Internet Research Task Force
IRTF SMUG Internet Draft
draft-irtf-smug-subsetdifference-00.txt
July 2001

Jeff Lotspiech(IBM)
Moni Naor(Weizmann Institute)
Dalit Naor(IBM)

Subset-Difference based Key Management for Secure Multicast

<draft-irtf-smug-subsetdifference-00.txt>

STATUS OF THIS MEMO

Abstract

This document describes a key management mechanism for multicast communication sessions that is based on the 'Subset-Difference' algorithm. The Subset-Difference algorithm is a new revocation scheme which allows a Center (such as a Group Controller/ Key Manager) to send a message so that *every* authorized receiver, but none of the revoked receivers, can decrypt. This message consists of only 2r keys, where r is the number of revoked group members.
In this draft we first describe this new revocation scheme, and then then discuss how it can be used for key management in Secure Multicast applications. Its main advantage is that it eliminates the need for a mechanism that allows individual updates in case a user did not receive or did not perform the required re-keing operations. This is particularly useful in settings with unreliable communication or high rates of packet loss. It also provides an elegant and efficient solution for the backward secrecy problem. The algorithm guarantees complete secure multicast communication even if all revoked users (non group-members)

collude their keys.

1 Introduction and Motivation

The problem of a Center (or Group Controller/Key Server) transmitting data to a large group of receivers so that only a predefined subset is able to decrypt the data is at the heart of a growing number of applications. Among them are pay-TV applications, IP multicast communication, secure distribution of copyright-protected material (e.g. music) and audio streaming. The area of Broadcast Encryption deals with methods to efficiently broadcast information to a dynamically changing group of users who are allowed to receive the data. It is often convenient to think of it as a Revocation Scheme, which addresses the case where some subset of the users are excluded from receiving the information.

One special case is when the receivers are stateless. In such a scenario, a (legitimate) receiver is not capable of recording the past history of transmissions and change its state accordingly. Instead, its operation must be based on the current transmission and its initial configuration. Stateless receivers are particularly important for the case where the receiver is a device that is not constantly on-line, such as a media player (e.g. a CD or DVD player where the ``transmission" is the current disc), a satellite receiver (GPS).

Recently a new revocation scheme, called the 'Subset-Difference' scheme, has been proposed [NNL01]. The Subset-Difference algorithm is especially suitable for stateless receivers. Its main advantage over the previous is that it requires to transmit only 2r (or 1.25 on average) keys instead of 2rlogN keys in order to revoke r users from a set of N users, regardless of the coalition size, while maintaining a single decryption at the user's end. In return, it requires every receiver to store log^2N keys instead of logN keys. The receiver needs to employ 1 decryption for every re-keying event plus logN applications of a pseudo-random generator.

In Multicast applications the Center can be viewed as the Group Controller/Key Server which transmits key updates. Receivers are typically perceived as statefull, namely they may update their configuration between transmission. However, this requires all users to be connected during a re-keying event and to change the internal state accordingly. This is not a realistic assumption in scenarios with a high loss rate of packets. The natural approach to solve this problem is to introduce a mechanism that allows an individual update [RFC2627][WGL]. Taking the stateless approach gets rid of the need for such mechanism.

This draft introduces the new Subset-Difference revocation algorithm(in Sections 2 and 3) and shows its applicability to the problem of Secure Multicast(Section 4).
Specifically, it proposes a few modes in which the new Subset-Difference revocation scheme can be used to provide secure group communication. When applied to Key Management in multicast scenarios, it requires a transmission of 2r keys at a re-keying event, where r is the number of revoked-members. This automatically

(i) avoids individual re-keying events due to lack/failure in communication,

(ii) achieves backward secrecy in a straightforward manner.

Notation: We denote by N the total number of users in the system and by r the size of the revoked set R.

1.1 Previous Work

The area of Broadcast Encryption was first formally studied (and coined) by Fiat and Naor in [FN] and has received much attention since then. In principle any scheme that works for the connected mode, where receivers can remember past communication, may be converted to a scheme for stateless receivers (such a conversion may require to include with any transmission the entire `history' of revocation events). An overview of previous algorithms to this problem, especially when adapted to the stateless receiver scenario, can be found in [NNL].

