-edges.

A *canonical edge*, or *c-edge*, corresponding to a useful h-edge, $V \overset{\mathrm{h}}{\to} v$, is a simple directed edge from the canonical node, v_c, of that h-edge to v and is denoted $v_c \overset{\mathrm{c}}{\to} v$. The definitions of canonical nodes and edges are both specific to the configuration $\mathcal{C}$.

Definition 5. *Let $\mathcal{C} = (\mathcal{M}, \mathcal{N}, \mathcal{H})$ be a configuration of the multicast game. A canonical path or a c-path from a node v_1 to another node v_2 ($v_1, v_2 \in \mathcal{V}$) under $\mathcal{C}$, denoted $v_1 \leadsto_{\mathcal{C}} v_2$, is a path consisting of zero or more t-edges and c-edges such that all nodes on this path are h-reachable from v_1.*

At this point it is not clear whether a canonical path must exist from any node v_1 to any other node v_2. Indeed, this does not hold for every pair (v_1, v_2). The following lemma characterizes the existence of canonical paths for certain pairs of nodes - a canonical path from v_1 to v_2 must exist if v_2 is h-reachable from v_1 but is not h-reachable from the set $\mathcal{N}$.

Lemma 1. *For any configuration $\mathcal{C} = (\mathcal{M}, \mathcal{N}, \mathcal{H})$ and any two nodes $v_1, v_2 \in \mathcal{V}$, if $\{v_1\} \Rightarrow_{\mathcal{C}} v_2$ and $\mathcal{N} \not\Rightarrow_{\mathcal{C}} v_2$, then there exists a c-path from v_1 to v_2 under $\mathcal{C}$.*

Proof. Let $R(v_1) \subseteq \mathcal{V}$ denote the set of all nodes which are h-reachable from v_1 (here, and everywhere else in the proof, h-reachable means h-reachable under $\mathcal{C}$). Let $B \subseteq R(v_1)$ be the set of *bad* nodes such that for all $v_2 \in B$, v_2 is not h-reachable from $\mathcal{N}$ and yet, there exists no c-path from v_1 to v_2. Let $G = R(v_1)\backslash B$ (the set of *good* nodes). We claim that either the set of bad nodes is empty or (if not so) v_1 is in it (i.e. $B = \phi$ or $v_1 \in B$).

Suppose this is not the case i.e. suppose that B is non-empty and it still doesn't contain v_1. Then for all nodes in B to be h-reachable from v_1, there exists some node $v_2 \in B$ such that one of the following conditions hold *(i)* For some $v \in G$, $v \xrightarrow{\text{t}} v_2 \in \mathcal{T}$; *(ii)* For some $V \subseteq G$, $V \xrightarrow{\text{h}} v \in \mathcal{H}$. Since any $v_2 \in B$ is not h-reachable from $\mathcal{N}$, an h-edge incident on it must be useful and thus, must have a c-edge corresponding to it. So, if B is non-empty and doesn't contain v_1 there must exist a t-edge or a c-edge from some node $v \in G$ to some node $v_2 \in B$. By the definition of B there exists no c-path from v_1 to such a v_2. Which means there must not be a c-path from v_1 to v as well (else joining such a path with the edge between v and v_2 would give us a c-path from v_1 to v_2). At the same time v must not be h-reachable from $\mathcal{N}$ for that would imply $\mathcal{N} \Rightarrow_{\mathcal{C}} v_2$. Both these two conditions qualify v to be a member of B, which it is not. We, thus, conclude that the set B is either empty or contains the node v_1. If B is an empty set, we're done. If it isn't and it contains v_1, then the definition of B is defied since there exists a trivial c-path (with 0 edges) from v_1 to itself. Thus, the set B must be empty and the lemma holds. ∎

