

Key Management for Large Dynamic Groups:
One-Way Function Trees and Amortized Initialization
<draft-irtf-smug-groupkeymgmt-oft-00.txt>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

Copyright Notice

Copyright (C) The Internet Society (2000). All Rights Reserved.

Abstract

We present and implement a scalable method for establishing group session keys for secure large, dynamic groups such as multicast sessions. Our method is based on a novel application of One-Way Function Trees (OFTs). The number of keys stored by group members, the number of keys broadcast to the group when new members are added or evicted, and the computational efforts of group members, are logarithmic in the number of group members. The method provides perfect forward and backward security: evicted members cannot read future messages, even with collusion by arbitrarily many evicted members, and newly admitted group members cannot read previous messages. In comparison with the Logical Key Hierarchy (LKH) of Wallner et al., our algorithm roughly halves the number of bits that need to be broadcast to members in order to re-key after a member is added or evicted. In addition, and unlike LKH, our algorithm has the option of being member contributory in that members can be allowed to contribute entropy to the group key. Running on a Pentium with 64 MB of RAM, our prototype has handled groups with up to 100,000 members.

Contents

| | | |
|-----|---|----|
| 1. | Introduction | 02 |
| 2. | Previous Work | 03 |
| 2.1 | Key Establishment Algorithms for Large Groups | 03 |
| 2.2 | Managing Large Groups | 05 |
| 2.3 | Authentication in Large Groups | 06 |
| 2.4 | Multicast Security | 06 |
| 3. | Group Operations and Their Security Requirements | 06 |
| 3.1 | Operations | 07 |
| 3.2 | Security Requirements | 07 |
| 4. | One-Way Function Trees | 08 |
| 4.1 | Structure of an OFT | 09 |
| 4.2 | Algorithms | 11 |
| 4.3 | Properties | 13 |
| 4.4 | Enhanced OFT+ and LKH+ Algorithms | 14 |
| 4.5 | Comparisons with Other Leading Key-Establishment Methods | 15 |
| 5. | Design Choices, Implementation Issues, and Experience | 18 |
| 5.1 | Design Choices | 18 |
| 5.2 | Implementation Issues and Experience | 18 |
| 6. | Amortized Group Induction | 20 |
| 6.1 | Induction Model | 20 |
| 6.2 | Induction Algorithm | 21 |
| 6.3 | An Example of Induction | 22 |
| 7. | Conclusion and Open Problems | 22 |
| 8. | Security Considerations | 24 |
| 9. | References | 24 |
| 10. | Acronyms and Abbreviations | 31 |
| 11. | Authors' Addresses | 31 |
| 12. | Acknowledgments | 32 |

1. Introduction

Efficiently managing cryptographic keys for large, dynamically changing groups is a difficult problem. Every time a member is evicted from a group, the group key must change; it may also be required to change when new members are added. The members of the group must be able to compute a new key efficiently, while arbitrary coalitions of evicted members must not be able to obtain it. Communication costs must also be considered.

Real-time applications, such as secure audio and visual broadcasts, pay TV, secure conferencing, military command and control, and controlling access to the Global Positioning System (GPS) service need very fast re-keying so that changes in group membership are not disruptive. To deal with large group sizes (e.g. 100,000 members), we seek solutions whose re-keying operations "scale" well in the sense that time, space, and broadcast requirements of the method grow at most logarithmically in the group size. Key management for these applications should be able to take advantage of efficient broadcast channels, such as radio broadcast and Internet multicast.

We present a new practical algorithm for establishing shared, secret group communications keys in large, dynamic groups. Our algorithm [McG98, Hard97, Bal98, Bal98b], which is based on a novel application of one-way function trees, scales logarithmically in group size. In comparison with previously published methods, our algorithm approximately halves the required broadcast size. Significantly, our algorithm has the option of being member contributory in that members can be allowed to contribute entropy to the group key. This document describes our algorithm, compares it with other approaches, and discusses implementation issues based on our implementation of this algorithm.

2. Previous Work

The broad and challenging problem of establishing keys for large dynamic groups touches upon a large body of previous work, including algorithms for key establishment, group management, authentication in large groups, and multicast security. As for key management, unfortunately very little work has been done on this critical problem in the open literature (e.g. see Schneier [Sch96, pp. 169-187]).

2.1 Key Establishment Algorithms for Large Groups

Prior methods for establishing keys in groups fall roughly into five categories: simple methods that scale linearly in group size, information-theoretic approaches, algorithms based on group Diffie-Hellman key exchange, hybrid methods that trade off information-theoretic security for reduced storage requirements, scalable methods based on hierarchies, and other techniques. Sherman [Bal98, Section 7], Kruus [Kru98], Menezes [Men97], and Just [Jus94] survey some of these methods.

Unfortunately, the information-theoretic approaches [Blu92, Sti96, Chi89] require exponential space to achieve forward secrecy against arbitrarily many colluding evicted members. Although the group Diffie-Hellman methods [Ste96-7, Bur97, Bur94, Ate98] offer distributed functionality, which might be attractive for some applications, they typically suffer from a linear number of expensive public-key operations. Similarly, the hybrid approaches [Fia93, Ber91] scale linearly or worse. As for other techniques (e.g. [Blo90, Gon89]) that do not fall nicely into any of the above categories, we have not found any that provide adequate security.

