Efficient Communication-Storage Tradeoffs for Multicast Encryption

Ran Canetti[1], Tal Malkin[2⋆], and Kobbi Nissim[3]

[1] IBM T. J. Watson Research Center, Yorktown Height, NY, 10598,
canetti@watson.ibm.com
[2] Laboratory for Computer Science, Massachusetts Institute of Technology,
545 Technology Square, Cambridge, MA 02139,
tal@theory.lcs.mit.edu
[3] Dept. of Computer Science and Applied Math, Weizmann Institute of Science,
Rehovot 76100, Israel,
kobbi@wisdom.weizmann.ac.il

Abstract. We consider re-keying protocols for secure multicasting in a dynamic multicast group with a center. There is a variety of different scenarios using multicast, presenting a wide range of efficiency requirements with respect to several parameters. We give an upper bound on the tradeoff between storage and communication parameters. In particular, we suggest an improvement of the schemes by Wallner *et al.* and Wong *et al.* [13,14] with sub-linear center storage, without a significant loss in other parameters.
Correctly selecting the parameters of our scheme we can efficiently accommodate a wide range of scenarios. This is demonstrated by Applying the protocol to some known benchmark scenarios.
We also show lower bounds on the tradeoff between communication and user storage, and show that our scheme is almost optimal with respect to these lower bounds.

1 Introduction

Multicast communication (and, in particular, IP multicast routing) is an attractive method for delivery of data to multiple recipients. The motivation for multicast communication is its efficiency – multicast group users get the same message simultaneously, hence the reduction of both sender and network resources. A wide range of applications benefit from efficient multicast: interest groups, file and real-time information update, video multi-party conferences, on-line games and pay TV are few examples.

Securing multicast communication is non-trivial and poses a number of challenges, ranging from algorithmic problems, through system and communication design, to secure implementation. (See overview in [5,4].) The main security concerns are typically *access control* — making sure that only legitimate members

⋆ Supported by DARPA grant DABT63-96-C-0018. Part of this work was done while the author was visiting the IBM T.J. Watson Research Center.

© Springer-Verlag Berlin Heidelberg 1999

of a multicast group have access to the multicast group communication, *source authentication* — verifying that received multicasted data is unmodified and originates with the claimed source, and *maintaining availability* — protecting against denial-of-service and clogging attacks.

This paper focuses on providing access control for multicast communication. The standard technique to this end is to maintain a common key that is known to all the multicast group members, but is *unknown* to non-members. All group communication is then encrypted using the shared key. (We remark that long-term secrecy is typically not a concern for multicast communication; encryption is used mainly for obtaining short-term access control.) The main problem here is *key management* — how to maintain the invariant that all group members, and only them, have access to the group key in a group with dynamic membership. We limit ourselves to the case where there is a centralized *group controller* (or, *group center*) who handles the task of key management. Whenever a member joins or leaves the group, the group key needs to be changed and the new key needs to be let known to all members.

We concentrate on efficient schemes for this *re-keying* problem. In particular, we show a *tradeoff* between communication and storage parameters for the group controller and members, and provide nearly optimal upper and lower bound for some of these parameters. Our protocol is parameterized in terms of the tradeoff, allowing different choices of parameters to result in a variety of scheme performances. This makes the protocol suitable for different applications and scenarios. The works of [13,14] on efficient re-keying schemes are the starting point for this work.

1.1 Security of Re-keying Schemes

A standard security requirement from the data encryption mechanism is *semantic security* [8] of the group communication. Assuming the usage of appropriate (semantically secure) encryption schemes, this requirement reduces to the semantic security of the group session key k_s, shared by the group members. I.e. it is required that an adversary cannot distinguish the real session key from a random key.

If the only operation allowed is joining new users to the group, the re-keying problem is solved by simply giving the session key k_s to the new users. If backward privacy is also required (i.e. new users should not have access to past messages), then a new session key k_s^{new} may be selected and given to the new users, and $E_{k_s}(k_s^{new})$ is multicasted. (Alternatively, the new key can be locally computed as a pseudorandom function of the old key.)

Removing users from the group requires the change of k_s (and possibly other data) to guarantee the semantic security of the new key against any coalition of removed users. It is stressed that security is required against *any* coalition of removed users. In particular, we do not assume any limit on the size or structure of the coalition.

To be able to focus on the re-keying problem we assume authenticated and reliable communication, or more specifically that the messages sent by the group center arrive at their destination and messages are not modified, generated, or replayed by an adversary. These concerns should be addressed separately.

1.2 Efficiency of Re-keying Schemes

Efficiency of multicast re-keying schemes is measured by several parameters: (i) communication complexity, (ii) user storage and (iii) center storage and (iv) time complexity. In this paper we concentrate on the communication and storage complexity measures (of course, without letting the time complexity be infeasible).

Communication complexity is probably the most important measure, as it is the biggest bottleneck in current applications. (Indeed, reducing communication is the main motivation for using multicast technology.)

Reducing the *center storage* enables small memory in the security module (which is responsible for key management). This module is typically separate from the module(s) handling group membership; this latter task typically requires special handling of each member upon joining and leaving the group, and is left out of scope of this work. The module separation can be either logical or physical. Furthermore, for large groups the membership module may consist of several disparate components handling different regions, while the key management module remains centralized. Also, the performance and latency requirement from the key-management module may be more stringent.

