Multicast Security: A Taxonomy and Some 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 sound security mechanisms for multicast communication. Yet, existing security protocols for multicast offer only partial solutions.**

We first present a taxonomy of multicast scenarios on the Internet and point out relevant security concerns. Next we address two major security problems of multicast communication: *source 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 solution to the key revocation problem.

I. Introduction

The popularity of multicast has grown considerably with the wide use of the Internet. Examples include Internet video transmissions, news feeds, stock quotes, software updates, live multiparty conferencing, on-line video games and shared whiteboards. 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 [17]). But this solution is prohibitively inefficient in most multicast scenarios. In particular, 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.

This work. First, we present a taxonomy of multicast security concerns and scenarios, with a strong emphasis on IP multicast[1]. 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 in other settings. 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.

Next we concentrate on a problem that emerges as a serious bottleneck in multicast security: *source* and *message* authentication. Known attempts to solve multicast security problems (e.g., [16], [22], [3], [28], [29], [21]) 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 source 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.

We present solutions to the source authentication problem based on shared key mechanisms (namely, Message Authentication Codes — MACs), where each member has 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. Our main savings are in the time to generate signatures.

The basic source authentication scheme for a single sender draws from ideas of [2], [11]: 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 to this 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.
- Using similar parameters to those of the basic scheme, one can guarantee that each good member has *many* keys that are

*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 for Computer Science, MIT. Research supported by DARPA contract DABT63-96-C-0018. Email:miccianc@theory.lcs.mit.edu
¶Dept. of AM and CS, Weizmann Institute of Science. Research supported by ESPRIT working group RAND2. Email: naor@wisdom.weizmann.ac.il
‖Dept. of AM and CS, Weizmann Institute of Science. Research supported by an Eshkol Fellowship from the Israeli Ministry of Science. Email: bennyp@wisdom.weizmann.ac.il

[1] This survey is also the basis for a recent internet-draft [8], prepared for the Secure Multicast Group (SMuG) of the IRTF [26]. See also [15] for an earlier, more basic discussion of secure multicast issues.

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 can be reduced to only ℓ *bits*. (Also, such MAC functions may be more efficient than regular MACs.)

- A very similar method allows for *many senders* to use the same structure 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 not known to any (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.

Finally, we consider the *membership revocation* problem. When a member leaves a multicast group it might be required to change the group key in a way that the leaving member does not learn the new key. A relatively efficient solution to this problem has been recently proposed [28], [29]. We present an improvement to this solution, that saves half of the communication overhead. (When a new member joins, the group might have to be re-keyed as well, in order to prevent the joining member from understanding previous group communication. This is a much simpler task: the group controller simply multicasts the new key encrypted with the previous group key.)

Organization. In Section II we list and discuss multicast security issues, in several common scenarios. In Section III we present our multicast authentication schemes, and in Section IV we present our improvements over past mechanisms for membership revocation.

II. 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 become effective? 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)?

Next, what 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) or 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. 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. We distinguish two types of secrecy: *Ephemeral secrecy* means preventing non group-members from easy access to the transmitted data. Here a mechanism that only delays access may be sufficient. *Long-term secrecy* means protecting the confidentiality of the data for a long period of time. This type of secrecy is often not needed for multicast traffic.

Authenticity may take two flavors: *Group authenticity* means that each group member can recognize whether a message was sent by a group member. *Source authenticity* means that it is possible to identify the particular sender within the group. It may 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.

Access control, or making sure that only registered and legitimate parties have access to the communication addressed to the group, is usually obtained by maintaining ephemeral secrecy of the data. Enforcing access control also involves authenticating potential group members. The access control problem becomes considerably more complex if members may join and leave with time.

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-enabled 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 in general beneficial to distribute the security tasks as much as possible.

A natural approach for distributing trust in multicast security centers is to use *threshold cryptography* [9], [13] and *proactive security* [7] techniques to replace 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 IV.

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.

