Multicast Security: A Taxonomy and Efficient Constructions

Ran Canetti* Juan Garay† Gene Itkis‡ Daniele Micciancio§ Moni Naor¶

Benny Pinkas ‖

Abstract

Multicast communication is becoming the basis for a growing number of applications. It is therefore critical to provide founded security mechanisms for multicast communication. Yet, existing security protocols for multicast offer only very partial solutions.

We first present a taxonomy of multicast scenarios on the Internet and point out the relevant security concerns. Next we identify two major security problems of multicast communication: *individual authentication*, and *key revocation*.

Maintaining authenticity in multicast protocols is a much more complex problem than for unicast; in particular, known solutions are prohibitively inefficient in many cases. We present a solution that is reasonable for a range of scenarios. Our approach can be regarded as a 'midpoint' between traditional Message Authentication Codes and digital signatures. We also present an improved and very efficient solution to another prevailing problem for multicast protocols, namely the key revocation problem.

*IBM T.J. Watson research center. Email: canetti@watson.ibm.com

†Lucent Bell-Labs. Email: garay@research.bell-labs.com

‡News Data Systems. Email: gitkis@ndc.co.il

§Laboratory of computer science, MIT. Email:miccianc@theory.lcs.mit.edu

¶Dept. of AM and CS, Weizmann Institute of Science. Email: naor@wisdom.weizmann.ac.il

‖Dept. of AM and CS, Weizmann Institute of Science. Email: bennyp@wisdom.weizmann.ac.il

1 Introduction

The popularity of multicast has grown considerably with the wide use of the Internet. Examples include Internet video transmissions, stock quotes, news feeds, software updates, live multi-party conferencing, on-line video games and shared white-boards. Yet, security threats on the Internet have flourished as well. Thus the need for secure *and efficient* multicast protocols is acute.

Multicast security concerns are considerably more involved than those regarding point-to-point communication. Even dealing with the 'standard' issues of message authentication and secrecy becomes much more complex; in addition other concerns arise, such as access control, trust in group centers, trust in routers, dynamic group membership, and others.

A trivial solution for secure multicast is to set up a secure point-to-point connection between every two participants (say, using the IP-Sec protocol suite [22]). But this solution is prohibitively inefficient in most multicast scenarios[1]. Furthermore it obviates the use of multicast routing. Instead, we are looking for solutions that mesh well with current multicast routing protocols, and that have as small overhead as possible. In particular, a realistic solution must maintain the current way by which *data packets* are being routed; yet additional control messages can be introduced, for key exchange and access control.

1.1 This work

First, we offer a taxonomy of multicast security concerns and scenarios, with a strong emphasis on IP multicast[2]. It soon becomes clear that the scenarios are so diverse that there is little hope for a unified security solution that accommodates all scenarios. Yet we suggest two 'benchmark' scenarios that, besides being important on their own, have the property that solutions for these scenarios may be a good basis for other scenarios. In a nutshell, one scenario involves a single sender (say, an on-line stock-quotes distributor) and a large number of recipients (say, hundreds of thousands). The second scenario is on-line virtual conferencing involving up to few hundreds of participants, where many (or all) of the participants may be sending data to the group. We elaborate within.

Next we concentrate on a problem that emerges as a serious bottleneck in multicast security: individual sender and message authentication. Known attempts to solve multicast security problems (e.g., [20, 21, 28, 3, 33, 34, 27]) concentrate on the task of sharing a *single key* among the multicast group members. These solutions are adequate for encrypting messages so that only group members can decrypt. However, the single shared key approach is inadequate for authentication, since a key shared among *all* members cannot be used to differentiate among senders in the group. In fact, the only known solutions for multicast authentication involve heavy use of public key signatures — and these involve considerable overhead, especially in the work needed to generate signatures. (Here *stream signatures* [17] could reduce much of the overhead; however their techniques do not tolerate packet loss; see more below).

We present solutions to the individual authentication problem based on shared key mechanisms (namely, Message Authentication Codes — MACs). Yet, we let each member have a *different* set of keys. We first present a basic scheme and then gradually improve it to a scheme that outperforms public-key signatures in several common scenarios, where our main savings are in the time to *generate* signatures.

[1] We remark that although there are solutions to point-to-point keying between any pair among n users, which require only $(O(n))$ keys [19] and no public key cryptography, they are still not scalable.

[2] See also [18] for an earlier, more basic discussion of secure multicast issues.

The basic individual authentication scheme for a single sender draws from ideas of [2, 13]: the sender holds a set of ℓ keys and attaches to each packet ℓ MACs – each MAC computed with a different key. Each recipient holds a subset of the ℓ keys and verifies the MAC according to the keys it holds. Appropriate choice of subsets insures that with high probability no coalition of up to w colluding bad members (where w is a parameter) know all the keys held by a good member, thus authenticity is maintained. We present several enhancements for the scalability of the authentication scheme:

- A considerable gain in the computational overhead of the authentication scheme is achieved by noticing that the work needed for computing some known MAC functions on the same input and ℓ different keys is far less than the ℓ times the work to compute a single MAC. This is so since the message can first be hashed to a short string using key-less collision-resistant hashing.

- A second improvement enables the use of MACs with a single bit output. Such MAC functions can be very efficient to compute, and the communication overhead of the MAC outputs is also greatly reduced.

  We show that using similar parameters to those of the basic scheme, one can guarantee that each good member has *many* keys that are known only to itself and to the sender. In order to break the scheme an adversary has to forge *all* the MACs computed with these keys. Thus it is enough that the sender attaches to the message only a *single bit* out of each generated MAC (as long as this bit cannot be successfully 'predicted' without knowing the key – see elaboration within). Consequently, the total length of the tag attached to the message is now only ℓ *bits*.

- A third improvement notices that a very similar method allows for *many senders* to use the same set of keys — each sender will hold a different subset of keys, making sure that with high probability each sender-recipient pair shares a sufficient number of keys that are known to no (small enough) bad coalition.

- It is further possible to increase security by making sure that no coalition of *senders* can forge messages, only large coalitions of recipients can. This property is beneficial when the recipients are relatively trusted (say, these are network routers). It is achieved by differentiating between *primary* and *secondary* keys. A sender only receives secondary keys, while primary keys are only held by the recipients. Each secondary key is derived by applying a pseudorandom function (e.g., a block cipher or keyed hash), keyed by the corresponding primary key, to the sender's public identity. Each recipient can now compute the relevant secondary keys and verify the MACs; yet, no coalition of senders knows even a single key other than its legitimate set of keys. Similar techniques constitute an alternative method for achieving the third improvement.

We also consider the security problem of *membership revocation*. When membership in the multicast group changes it might be required to change the group key. When a new user joins the multicast group the group key might have to be re-keyed in order to prevent the joining member from understanding previous group communication. This re-keying can be achieved with negligible communication overhead by simply sending the the new key to the old group members encrypted with the previous group key. The problem is much more complicated when a user should be removed from the group. The group key should be changed but since the excluded member knows the value of this key it cannot be used to encrypt the new key. We present a considerable improvement to current solutions for group re-keying, in both efficiency and security.

Organization: In section 2 we list and discuss multicast security issues, in several common scenarios[3]. In Section 3 we present our multicast authentication schemes, and in Section 4 we present our improvements over past mechanisms for membership revocation.

2 Multicast Security Issues

We overview salient characteristics of multicast scenarios, and discuss the relevant security concerns. The various scenarios and concerns are quite diverse in character (sometimes they are even contradictory). Thus it seems unlikely that a single solution will be satisfactory for all multicast scenarios. This situation leads us to suggest two benchmark scenarios for developing secure multicast solutions.