The logical-tree-hierarchy (LKH) scheme, suggested independently by Wallner et. al. [RFC2627] and Wong et. al. [WGL], is designed for the connected mode for multicast re-keying applications. It revokes a single user at a time, and updates the keys of all remaining users. This requires a transmission of 2logN keys to revoke a single user, or 2rlogN keys to revoke r users; each user should store log N keys and the amount of work each user should do is log N encryptions or rlogN for r revocations (the expected number is O(r) for an average user). These bounds are improved in [CGIMNP],[CMN],[MS], but unless the storage at the user is extremely high they still require a transmission of length rlogN.

2. Subset-Cover Algorithms

This section describes a family of revocation algorithms called the Subset-Cover algorithms, which the 'Subset-Difference' belongs to.

2.1 Preliminaries - Problem Definition

Let N be the set of all users |N|=n, and R be a group of r users whose decryption privileges should be revoked. The goal of a revocation algorithm is to allow a center to transmit a message

M to all users such that any non-revoked user u can decrypt the message correctly, while even a coalition consisting of all members of R cannot decrypt it.

A system consists of three parts: (1) An initiation scheme, which is a method for assigning the receivers secret information that will allow them to decrypt. (2) The broadcast algorithm - given a message M and the set R of users that should be revoked, outputs a ciphertext message M' that is broadcast to all receivers. (3) A decryption algorithm - a (non-revoked) user that receives ciphertext M' using its secret information should produce the original message M. If receivers are stateless, the output of the decryption should be based on the current message and the secret information only.

2.2 A Subset-Cover Algorithm and its Components

A Subset-Cover algorithm defines a collection of subsets $S_1, \ldots, S_w$ of users. Each subset S_j is assigned (perhaps implicitly) a long-lived key L_j; each member u of S_j should be able to deduce L_j from its secret information. Given a revoked set R, the remaining users $N \setminus R$ are partitioned into disjoint subsets $S_{i1}, \ldots, S_{im}$ so that the union of these subsets is $N \setminus R$ and a session key K is encrypted m times with $L_{i1}, \ldots, L_{im}$.

Specifically, the algorithm uses two encryption schemes:

i. A method $F_K: \{0,1\}^* \to \{0,1\}^*$ to encrypt the message itself. The key K used will be chosen fresh for each message M - a session key - as a random bit string. F_K should be a fast method and should not expand the plaintext. The simplest implementation is to Xor the message M with a stream cipher generated by K.

ii. A method to deliver the session key to the receivers, for which we will employ an encryption scheme. The keys L here are long-lived. The simplest implementation is to make $E_L:\{0,1\}^l \to \{0,1\}^l$ a block cipher.

The algorithm consists of three components:

i. Scheme Initiation :

Every receiver u is assigned private information I_u. For all $1 \le i \le w$ such that u is in S_i, I_u allows u to deduce the key L_i corresponding to the set S_i. Note that the keys L_i can be chosen as a function of other (secret) information, which is the case in the subset-difference algorithm.

ii. The Broadcast Algorithm at the Center:

- Choose a session encryption key K.
- Given a set R of revoked receivers, the center finds a partition of the users in $N \setminus R$ into disjoint subsets $S_{i1}, \ldots, S_{im}$.

Let L_i1, ...L_im be the keys associated with the above subsets.
- The center encrypts K with keys L_i1, ..., L_im
and sends the ciphertext

[i1,i2,...,im, E_L_i1(K), E_L_i2(K),...,E_L_im(K)], F_K (M)

The portion in square brackets preceding F_K (M) is called the 'Header' and F_K (M) is called the 'Body'.

iii. The Decryption step at the receiver u, upon receiving a broadcast message
[i1, i2,...,im, C_1, C_2,..., C_m] , M'

- Find ij such that u is in S_ij (in case u is revoked the result is null).
- Extract the corresponding key L_ij from I_u.
- Compute D_L_ij(C_j) to obtain K.
- Compute D_K(M') to obtain and output M.