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 derived keys) 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 (e.g. a news agency that broadcasts news-feeds and stock-quotes 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 into several sources in different locations. The recipients are typically lower-end machines with limited resources. Consequently, any security solution should be optimized for efficiency at the recipient side.

Although the life-time of the group is usually very long group membership is typically 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 typically 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 news).

A natural solution for this scenario may have a group management center that handles access control and key management. (To scale the solution to a larger number 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.

A simple and popular variant of this scenario, file transmission and updates, typically has static group membership and does not require on-line delivery of data.

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, cryptographically disconnecting it from the group is often not crucial. 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, 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. Virtual conferencing applications are also 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.

III. Efficient Authentication Schemes

We concentrate on two approaches to authentication: public key signatures, and MACs. (We do not address information-theoretic authentication mechanisms, such as [10], [25], [6], which are inherently inefficient for groups of non-trivial size.)

Public key signatures are perhaps the most natural mechanism for multicast authentication. Yet, signatures are typically long, and computing and verifying each signature requires a significant computational overhead. Applying signatures to authenticate streams of data was investigated in [14], who proposed a chaining mechanism that requires a single signature per stream. These constructions do not tolerate packet loss, and are thus incompatible with IP multicast. Alternatively, [30] suggested using tree-based hashing to authenticate streams. This approach is a little less efficient, and incurs some latency, but it better tolerates packet loss.

As an alternative to public key signatures, we propose an authentication method based on *message authentication codes* (MACs). A MAC is a function which takes a secret key k and a message M and returns a value $\text{MAC}(k, M)$. Very informally, a MAC scheme is *unforgeable* if an adversary that sees a sequence $\{M_i, \text{MAC}(k, M_i)\}$ where the M_i's are adaptively chosen, but does not know k, has a negligible probability to generate $\text{MAC}(k, M)$ for *any* $m \notin \{M_i\}$.

While MACs are typically much more efficient to generate and verify than digital signatures, they require that all potential verifiers have access to a shared key, k. This property makes MACs seemingly insufficient for achieving source authentication: any potential receiver who has the key k can "impersonate" the sender. We present new MAC-based authentication methods which achieve source authentication, and are more efficient than public key based authentication (especially in the time to generate signatures). We first present a description of a basic scheme, followed by several variants and improvements (see sketch in the Introduction).

We analyze the following salient resources for all the schemes we present: The *running time* required to authenticate a message and to verify an authentication, denoted T_S and T_V, respectively. The *length of the keys* that the authenticator and the verifier should store, denoted M_S and M_V, respectively. The *length of the authentication message* (the MAC or the signature), denoted C. These resources are obviously related to the latency, secure memory and bandwidth overhead parameters discussed in Section II.

Per-message unforgability 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 (e.g., 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 (but does not know a-priori whether it will be able to authenticate the message). 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^{-20} up to 2^{-10}).

A bit more formally, we say that a MAC scheme is *q-per-message unforgeable* if no (probabilistic 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 decide to quit or to gamble. If it quits it receives a payment of \$0. Otherwise, it chooses a message $m \notin \{m_1, \ldots, m_k\}$ and tries to guess the value of the MAC on m. The adversary receives $\$(1-q)$ if its guess is correct, and pays $\$q$ otherwise. In other words the adversary may guess correctly the value of the MAC with probability at most q, but (except with negligible probability) won't "know" whether its guess is correct.

We believe that for most systems a small (although non-negligible) per-message unforgeability (say, $q = 2^{-20}$) is sufficient. Note that per-message unforgeability is a weaker security property than standard unforgeability, in the sense that any scheme that is unforgeable in the standard sense is also q-per-message unforgeable (for any non-negligible value of q). The converse does not necessarily hold.