Multicast group characteristics

We list salient parameters that characterize multicast groups. These parameters affect in a crucial way which security architecture should be used. The **group size** can vary from several tens of participants in small discussion groups, through thousands in virtual conferences and classes, and up to several millions in large broadcasts. **Member characteristics** include computing power (do all members have similar computing power or can some members be loaded more than others?) and attention (are members on-line at all times?).

A related parameter is **membership dynamics:** Is the group membership static and known in advance? Otherwise, do members only join, or do members also leave? how frequently does membership change and how fast should changes be updated? Also, is there a **membership control** center that has information about group membership? Finally what is the expected **life time** of the group? several minutes? days? unbounded?

Another parameter is the **number and type of senders:** Is there a single party that sends data? several such parties? all parties? Is the identity of the senders known in advance? Are *non-members* expected to send data?

Another parameter is the **volume and type of traffic:** Is there heavy volume of communication? Must the communication arrive in real-time? what is the allowed latency? For instance, is it data communication (less stringent real-time requirements, low volume)? audio (must be real-time, low volume)? video (real-time, high volume)? Also, is the traffic bursty?

Another parameter that may become relevant is the routing algorithm used. For instance, a security mechanism may interact differently with dense-mode and sparse-mode routing. Also, is all routing done via a single server or is it distributed?

Security requirements and trust issues

The most basic security requirements are **secrecy** and **authenticity.** Secrecy usually means that only the multicast group members (and all of them) should be able to decipher transmitted data. Authenticity may take two flavors: Group authenticity means that each group member can recognize whether a message was sent by a group member. Individual authenticity means that it is possible to

[3]This survey is also the basis for a recent internet-draft [10].

identify the particular sender within the group. It may also be desirable to be able to verify the origin of messages even if the originator is not a group member.

Other concerns include several flavors of **anonymity** (e.g., keeping the identity of group members secret from outsiders or from other group members, or keeping the identity of the sender of a message secret). A related concern is protection from traffic analysis. A somewhat contradictory requirement is **non-repudiation,** or the ability of receivers of data to prove to third parties that the data has been transmitted.

An additional concern is **access control.** It is often desirable to control the group membership, and sometimes also keep track on the amount of usage of each member (say, for billing purposes). Here the problem becomes more complex if members may join with time, and even more complex if members may leave the group (and then the group has to make sure that the leaving members lose the cryptographic abilities reserved to members).

Lastly, maintaining **service availability** is ever more relevant in a multicast setting, since clogging attacks are easier to mount and are much more harmful. Here protection must include multicast routers as well as end-hosts.

Trust issues. In simple scenarios there is a natural *group owner* that can be trusted to manage the group security. Typical roles are access control, logging traffic and usage, and key management. (It may be convenient, but not necessary, to identify the group owner with the *core* used in some multicast routing protocols, e.g. in [3]) In other cases no single entity is totally trusted; yet different entities can be trusted to perform different tasks (for instance, the access-control entity may be different than the entity that distributes keys). In addition, basing the security of the entire group on a single service makes the system more vulnerable. Thus, it is beneficial, in general, to distribute the security tasks as much as possible.

A natural approach for distributing trust in multicast security centers is to use techniques of *threshold cryptography* [11, 16] and *proactive security* [9] for replacing a single center with a distributed service with no single point of failure. This is an interesting topic for future research.

Performance

Performance is a major concern for multicast security applications. The most immediate costs that should be minimized are the latency and work overhead per sending and receiving data packets, and the bandwidth overhead incurred by inflating the data packets via cryptographic transformations. Secure memory requirement (e.g., lengths of keys) is a somewhat less important resource, but should also be minimized. Here distinction should be made between the load on strong server machines and on weak end-users.

Other performance overheads to be minimized include the group management activity such as group initialization and member addition and deletion. Here member deletion may cause severe overhead since keys must be changed in order to ensure revocation of the cryptographic abilities of the deleted members. We elaborate in Section 4.

An additional concern is possible congestion, especially around centralized control services at peak sign-on and sign-off times. (A quintessential scenario is a real-time broadcast where many people join right before the broadcast begin and leave right after it ends.)

Another performance concern is the work incurred when a group member becomes active after being dormant (say, off-line) for a while.

2.1 Benchmark Scenarios

As seen above, it takes many parameters to characterize a multicast security scenario, and a large number of potential scenarios exist. Each scenario calls for a different solution; in fact, the scenarios are so different that it seems unlikely that a single solution will accommodate all. This is in sharp contrast with the case of unicast security, where a single architectural approach — public-key based exchange of a key, followed by authenticating and encrypting each packet using keys derived from this key — is sufficient for most scenarios.

In this section we present two very different scenarios for secure multicast, and sketch possible solutions and challenges. These scenarios seem to be the ones that require most urgent solutions; in addition, they span a large fraction of the concerns described above, and solutions here may well be useful in other scenarios as well. Thus we suggest these scenarios as benchmarks for evaluating security solutions.

Single source broadcast

Consider a single source that wishes to continuously broadcast data to a large number of recipients. The source can be a news agency that broadcasts stock-quotes and news-feeds to paying customers. Such applications are common in the Internet today but they still typically rely on unicast routing and have few or no security protections.

Here the number of recipients can be hundreds of thousands or even millions. The source is typically a top-end machine with ample resources. It can also be parallelized or even split to several sources in different locations. The recipients are typically lower-end machines with limited resources. Consequently, and security solution must optimize for efficiency at the recipient side.

Although the life-time of the group is usually very long, the group membership is dynamic: members join and leave at a relatively high rate. In addition, at peak times (say, before and after important broadcasts) a high volume of sign-on/sign-off requests are expected.

The volume of transmitted data may change considerably: if only text is being transmitted then the volume is relatively low (and the latency requirements are quite relaxed); if audio/video is transmitted (say, in on-line pay-TV) then the volume can be very high and very little latency is allowed.

Authenticity of the transmitted data is a crucial concern and should be strictly maintained: a client must never accept a forged stock-quote as authentic. Another important concern is preventing non-members from using the service. This can be achieved by encrypting the data; yet the encryption may be weak since there is no real secrecy requirement, only prevention from easy unauthorized use. Regarding trust, here there is a natural group owner that manages access-control as well as key management. However, the sender of data may be a different entity (say, Yahoo broadcasting Reuters stock-quotes in its home-page).

A natural solution for this scenario may have a group management center that handles access control and key management. (To scale the solution to larger numbers of recipients the center can be distributed, or a hierarchal structure can be introduced.) It is stressed that the center handles only 'control traffic'. The data packets are routed using current multicast routing protocols. Encryption can be done using a single key shared by all members. Yet, two main cryptographic problems remain: How to authenticate messages, and how to make sure that a leaving member loses its ability to decrypt.

Clearly, a single key shared by all members cannot be used to authenticate the source. In the next section we describe techniques for authenticating individual sources using traditional MACs with a set of shared keys.

Member revocation, or making sure that a leaving member loses its ability to decrypt, inevitably involve choosing a new encryption key and distributing it to all remaining members. In Section 4 we describe current techniques and improve on them somewhat.

Virtual Conferences

Typical virtual conference scenarios include on-line meetings of corporate executives or committees, town-hall type meetings, interactive lectures and classes, and multiparty video games. A virtual conference involves several tens to hundreds of peers, often with roughly similar computational resources. Usually most, or all, group members may a-priori wish to transmit data (although often there is a small set of members that generate most of the bandwidth).

The group is often formed per event and is relatively short-lived. Membership is usually static: members usually join at start-up, and remain signed on throughout. Furthermore, even if a member leaves it is often not crucial to cryptographically disconnect it from the group. Bandwidth and latency requirements vary from application to application, similarly to the case of single source broadcast.