Below we specify the implementation of these components by the Subset-Difference algorithm. The algorithm is evaluated based upon three parameters:
- Message Length - the length of the header that is attached to F_{K}(M), which is proportional to m, the number of sets in the partition covering N\R.
- Storage size at the receiver - how much private information (typically, keys) does a receiver need to store. For instance, I_u could simply consist of all the keys of the subsets that u belongs to, or if the key assignment is more sophisticated it should allow the computation of all such keys.
- Message processing time at receiver. We often distinguish between decryption and other types of operations.

2.3 Comparison to the Logical Key Hierarchy (LKH) approach of [RFC2627] and [WGL]

Readers familiar with the Logical Key Hierarchy approach, in particular the tree method of [RFC2627] and [WGL], may find it instructive to compare it with the Subset-Cover algorithm approach. In LKH receivers are viewed as leaves in a full binary tree and are grouped into subsets according to subtrees in the full tree. An independent label is assigned to each node in the binary tree, thus providing a key for each subset of leaves that form a subtree. However, these labels are used quite differently - the re-keying employed by the LKH scheme changes some of these labels at every revocation. In the Subset-Cover framework, labels correspond to the long-term keys and therefore are static (never change); what changes is a single session key. The separation of the labels and the session key has a consequence on the message length as shown in [NNL].

One can extend the LKH approach to handle r revocations at a time in the following manner: For a batch of r revocations, no label is changed more than once, i.e. only the "latest" value is transmitted and used. We call it the 'clumped re-keying method'. In this variant the number of encryptions is roughly rlongN/r, but it requires log N decryptions at the user, (as opposed to a single decryption in our framework).

3. The Subset-Difference Algorithm

The subset-difference algorithm falls under the subset-cover framework. Its main advantage is that it partitions the non-revoked users into at most 2r-1 subsets (or 1.25r on average), whereas previously known algorithm required O(rlog N) subsets, effectively reducing the message length accordingly. In return, the number of keys stored by each receiver is 0.5log^2N, an increase by a factor of of 0.5log N from previous known algorithms. The key characteristic of the Subset-Difference method, which essentially leads to the reduction in message length, is that in this method any user belongs to a substantially large number of subsets (O(N) subsets). The challenge is to devise an efficient procedure to succinctly encode this large set of keys at the user.

3.1 The subset description

The receivers are viewed as leaves in a complete binary tree. The collection of subsets S_1,..., S_w defined by this algorithm corresponds to subsets of the form "a group of receivers G_1 minus another group G_2", where G_2 is a subset of G_1. The two groups G_1, G_2 correspond to leaves in two full binary subtrees. Therefore a valid subset S is represented by two nodes in the tree (v_i, v_j) such that v_i is an ancestor of v_j. We denote such subset as S_{i,j}.

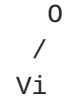
- A leaf u is in S_{i,j} iff it is in the subtree rooted at v_i but not in the subtree rooted at v_j,

or in other words

- u is in S_{i,j} iff v_i is an ancestor of u but v_j is not.

Figure 1 depicts S_{i,j}. Note that every subtree is also a subset in this collection: specifically, a subtree appears here as the difference between its parent and its sibling. The only exception is the full tree itself, and we will add a special subset for that. We postpone the description of the key assignment till later; for the time being assume that each subset S_{i,j} has an associated key L_{i,j}.

Figure 1 - The Subset Difference Method.



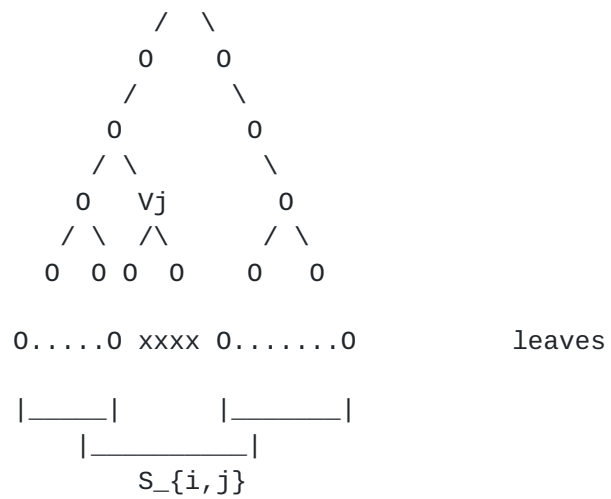


Figure 1: Subset S_{i,j} contains all leaves O that are not excluded (x'ed).