A. The Basic Authentication Scheme for a Single Source

Let w be the maximum number of corrupted users. The basic scheme proceeds as follows:

- The source of the transmissions (S) knows a set of $\ell = e(w+1)\ln(1/q)$ keys, $R = \langle K_1, \ldots, K_\ell \rangle$.
- Each recipient u knows a subset of keys $R_u \subset R$. Every key K_i is included in R_u with probability $1/(w+1)$, independently for every i and u [2].
- Message M is authenticated by S with each key K_i using a MAC and $\langle \text{MAC}(K_1, M), \text{MAC}(K_2, M), \ldots, \text{MAC}(K_\ell, M) \rangle$ is transmitted together with the message.
- 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 rejects the message.

[2]Notice that this can be accomplished by using a $(w+1)$-wise independent mapping from users to subsets.

The performance parameters are the following. The source must hold $M_S = \ell = e(w+1)\ln(1/q)$ basic MAC keys. Each receiver expects to hold $M_V = e\ln(1/q)$ MAC keys.[3] The communication overhead per message is $C = e(w+1)\ln(1/q)$ MAC outputs. The running time overhead is $T_S = e(w+1)\ln(1/q)$ MAC computations for the source and only $T_V = e\ln(1/q)$ MAC computations for a receiver.

Theorem 1: 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)[4].

Proof (sketch): For every user u and any coalition of w users, the probability that a specific key is good (i.e. contained in a user's subset, but not in the subset of any of the w members of the coalition) is ' $g = \frac{1}{w+1}\left(1 - \frac{1}{w+1}\right)^w = \frac{1}{(w+1)\left(1+\frac{1}{w}\right)^w} > \frac{1}{e(w+1)}$. Therefore, the probability that R_u is completely covered by the subsets held by the coalition members is $(1-g)^\ell < (1 - \frac{1}{e(w+1)})^{e(w+1)\ln(1/q)} < e^{-\ln(1/q)} = q$. If R_u is not covered, the set $\{MAC(K_i, M)\}_{i \in R_u}$ contains at least one MAC for which the coalition does not know K_i. The probability of computing it correctly is at most q'. By union bound, the probability that the coalition can authenticate M to u is at most $q + q'$. □

Notice that when the keys of a user are not covered by the coalition, the coalition cannot check in advance (off-line) whether it can authenticate a specific message. Therefore the probability q' of authenticating a message by breaking a MAC can be rather large (e.g. even $q' = 2^{-10}$ might be reasonable for many applications).

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 [11].

The security is against an arbitrary, but fixed, coalition of up to w corrupt recipients. Notice that it is possible to construct schemes which are secure against any coalition of size w as follows. Let $q = (n \cdot \binom{n}{w})^{-1}$ (i.e. 1 over the number of possible combinations of coalitions and users). By a probabilistic argument, there exists 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 less than $e(w+1)^2 \ln n$ keys, and each recipient has a subset of expected size less than $e(w+1)\ln n$.

B. Smaller Communication Overhead

We now describe a scheme with a lower communication overhead. The idea behind it is that using just four times as many keys as in the basic scheme, one can ensure that the coalition does not know $\log(1/q)$ of the user's keys. Each key can therefore 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.

Recall the basic scheme: it limits the success probability of a corrupt coalition to be $q + q'$, where q' is the per-message unforgeability. The MAC output must be at least $\log_2(1/q')$ bits long. Therefore, assuming $q' = q$, the communication overhead is $C > e(w+1)\ln^2(1/q)$ *bits*. The improved scheme achieves a communication overhead smaller than $4e(w+1)\ln(1/q)$ *bits*.

The improved scheme uses a MAC with a single bit output. (Current constructions of MACs have much larger outputs, but our schemes can use a single bit of this output. It might also be possible to design a special-purpose MAC function with a single bit output, which would be more efficient than standard constructions.) For simplicity of 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 that the user accepts an unauthentic message from the coalition is at most[5] q. In the suggested scheme the source uses $\ell = 4e(w+1)\ln(1/q)$ keys where each key is included in a user's subset with probability $1/(w+1)$.