Using our scheme, the center storage may indeed be sub-linear, thereby improving on the the best previously known schemes [13,14], without a significant change in other parameters. E.g. with current technology, for a million users multicast group, our reduction enables a security module with all its storage in fast cache memory, making it considerably more efficient.

The motivation for reducing *user storage* stems from applications in which the users are low-end, and have severe memory restrictions (e.g. when the multicast group consists of cable TV viewers and the user module resides in the cable converter unit).

Since there is a large number of potential multicast scenarios it seems unlikely that a single solution will fit all scenarios. This motivates a *tradeoff* between efficiency parameters. Simple solutions suggest that such a tradeoff exists: (i) One extreme is a center that shares, in addition to the session key, a distinct symmetric key with each user. When a user is removed, the center sends new symmetric keys and a new session key to each of the users separately. Thus, user storage is minimal but the communication costs are proportional to the number of group users. (ii) An opposite extreme is having a key for every possible subset of users, where every potential user gets all the keys for the subsets that contain her. Whenever a user is removed, the session key is set to the key of the remaining subset of users. The length of the re-keying message of this solution is optimal (it suffices to declare each removed user), but the number of keys held by each user is clearly prohibitive (at least 2^{n-1} keys, where n is the group size).

Our goal is to study the tradeoff between communication and storage, and construct schemes which are flexible enough to fit a variety of scenarios, in a way that is provably optimal (or close to optimal).

We achieve this goal with respect to the tradeoff between communication and user storage. For the tradeoff between communication and center storage, our upper bound is better than all previously known schemes. Proving a lower bound on the latter tradeoff remains an intriguing open problem.

1.3 Summary of Results

We give an upper bound on the tradeoff between user storage, center storage and communication, and a lower bound relating user storage and the minimal communication. The gap between the bounds is at most logarithmic in the size of the group. Moreover, for a natural class of protocols, including all currently known ones, the gap is closed, namely our scheme is optimal for this class. Thus, our upper bound is nearly optimal with respect to our lower bound, in a strong sense. Our upper bounds are based on the re-keying schemes of Wallner *et al.* and Wong *et al.* [13,14], with improvements of [4] and McGrew and Sherman [10]. These schemes communicate $\log n$ encrypted keys per update, and require linear center storage ($2n - 1$ keys), and logarithmic user storage ($\log n$ keys).

Upper Bound We give an upper bound (i.e. a protocol) which allows trading center storage with communication, with the restriction that communication is lower bounded as a function of user storage. Specifically, for a group of n users with user storage of $b + 1$ keys, the communication is $O(bn^{1/b} - b)$ encrypted keys. Center storage multiplied by communication length is roughly $O(n)$.

One instance yields $O(\log n)$ communication, $O(\log n)$ user storage, $O(\frac{n}{\log n})$ center storage. This is the first scheme with center storage sub-linear in n. Other instances are suitable for different applications, as we demonstrate by applying our scheme to benchmark scenarios.

In practice, re-keying protocols may be used in "batch mode", where the center does not immediately perform updates, but rather waits until several updates accumulate and perform all of them at once. (This is acceptable for most applications.) Doing this allows in many cases (such as in our scheme) significant savings in the communication. However this paper focuses on updates one-by-one, as this is the worst case scenario.

Lower Bounds We first give a lower bound on the communication of re-keying protocols as a function of user storage. We prove that if each user holds at most $b + 1$ keys, the communication costs are at least $n^{1/b}$ encrypted messages.

We further consider the class of *structure preserving protocols* (to which currently known schemes belong [13,4]). Intuitively, structure preserving protocols are those that maintain the property of "u_1 knows m keys which u_2 doesn't" across updates. That is, if user u_1 holds m keys which are not known to user u_2, then after deleting a user $u_3 \notin \{u_1, u_2\}$ and performing the necessary updates, u_1 still holds about m keys not known to u_2. For structure preserving protocols, we show a tight (up to small constant factors) lower bound of $bn^{1/b} - b$ messages (matching our upper bound protocol).

The lower bound is for algorithms that use a "generic" key encryption mechanisms. Formally, we assume a "black-box encryption service" that is the only means of encryption (i.e., the algorithm should provide perfect secrecy in the idealized model). Consequently, the implication of the lower bounds is that in order to achieve more efficient protocols than ours one would have to use specific properties of a particular encryption system, such as exploit algebraic properties of the keys used.

1.4 Related Work

A different approach to solving the problem of allowing only legitimate users to access multicasted data is put forward by Fiat and Naor [6]. In their formalization, a center uses a broadcast channel to transmit messages to a group of users $\mathcal{U}$. There are two *pre-specified* sets: (i) collection $\mathcal{S} \subseteq 2^{\mathcal{U}}$ of legal subsets of recipients, and: (ii) collection $\mathcal{C} \subseteq 2^{\mathcal{U}}$ of possible "bad" coalitions. The goal is to enable the center to communicate data secretly to a given set $S \in \mathcal{S}$ of users, while preventing any coalition from $\mathcal{C} - S$ to gather information on the data. Any such mechanism can be used to establish a group key and thus provides a solution to the re-keying problem.