3.2 The Cover

For a given set R of revoked receivers, let u_1,...,u_r be the leaves corresponding to the elements in R. The Cover is a collection of disjoint subsets S_{i1,j1}, S_{i2,j2},...,S_{im,jm} which partitions N\R. Below is an algorithm for finding the cover, followed by an example, and an analysis of its size (number of subsets).

Finding the Cover:

The method partitions N\R into disjoint subsets S_{i1,j1}, S_{i2,j2},...,S_{im,jm} as follows: consider the backbone tree T induced by the set R of vertices and the root, i.e. the minimal subtree of the full binary tree that connects all the leaves in R. We build the subsets collection iteratively, maintaining a the backbone tree T with the property that any u in N\R that is below a leaf of T has been covered. We start with the initial backbone tree T and then iteratively remove nodes from T (while adding subsets to the collection) until T consists of just a single node:

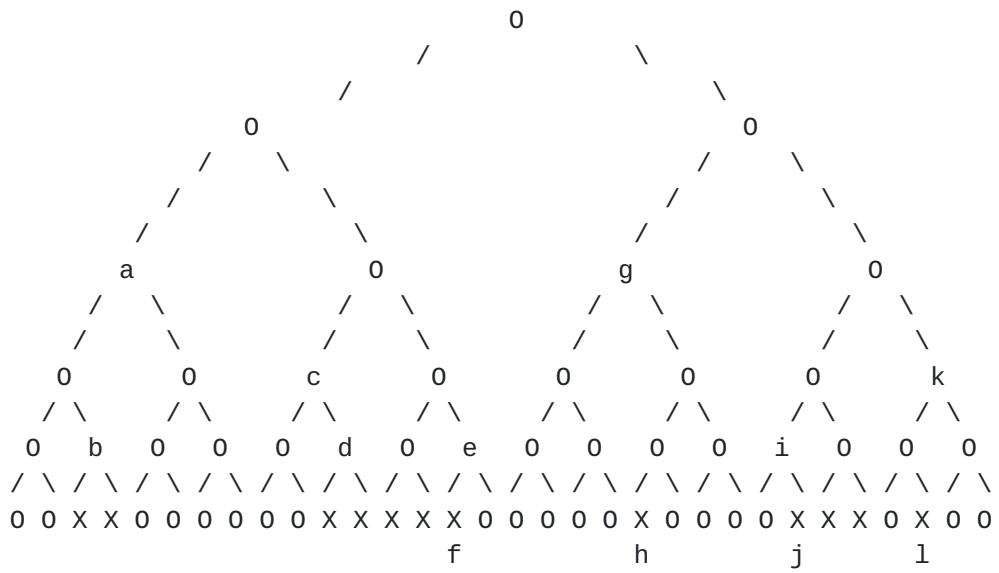
(i) Find two leaves v_i and v_j in T such that the least-common-ancestor v of v_i and v_j does not contain any other leaf of T in its subtree. Let v_l and v_k be the two children of v such that v_i a descendant of v_l and v_j a descendant of v_k. (If there is only one leaf left, make v_i=v_j to the leaf, v to be the root of T and v_l=v_k=v.)
(ii) If v_l is not v_i then add the subset S_{l,i} to the collection; likewise, if v_k is not v_j add the subset S_{k,j} to the collection.
(iii) Remove from T all the descendants of v and make it a leaf.