Canonical Graphs We focus our attention on secure configurations from now on. Let $\mathcal{C} = (\mathcal{M}, \mathcal{N}, \mathcal{H})$ be a secure configuration with secret node v_s. By the definition of a secret node and by Lemma 1, for every $v_m \in \mathcal{M}$ there must exist a canonical path from v_m to v_s. For every $v_m \in \mathcal{M}$ select a c-path $P_m \equiv v_m \rightsquigarrow_{\mathcal{C}} v_s$. The canonical graph for $\mathcal{C}$, denoted $G(\mathcal{C})$, is defined as the graph formed by superimposing the c-paths P_m associated to the member nodes $v_m \in \mathcal{M}$. While superimposing paths, if there is more than one c-paths containing an edge between the same two nodes and if at least one of these edges is a c-edges then we insert a single c-edge between the nodes in $G(\mathcal{C})$, and if all these edges are t-edges, then we insert a single t-edge between the nodes. If there is no edge (a c-edge or a t-edge) between any two nodes then there is no edge between them in $G(\mathcal{C})$ also. Note that in the graph $G(\mathcal{C})$, there may be more than one paths from any member node v_m to v_s, but only one of them corresponds (modulo replacement of t-edges by c-edges) to the canonical path P_m associated to v_m. Figure 1(a) shows a toy example of a canonical graph for a configuration with three members nodes $\{v_1, v_2, v_3\}$ and secret node v_s.

For each member node, v_m, in $\mathcal{M}$, we define the *incidence weight* of v_m in the graph $G(\mathcal{C})$ as the number of c-edges in this graph incident on any node along the c-path $P_m \equiv v_m \rightsquigarrow_{\mathcal{C}} v_s$. This is at least equal to the number of c-edges on the c-path itself. The *maximum incidence weight* of the graph $G(\mathcal{C})$ is the maximum among the incidence weights of all member nodes in it. A useful property on

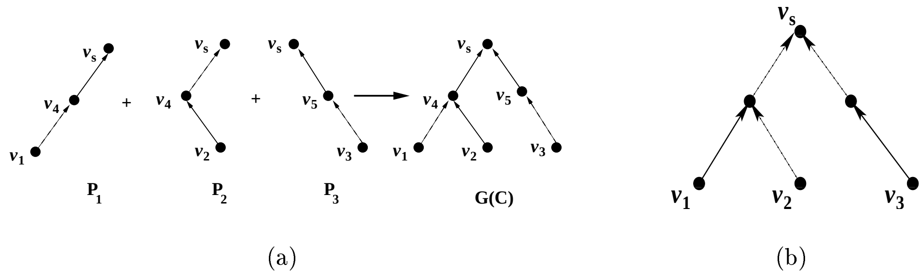

Fig. 1. Canonical Graphs : Figure (a) shows the construction of a canonical graph for a configuration with three member nodes $\{v_1, v_2, v_3\}$ and secret node v_s. c-edges are shown by dark lines while t-edges are shown by dotted ones. Path P_i goes from member i to v_s. Note that there is a c-edge between v_4 and v_s in P_1 and a t-edge between the same two nodes in P_2; the final graph has a c-edge between these nodes because of the higher precedence given to c-edges. In this graph, v_1, v_2 and v_3 have incidence weights 3,3 and 2 respectively. Figure (b) shows an example of a graph that cannot be a canonical graph since the topmost node, v_s, has two t-edges enterring it. This restriction on t-edges will be crucial in proving Lemma 2.

the maximum incidence weight of any canonical graph is given by the following lemma.

Lemma 2. *Let $\mathcal{C} = (\mathcal{M}, \mathcal{N}, \mathcal{H})$ be a secure configuration such that $|\mathcal{M}| = n$. Then, any canonical graph for $\mathcal{C}$ has maximum incidence weight at least $\lceil \log_2 n \rceil$.*

Proof. We shall prove something stronger than what the lemma states. We say that a node $v \in \mathcal{V}$ is *$\mathcal{M}'$-secure* under $\mathcal{C}$ (for some $\mathcal{M}' \subseteq \mathcal{M}$) if $\mathcal{N} \not\Rightarrow_{\mathcal{C}} v$ and for all nodes $u \in \mathcal{M}'$, $u \Rightarrow_{\mathcal{C}} v$. For a set $\mathcal{M}' \subseteq \mathcal{M}$ and a node v which is $\mathcal{M}'$-secure under $\mathcal{C}$, we define a *sub-canonical graph* for $\mathcal{C}$ *over $\mathcal{M}'$ and v*, denoted $G_{\mathcal{C}}(\mathcal{M}', v)$, as a graph formed by superimposing c-paths from nodes in $\mathcal{M}'$ to v, one c-path being selected for every node in $\mathcal{M}'$. The set of c-paths from nodes in $\mathcal{M}'$ to v used for constructing $G_{\mathcal{C}}(\mathcal{M}')$ is denoted $P(G_{\mathcal{C}}(\mathcal{M}', v))$. As a special case, observe that any canonical graph for $\mathcal{C}$ is a sub-canonical graph over $\mathcal{M}$ and v_s.