The [6] solution is radically different than ones discussed here. In particular, it allows encrypting multicast communication even without requiring all users to have a single common key; in addition, joining and leaving of members does not necessarily require *any* action by the other members. However, their solution assumes in a critical way some bound on the size or structure of the coalition of adversarial non-members. This work considers schemes where no such assumptions are made.

There have been some works in broadcast encryption models that consider lower bounds on storage and communication, and show that both cannot be simultaneously low. Luby and Staddon [9] allow arbitrary coalitions, but restrict the possible subsets of recipients to be all sets of certain size $n - m$. In this model they study the tradeoff between the number of keys held by each user, and the number of transmissions needed for establishing a new broadcast key. They assumed a security model that allows translating the problem to a combinatorial (set theoretic) problem. Their lower bound states that either the number of transmissions is very high, or the number of keys held by every user is high.

Blundo, Frota Mattos and Stinson [2] and Stinson and Trung [12] study communication storage tradeoff in a model of *unconditionally secure* broadcast encryption [2] by providing some upper and lower bounds for key pre-distribution schemes (e.g. [3,11]) and broadcast encryption. This model further differs from ours in that information theoretic security is required, and storage and communication are measured in terms of amount of secret *information* stored by each user, and the broadcast *information rate.*

Organization In Section 2 we describe our communication and encryption model. The upper bound scheme is described in Section 3. Finally, we prove lower bounds on the tradeoff between user storage and communication in Section 4.

2 Preliminaries

Let $\mathcal{U}$ denote the universe of all possible users[1], and GC denote the group center. We consider a set $M = \{u_1, \ldots, u_n\} \subseteq \mathcal{U}$, called the *multicast group* (for simplicity, $GC \notin M$). A session key k_s is initially shared by all users in M and by GC (and is not known to any user $v \notin M$). In addition, other information may be known to the users and the center. We abstract away the details of the initialization phase by which the users get their initial information. In particular we may assume that each user joining M has an authenticated secure unicast channel with the center GC for the purpose of initialization. (In practice this may be obtained by using a public key system.) After the initialization phase, and throughout the lifetime of the system, the only means of communication with group members is via a multicast channel on which the group center may broadcast messages that will be heard by all users in $\mathcal{U}$. Our goal is to securely update the session key when the group M changes, so that all users in the group, and only them, know the session key at any given time.

A *multicast protocol* specifies an algorithm by which the center may update the session key (and possibly other information) for the following two update operations on M:

- *remove*(U) where $U \subseteq M$. The result is the removal of users in U from the multicast group: $M^{new} = M \setminus U$.
- *join*(U) where $U \subseteq \mathcal{U}$. The result is the joining of users in U to the multicast group: $M^{new} = M \cup U$.

Since the worst case for the re-keying protocol is when $|U| = 1$, from now on we assume $|U| = 1$ and measure the efficiency of our protocols accordingly. In our description we focus on the removal of users from the multicast group, since dealing with joining users is much simpler and can be done with virtually no communication overhead.

Since we do not want to consider specific private key encryption and their particular properties, we concentrate on a general key-based model, where the cryptographic details are abstracted away. This is modeled by a publically available black-box pair E, D, such that E given as inputs a key k and a message m outputs a *random* ciphertext $c = E(k, m)$; given a ciphertext c and a key k, the decryption algorithm D outputs the plaintext m. (We assume that the encryption is deterministic; that is, two applications with the same message and key will result in the same ciphertext. Probabilistic encryption can be built upon E, D in straightforward ways.) This model guarantees that, when multicasting a message encrypted with a key k, any user holding k will be able to decrypt, and any coalition of users that does not hold k gains no information from hearing the ciphertext. To formalize our requirement that all encryption and decryption is being done via the black-box pair E, D, we let the adversary be computationally *unbounded.* A lower bound in our model means that any scheme which beats the

[1] There is no need to a-priori have an explicit representation of $\mathcal{U}$. For example, $\mathcal{U}$ may be the set of all users connected to the Internet.

bound must be based on a particular encryption scheme and its particular (We remark that, although this model is formalized with the lower bounds in mind, our re-keying schemes can be proven secure even in this model.)

Multicast Encryption Protocols We define the model of key-based multicast as follows. Let l be a security parameter, and let the number of users n be polynomial in l. Let $K \subset \{0,1\}^l$ be a set of *keys*. Each user $u_i \in M$ holds a subset $K(u_i) \subseteq K$ of keys. In particular, there is a "session key" $k_s \in K$ such that every $u \in M$ holds k_s. For a set of users $U \subseteq M$ we define $K(U) = \bigcup_{u \in U} K(u)$. We say that a set $U \subseteq M$ holds a key $k \in K$ if $k \in K(U)$.

In response to a request for update operation the group center (following a given protocol) sends a multicast message that results in changed group keys (and possible other keys). For a key $k \in K$ and a string $m \in \{0,1\}^l$, the group center GC may send over the broadcast channel the ciphertext $E_k(m)$. Users holding k may decrypt and obtain m. After all the ciphertexts for an update have been broadcasted by the center, the users who can decrypt ciphertexts do so, and follow the protocol specification to update their keys. The new total set of keys is denoted by K^{new}.