All performance parameters are multiplied by four. The source must store $M_S = \ell = 4e(w+1)\ln(1/q)$ basic MAC keys. Each receiver expects to store $M_V = 4e\ln(1/q)$ basic MAC keys. The communication overhead is $C = 4e(w+1)\ln(1/q)$ *bits* per message. The source must compute $4e(w+1)\ln(1/q)$ MACs, whereas each receiver expect to compute $4e\ln(1/q)$ MACs.

Theorem 2: Consider a MAC with a single bit output that is $\frac{1}{2}$-per-message unforgeable, and consider the above scheme using this MAC and $\ell = 4e(w+1)\ln(1/q)$ keys. Then for every user u and coalition of other w corrupt users, it holds with probability $1 - q$ that the resulting scheme is q-per-message unforgeable (with respect to the coalition and u).

Proof (sketch): The probability that a specific key is good is $g > \frac{1}{e(w+1)}$ as before. Since the MAC is $\frac{1}{2}$-per-message unforgeable, the coalition cannot guess with probability better than $1/2$ the output of a MAC whose key it does not know. Therefore the expected success probability of a corrupt coalition is $\sum_{i=0}^{\ell} \binom{\ell}{i} g^i (1-g)^{\ell-i} 2^{-i} = (1 - g/2)^\ell < q^2$. By Markov inequality, with probability at most q the coalition has a probability greater than q to compute all MACs with key in R_u. In other words, with probability $1 - q$ the scheme is q-per-message unforgeable. □

C. Multiple Dynamic Sources

The schemes presented above can be easily extended to enable *any* party to send authenticated messages. The global set of ℓ keys is $w + 1$ times bigger than in the single source scheme, and every party receives a random subset R_u of these keys. Keys

[3] A straightforward modification of this scheme allows each member to have a *fixed* number of keys.

[4] A similar result holds with respect to per-message unforgability. That is, if the MAC is q'-per-message unforgeable then for any user u and coalition of other w corrupt users, it holds with probability $1 - q$ that the resulting scheme is q'-per-message unforgeable with respect to the coalition and the user.

[5] 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 (see [23]) 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 $1/2 + m\varepsilon$.

are included R_u independently at random with probability $\frac{1}{w+1}$. When a party u sends a message, it authenticates it with all the keys in R_u, and every receiving party v verifies the authentications that were performed with the keys in $R_u \cap R_v$. It is straightforward to verify that the resulting schemes are as secure as the single source schemes. Note that the (average) communication and computation overheads are not changed. The mapping of users to subsets can be done with a public $(w+2)$-wise independent hash function.

Following, we present a better method which supports a *dynamic* set of sources and has 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. The keys of sources do not help such coalitions. This property is especially useful if receivers are more trusted than senders, as might be the case for example if the receivers are network routers.
- The scheme provides a computational (rather than an information theoretic) security against revealing to a coalition all the keys in the intersection of a source and a receiver's subsets.

The scheme uses a family of pseudo-random functions $\{f_k\}$ (see [20] 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 III-A or the communication efficient scheme of Section III-B.

Initialization: The scheme uses ℓ *primary keys* $\langle k_1, \ldots, k_\ell \rangle$, where ℓ is 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 primary keys. Every primary key k_i is included in R_v with probability $1/(w+1)$.

Source Initialization: Every party u which wishes to send messages receives a set of *secondary keys* $S_u = \langle f_{k_i}(u), f_{k_2}(u), \ldots, f_{k_\ell}(u) \rangle$. This set can be sent 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 secondary 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 secondary keys of u with primary key in 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 secondary key for each of the primary keys in their subset

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 secondary 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. It also enables to give sources dedicated keys for authenticating different messages. An attractive application of these properties is to give the source which is designated to broadcast at time T the set of secondary keys $f_k(T)$, and require it to use them to authenticate its broadcast at that time. This approach ensures that sources can only send information to the group in their designated time slots.