We hypothesize that for all $i > 0$, if there exists a pair $(\mathcal{M}', v')$ where $\mathcal{M}' \subseteq \mathcal{M}$, $|\mathcal{M}'| = i$ and v' is $\mathcal{M}'$-secure then for every such pair $(\mathcal{M}', v')$ the maximum incidence weight of any graph $G_{\mathcal{C}}(\mathcal{M}', v')$ is at least $\lceil \log_2 i \rceil$. This hypothesis clearly implies the above lemma.

The proof uses an inductive argument on i. For the base case observe that a single node in the set $\mathcal{M}$ is a trivial sub-canonical graph with $\lceil \log_2 1 \rceil = 0$ c-edges. Suppose that for some $j > 1$ and for all $i < j$, the maximum incidence weight of any graph, $G_{\mathcal{C}}(\mathcal{M}_i, v_i)$, with $\mathcal{M}_i \subseteq \mathcal{M}$ and $|\mathcal{M}_i| = i$, if there exists such a graph, is at least $\lceil \log_2 i \rceil$. Suppose there exists a pair $(\mathcal{M}_j, v_j)$ such that, $\mathcal{M}_j \subseteq \mathcal{M}$, $|\mathcal{M}_j| = j$ and v_j is $\mathcal{M}_j$-secure under $\mathcal{C}$. Consider the graph

$G_{\mathcal{C}}(\mathcal{M}_j, v_j)$ and let v_r be the (unique) node in this graph (v_r may be the same as v_j) such that v_r has in-degree greater than 1, say m, and all nodes on the path from v_r to v_j have in-degree exactly 1 (By in-degree of a node we mean the number of c-edges *and* t-edges in $G_{\mathcal{C}}(\mathcal{M}_j, v_j)$ incident on it). Since v_j is not h-reachable from $\mathcal{N}$ and since a c-edge is defined only for a pair of nodes both of which are not h-reachable from $\mathcal{N}$, v_r must also not be h-reachable from $\mathcal{N}$. Let $v_1, v_2, \cdots, v_m$ be the nodes which point to v_r and $\mathcal{M}_1$, $\mathcal{M}_2 \cdots \mathcal{M}_m$ be the sets of member nodes in $\mathcal{M}_j$ for which the canonical paths to v_j go through $v_1, v_2 \cdots v_m$ respectively. Since v_r is not h-reachable from $\mathcal{N}$ under $\mathcal{C}$, none of the nodes $v_1, v_2, \cdots, v_m$ must be so, too. It is not hard to see that, for any $l \in \{1, \cdots, m\}$, the graph formed by superimposing the portion of the c-paths from nodes in $\mathcal{M}_l$ upto the node v_l is also a sub-canonical graph for $\mathcal{C}$ (over $\mathcal{M}_l$ and v_l). Furthermore, there exists some $l' \in [m]$ such that $|\mathcal{M}_{l'}| \geq \lceil j/m \rceil$. From the induction hypothesis, the maximum incidence weight the graph $G_{\mathcal{C}}(\mathcal{M}_{l'}, v_{l'})$ is at least $\lceil \log_2 \left(\frac{j}{m}\right) \rceil$. Finally, the maximum incidence weight of $G_{\mathcal{C}}(\mathcal{M}_j, v_j)$ must be at least equal to the maximum incidence weight of $G_{\mathcal{C}}(\mathcal{M}_{l'}, v_{l'})$ plus the number of c-edges incident on v_r in $G_{\mathcal{C}}(\mathcal{M}_j, v_j)$. A crucial observation is that there can be at most one t-edge incident on v_r, which means at least $m-1$ out of the m edges incident on it must be c-edges. Thus, the maximum incidence weight of $G_{\mathcal{C}}(\mathcal{M}_j, v_j)$ is at least $\min_{m \in [j]} \lceil \log_2 \left(\frac{j}{m}\right) \rceil + m - 1$ which is not less than $\lceil \log_2 j \rceil$. ▮

5.2 The Main Theorem

We consider multicast games in which the group center, C, always maintains the game in a secure configuration. The following theorem establishes a logarithmic lower bound on the amortized cost of the moves performed by C, when the moves of A are adversarially chosen. The lower bound holds for any initial configuration, and even if A only issues `replace` operations that do not affect the size of the group. This lower bound directly implies a $\lceil \log_2 n \rceil$ lower bound on the amortized communication complexity of any secure group key distribution protocol.