Authenticity of data *and sender* is the most crucial security concern. In some scenarios maintaining secrecy of data and anonymity of members may be crucial as well; in many other scenarios secrecy of data is not a concern at all. Although there is often a natural group owner that may serve as a trusted center, it beneficial to distribute trust as much as possible.

Also here a simple approach to a solution uses a server that handles access control and key management. Encryption, when needed, can be dealt with as above. Yet here the performance requirements from the authentication mechanism are very different. In particular, in contrast with the single sender scenario, here signing data packets may be prohibitively slow on the sender's machine. In addition, there are far less receivers, and the group members may be somewhat more trustworthy. Also virtual conferencing applications are typically more tolerant to occasional and local authentication errors. These considerations point to an alternative approach to solving the multicast authentication problem. In the next section we describe this alternative approach.

3 Efficient Authentication Schemes

There are three approaches to authentication: information theoretic authentication, public key signatures, and MACs. Attempts at designing information theoretically secure (also called unconditionally secure) authentication schemes for large groups of receivers [12, 31] are inherently inefficient. Some of the suggested schemes could only be used for a single authentication, and they all have storage (distribution) requirements which are in the same order as the total number of messages that can be authenticated. Another information-theoretic scheme, of Blundo et al [7], can be adapted for authentication, but requires storage which is exponential in the size of the multicast group; therefore it cannot be applied for large groups.

Signatures: Public key signatures can be used for authentication. Such signatures are typically long and require a significant computational overhead for computing the signature as well as for verifying it. Their application for authenticating streams of data was investigated in [17] who proposed a chaining mechanism that requires a single signature per stream. The constructions of signature based authentication schemes for streams do not tolerate packet loss, and are thus incompatible with IP multicast. One solution for that problem is to use costly re-transmission mechanisms. An alternative approach which we propose is the following: consider the stream where for each packet its one-way hash is concatenated. Then use an *erasure-correction code* to protect these hashes (which are the values signed). The "Priority Encoding Transmission" (PET) error-correction scheme [1] is especially appropriate for this task since it can efficiently provide different error-correction capabilities for different parts of the transmitted message with a minimal bandwidth overhead. It is therefore possible to efficiently ensure that the parts of the message which are critical for authentication have good error-recovery capabilities. Therefore if sufficiently many packets arrive then it is possible to verify the authenticity of each additional packet that arrives (even if out of order).

A different proposal for dealing with packet loss was very recently proposed in [35] where other chaining schemes were considered. A disadvantage of their of there proposal is that a large number of bits should be added to each packet.

As for the signature scheme used, [17] considered RSA signatures and [35] examined the the Fiat-Shamir signature scheme [15] (which has fast signature generation time). We remark that elliptic curves based signature schemes [23] are also good candidates for use in authentication schemes for streams, due to their short signature length.

For the remainder we discuss authentication methods based on MACs. We present new individual authentication methods which are more efficient (especially in the time to generate signatures) than public key based authentication. We first present a description of a basic scheme, followed by several major improvements (see a sketch of these improvements in the Introduction).

We describe authentication schemes based on MACs. A MAC is a function which takes a secret key k and a message M and returns a value $\mathrm{MAC}(k, M)$. Very informally, it has the property that an adversary that sees a sequence $\{M_i, \mathrm{MAC}(k, M_i)\}$ but does not know k, has a negligible probability to generate $\mathrm{MAC}(k, M)$ for *any* $m \notin \{M_i\}$.[4] MACs are much more efficient than signatures.

The schemes use a set of keys which are known to the source of the transmissions, and supply users with subsets of these keys in a way which limits the probability that a coalition of users knows all the keys of another user. The source computes for each packet MACs with each of its keys, and attaches these MACs to the transmitted packet. Each receiver verifies the MACs which correspond to the keys in its possession. A corrupt coalition of users cannot authenticate a message to another user without knowing all the keys in its subset or breaking the MAC.

We consider the following salient resources consumed by an authentication scheme, which we analyze for all the schemes we present: The *running time* required to authenticate a message and to verify an authentication, T_S and T_V, respectively. The *length of the keys* that the authenticator and the verifier should store, M_S and M_V, respectively. The *length of the authentication message* (the MAC or the signature), C.

The security of MAC schemes

We distinguish between two types of attacks against a MAC scheme. One is a complete break, where the attacker can authenticate any message of its choice (say, a key recovery attack). The other attack allows the attacker to *randomly* authenticate false messages; here the attacker can authenticate a given message with some fixed and small probability. Our schemes do not allow complete break with higher probability than the underlying MAC scheme. Yet, we do allow for random authentication errors with non-negligible probability (say, 2^{-10} to 2^{-20}).

A bit more formally, we say that a MAC scheme is *q-secure per message* if no (probabilistic

[4] We do not specify the type of queries that the adversary can use. It might be able to see MACs with k of messages chosen by other parties, or might be able to choose itself the messages M' for which it would see $\mathrm{MAC}(k, M')$. The properties of whatever definition that is chosen propagate to our construction of an authentication scheme.

polynomial-time) adversary has a positive expected payoff in the following guessing game: the adversary can ask to receive the output of the MAC on a sequence of messages $m_1, \ldots, m_k$ of its choice and then the adversary can decide to quit or to gamble. If it quits it receives a payment of \$0 and if it gambles the adversary should choose a message $m \notin \{m_1, \ldots, m_k\}$ and guess the value of the MAC on this message. The adversary receives $\$(1-q)$ if it is correct, and pays $\$q$ if it incorrect. In other words the adversary may guess correctly the value of the MAC with probability of at most q, but (except with negligible probability) would not "know" that it has guessed correctly.

We believe that for most (but not all) systems a q-security per message where q is small but not negligible is sufficient (e.g. $q = 2^{-20}$). Note that the definition of "q-secure per message" assures that the probability of a complete break is exponentially small. We call q the *authentication security per message*.

3.1 The Basic Authentication Scheme for a Single Source

The basic scheme proceeds as follows:

- Denote by S the source of the transmissions. S knows a set of ℓ keys, $R = \langle K_1, \ldots, K_\ell \rangle$.
- Each of the recipients knows a subset of this set of keys: recipient u knows the subset $R_u \subset R$.
- When S sends a message M it authenticates it with each of the keys, using a MAC. That is, a message M is accompanied with $\langle \mathrm{MAC}(K_1, M), \mathrm{MAC}(K_2, M), \ldots, \mathrm{MAC}(K_\ell, M) \rangle$.
- Each recipient u verifies all the MACs which were created using the keys in its subset R_u. If any of these MACs is incorrect then u reject the message.

ANALYSIS: Assume that an adversary knows the keys of a coalition of w corrupt users $C = \langle u_1, u_2, \ldots, u_w \rangle$, and tries to send a message M' (which was not sent before) and authenticate it as a message sent by S. For every recipient u for which $R_u \not\subseteq \cup_{i=1}^{w} R_{u_i}$, the probability with which u accepts the message is at most the probability of breaking a single MAC.

Let the total number of keys be $\ell = ew\ln(1/q)$, and for every $\langle$recipient,key$\rangle$ pair let the probability that the key is included in the subset of the recipient be $1/(w+1)$, independently of other pairs (it is therefore enough to use an $(w+1)$-wise independent mapping of users to subsets). Then it is straightforward to verify that for every user u and any coalition of w users, the probability that R_u is completely covered by the subsets of the coalition is at most q. The overhead of this scheme is described in theorem 1.