D. Signatures vs. MACs: a rough performance comparison

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

Consider for example RSA signatures with an 1024 bit modulus. Recent measurements indicate that on a fast machine (200MHz power pc) a signature (authentication) takes $T_S = 1/50$s and verification time is $T_V = 1/30,000$s.[6] 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 down of the input 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 [19], [4] 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 [4] 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 authentication schemes. In particular, these MAC functions would make use of the fact that they 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 schemes based on HMAC.

Table I compares the overhead of RSA and DSS signatures to the overhead of the suggested authentication schemes with some specific parameters. The communication overhead of the basic and improved schemes are based on using only 10 bits out of each MAC.

The table describes the number of authentications and verifications that can be performed per second, the communication

[6] The numbers here are for highly optimized RSA code with verification exponent 3. Verification using standard RSA code is considerably slower.

[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.

	Auth.	Ver.	Comm.	Source Key	Receiver Key
Units	(ops/sec)	(ops/sec)	(bits)		
RSA[6] 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
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 I

A PERFORMANCE COMPARISON OF AUTHENTICATION SCHEMES.

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 per-message unforgeability of $q = 10^{-3}$ against coalitions of up to ten corrupt users. Next 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).

It is seen that the signing time is much shorter in our scheme than with public key signatures. The verification time is comparable to (highly optimized) RSA and much faster than DSS.[8]

IV. Dynamic Secrecy – User Revocation

Secret group communication can be achieved by encrypting messages with a group key. This raises the question of how to add or remove users from the group. When a new member joins the group, the common key can be sent to the new member using secure unicast. Alternatively, if the previous communications should be kept secret from the new user, a new common key can be generated and sent to the old group members (encrypted with the old common key) and to the new member (using secure unicast). User deletion is more problematic. Obviously, it is not enough to just ask members who leave the group to delete their group key, and it is essential to change the key with which group communication is encrypted in order to conceal future communications from former group members. This problem is known as *user revocation* or *blacklisting*, and is particularly important in applications like pay-per-view in which only paying customers should be allowed to receive transmissions.

We survey some solutions for the member deletion problem, describe a particularly appealing construction from [28], [29] based on binary trees, and present an improved construction with reduced communication overhead. We also show how our construction is more resistant to a certain kind of attack.

A. Some User Revocation Schemes

A trivial solution for the member revocation problem is for each group member to share a individual secret key with a center which controls the group. When a member is deleted from the group, the center chooses a new common key to encrypt future multicast messages, and sends it to every group member, encrypted with the respective individual secret keys. This solution does not scale up well since a group of n members requires a key renewal message with $n - 1$ new keys.

A more advanced solution was suggested by Mittra [22]. It divides the multicast group into subgroups which are arranged in a hierarchical structure and each has a special group controller. The user revocation overhead is linear in the size of a subgroup. However, this solution introduces group controllers in every subgroup which form many possible points of failure, both for availability and for security.

There are also suggestions to use public key technology, namely generalized Diffie-Hellman constructions, to enable communication efficient group re-keying (e.g. [27]). 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 [12] 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. A very nice feature of this scheme is that the overhead of a rekeying message does not depend on the number of users that are removed from the group. The communication overhead of the scheme 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. The scheme also requires each user to store $O(\log k \log(1/p))$ keys. The main drawback in applying it 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. 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).

B. A Tree Based Scheme

Tree based group rekeying schemes were suggested by Wallner et al. [28] (who used binary trees), and independently by Wong et al. [29] (who consider the degree of the nodes of the tree as a parameter). We concentrate on the scheme of [28] since

[8] 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.

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 scheme when 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 if a user misses some control packet relative to a user deletion operation (e.g., if it temporarily gets disconnected from the network), it needs to either ask for all the missed control packets, or incur in a communication overhead comparable to a user addition operation.

We now describe the scheme of [28]. 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.

The group is initialized as follows. Users are associated to the leaves of a tree of height $\log n$ (see Figure 1). The group controller associates a key k_v to every node of the tree, and sends to each user (through a secure channel) the keys associated to the nodes along the path connecting the user to the root. For example, in the tree of Figure 1, user u_0 receives keys k_{000}, k_{00}, k_0 and k. Notice that the root key k is known to all users and can be used to encrypt group communications.