Theorem 1. *For every strategy of player C and initial configuration $(\mathcal{M}_0, \mathcal{N}_0, \mathcal{H}_0)$, there exists a strategy of player A consisting of* `replace` *operations only, such that for any $t \geq 1$, the amortized cost, $\tilde{c}_t$, of the first t moves of C is at least $\lceil \log_2 n \rceil - |\mathcal{H}_0|/t$, where $n = |\mathcal{M}_0|$ is the size of the group. In particular, the asymptotic amortized cost of the moves of C is*

$$\lim_{t \to \infty} \tilde{c}_t \geq \lceil \log_2 n \rceil.$$

Proof. Let $(\mathcal{M}_i, \mathcal{N}_i, \mathcal{H}_i)$ be the sequence of configurations following each move by C. We know by assumption that all configurations are secure. Notice that for all i, $\mathcal{H}_{i-1} \subseteq \mathcal{H}_i$, and the cost of each move by C equals $c_i = |\mathcal{H}_i| - |\mathcal{H}_{i-1}|$. The moves of A are chosen as follows. All moves are *replace* moves that substitute one of the current member nodes with a non-member node. In particular, the size of the group is always equal to $n = |\mathcal{M}_0| = |\mathcal{M}_i|$. By Lemma 2, the maximum incidence weight of the canonical graph, $G(\mathcal{C}_i)$, for any configuration $\mathcal{C}_i = (\mathcal{M}_i, \mathcal{N}_i, \mathcal{H}_i)$ is at least $\lceil \log_2 n \rceil$. Let v_i be a member node achieving the maximum incidence weight in $G(\mathcal{C}_i)$. In his ith move, player A replaces the member node v_i with a new node from $\mathcal{N}_i$ that never was a member node before.

For the configuration, $\mathcal{C}_i$, consider the graph, $G(\mathcal{C}_i)$, and the c-path $v_i \rightsquigarrow_{\mathcal{C}_i} v_s$ in this graph (here, v_s is a secret node for $\mathcal{C}_i$). Let $I_{\mathcal{C}_i}(v_i)$ be the set of c-edges in $G(\mathcal{C}_i)$ which are incident on the nodes in $v_i \rightsquigarrow_{\mathcal{C}_i} v_s$ and let $H_{\mathcal{C}_i}(v_i)$ be the set of (useful) h-edges corresponding (uniquely) to the c-edges in $I_{\mathcal{C}_i}(v_i)$. The key observation is that once v_i gets labeled as a non-member node (and its label doesn't change after that), the nodes on the c-path $v_i \rightsquigarrow_{\mathcal{C}_i} v_s$ become h-reachable from the set of non-member nodes under *any* configuration of the game following $\mathcal{C}_i$. This implies that all h-edges in $H_{\mathcal{C}_i}(v_i)$ become (and remain) useless for all configurations from time i onwards, since they are incident on $v_i \rightsquigarrow_{\mathcal{C}_i} v_s$.

Since v_i is a node with maximum incidence weight in $G(\mathcal{C}_i)$, there are at least $\lceil \log_2 n \rceil$ c-edges in $I_{\mathcal{C}_i}(v_i)$ and an equal number of h-edges in $H_{\mathcal{C}_i}(v_i)$. So, each time A performs a move, the number of useless h-edges increases by $\lceil \log_2 n \rceil$, and after t move there are at least $t \cdot \lceil \log_2 n \rceil$ useless h-edges in $\mathcal{C}_t$. Clearly, the number of useless h-edges cannot be greater than the number of h-edges in the final configuration, i.e.,

$$\begin{aligned} t \cdot \lceil \log_2 n \rceil &\leq |\mathcal{H}_t| \\ &= |\mathcal{H}_0| + \sum_{i=1}^{t} |\mathcal{H}_i \setminus \mathcal{H}_{i-1}| \\ &= |\mathcal{H}_0| + \sum_{i=1}^{t} c_i \end{aligned}$$

where c_i is the cost of the ith move performed by C. From this, we immediately get the desired bound on the amortized cost of C's moves : $\frac{\sum_{i=1}^{t} c_i}{t} \geq \lceil \log_2 n \rceil - \frac{|\mathcal{H}_0|}{t}$ ∎