Theorem 1 *Consider the above scheme using $\ell = ew\ln(1/q)$ MACs with different keys, and assume that the probability of computing the output of a MAC without knowing the key is at most q'. Let u be a user. Then the probability that a coalition of w corrupt users can authenticate a message M to u is at most $q + q'$ (the probability is taken over the choice of key subsets and over the message).*

The lengths of the keys are expected to be $M_S = \ell = ew\ln(1/q)$ basic MAC keys for the source, and $M_V = e\frac{w}{w+1}\ln(1/q)$ MAC keys for the receiver. The communication overhead per message is $C = ew\ln(1/q)$ MAC outputs. The running time overhead is $T_S = w\ln(1/q)$ MAC computations for the server and only about $T_V = e\ln(1/q)$ MAC computations for the receiver.

Proof: Assume that the probability that the coalition can authenticate a message is more than $q + q'$. This probability is at most q (coalition keys cover all the user keys), plus $(1-q)$ times the probability of generating $MAC(k, M)$ without knowing k. Therefore the probability of generating $MAC(k, M)$ without knowing k must be greater than q'. A contradiction. $\square$

A nice feature of this construction is that the complexity does not depend on the total number of parties but rather only on the maximum size of a corrupt coalition and the allowed error probability. We remark that a similar idea was previously used by Fiat and Naor for broadcast encryption (described in [2]) and by Dyer et al. for pairwise encryption [13].

The security is against coalitions of up to a certain number of corrupt recipients. It is possible to construct schemes which are secure against any coalition, but are less efficient[5].

The key allocation only guarantees that the union of the keys of a coalition does not cover the keys of a user. The security of the scheme depends on the security of the MAC, since an insecure MAC enables an adversary to generate correct MAC outputs even without knowing the required keys. Stating the result of the theorem more clearly, the scheme guarantees that the event that all the keys of a user are known to a corrupt coalition happens with probability at most q. In this case the coalition can authenticate any message to the user. (It is therefore preferable that the mapping between users and keys is kept secret to prevent the corrupt coalition from identifying to which users it can authenticate false messages). Now consider a user whose keys are not covered by the coalition: messages to this user can be authenticated by the coalition with probability at most q'. Even more, in this case the coalition cannot check in advance (off-line) whether it can authenticate a specific message. Therefore q', the authentication security per message, can be rather large (e.g. even $q' = 2^{-10}$ might be reasonable for many applications).

IMPROVED SECURITY: A disadvantage of this type of security with a moderate value of q is that the probability that there exists a user which a coalition can *always* fool (because it knows all of its keys) might be non-negligible. It is desirable to reduce the probability of such events. A simple approach that achieves this property is to use c times as many keys as in the previous schemes (where c is a small constant), and require the sender of a message to authenticate it with a random subset of $1/c$ of its keys (the subset should be chosen by a publicly known $(w+1)$-wise independent hash function of the time of day). The number of keys held by each party is c times as large but the communication, generation, and verification overheads are as before. The choice of the parameter c defines a parameter d such that the following statement holds for every user and every coalition of up to w corrupt users: *with probability at most q^d the user has less than $\log_c(1/q)$ keys which are not covered by the coalition.* Then at each time and for each receiver there is a probability q that the coalition does not cover all the keys which are currently in use by the receiver.

3.2 Smaller Communication Overhead

We now describe a scheme with a lower communication overhead. The idea behind it is that at a similar cost to that of the basic scheme it is possible to ensure that the coalition does not know $\log(1/q)$ keys of the user. Therefore each key can be used to produce a MAC with a *single bit output* and the communication overhead is improved. The coalition would have to guess $\log(1/q)$ bits to create a false authentication and its probability of success is as before.

The basic scheme limited the success probability of a corrupt coalition to be $q + q'$, where q' is the authentication security per message. It is thus most reasonable to use $q' = q$. The MAC output must be at least $\log(1/q')$ bits long, which makes the communication overhead to at least $C = dew\log^2(1/q)$ *bits* where $d = \ln 2$. The improved scheme achieves a communication overhead of only $4ew\ln(1/q)$ *bits*.

[5] Here is a construction in which no coalition knows the key of any user: Straightforward probabilistic arguments show that there is a system for n recipients in which the subset of no user is covered by the union of the subsets of a coalition of size w. The system has a total of $O(w^2\ln(n))$ keys, where each recipient has a subset of expected size $O(w\ln(n))$ (this corresponds to $q = (n \cdot \binom{n}{w})^{-1}$.

The suggested scheme uses a MAC with a single bit long output. Current constructions of MACs have much larger outputs. A recent work [29] shows that it is possible to generate a single bit pseudo-random function from *any* MAC (or unpredictable function). It might also be possible to design a designated MAC function with a single bit output. For the simplicity of the exposition assume that for this MAC $q' = 1/2$. If the keys of a corrupt coalition do not cover $\log(1/q)$ keys of a user's subset then the probability of the coalition succeeding in sending false messages to this user is at most[6] q. In the suggested scheme the source uses $\ell = 4ew\ln(1/q)$ keys where each key is chosen to a user's subset with probability $1/(w+1)$. The following result is achieved:

Theorem 2 *Consider a MAC with a single bit output for which the probability of generating* $\mathrm{MAC}(k, M)$ *without knowing the key* k *is* $1/2$, *and consider the above scheme using this MAC with* $\ell = 4ew\ln(1/q)$ *keys. Let* u *be a user. Then the probability that a coalition of* w *corrupt users can authenticate a message* M *to* u *is at most* $2q$ *(the probability is taken over the choice of the coalition, the user, and the message).*

The length M_S (M_V) *of the key of the source (a user) is* $\ell = 4ew\ln(1/q)$ *(expected to be* $M_V = 4e\frac{w}{w+1}\ln(1/q)$*) basic keys of a MAC with a single bit output. The communication overhead per message is* $C = 4ew\ln(1/q)$ **bits**. *The source should compute* $4ew\ln(1/q)$ *MACs, whereas each receiver should compute only about* $4e\ln(1/q)$ *MACs.*

Proof (sketch): When keys are selected to the subsets with probability $1/(w+1)$ the probability that a specific key is contained in a user's subset and is not covered by any of the w members of the coalition is $1/(ew)$. Therefore the expected success probability of a corrupt coalition is $\sum_{i=0}^{\ell}\binom{\ell}{i}(ew)^{-i}(1-1/(ew))^{\ell-i}2^{-i} = (1-1/(2ew))^{\ell} \approx \exp(-\ell/(2ew))$. If we require this expected probability to be at most q^2 then (following from the Markov inequality) with probability at most q it happens that a coalition has a probability greater than q to succeed in sending corrupt shares to a recipient. $\square$

3.3 Multiple Sources

The schemes we presented can be readily extended to schemes which enable any party to send authenticated messages. In the extended schemes there is a *global* set of ℓ keys and every party knows a random subset R_u of these keys. When a party u sends a message it authenticates it with all the keys in R_u. Every receiving party v verifies the authentications that were performed with the keys in the intersection between its subset and the subset of the sending party, i.e. with the keys in $R_u \cap R_v$. It is straightforward to see that by using a total of $w+1$ as many keys as in the single source schemes (and then each recipient's subset is also $w+1$ times larger), and by choosing keys to a party's subset independently at random with probability $1/(w+1)$, the schemes are as secure as the single source schemes. Note that the communication and computation overheads are as in the single source schemes since any source is expected to have the same number of keys as before. The mapping of users to subsets can be obtained using a public $(w+2)$-wise independent hash function. Then every source can simply attach to its message a sequence of MAC outputs with all the keys in its subset (without having to specify which key was used to produce each MAC), and each receiver can individually find which of these MACs to check.