For the definition of security, we consider an adaptive adversary who may, repeatedly and in an arbitrary order, submit update (remove/join) operations to the center for subsets of his choice, and break into users $u \in \mathcal{U}$ of his choice (thereby getting all of u's information).

We say that a multicast system is *secure* if for any adversary, after any sequence of operations as above, if the adversary has not broken into any user who was in the multicast group while a key k_s was the session key, then the adversary has no advantage in distinguishing k_s from a random key. Note that this definition implies *backward security* as well, since the adversary is not allowed to learn any information about a previous session key, unless he broke into a user who legitimately belonged to the group at the time that key was used). We also do not put a restriction on the number of users the adversary may break into.

Finally, by convention, when performing a *remove*(U) operation, all keys in $K(U)$ are removed from K^{new} (since we require arbitrary resilience, it can be shown that there is no advantage in using a key of a removed user to broadcast a message, and thus these keys may be removed). In particular, k_s is also removed, and thus a new key must resume the special role of a session key k_s^{new}.

The *communication complexity* of an update operation is measured by the number of ciphertexts that need to be broadcasted by the center per update (for the worst case choice of update), and is denoted by $c(n)$ for a group of size n. The *storage* is measured by the number of keys that need to be stored.

3 A Re-keying Scheme

We start by describing two schemes that our construction will be built upon. The first (described in Section 3.1) is a simple scheme achieving minimal (constant) storage for the center and each user, but highly inefficient (linear) communication complexity. The second (described in Section 3.2) is a widely used scheme by Wallner *et al.* and Wong *et al.* [13,14] (with an improvement of [4]), which we call the basic tree scheme. This scheme achieves logarithmic communication complexity and logarithmic storage for each user, but linear storage for the center. We then show (in Section 3.4) how the basic tree scheme can be generalized and combined with the minimal storage scheme, so as to achieve an improved scheme with a tradeoff between the parameters. As a special case, we get a reduction of the center storage in the tree scheme by a logarithmic factor.

3.1 A Minimal Storage Scheme

We describe a simple scheme, which requires the smallest possible amount of storage – two keys for the center and each user[2], but is very communication intensive, requiring $(n-1)$ ciphertext sent per removal of a user. We will later use this scheme as a building block in our construction.

In this scheme each user u holds the session key k_s, and a unique symmetric key k_u not known to any other user. The center should be able to generate the keys of all users, which is possible by holding a single secret key r, an index to a pseudo-random function f_r [7] (which can be constructed from the same black-box used for encryption). The keys can be generated by applying the function to the user's index, namely $k_u = f_r(u)$.

When a group of users U is removed from the group, the center chooses a new session key k_s^{new}, and sends it to each user, by broadcasting the ciphers $E_{k_u}(k_s^{new})$ for all $u \in M^{new} = M \setminus U$.

The security of this scheme is based on the security of the encryption scheme and pseudo-random function. The parameters are summarized in Table 1.

3.2 The Basic Tree Scheme

We describe the scheme by Wallner *et al.* and Wong *et al.* [13,14] (with the improvement of [4]). For a detailed description, we refer the reader to [13,14,4].

The group center creates a balanced binary tree with at least n leaves and assigns a l-bit random key to every node. Let k_ϵ denote the key assigned with the tree root v_ϵ. Denote the left and right children of node v_σ by $v_{\sigma 0}, v_{\sigma 1}$ and their assigned keys by $k_{\sigma 0}, k_{\sigma 1}$ respectively (i.e. the left and right children of the node indexed by σ are indexed by σ concatenated with 0 or 1 respectively). Every user in M is assigned a leaf and is given the $\log n + 1$ keys assigned to nodes on the path from the root to this leaf. Since k_ϵ is known to all group members it is used as the session key: $k_s = k_\epsilon$.

Notation Let $\sigma \in \{0,1\}^*$. Denote by σ^i the string resulting by erasing the i rightmost bits of σ. Denote by $flip(\sigma)$ the string resulting by flipping the rightmost bit of σ.

Let $G : \{0,1\}^l \to \{0,1\}^{2l}$ be a pseudo random generator that doubles the size of its input [15,1]. Let $G_L(x), G_R(x)$ be the left and right halves of $G(x)$ respectively. Upon removal of a user u_σ, The group center chooses a random number $r_{\sigma^1} \in_R \{0,1\}^k$. For $i = 1, \ldots, \log n$ the group center sets $k_{\sigma^i}^{new}$ to $G_L(r_{\sigma^i})$, sets $r_{\sigma^{i+1}}$ to $G_R(r_{\sigma^i})$ and broadcasts $E_{k_{flip(\sigma^{i-1})}}(r_{\sigma^i})$.

[2] This is minimal by Corollary 3 in the next section.