6 Extensions to Our Model

In this section, we address some of the possible extensions and modifications one could make to our model for secure group key distribution described in Sect. 3. Some of these extensions yield models that are equivalent to the model we have already described while others lead to interesting open problems for the group key distribution problem.

1. Allowing Message Pairs: We have proved a lower bound for protocols where the rekey messages sent by the group center consist of a single key encrypted with multiple other keys. It is easy to see that our lower bound also applies to more general protocols where every rekey message can also consist of "pairs" of other rekey messages (i.e. protocols in which the grammar for messages also includes a rule $\mathbf{M} \rightarrow (\mathbf{M}, \mathbf{M})$). Allowing messages pairs does not affect communication complexity in any way.

2. Groups without Simultaneous Leave and Join: Our lower bound for group key distribution is proved using a sequence of simultaneous `join` and `leave` operations (which we refer to as a `replace` operation) performed on the group by an adaptive adversary. One reason for having `replace` operations is that they simplify our analysis considerably (by helping us keep the group size constant over time). In groups where `replace` operations are not allowed, the bound that we get using our technique is $\log_2(n)/2$. We remark that it is possible to construct a practical protocol (without replace operations) in which every individual `join` and `leave` can be performed at the cost of $\log_2(n)/2$ multicast messages and $\log_2(n)/2$ *unicast* messages (this can be done by combining the protocol of [3] with ideas from [13]). The 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. We are unable to resolve this question at the moment and leave it open for future work.

3. Other Cryptographic Primitives: Our model for secure group key distribution allows the usage of iterated encryption and pseudorandom generation for the center's rekey messages and the best known protocols for this problem also use just these cryptographic primitives. It would be interesting to find out if better protocols can be constructed using other cryptographic primitives (for eg., pseudorandom functions, secret sharing) or whether our lower bound can be extended to even more general classes of protocols that allow the usage of such primitives[6].

Analyzing Upper Bounds. The model for secure 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 efficiency issues which we ignore in our framework (note that ignoring such issues only helps to *strengthen* our lower bound). 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. Also, we do not address the issue of storage limitations of the users or the group center. In practice, the key update should be made possible not only with minimal communication overhead but also with minimal storage requirements for the users.

References

1. M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[6] We note that some known protocols do make use of pseudorandom functions (e.g., [13]), but not in a substantial way, meaning that whatever they do can be easily achieved using pseudorandom generators instead.

2. 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 Internarional 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-Verlag.
3. 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.
4. 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-Verlag.
5. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
6. A. Fiat and M. Naor. Broadcast encryption. In D. R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, Proceedings of the 13th annual international Cryptology conference*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491, Santa Barbara, California, USA, Aug. 1993. Springer-Verlag.
7. R. Gennaro. A protocol to achieve independence in constant rounds. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):636–647, 2000.
8. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33:792–807, 1986.
9. H. Harney and C. Muckenhirn. Group key management protocol (GKMP) architecture. Request for Comments 2094, Internet Engineering Task Force, July 1997.
10. H. Harney and C. Muckenhirn. Group key management protocol (GKMP) specification. Request for Comments 2093, Internet Engineering Task Force, July 1997.
11. 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-Verlag.
12. D. Micciancio and B. Warinschi. Completeness theorems for the abadi-rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004. Preliminary version in WITS 2002.
13. A. Perrig, D. X. Song, and J. D. Tygar. ELK, A New Protocol for Efficient Large-Group Key Distribution. In *IEEE Symposium on Security and Privacy*, pages 247–262, Oakland, CA, USA, May 2001. IEEE.
14. J. Snoeyink, S. Suri, and G. Varghese. A lower bound for multicast key distribution. In *INFOCOM 2001. Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 422–431, New York, NY, Apr. 2001. IEEE.
15. 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.
16. 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 version in SIGCOMM 1998.