An alternative description of the cover algorithm is as follows:

Consider maximal chains of nodes with outdegree 1 in T. More precisely, each such chain is of the form [v_i1, v_i2,... v_il] where (i) all of v_i1, v_i2,...,v_il-1 have outdegree 1 in T (ii) v_il is either a leaf or a node with outdegree 2 and (iii) the parent of v_i1 is either a node of outdegree 2 or the root. For each such chain where l<=2 add a subsets S_{i1,il} to the cover. Note that all nodes of outdegree 1 in T are members of precisely one such chain.

3.2.1 Example.

For example, consider the 5-level tree depicted below with 32 leaves, 12 of which are revoked (marked with X). The cover consists of 6 subset-differences which are
S_{a,b} , S_{c,d} , S_{e,f} , S_{g,h} , S_{i,j} , S_{k,l}



In the above example, subset S_{g,h} demonstrates the case in which the reduction in cover size obtained due the definition of subsets as *differences* is the most dramatic. The revoked leaf h is a 'singleton' within the subtree rooted at g; hence a single subset S_{g,h} suffices to cover the remaining leaves in the subtree. A method that groups leaves according to subtrees would require 3 subsets (or in general, the order of logN subsets) to cover these remaining leaves.

3.3 The cover size:

A cover can contain at most 2r-1 subsets for any set of r revocations. Every iteration increases the number of subsets by at most two (in step (2)) and reduces the number of the Steiner

leaves by one (in Step (3)), except the last iteration
that may not reduce the number of leaves but adds only one subset.
Starting with r leaves, the process generates the total of 2r-1
subsets. Moreover, every non-revoked u is in exactly one
subset, the one defined by the first chain of nodes of outdegree
1 in ST(R) that is encountered while moving from u towards
the root. This encounter must hit a non-empty chain, since the
path from u to the root cannot join ST(R) in an outdegree 2
node, since this implies that u is in R.

The above analysis is a worst-case analysis and
there are instances which actually require 2r-1 sets. However,
if the set of revoked leaves is random, then both analytical
analysis and experimental results show tighter bounds. In fact,
the average number of subsets in a cover is 1.25r.

3.4 Key assignment to the subsets

We now define what information each receiver must store. If
each receiver needs to store explicitly the keys of
all the subsets it belongs to, the storage requirements would
expand tremendously: consider a receiver u; for each complete
subtree T_k it belongs to, u must store a number of keys
proportional to the number of nodes in the subtree T_k that are
not on the path from the root of T_k to u. There are
log N such trees, one for each height 1<=k<=log N,
yielding the total of O(N) keys.

We therefore devise a key assignment method that requires a
receiver to store only O(log N) keys per subtree, for the total
of O(log^2 N) keys.

While the total number of subsets to which a user u belongs is
O(N), these can be grouped into logN clusters defined by the
first subset i (from which another subsets is subtracted). The
way we proceed with the keys assignment is to choose for each
1<=i<=N-1 corresponding to an internal node in the full
binary tree a random and independent value LABEL_i. This
value should *induce* the keys for all legitimate subsets of
the form S_{i,j} using pseudo-random functions.

Let G be a (cryptographic) pseudo-random sequence generator
that *triples* the input, i.e. whose output length is three times
the length of the input; let G_L(S) denote the left third of the
output of G on seed S, G_R(S) the right third and G_M(S)
the middle third.

Consider now the subtree T_i (rooted at v_i). We will use the
following top-down labeling process: the root is assigned a label
LABEL_i. Given that a parent was labeled S, its two
children are labeled G_L(S) and G_R(S) respectively. Let

LABEL_{i,j} be the label of node v_j derived in the subtree T_i from LABEL_i. Following such a labeling, the key L_{i,j} assigned to set S_{i,j} is G_M of LABEL_{i,j}. Note that each label induces three parts:

G_L - the label for the left child,
G_R - the right child, and
G_M - the key at the node.

The process of generating labels and keys for a particular subtree is depicted in Figure 2. For such a labeling process, given the label of a node it is possible to compute the labels (and keys) of all its descendants. On the other hand, without receiving the label of an ancestor of a node, its label is pseudo-random and for a node j, given the labels of all its descendants (but not including itself) the key L_{i,j} is pseudo-random (LABEL_{i,j}, the label of v_j, is not pseudo-random given this information simply because one can check for consistency of the labels). It is important to note that given LABEL_i, computing L_{i,j} requires at most log N invocations of G.