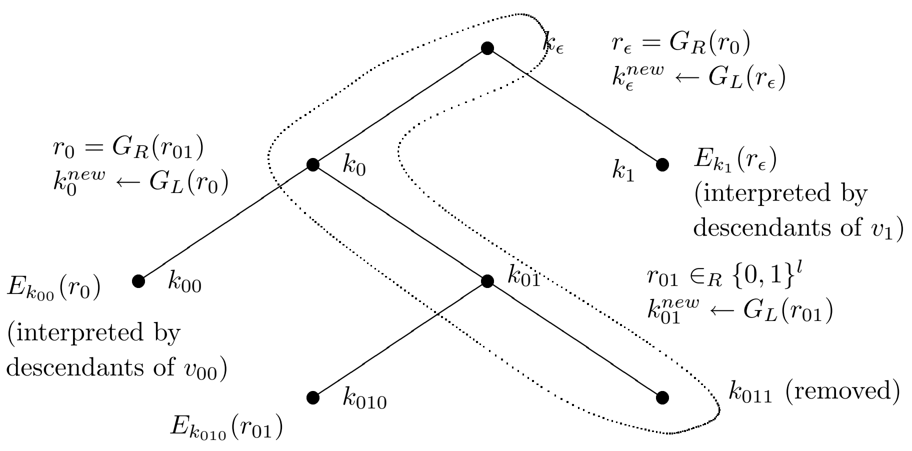

Fig. 1. The basic tree scheme actions when holder of k_{011} is removed. (The figure shows only tree nodes that affected by the removal.)

E.g. if u_{011} is removed (see Figure 1), r_{01} is chosen at random, k_{01}^{new} is set to $G_L(r_{01})$, r_0 is set to $G_R(r_{01})$ and r_{01} is broadcasted encrypted with k_{010}. Then, k_0^{new} is set to $G_L(r_0)$, r_ϵ is set to $G_R(r_0)$ and r_0 is broadcasted encrypted with k_{00}. Finally, the new session key $k_s^{new} = k_\epsilon^{new}$ is set to $G_L(r_\epsilon)$ and r_ϵ is broadcasted encrypted with k_1. Now, every user can compute the changed keys on his root-to-leaf path.

The basic tree scheme parameters appear in Table 1.

	minimal storage scheme	*basic tree scheme*
user storage	2	$\log n + 1$
center storage	2	$2n - 1$
communication	$(n-1)$	$\log n$

Table 1. Parameters of the basic schemes.

3.3 On the Storage Requirements of the Group Center

On first glance, reducing the center storage requirements in the tree scheme may proceed as follows. Instead of having the center keep all keys on the tree, the keys may be generated from a single key, say by applying a pseudo-random function, and the center will keep only this secret key. However, this idea does not seem to work, since when an update occurs, the center will have to change the secret key, requiring changing the entire tree, thus bringing the communication to linear.[3]

[3] Alternatively, the secret key may stay the same, but some counter be changed for every update. However, this is only useful if we require threshold security (requiring

In the next subsection we reduce the center storage to $\frac{n}{\log n}$. Further reducing the center storage, or alternatively proving it impossible, remains an interesting open problem.

3.4 Combined a-ary Tradeoff Scheme

The basic tree shown in the previous paragraph may be naturally generalized from binary trees to a-ary trees. We combine this generalization with the minimal storage scheme to create our tradeoff scheme. There are two parameters of the construction (i) a - the degree of the tree internal nodes, and (ii) m - the size of user subsets to which the minimal storage scheme is applied. The parameters determine the number of keys given to every user and the communication costs for an update operation. Details follow.

Divide the multicast group users to disjoint subsets of size m: $U_1, \ldots, U_{n/m}$, $\cup_{i=1}^{n/m} U_i = M$. The group center constructs an a-ary tree of height $b = \lceil \log_a \lceil \frac{n}{m} \rceil \rceil$ (i.e. the tree has at least n/m leaves). Assign subset U_i with the ith leaf of the tree. As in the basic tree scheme, a random key is assigned with each tree node.

For $m = 1$, $U_i = \{u_i\}$, the scheme is a simple generalization of the basic tree scheme to a-ary trees. For $m > 1$, we combine the basic tree scheme and the minimal storage scheme as follows. Every user $u \in U_i$ is given the b keys assigned to the nodes on the path from the root to the ith leaf. The center holds all these keys, as well as secret keys r_i for each leaf i (r_i's are not known to any user). r_i is used as the seed for the minimal storage scheme between the group center and U_i, namely r_i is used for generating a unique private key for every $u \in U_i$. Whenever a user $u \in U_i$ is removed, the keys on the path from the ith leaf to the root are changed. The center sends to every user in $U_i \setminus \{u\}$ the new key for the ith leaf as in the minimal storage scheme, and then sends the ciphertexts necessary to update the path to the root as in the basic tree scheme.

The security of this scheme follows from the security of the minimal storage scheme and the basic tree scheme (based on the security of the pseudorandom function). The parameters of the scheme appear in Table 2.

	general m, a	*Example 1*	*Example 2*
user storage	$\log_a(\frac{n}{m}) + 1$	$O(\log n)$	2
center storage	$\frac{n}{m} \cdot \frac{a}{a-1}$	$O(\frac{n}{\log n})$	$n^{1/2} + 1$
communication	$m - 1 + (a-1)\log_a(\frac{n}{m})$	$O(\log n)$	$2n^{1/2} - 2$

Table 2. Parameters of the tradeoff scheme. Note that setting $a = m = n$ gives the minimal storage scheme and setting $m = 1, a = 2$ gives the basic tree scheme. In Example 1, $a = 2, m = O(\log n)$, in Example 2, $a = m = n^{1/2}$.

storage which is linear in the size of the coalition). For the strong notion of security against arbitrary coalitions, this would again require linear storage from the center.