[6] More formally, assume that it is not possible to distinguish in polynomial time between the output of the MAC and a random bit with probability better than $1/2+\varepsilon$. Then, as [29] assures, one can use a "hybrid argument" to show that it is not possible to distinguish between m MAC outputs and an m bit random string with probability better than $1/2 + m\varepsilon$.

3.4 Dynamic Sources

We describe here a new scheme for multiple sources with the following properties:

- The total number of keys is as in schemes for a single source, but every party can send authenticated messages.
- The scheme does not require the set of sources to be defined in advance or to contain all parties. Rather, it allows to *dynamically* add sources.
- The scheme distinguishes between the set of sources and the set of receivers. Only coalitions of more than w *receivers* can send false authenticated messages. Sources do not help such coalitions. This property is useful if receivers are more trusted than senders, as might be the case for example if the receivers are network routers and sources are common users.
- The scheme only provides a computational (rather than an information theoretic) security that not all keys in the intersection of a source and a receiver's subsets are known to a coalition. However this is not a decrease in security compared to the previous constructions since the keys are used in MACs which provide only computational security.

The new scheme uses a family of pseudo-random functions $\{f_k\}$ (see [26] or [4] for a discussion of pseudo-random functions). It is based on a single source scheme and can be built upon the basic scheme we described in Section 3.1 or the communication efficient scheme of Section 3.2.

Initialization: The scheme uses ℓ base keys $\langle k_1, \ldots, k_\ell \rangle$, where ℓ is exactly as in the single source schemes ($\ell = O(w \log(1/q))$). Each key k_i defines a pseudo-random function f_{k_i}.

Receiver Initialization: Each party v which intends to receive messages obtains a subset R_v of base keys. Every base key k_i is selected to R_v with probability $1/(w+1)$.

Source Initialization: Every party u which wishes to send messages receives a set of keys, *the set of second generation keys of* u, $S_u = \langle f_{k_i}(u), f_{k_2}(u), \ldots, f_{k_\ell}(u) \rangle$. Note that this set can be sent at any time after the system has been set-up, and the identity or the number of sources does not have to be defined in advance.

Message Authentication: When a party u sends a message M it authenticates it with all the second generation keys in S_u. That is, $\forall\, k \in S_u$ it computes and attaches a MAC of M with k.

Every receiving party v computes all the second generation keys of u which it can generate from R_v. Namely, it computes the set $f_{R_v}(u) = \{f_k(u) | k \in R_v\}$. It then verifies all the MACs which were computed using these keys.

The number of keys which are used and stored is as in the single source scheme. The work of the sources is as in the previous schemes, and receivers only have the additional task of evaluating f to compute a second generation key for each of the base keys in their subset (if consecutive authenticated packets are received from the same source the receiver should compute the second generation keys only when the first packet is received and use them for all the packets).

A very useful property of this scheme is that it enables a *dynamic* set of sources. New parties can be allowed to send authenticated messages by giving them a corresponding set of second generation keys. Another useful property of the scheme is that the set of sources can be separated from the set of receivers, and no coalition of sources can break the security. This property is useful is receivers are more trusted than sources. It also enables to give sources dedicated keys for authenticating different messages. An attractive application of these properties is described next:

An Application of a Dynamic Set of Sources

The scheme described above enables the set of sources to be dynamic, and prevents coalition of sources from breaking the security. The following application uses these two properties: Assume that it is required to authenticate the transmissions of a TV channel. The channel usually receives programs from other companies which produce them. Traditionally these parties send the programs to the center of the TV channel and then the programs are being transmitted from there to their viewers. It is possible instead that each production company will transmit its programs to the TV channel viewers (at the appropriate time) directly from its site, possibly using Internet multicast. Then the channel can give each of its viewers a subset of base keys R_v, and give the production company which should transmit a program at time t on date d the set $S_{d,t} = \langle f_{k_1}(s,d), \ldots f_{k_\ell}(s,d)\rangle$ of second generation keys. The company will authenticate the program using the keys in this set. Each viewer v will then compute the set $f_{R_v}(d,t)$ and check the MACs performed with these keys. Note that although every production company might receive many authentication keys (for many programs at different dates), no coalition of these companies can authenticate programs for which they do not receive authentication keys from the TV channel. The TV channel does not have to decide on the program table in advance, but rather at any time which is convenient.

3.5 Signatures vs. MACs: a rough performance comparison

Compared to the performance of public key signatures, our authentication schemes reduce the running time of the authenticator dramatically. The running time of the verifier and the communication overhead are of the same order as with public key signatures (the exact comparison depends on the size of the corrupt coalitions against which the schemes operate).

Recall the notation we use for the resources consumed by authentication schemes: The *running time* required to authenticate a message and to verify an authentication, T_S and T_V, respectively. The *length of the keys* that the authenticator and the verifier should store, M_S and M_V, respectively. The *length of the authentication message* (the MAC or the signature), C.

Consider for example RSA signatures with an 1024 bit modulus. Recent measurements indicate that a fast machine (200MHz power pc) signature (authentication) takes $T_S = 1/50$ sec. and verification time is $T_V = 1/30,000$ sec. For 768-bit DSS on a similar platform the numbers are roughly $T_S = 1/40$ and $T_V = 1/70$. In comparison, an application of the compression function of MD5 takes about $1/500,000$ of a second; an application of DES takes roughly the same time. Future block ciphers and hash functions are expected to be considerably faster.

The schemes we introduce require the parties to apply many MACs with different keys to the same message. Current constructions of MACs achieve both a hash of the input down to the required output size, and a keyed unpredictable output. For the suggested schemes it is preferable to perform a *single* hash down of the message, and then compute MACs of the hashed down value[7]. Regarding HMAC [25, 5] as a reference MAC function, this implies that only one of HMAC's two nested keyed applications of a hash function should be used. In the terms of [5] this corresponds to defining ℓ MACs with keys $k_1, \ldots, k_\ell$ as $\{NMAC_{k,k_i}\}_{i=1}^{\ell}$, where the key k is common to all functions. Therefore in comparisons to public key operations we assume that a MAC takes a single application of a compression function of the hash function in use (say, MD5), or equivalently a single application of a block-cipher such as DES.

Furthermore, we believe that more efficient MACs could be designed for our suggested multicast authentication schemes. In particular, these MAC functions would make use of the fact that they

[7] The initial hash down is also performed for public key signatures, since messages should be reduced to the size of the public key modulus. Therefore we omit its computation time from the running time overhead of our schemes.

can have a single bit output, and would have small amortized complexity (for evaluations of the function on the same input and many keys). Authentication schemes based on such functions should be considerably more efficient than the schemes based on HMAC.

In Table 1 we compare the overhead of RSA and DSS signatures to the overhead of the suggested authentication schemes with some specific parameters. As mentioned above, we estimate that using current technology about $500,000$ MACs can be evaluated in a second (here we take into account the fact that each message has to MACed with a number of different keys. Also, the communication overhead of the basic and improved schemes are based on using only 10 bits out of each MAC. The improved scheme is taken with parameter $c = 2$.)

	Auth.	**Ver.**	**Comm.**	**Source Key**	**Receiver Key**
Units	(ops/sec)	(ops/sec)	(bits)		
RSA, 1024 bits	50	30,000	1024	2048 bits	1024 bits
DSS, 768 bits	70	40	1536	1536 bits	1536 bits
Basic scheme $w = 10, q = 10^{-3}$	$2,650$	$26,500$	1900	190 MAC keys	19 MAC keys
Improved Security $w = 10, q = 10^{-3}$	$2,650$	$26,500$	1900	380 MAC keys	38 MAC keys
Low Comm. $w = 10, q = 10^{-3}$	660	$6,600$	760	760 MAC keys	76 MAC keys
Perfect Sec. $n = 10^4, q' = 10^{-3}$	200	2000	$25,000$	2500 MAC keys	250 MAC keys

Table 1: A performance comparison of authentication schemes.