Figure 2 - Key Assignment to the Subsets.

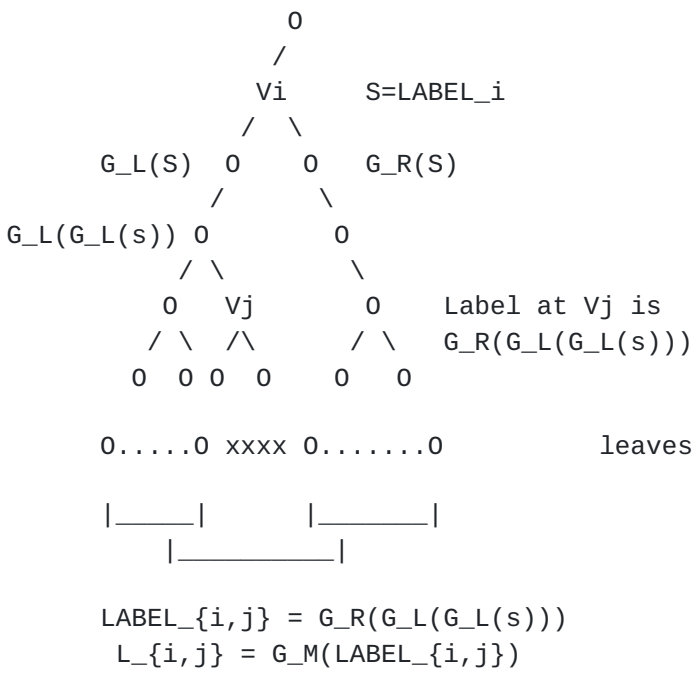


Figure 2 - Generation of LABEL_{i,j} and the key L_{i,j}.

We now describe the information I_u that each receiver u gets in order to derive the key assignment described above. For each subtree T_i such that u is a leaf of T_i the receiver u should be able to compute L_{i,j} iff j is not an ancestor of u. Consider the path from v_i to u and let v_i1,v_i2,...,v_ik be the nodes just ``hanging

off" the path, i.e. they are adjacent to the path but not ancestors of u (see Figure 3). Each j in T_i that is not an ancestor of u is a descendant of one of these nodes. Therefore if u receives the labels of v_i1,v_i2,...,v_ik as part of I_u, then invoking G at most log N times suffices to compute L_{i,j} for any j that is not an ancestor of u.

Figure 3 - Key Assignment to Receivers

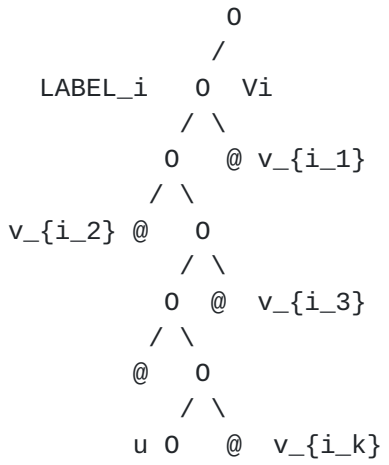


Figure 3 - Leaf u receives the labels of v_{i_1},...,v_{i_k} (marked with @) that are induced by the label LABEL_i of v_i.

As for the total number of keys (in fact, labels) stored by receiver u, each tree T_i of depth k that contains u contributes k-1 keys (plus one key for the case where there are no revocations), so the total is 0.5log^2 N +0.5log N + 1.

3.5 Decryption Step:

At decryption time, a receiver u first finds the subset S_{i,j} such that u is in S_{i,j}, and computes the key corresponding to L_{i,j}. The evaluation of the subset key takes now at most log N applications of a pseudo-random generator. After that, a single decryption is needed.

3.5 Hierarchical Revocation

Suppose that the receivers are grouped in a hierarchical manner, and that it is desirable to revoke a group that consists of the subordinates of some entity, without paying a price proportional to the group size (for instance all the players of a certain manufacturer). The subset-difference algorithm lends

itself to hierarchical revocation naturally, given the tree structure. If the hierarchy corresponds to the tree employed by the methods, then to revoke the receivers below a certain node counts as just a single user revocation.