Denote the center storage by s_{GC}, the user storage by $b+1$ (i.e. $b = \log_a(\frac{n}{m})$, or equivalently $a = \frac{n}{m}^{1/b}$) and the communication by $c = c(n)$. The tradeoff scheme allows trading center storage and communication costs, subject to the restriction that communication costs are lower bounded as a function of user storage. Specifically:

Theorem 1. *There exist secure multicast encryption protocols such that*

1. $s_{GC} \cdot c = \Theta(n)$.
2. $c = \Theta(bn^{\frac{1}{b}})$.

These bounds follow from the parameters of our scheme in Table 2.

Thus, our scheme is flexible enough to deal with a large range of applications, adjusting the parameters accordingly (see, for exampele, [4,5] for a discussion and two very different benchmark scenarios).

In particular, it follows that using our scheme the center storage may be reduced by a factor of $\log n$ with respect to the storage in [13,14,4]. Further reduction in the center storage is achieved by noticing that the center need not hold an *explicit* representation of keys, instead it can hold a shorter representation from which it is possible to compute the keys efficiently. Consider, for instance, the case where the group center holds a secret key r to a pseudo-random function $f_r : \{0,1\}^l \rightarrow \{0,1\}^l$, and a counter *cnt* which is initially set to zero. Set $m > 2$. When a user in U_i is removed, the center uses $r_i = f_r(cnt)$, stores *cnt* in the leaf corresponding to U_i and advances *cnt*. All the nodes on the path from the ith leaf to the root store a pointer to leaf i. This way, the center may compute any key in the tree via one application of f_r and $O(\log_a \frac{n}{m})$ applications of the pseudo random generator G.

As an example, consider a group with a million users using DES (7-bytes keys). In the basic construction, the needed center memory is $2 \cdot 10^6 \cdot 7 =$ 14Mbytes. Using our construction with a 4-bytes counter reduces the center memory to $2 \cdot 10^6 \cdot 4/20 =$ 400Kbytes, which is small enough to be put in a fast cache memory.

4 Lower Bounds

In this section we describe lower bounds on the amount of storage and the communication complexity per update (both measured in units of l bits, namely the key size), and the relation between the two. We begin by observing simple lower bounds on the user storage and the number of keys in the system.

Lemma 2. *For any secure multicast encryption protocol,* $\forall U \subseteq M\ \exists k \in K$ *such that* $k \in K(U)$ *but* $\forall v \in M \setminus U,\ k \notin K(v)$ *(every subset of users has a key which does not belong to any other user outside the subset).*

Proof. Assume for contradiction that there exists a subset $U \subseteq M$ such that $\forall k \in K(U),\ k \in K(M \setminus U)$. That is, every key held by users in U is also held by some user in $M \setminus U$. It follows that any multicast message which is understood by someone in U is also understood by the coalition $M \setminus U$. Consider the operation *remove*$(M \setminus U)$ (whether done by removing the users one by one, or a more general removal of the whole subset). By the above, there is no way to provide U with a new session key that is not known to the coalition $M \setminus U$, and thus this update operation cannot be performed securely, yielding a contradiction. □

Corollary 3. *For any secure multicast encryption protocol,*

1. *Every user $u \in M$ must hold at least two keys: a unique key k_u known only to u and GC, and the session key k_s.*
2. *The total number of keys in the system is $|K| \geq n+1$.*

We now turn to prove lower bounds regarding the tradeoff between communication and user storage. Consider any given secure multicast encryption protocol. Recall that n denotes the number of users in the multicast group M, and $c(n)$ the denotes the maximal communication complexity required for re-keying following a deletion of a user from the group. We let $b(n)+1$ denote the maximal number of keys, including the session key, held by any user in M (for convenience, we sometimes omit the argument n from the notation of b). We will prove bounds on the relation between $b(n)$ and $c(n)$.

We start with the special case of $b(n) = 1$, namely for a system where each user holds only one key in addition to the session key. This case will be used in the following general theorems.

Lemma 4. *If the maximal number of keys held by each user is $b(n)+1 = 2$, then the re-keying communication costs satisfy $c(n) \geq n-1$.*

Proof. Since each user u holds at most two keys, by Corollary 3 these must be the session key k_s and a unique key k_u known only to u. When a user is removed, the other $n-1$ users must be notified in order to establish the new session key. But since k_s is known to the removed user it cannot be used, forcing the center to use the unique keys k_u for each user who stays in the group, requiring one message per user, for a total of $n-1$ messages. □

The minimal storage scheme presented in Section 3.1 matches the above lower bound.

Theorem 5. *Let $b(n)+1$ be the maximal number of keys held by any user in M. Then, the re-keying communication costs satisfy $c(n) \geq n^{1/b(n)} - 1$.*

Proof. The proof is by induction on b. The base case, $b(n) = 1$, is proved in Lemma 4. For $b(n) > 1$, denote by t_k the number of users holding key k. Denote by k_{max} a key other than the session key, such that $t = t_{k_{max}}$ is maximal.