In order to remove a user u from the group, the group controller performs the following operations. For all nodes v along the path from u to the root, a new key k'_v is generated. New keys are encrypted as follows. Key $k'_{p(u)}$ is encrypted with key $k_{s(u)}$, where $p(u)$ and $s(u)$ denote respectively the parent and sibling of u. For any other node v along the path from u to the root (excluded), key $k'_{p(v)}$ is encrypted with keys k'_v and $k_{s(v)}$. All encryptions are sent to the users. For example, in order to remove user u_0 from the tree of Figure 1 the following encryptions are transmitted (see Figure 2): $E_{k_{001}}(k'_{00}), E_{k'_{00}}(k'_0), E_{k_{01}}(k'_0), E_{k'_0}(k'), E_{k_1}(k')$. It is easy to verify that each user can decrypt only the keys it is entitled to receive.

C. The Improved Scheme

The improved scheme reduces the communication overhead of [28] by a factor of two, from $2\log n$ to only $\log n$. The initialization of the scheme is the same as in [28]. We now describe the user revocation procedure. Let G be pseudo-random generator which doubles the size of its input [5]. Denote by $L(x), R(x)$ the left and right halves of the output of $G(x)$, i.e., $G(x) = L(x)R(x)$ where $|L(x)| = |R(x)| = |x|$. To remove a user u, the group controller associates a value r_v to every node v along the path from u to the root as follows: It chooses $r_{p(u)} = r$ at random and sets $r_{p(v)} = R(r_v) = R^{|u|-|v|}(r)$ for all other v (where $p(v)$ denotes the parent of v). The new keys are defined by $k'_v = L(r_v) = L(R^{|u|-|v|-1}(r))$. Notice that from r_v, one can easily compute all keys $k'_v, k'_{p(v)}, k'_{p(p(v))}$ up to the root key

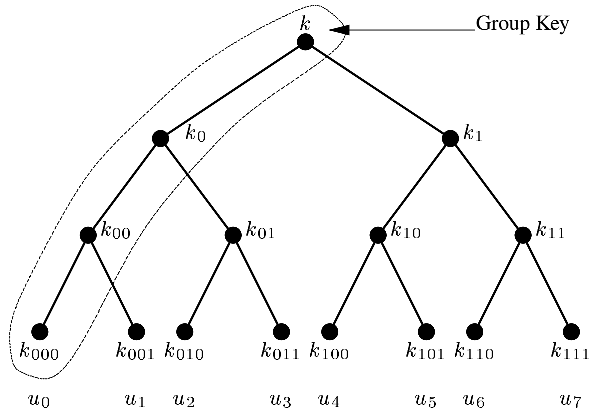

Fig. 1. The tree key data structure (the keys of u_0 are encircled).

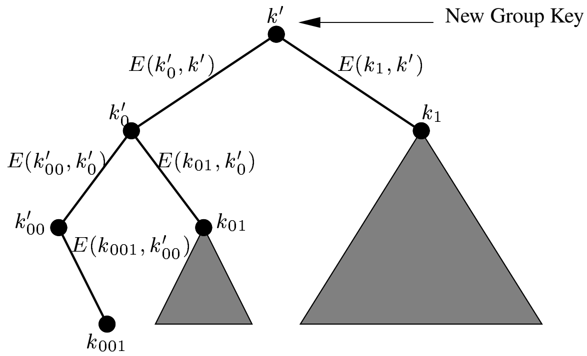

Fig. 2. Key revocation in the basic scheme.

k'. Finally each value $r_{p(v)}$ is encrypted with key $k_{s(v)}$ (where $s(v)$ denotes the sibling of v) and sent to all users. For example, in order to remove user u_0 from the tree of Figure 1, we send encryptions $E_{k_{001}}(r), E_{k_{01}}(R(r)), E_{k_1}(R(R(r)))$. One can easily verify that, under the assumption that G is a cryptographically strong pseudo-random generator, each user can compute from the encryptions all and only the keys it is entitled to receive.