It can be shown that we can remove any collection of m subsets and cover the rest with 3m-1 subsets. Hence, the hierarchical revocation can be performed by first constructing m sets that cover all revoked devices, and then covering all the rest with 3m-1, yielding the total of 4m sets.

3.5 Storage at the Key Server

In the proposed algorithm a unique label is associated with each node in the tree. Storing these labels explicitly at the Center can become a serious constraint. However, these labels can be generated at the center by applying a secret pseudo-random function on the name of the node without affecting the security of the scheme. This reduces the storage required by at the Key Server to the single key of the pseudo-random function.

Furthermore, it may be desirable to distribute the center between several servers with the objective of avoiding a single or few points of attack. In such a case the distributed pseudo-random functions of [NPR] may be used to define the labels.

4. Applications to Multicast

4.1. Stateless receivers in Multicast

In Secure multicast, the Center corresponds to the Group Controller/Key Server. Typically, in such applications it is assumed that receivers may update their keys [WGL][RFPC2527]. This update is referred to as a re-keying event and it requires all users to be connected during this event and change their internal state (keys) accordingly. However, even in the multicast scenario it is not reasonable to assume that all the users receive all the messages and perform the required update. Therefore some mechanism that allows individual update must be in place. Taking the stateless approach eliminates the need for such a mechanism, as discussed below. In case the number of revocations is not too large this may yield a more manageable solution. This is especially relevant when there is a single source for the sending messages or when public-keys are used.

4.2. Backward secrecy

Revocation in itself lacks backward secrecy in the following sense: a constantly listening user that has been revoked from the system records all future transmission (which it can't decrypt

anymore) and keeps all ciphertexts. At a later point it gains a valid new key (by re-registering) which allows decryption of all past communication. Hence, a newly acquired user-key can be used to decrypt all past session keys and ciphertexts. The way that LKH multicast key assignment protocols propose to achieve backward secrecy is to perform re-keying when new users are added to the group (such a re-keying may be reduced to only one way chaining, known as {LKH+}), thus making such operations non-trivial. We point out that in the subset-difference algorithm it may be easier: At any given point of the system include in the set of revoked receivers all identities that have not been assigned yet. As a result, a newly assigned user-key cannot help in decrypting an earlier ciphertext. Note that this is feasible since we assume that new users are assigned keys in a consecutive order of the leaves in the tree, so unassigned keys are consecutive leaves in the complete tree and can be covered by at most log N sets. Hence, the unassigned leaves can be treated with the hierarchical revocation technique, resulting in adding at most log N revocations to the message.

We note that, unlike the methods of [WGL][RFPC2527], the keys associated with a revoked leaf can not be re-assigned to a new user at a later point. This requires the initial design to have some assumption the maximum number of anticipated users throughout the lifetime of the system. A tree height of 32 provides an ample number of users (about 4 billion of them) that is appropriate for most systems.

4.3. Using the Subset-Difference Revocation for Group Communication

There are a number of modes in which the Subset-Difference algorithm can be applied.

4.3.1. The Centralized Group-Key Update Mode

> In the most natural mode, we assume the existence of a Group Controller/Key Server which issues a new "Group Key" when necessary. The receivers communicate among each other using the current "Group Key". A new Group key is issued to all by multicasting a new Header (recall that a Header is essentially a new session key, encrypted with all subsets in the cover). The header is generated by the Group Controller/Key Server, while revoking all receivers that are currently *not* members of the group, including the 'unassigned' ones (leaves that have not been assigned yet to actual receivers; recall that this can be done 'cheaply' with our method).
>
> A group-key update requires bandwidth of at most 2r keys (or 1.25r keys on the average). This operation should be done every time an existing member is revoked, or terminated, or when a new member joins the group.

Every member who has been disconnected for some time and missed (possible a few) group updates can simply request the current Header from the Group Controller/Key Server. Note that this Header can be sent in the clear, without any point-to-point encryption, provided that the server is authenticated, as only currently authorized receivers can decrypt the Header.