With this performance it seems that the signing time is much shorter in our scheme that with public key signatures, and the verification time is of the same order as for RSA (and much faster than DSS). However, as mentioned above, we believe that much more efficient MACs can be used for our schemes. In particular, the *Low Communication* variant requires a MAC with only a single bit output, which might be further improved.

In addition note that if public key signatures are used for authentication then each receiver should store the verification keys of all sources, or alternatively the verification keys should be certified by a certification authority and then the length of the authentication message and the verification times are doubled.

The table describes the number of authentications and verifications that can be performed per second, the communication overhead in bits, and the length of the key used by the source and the receivers. The first two rows are for RSA and DSS signatures. The third row provides an estimate for our basic authentication scheme, providing security of $q = 10^{-3}$ against coalitions of up to ten corrupt users. The fourth row is for a scheme which uses more keys and thus reduces the probability that the coalition knows all the keys of a user, but each authentication is performed with the same number of keys. Then we present the performance of the communication efficient variant, in which each MAC has a single bit output. Last is the performance of a scheme which guarantees that no coalition knows all the keys of any user (its overhead seems too large to justify its use).

4 Dynamic Secrecy – User Revocation

Once some common information has been set up for a group, secrecy can be obtained by encrypting the messages that are sent to the group members. When a new member joins the group the common key can be sent to it through using secure unicast. Alternatively, if the new member should not be able to understand previous group communications then a new common key should be generated and sent to the old group members (say, encrypted with the old common key) and to the new member. The main problem arises when a member leaves the group. This member should not be able to receive and comprehend future group transmissions. It should be assumed that some members might try to receive and understand the group communication even after they leave the group, for example in applications like pay-per-view in which only paying customers should be able to receive transmissions. It is therefore not sufficient to ask members who leave the group to delete the group key which they hold and to stop receiving the transmissions but rather it is essential to change the key with which group communication is encrypted. This problem is known as *user revocation* or *blacklisting*.

Following we survey some solutions for the member deletion problem, and show how to reduce the communication overhead of a particularly appealing construction based on binary trees [33]. We also describe how to enhance this construction against a new type of attack.

4.1 User Revocation Schemes

A trivial solution for the member revocation problem is for each group member to share a secret key with a center which controls the group. When a member is deleted from the group the center chooses a new common key with which future multicast messages should be encrypted, and sends this key to every group member, encrypted with the secret key that is shared by the center and the member. This solution does not scale up well since a group of n members requires a key renewal message with $n-1$ new keys. This overhead prevents this solution from being applicable to large scale applications or even to smaller scale applications in which the groups are dynamic and members are leaving groups very often.

A more advanced solution was suggested by Mittra [28]. In his scheme the multicast group is divided into subgroups which are arranged in a hierarchical structure. There is a single *group security controller* but each subgroup has a *group security intermediary* which is responsible for maintaining security in the subgroup. A different key is associated with each subgroup and known to all its members, but its security intermediary also knows the key of a subgroup in an upper layer of the hierarchy. When a group member is deleted the only key to be changed is the key of the subgroup to which it belonged, and the communication overhead is relatively small (the situation is more complex if a subgroup security intermediary leaves the group). However, this solution introduces many possible points of failure, the security intermediaries. A corrupt or a malfunctioning intermediary affects the operation and the security of the subset of the network for which it is responsible, or even of the entire network.

There are suggestions to use public key technology, namely generalized Diffie-Hellman constructions, to enable communication efficient group rekeying [8, 32]. However, for a group of n members these suggestions require $O(n)$ exponentiations. For most applications this overhead is far too high to be acceptable in the near future.

A totally different solution was suggested by Fiat and Naor [14] and was motivated by pay-TV applications. It enables a single source to transmit to a dynamically changing subset of legitimate receivers from a larger group of users, such that coalitions of at most k users cannot decrypt the transmissions unless one of them is a member in the subset of legitimate receivers. This scheme can be used to send new group keys when a group member leaves. The communication overhead of this re-keying messages is $O(k \log^2 k \log(1/p))$, where p is an upper bound on the probability that a coalition of at most k users can decrypt a transmission to which it is not entitled. This scheme also requires each user to store $O(\log k \log(1/p))$ keys. The main drawback in applying this scheme for Internet applications is that the security is only against coalitions of up to k users, and the parameter k substantially affects the overhead of the scheme. If k is not too large then it might not be too hard to organize a group of more than k colluders, and on the other hand large values of k incur a large overhead. It should also be noted that this scheme is only suitable for a single source of transmission, but this obstacle might be overcome if all users trust the owner of the group and all communication is sent through a unicast channel to this owner and from there multicasted to the group (as is the case for example in CBT routing).

4.2 A Tree Based Scheme

Tree based group rekeying schemes were suggested by Wallner et al. [33] (who used binary trees), and independently by Wong et al. [34] (who consider the degree of the tree nodes as a parameter). We concentrate on the scheme of [33] since it requires a smaller communication overhead per user revocation. This scheme applied to a group of n users requires each user to store $\log n + 1$ keys. It uses a message with $2 \log n - 1$ key encryptions in order to delete a user and generate a new group key. This process should be repeated for every deleted user. The scheme has better performance than the Fiat-Naor broadcast encryption scheme in case the number of deletions is not too big. It is also secure against any number of corrupt users (they can all be deleted from the group, no matter how many they are). A drawback of this scheme is that a group member must receive *all* the update messages that are sent by the group owner when users are deleted, in order to continue to be able to decrypt the group transmissions. If a member does not receive messages for a while, for example when it is disconnected from the network, it must first receive all the update messages that were sent while it was disconnected. This might incur a considerable overhead and also affect the reliability of the scheme.

We next present an improvement to the scheme of [33] which halves the communication overhead. Although this is only an improvement by a (relatively small) constant it might be significant for many applications[8]. In addition we present a new type of attack to which the scheme of Wallner et al. is susceptible, and against which our scheme is secure.

4.2.1 The basic scheme

The following scheme was suggested in [33]:

Setting: Let $u_0, \ldots, u_{n-1}$ be n members of a multicast group (in order to simplify the exposition we assume that n is a power of 2). They all share a group key k with which group communication is encrypted. There is a single group controller, which might wish at some stage to delete a user from the group and enable the other members to communicate using a new key k', unknown to the deleted user.

Initialization: The group controller associates the group members to leaves of a binary tree in the following way. The root of the tree is marked v_ε. The two descendants of a node v_i are marked v_{i0} and v_{i1}. The tree is of depth $\log n$, and user u is associated with the *leaf* v_u (the marking is

[8] Independently, McGrew and Sherman [27] have presented a tree based rekeying scheme which has the same overhead as ours. However, the security of their scheme is based on non-standard cryptographic assumptions and is not rigorously proven. In comparison, the security of our scheme can be rigorously proven based on the widely used assumption of the existence of pseudo-random generators [36, 6].