Advantages of the new scheme: This construction halves the communication overhead of the basic scheme to only $\log n$, 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. Sup-

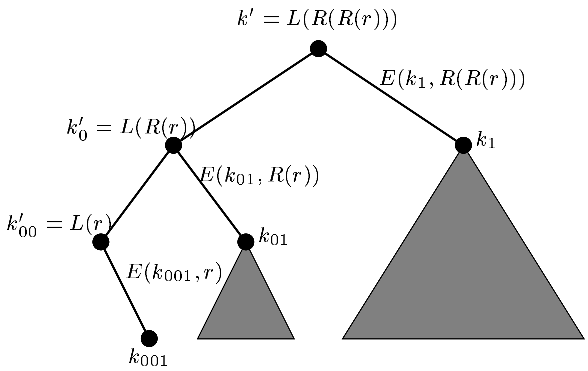

Fig. 3. Key revocation in the improved scheme.

pose that there is an adversary which can break encryptions performed with a subset of the key space (for example keys in 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 [28] is used, the adversary might corrupt the method by which the group controller generates keys in such a way that the root key would always be chosen from the "weak" subspace. However, if our scheme is used, and the pseudo-random generator $G(x) = L(x)R(x)$ is cryptographically strong, then it will be hard to find values r such that the root key $k = L(R(R(\cdots(r)\cdots)))$ is weak.

Independently, McGrew and Sherman [21] 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 [5].

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, J. Bloemer, J. Edmonds, M. Luby and M. Sudan, "Priority Encoding Transmission", *IEEE Tran. on Inf. Th.*, Vol. 42, No. 6, Nov. 1996.

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

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

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

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

[6] 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*, LNCS vol. 740, Springer-Verlag, 1993, 471–486.

[7] R. Canetti, R. Gennaro, D. Herzberg and D. Naor, "Proactive Security: Long-term protection against break-ins", CryptoBytes, Vol. 3, No. 1, Sprint 1997.

[8] R. Canetti and B. Pinkas, "A taxonomy of multicast security issues", internet draft `draft-canetti-secure-multicast-taxonomy--01.txt`, November 1998.

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

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

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

[12] A. Fiat and M. Naor, "Broadcast Encryption", *Advances in Cryptology – CRYPTO '92*, LNCS vol. 839, 1994, pp. 257–270.

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

[14] R. Gennaro and P. Rohatgi, "How to sign digital streams", *Advances in Cryptology – CRYPTO '97*, LNCS, vol. 1294, Springer-Verlag, 180–197.

[15] L. Gong and N. Schacham, "Elements of Trusted Multicasting," TR SRI-CSL-94-03, Computer Science Laboratory, SRI International, May 1994.

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

[17] *IP Security Protocol (ipsec)*, IETF working group. URL: `http://www.ietf.org/html.charters/ipsec-charter.html`

[18] H. Krawczyk, "SKEME: A Versatile Secure Key Exchange Mechanism for Internet", in *IEEE 1996 Symp. on Network and Dist. Systems Security*.

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

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

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

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

[23] M. Naor and O. Reingold, "From Unpredictability to Indistinguishability: A Simple Construction of Pseudo-Random Functions from MACs", *Advances in Cryptology – Crypto '98*, LNCS vol. 1462, Springer-Verlag, 1998, pp. 267–282.

[24] R. L. Rivest, *The MD5 Message Digest Algorithm*, RFC 1321, April 1992.

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

[26] The Secure Multicast Group of the Internet Research Task Force, see `http://www.ipmulticast.com/community/smug/`

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

[28] D.M. Wallner, E.J. Harder and R.C. Agee, "Key Management for Multicast: Issues and Architectures". "draft-wallner-key-arch-00.txt".

[29] C.K. Wong, M. Goudaand S. Lam, "Secure Group Communications Using Key Graphs", *Sigcomm '98*.

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