The later is the main advantage of the proposed scheme over the traditional LKH approach, which requires to transmit to a disconnected member *all the history* of re-keying events since the receiver has last been connected (logN keys per re-keying event). However, a group-key update now requires bandwidth of 2r keys, instead of 2logN keys.

4.3.2 The 'Floating Header' Mode

Again, we assume the existence of a current "Group Key", and its corresponding Header, at any given time. However, in this mode the Header is attached to *every* message in the system (not only those involving the Key Server) by the party sending the message.

With this approach, Group Key updates take place naturally; whenever needed, the Group Controller issues a new Header by multicasting to all and connected receivers update their current Header. It does not require a disconnected server wishing to listen to incoming message to update its Header upon reconnecting - it can simply wait for the next message to arrive. The only exception occurs when such receiver needs to send a message before it gets a new one; this involves a request for a Header from any connected member of the group, including but not necessarily, the Key Server itself. Note that a message authentication is in place, as the Header which is part of the message needs to be authenticated.

The disadvantage of this mode is the additive increase in bandwidth - now every message requires 2r extra keys, in addition to the message body itself.

4.3.3 The Hybrid Mode

The hybrid mode does not attach the Header with every message; it assumes that all authorized parties have an access to the current, most updated, group-key. It therefore calls for a mechanism for an individual to obtain the most current group key.

A possible solution is for the Key Server to have a certain

authenticated location (e.g. a url) that 'posts' the current Header. In the straightforward implementation, a sender is then required to fetch the Header from this location before sending the message and encrypt the message with the session key provided by the Header; analogously, the receiver fetches the recent Header and decrypts the message.

This approach avoids the need for a synchronized re-keying event; it puts the burden on the senders/receivers to obtain the most current group key for every encryption and decryption, including a receiver that has been disconnected for some time. The disadvantage is the fact that the 'Header-posting' location has now become a bottleneck.

Few optimization attempts are in place here. The most important one is to reduce the load on the posting location via mirroring or other techniques. Alternatively, one may try to minimize the accesses to the location by putting a mechanism allowing a receiver to quickly determine whether the posted Header had changed since its last lookup, or whether its most updated session key decrypts the message.

5. REFERENCES

[CGIMNP] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas, Multicast Security: A Taxonomy and Some Efficient Constructions. Proc. of INFOCOM '99, Vol. 2, pp. 708--716, New York, NY, March 1999.

[CMN] R. Canetti, T. Malkin, K. Nissim, Efficient Communication-Storage Tradeoffs for Multicast Encryption. EUROCRYPT 1999: pp.\ 459--474.

[FN] A. Fiat and M. Naor, Broadcast Encryption. In "Advances in Cryptology" - CRYPTO '93, Lecture Notes in Computer Science 773, Springer, 1994, pp.\ 480---491.

[MS] D. McGrew, A.T. Sherman, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees", submitted to IEEE Transactions on Software Engineering (May 20, 1998).

[NNL] D. Naor, M. Naor and J. Lotspiech, Revocation and Tracing Schemes for Stateless Receivers, to appear in Crypto 2001. A full version of paper appears in http://www.wisdom.weizmann.ac.il/~naor/

[NPR] M. Naor, B. Pinkas and O. Reingold, Distributed Pseudo-random Functions and KDCs, Advances in Cryptology - EUROCRYPT 1999, Lecture Notes in Computer Science, vol.\ 1592 Springer, 1999, pp.\ 327--346.

[RFC2627] D. M. Wallner, E. Harder, R. C. Agee, Key Management for Multicast: Issues and Architectures, September 1998.

[WGL] C. K. Wong, M. Gouda and S. Lam, Secure Group Communications Using Key Graphs, SIGCOMM 1998.

Author Address:

Jeff Lotspiech
650 **Harry Rd., San Jose, CA 95120**
lotspiech@almaden.ibm.com
office: 408-927-1851
fax: 408-927-3497

Moni Naor
Weitzman Institute

Dalit Naor
IBM Corporation