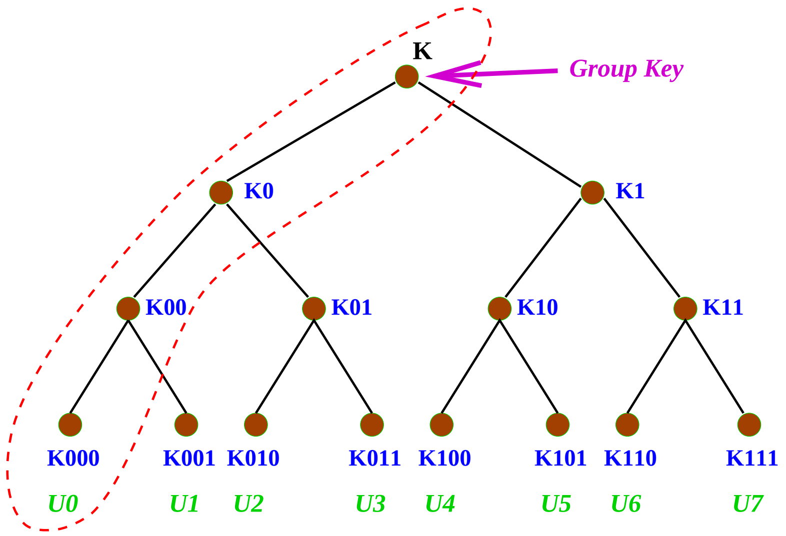

Figure 1: The tree key data structure (the keys of u_0 are encircled).

a $\log n$ bits binary number). The group controller associates every *node* v_i with a random key k_i. Each user u is given (through a secure channel) the keys of the nodes in the path from its leaf to the root, i.e. $\log n + 1$ keys. For example, if there are 8 users then user u_0 is associated with leaf v_{000} and receives keys $k_{000}, k_{00}, k_0, k_\varepsilon$ (see Figure 1). Since all users receive the root key k_ε it can be used as the group key and encrypt the group communication. (Note that the only purpose of the tree is to map users to keys, it is completely unrelated to the communication network).

User deletion: Suppose that the group controller needs to remove user u from the group. Without loss of generality assume that $u = u_0$ and that there is a total of 8 users and therefore u is mapped to leaf v_{000} (it is easier to describe the removal of a specific user than to describe the general case, but it is easy to infer the general case from this example). The group controller chooses new random values $k'_{00}, k'_0, k'_\varepsilon$ to the keys in the path from leaf v_{000} to the root. It communicates them in the following way which enables every user $u' \neq u_0$ to learn the new values in the path from its leaf to the root (e.g. user v_7 will learn the new value k'_ε of the root key, whereas user v_1 will learn the new values $k'_{00}, k'_0, k'_\varepsilon$). (See Figure 2).

The group controller sends the value k'_{00} encrypted with k_{001}. It sends k'_0 encrypted with the new key of its left child, k'_{00}, and also encrypted with the (old and unchanged) key of its right child, k_{01}. It also sends k'_ε encrypted with k'_0, and k'_ε encrypted with k_1. The deleted user u_0 cannot decrypt any of this values whereas any other user can decrypt exactly all the values in the intersection between the path from its leaf to the root and the path from leaf v_{000} to the root. For example, user u_1 can decrypt k'_{00} using k_{001}, then decrypt k'_0 using k'_{00} and decrypt k'_ε using k'_0.

Analysis: Each user should store $\log n + 1$ keys. The group controller should send a message of $2\log n - 1$ key encryptions in order to remove a user from the group and change the group key. If more than a single user should be removed the above procedure is repeated for each of the removed users.

4.2.2 Smaller communication overhead

It is possible to decrease the communication overhead of the previous scheme by a factor of two. We first give a simple description of the construction using a function f with "good properties",

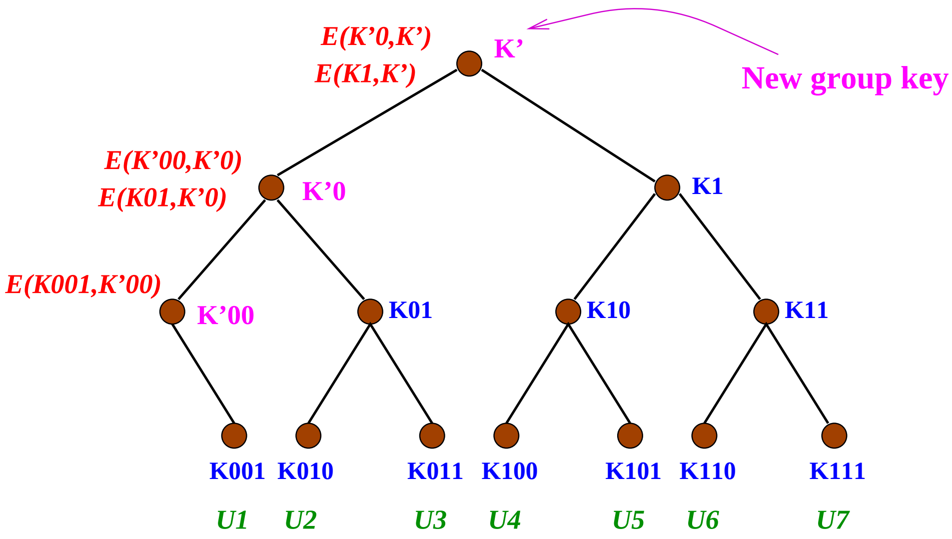

Figure 2: The transmission of new values to the keys surrendered to u_0.

and then give a formal description of the scheme, of the same complexity, for which we can prove security. The proposed scheme is also immune to an attack to which the basic scheme is vulnerable.

The initialization of the scheme is as in the scheme of [33]. Let f be a function with "good properties". Then when a user (say user u_0 of leaf v_{000}) is deleted, a random value r is chosen to be the key associated with its parent node v_{00}, i.e. $k'_{00} = r$. The new key of node v_0 is $k'_0 = f(r)$, and the new root key is $k'_\varepsilon = f(f(r))$. The group controller sends the value r encrypted with k_{001}, the value $f(r)$ encrypted with k_{01}, and the value $f(f(r))$ encrypted with k_1 (see figure 3). Each user can now decrypt only one of these values, but can also use f to generate from it all the values in the path from it to the root. For example, user u_1 can decrypt $k'_{00} = r$, and then can generate $k'_0 = f(r)$ and $k'_\varepsilon = f(f(r))$. The communication overhead is therefore only $\log n$.

Following we describe the scheme more formally. Let f be pseudo-random generator which doubles the size of its input [36, 6]. The security of the user deletion scheme can be formally reduced to the security of f. Denote by $f(x)|_L, f(x)|_R$ the left and right halves of the output of f on an input x. For our example the group controller generates a random value r_{00}, defines $k'_{00} = f(r_{00})|_L$ and $r_0 = f(r_{00})|_R$, and sends r_{00} encrypted with k_{001}. It defines $k'_0 = f(r_0)|_L$ and $r_\varepsilon = f(r_0)|_R$, and sends r_0 encrypted with k_{01}. It defines $k'_\varepsilon = f(r_\varepsilon)|_L$ and sends r_ε encrypted with k_1. Note that as in the "simplified" exposition, any user is able to compute exactly the new keys which it requires. For example, user u_1 decrypts r_{00} using k_{001}, and then computes $k'_{00} = f(r_{00})|_L$, $k'_0 = f(f(r_{00})|_R)|_L$, and $k'_\varepsilon = f(f(f(r_{00})|_R)|_R)|_L$.