On one hand, consider the set of t users holding the key k_{max}. By the induction hypothesis there exists a user holding k_{max} whose removal incurs re-keying communication costs at least $t^{\frac{1}{b-1}} - 1$, even if only the t users holding k_{max} are considered. On the other hand, when removing any user, the communication must be $c(n) \geq \frac{n}{t}$, since each message is an encryption under some key k which is understood by at most t users. It follows that the re-keying communication complexity is at least

$$c(n) \geq \max(t^{\frac{1}{b-1}} - 1, \frac{n}{t}) \geq \max(t^{\frac{1}{b-1}}, \frac{n}{t}) - 1 \geq n^{1/b} - 1$$

where the last inequality holds for any $1 \leq t \leq n$. □

For constant b the above bound is tight (upto a constant factor), and agrees with the scheme in Section 3. Otherwise, there is an $O(b)$ (and at most $O(\log n)$) gap between the above lower bound and the upper bound in Section 3.

In the following we consider a class of *structure preserving* re-keying protocols, defined below, that includes our protocol in Section 3 as well as the other known protocols. We show a tight lower bound (matching our upper bound) for this class, which is $c(n) \geq bn^{1/b}$. For the special case $b(n) = 2$ this bound holds even for protocols that are not structure preserving, and we find it useful to prove it in the following lemma. The proof follows the direction of the proof of Theorem 5 above with a more careful analysis.

Lemma 6. *If the maximal number of keys held by each user is $b(n) + 1 = 3$, then the re-keying communication costs satisfy $c(n) \geq 2\sqrt{n} - 2$.*

Proof. Each user u holds at most 3 keys, which by Corollary 3 must include the session key k_s, a unique key k_u, and a possible additional key. As before, let t denote the number of users holding a key k_{max} other than the session key, which is held by the maximal number of users. Consider the operation of removing one of the users holding k_{max}. All other $t - 1$ users holding k_{max} can only receive messages encrypted by their unique key, since the other two keys they are holding, k_{max} and k_s, were known to the removed user. This requires $t - 1$ messages. Since these messages are sent using unique keys, they do not give any information to the $n - t$ users not holding k_{max}, and thus additional messages should be sent to those users, requiring at least $\frac{n-t}{t}$ encryptions. Altogether,

$$c(n) \geq t - 1 + \frac{n-t}{t} = t + \frac{n}{t} - 2 \geq 2\sqrt{n} - 2$$

where the last inequality holds for any $1 \leq t \leq n$. □

An instance of tradeoff scheme (Example 2 in Table 2) matches the above lower bound.

Definition 7. *A protocol is* structure preserving *if $\forall U \subseteq M$ and $\forall v, v' \in M$ $(v \neq v')$, if there exists $k \in K$ such that $\forall u \in U$, $k \in K(u)$ but $k \notin K(v)$, then after the operation remove(v') there exists $k' \in K^{new}$ such that $\forall u \in U \setminus v'$, $k' \in K^{new}(u)$ but $k' \notin K^{new}(v)$.*

Intuitively, structure preserving protocols are those that maintain the property of "the set U has advantage over the user v" across updates, for any subset U and user v. That is, if there is a set of users U all sharing a key k, and a user v which does not have this key, then after removing another user v' (whether $v' \in U$ or not), the users U still holds some key k' that v does not hold.

Theorem 8. *For structure preserving protocols, the re-keying communication costs satisfy $c(n) \geq bn^{1/b} - b$, where $b+1$ denotes the maximal number of keys held by any user in M.*

Proof. The proof is by induction on b (using a stronger induction hypothesis described below). The base case $b = 1$ follows from Lemma 4. We have also proved the case $b = 2$ in the proof of Lemma 6, and in fact we use here the same idea as in the proof of Lemma 6. However, the difference is that for $b = 2$, the messages sent to the $t-1$ users holding k_{max} cannot be interpreted by anyone who does not hold k_max (since they are sent using unique keys), and thus they can be simply added to the messages sent to the users that do not hold k_max. In contrast, for $b > 2$, this is not necessarily true: some keys can be shared both by users holding k_{max} and users that do not hold k_max. Here we use the fact that the protocols is structure preserving and count the $t-1$ messages needed to update k_max which cannot be interpreted by users that do not hold k_max. Details follow.

We start by describing a process for selecting a user to be removed: we choose a maximal subset holding some key, then choose a maximal subset of this subset holding another key, and so on, going to smaller and smaller subset until we reach a single user. More formally, denote by $k^{b+1}_{max} = k_s$ (the session key), and $U^{b+1}_{max} = M$ (the entire multicast group). For $i = b, b-1, \ldots, 1$ let $k^i_{max} \notin \{k^{i+1}_{max} \cdots, k^{b+1}_{max}\}$ be a key that is held by a maximal number of users. Let U^i_{max} be the set of users holding k^i_{max}. At the end of the process $U^1_{max} = \{u\}$ is a singleton, since k^1_{max} is the unique key of a user u. Select to remove u.