Advantages of the new scheme: This construction halves the communication overhead of the basic scheme and its security can be rigorously proven. It has an additional advantage: In the scheme of Wallner et al the group controller chooses the group key (the root key), whereas is our construction this key is the output of a pseudo-random generator. Suppose that there is an adversary which can break encryptions performed with a subset of the key space (for example keys is which certain bits have a linear dependency), and furthermore that this adversary has gained temporary control over the group controller (e.g. when the controller was manufactured). Then if the scheme of [33] is used the adversary might corrupt the method by which the group controller generates keys such that the root key would always be chosen from the "weak" subspace. However, if our scheme is used and would utilize a secure pseudo-random generator f, then it is impossible

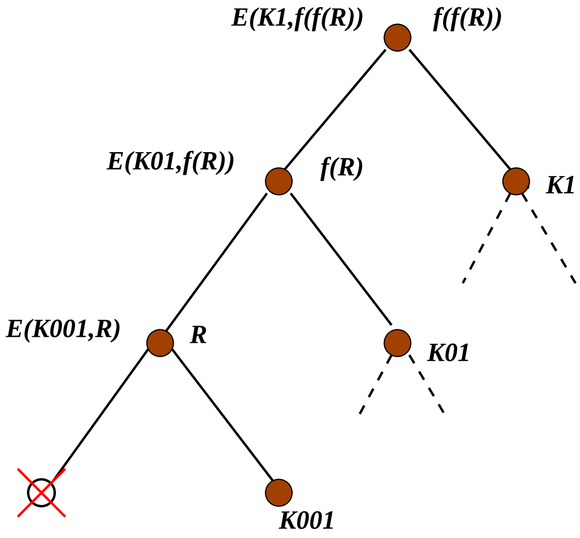

Figure 3: The transmission of new values to the keys in the improved scheme.

to find values r such that the root key $f(f(\cdots(r)\cdots)$ is in the weak key subset, except by running an exhaustive search.

Acknowledgments

We are very thankful to Shai Halevi, Rosario Gennaro and Tal Rabin, for discussions on the problems and possible solutions for multicast security.

References

[1] A. Albanese, Bloemer J., Edmonds J., Luby M., Sudan M. "Priority Encoding Transmission", *IEEE Transactions on Information Theory*, Vol. 42, No. 6, November 1996.

[2] N. Alon, "Probabilistic Methods in Extremal Finite Set Theory", in *Extremal Problems for Finite Sets*, 1991, 39–57.

[3] Ballardie, A. J., "A New Approach to Multicast Communication in a Datagram Network", Ph.D. Thesis, University College London, May 1995.

[4] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols", in *Proc. First Annual Conference on Computer and Communications Security* , ACM, 1993.

[5] M. Bellare, Canetti R. and Krawczyk H., "Keying Hash Functions for Message Authentication", *Advances in Cryptology – Crypto '96*, Lecture Notes in Computer Science vol. 1109, Springer-Verlag, 1996.

[6] M. Blum and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM J. on Comp.*, Vol. 13, 1984, 850–864.

[7] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung, "Perfectly-secure key distribution for dynamic conferences, *Advances in Cryptology – Crypto '92*, Lecture Notes in Computer Science vol. 740, Springer-Verlag, 1993, pp. 471–486.

[8] M. Burmester and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System", *Advances in Cryptology – Eurocrypt '94*, Lecture Notes in Computer Science vol. 950, Springer-Verlag, 1994, pp. 330–354.

[9] R. Canetti, R. Gennaro, D. Herzberg and Naor D., "Proactive Security: Long-term protection against break-ins", CryptoBytes.

[10] R. Canetti and B. Pinkas B., "A taxonomy of multicast security issues", internet draft `draft-canetti-secure-multicast-taxonomy-00.txt`, available at `ftp://ftp.ietf.org/internet-drafts/draft-canetti-secure-multicast-taxonomy-00.txt`, June 1998.

[11] Desmedt Y. and Frankel Y., "Threshold cryptosystems", *Advances in Cryptology – Crypto '89*, Lecture Notes in Computer Science vol. 435, Springer-Verlag, 1990, pp. 307–315.

[12] Y. Desmedt, Y. Frankel and M. Yung, "Multi-receiver/Multi-sender network security: efficient authenticated multicast/feedback", *IEEE Infocom '92*, pp. 2045–2054.

[13] M. Dyer, T. Fenner, A. Frieze and A. Thomason, "On Key Storage in Secure Networks", *J. of Cryptology*, Vol. 8, 1995, 189–200.

[14] A. Fiat A. and M. Naor, "Broadcast Encryption", *Advances in Cryptology – CRYPTO '92*, Lecture Notes in Computer Science vol. 839, 1994, pp. 257–270.

[15] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems", in *Advances in Cryptology – Crypto '86*, pp. 186–194, Springer-Verlag, 1987.

[16] Gemmel P., "An introduction to threshold cryptography", in *CryptoBytes*, Winter 1997, 7–12.

[17] Gennaro R. and Rohatgi P., "How to sign digital streams", *Advances in Cryptology – CRYPTO '97*, Lecture Notes in Computer Science, vol. 1294, Springer-Verlag, 1997, pp. 180–197.

[18] L. Gong and N. Schacham, "Elements of Trusted Multicasting," Technical Report SRI-CSL-94-03, Computer Science Labortatory, SRI International, Menlo Park, Ca., May 1994.

[19] L. Gong L. and D. Wheeler, "A Matrix Key Distribution Scheme," *Journal of Cryptology*, 2(2):51–59, 1990.

[20] Harney H. and Muckenhirn C., "Group Key Management Protocol (GKMP) Specification", *RFC 2093*, July 1997.

[21] Harney H. and Muckenhirn C., "Group Key Management Protocol (GKMP) Architecture", *RFC 2094*, July 1997.

[22] `http://www.ietf.org/html.charters/ipsec-charter.html`

[23] Koblitz N., "Elliptic curve cryptosystems", *Mathematics of Computation*, 48: 203-209, 1987.

[24] Krawczyk H., "SKEME: A Versatile Secure Key Exchange Mechanism for Internet", in *IEEE Proceedings of the 1996 Symposium on Network and Distributed Systems Security.*

[25] Krawczyk H., Bellare M. and Canetti R., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[26] Luby M., *Pseudorandomness and Cryptographic Applications*, Princeton University Press, 1996.

[27] McGrew D, Sherman A. T., "Key Establishment in Large Dynamic Groups Using One-Way Function Trees", manuscript, 1998.

[28] Mittra S., "Iolus: A Framework for Scalable Secure Multicasting", *Proc. of the ACM SIGCOMM '97*, Cannes, France, Sept. 1997.

[29] M. Naor, O. Reingold, "From Unpredictability to Indistinguishability: A Simple Construction of Pseudo-Random Functions from MACs", to appear, *Advances in Cryptology – Crypto '98.*

[30] Rivest R. L., *The MD5 Message Digest Algorithm*, Internet Request for Comments, April 1992, RFC 1321.

[31] Safavi-Naini R. and Wang H., "New results on Multi-receiver Authentication Code," *Advances in Cryptology – EUROCRYPT '98*, Lecture Notes in Computer Science, vol. 1403, Springer-Verlag, 1998, pp. 527–541.

[32] Steiner M., Tsudik G., Waidner M., "Diffie-Hellman key distribution extended to group communication", *3rd ACM Conf. on Computer and Communications Security"*, 1996.

[33] Wallner D.M., Harder E.J., Agee R.C., "Key Management for Multicast: Issues and Architectures". Avaliable at `ftp://ietf.org/internet-drafts/draft-wallner-key-arch-00.txt`

[34] Wong C. K., Gouda M., Lam S., "Secure Group Communications Using Key Graphs", to appear, *Sigcomm '98.*

[35] Wong C. K., Lam S., "Digital Signatures for Flows and Multicasts", The University of Texas at Austin, Department of Computer Sciences, Technical Report TR-98-15. July 1998.

[36] Yao A., "Theory and Applications of Trapdoor Functions", *Proc. 23rd IEEE Symp. on Foundations of Computer Science*, 1982, 80–91.