Lemma 9. *When removing a user according to the selection process described above, the communication re-keying costs satisfy $c(n) \geq t_2 + \frac{t_3}{t_2} + \cdots + \frac{t_{b+1}}{t_b} - b$, where $t_i = |U^i_{max}|$ (in particular, $t_{b+1} = n$).*

We prove the claim by induction on b. For $b = 1$ we simply need to prove $c(n) \geq t_2 - 1$ where $t_2 = n$, which follows from Lemma 4. For $b \geq 2$, let u be the user to be removed according to the selection process above. Consider the set U^b_{max}, which is a maximal-size set of users holding a key $k^b_{max} \neq k_s$. Since the protocol is structure preserving, after removing u there should be a key k' which is held by every user in $U^b_{max} \setminus \{u\}$, but not by any other user. Because of the way u was chosen, if $|U^b_{max} \setminus \{u\}| = t_b - 1 > 1$ then no such key k' unknown to u exists before the update, because otherwise the next maximal subset would be chosen as $U^{b-1}_{max} = U^b_{max} \setminus \{u\}$, and u would not be selected. Therefore, the center needs to send messages to generate this key. By the induction hypothesis, this requires communication of at least $t_2 + \frac{t_3}{t_2} + \cdots + \frac{t_b}{t_{b-1}} - (b-1)$, which cannot be interpreted by any user outside of U^b_{max}. Adding to it the communication costs for these outside users (in order to establish a new session key), sums up to

$$c(n) \geq t_2 + \frac{t_3}{t_2} + \cdots + \frac{t_b}{t_{b-1}} - (b-1) + \frac{n - t_b}{t_b} = t_2 + \frac{t_3}{t_2} + \cdots + \frac{n}{t_b} - b$$

as needed.

The only cases which we did not handle are those where $U^b_{max} \setminus \{u\}$ is small (empty or a singleton). If $U^b_{max} \setminus \{u\} = \phi$, by the maximality of U^b_{max}, each user holds only the session key and a unique key, and the bound of Lemma 4 can be applied. If it a singleton, any key other than the session key is held by at most two users, which implies that a message sent to the user in $U^b_{max} \setminus \{u\}$ (in order to update the session key) is encrypted by the unique key and cannot be interpreted by other users, thus the same calculation as above holds.

Thus, we have proved the claim. The theorem follows by observing that

$$t_2 + \frac{t_3}{t_2} + \cdots + \frac{n}{t_b} \geq bn^{1/b}$$

which can be proven by induction on b. Thus, $c(n) \geq bn^{1/b} - b$, and the proof is complete. □

Acknowledgments

We thank Moni Naor for pointing out the improvement using counters described at the end of Section 3.4.

References

1. M. Blum and S. Micali, How to generate cryptographically strong sequences of pseudorandom bits, SIAM J. Comput. **13** (1984), no. 4, 850–864.
2. C. Blundo, L. A. Frota Mattos and D. R. Stinson, Trade-offs between communication and storage in unconditionally secure schemes for broadcast encryption and interactive key distribution, in *Advances in cryptology—CRYPTO '96 (Santa Barbara, CA)*, 387–400, Lecture Notes in Comput. Sci., 1109, Springer, Berlin.
3. C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung, Perfectly secure key distribution in dynamic conferences, in *Advances in cryptology—CRYPTO '92*, 471–486, Lecture Notes in Comput. Sci., 740, Springer, Berlin.
4. R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas, Multicast Security: A Taxonomy and Efficient Authentication, Infocomm 1999.
5. R. Canetti and B. Pinkas, A Taxonomy of Multicast Security Issues, Internet draft <draft-canetti-secure-multicast-taxonomy-00.txt>, ftp://ftp.ietf.org/internet-drafts/draft-canetti-secure-multicast-taxonomy-00.txt.
6. A. Fiat and M. Naor, Broadcast Encryption, in *Advances in cryptology—CRYPTO '93 (Santa Barbara, CA)*, 480–491, Lecture Notes in Comput. Sci., 773, Springer, Berlin.
7. O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *JACM*, Vol. 33, No. 4, pages 792–807, 1986.
8. S. Goldwasser and S. Micali, Probabilistic encryption, J. Comput. System Sci. **28** (1984), no. 2, 270–299.
9. M. Luby and J. Staddon, Combinatorial Bounds for Broadcast Encryption, in K. Nyberg, editor, *Advances in Cryptology—EUROCRYPT '98 (Espoo, Finland)*, 512-526, Lecture Notes In Comput. Sci., 1403, Springer, Berlin.
10. McGrew D. A., and Sherman A. T., Key Establishment in Large Dynamic Groups using One-way Function Trees. Manuscript, 1998.

11. D.R. Stinson, On some methods for unconditionally secure key distribution and broadcast encryption, to appear in Designs, Codes and Cryptography.
12. D.R. Stinson and T. van Trung, Some new results on key distribution patterns and broadcast encryption, to appear in Designs, Codes and Cryptography.
13. D. M. Wallner, E. J. Harder and R. C. Agee, Key Management for Multicast: Issues and Architectures, Internet draft <draft-wallner-key-arch-01.txt>, ftp://ftp.ietf.org/internet-drafts/draft-wallner-key-arch-01.txt.
14. C. K. Wong, M. Gouda and S. S. Lam, "Secure Group Communication Using Key Graphs", SIGCOMM '98. Also, University of Texas at Austin, Computer Science Technical report TR 97-23.
15. A. C. Yao, Theory and applications of trapdoor functions, in *23rd annual symposium on foundations of computer science (Chicago, Ill., 1982)*, 80–91, IEEE, New York.