When the network is a tree, group Diffie-Hellman methods can require only a logarithmic number of operations [Bur97]. Nevertheless, distributed computation is not appropriate for all applications, and the basic unit of cost for all Diffie-Hellman methods still includes expensive public-key operations.

Therefore, for large groups, the leading candidates are the hierarchical methods, which scale logarithmically in group size.

Balenson, McGrew, Sherman expires February 25, 2001

[Page 03]

INTERNET-DRAFT

August 25, 2000

Recently, three hierarchical methods have been proposed:

the Logical Key Hierarchy (LKH) of Wallner, Harder, and Agee [Wal97, Harn99b],

the One-Way Function Tree (OFT) of McGrew and Sherman [McG98], and

the One-Way Function Chain (OFC) of Canetti et al. [Can99b, Section 4.2].

LKH is a top-down method in that it "pushes" new group keys down the key tree; OFT and OFC are bottom-up methods in that they derive new group keys from the leaves up to the root. In comparison with LKH, the bottom-up OFC and OFT methods broadcast approximately 50% less bits during a single member evict operation. Only OFT, however, has the option of being member contributory.

The first Internet Draft on the LKH algorithm appeared in July 1997, authored by Wallner, Harder, and Agee [Wal97]. The name LKH (suggested by Sherman) was adopted by Harney and Harder [Harn99b] in their 1999 implementation of the algorithm. In July 1997, Wong et al. [Won97, Won98] analyzed generalizations of LKH, and in 1998, Caronni et al. [Car98] independently published a similar idea to LKH. OFT was developed at TIS Labs at Networks Associates, Inc., as part of the DARPA-funded Dynamic Cryptographic Context Management (DCCM) Project [Bal98]. In November 1997, Sherman [Hard97] made the first presentation on OFT. The OFT algorithm is the subject of this document.

Importantly, in 1999, Canetti et al. [Can99b, Section 4.2] propose a variation of OFT, which we refer to as One-Way Function Chain (OFC). In OFC, there is always a functional relationship among the node secrets along the path in the key tree from some leaf to the root. OFC shares the broadcast reduction of OFT over LKH; additionally, OFC is slightly simpler than OFT. Canetti claims (without formal written proof) that both LKH and OFC can be proven secure in the sense that a computationally limited adversary cannot distinguish between an actual set of OFC messages versus a set of randomly chosen fake messages.

Recent work by Canetti et al. [Can99], which mentions OFT, gives a generalization and improvement of LKH based on a storage-communication tradeoff. They show that, for a group of n users with user storage of $b+1$ keys, re-keying can be done by broadcasting $O((b+1)n^{1/b}) - b$ keys with manager storage of approximately $O(n)$. They also prove a lower bounds of $b+1$ member storage and $n^{1/b}$ keys broadcast, which is tight for constant b . An interesting special

case of their tradeoff is $O(\lg n)$ member storage, $O(\lg n)$ keys broadcast, and $O(n/(\lg n))$ manager storage. [Note: " n^e " denotes n raised to the e -th power, and " \lg " denotes log base 2.]

Balenson, McGrew, Sherman expires February 25, 2001

[Page 04]

INTERNET-DRAFT

August 25, 2000

Selcuk, McCubbin, and Sidhu [Sel00] note that, if the a priori member eviction probabilities are known, then these probabilities can be used to advantage by placing members with high eviction probabilities near the root of any key tree. Poovendran and Baras [Poo99] also studied similar ideas. Unfortunately, for most applications, such probabilities are likely not to be known.

The hierarchical methods of Ballardie [Bal96, Bal97] and Harkins [Hark98a] are scalable but require trusted routers.

In addition, the linear Single Key Distribution Center (SKDC) approach is attractive for its simplicity -- at least for relatively small groups (e.g. see [Harn99a, Harn99c, Harn97a-b]).

2.2 Managing Large Groups

Managing large groups is important for group communications and for a variety of other applications, including distributed fault-tolerant computing and virtual private networks. Researchers have addressed important group-management issues including defining group membership, supporting distributed fault-tolerant applications, and effecting decentralized dynamic group management.

An important problem in group management is the problem of defining group membership. To support the design of secure, asynchronous, distributed, fault-tolerant systems, Michael Reiter [Rei94] devised a group-membership protocol that tolerates malicious corruption of up to one third of the participants. This protocol is useful in building systems that are robust against limited failures (e.g. hardware failure of some nodes), and that through threshold techniques distribute trust among two or more parties. By contrast, a single group controller is a single point of failure and hence not fault tolerant.

Although group keying is often used to secure group communications, another application of group keying arises in security architectures for distributed fault-tolerant computing. For example, Kenneth Birman [Rei93] and his research group at Cornell have studied how the notion of a secure process group can be used to effect secure, distributed, fault-tolerant computing. Their efforts include the ISIS, Horus [Hor], and Ensemble [Hay98] Systems, which provide a framework and toolkit for developing distributed applications.

Birman [Rod97] and his group have also applied similar ideas to design a virtual private network that can handle network faults, decentralized management, and dynamic membership. Unfortunately,

their "solution currently scales [only] to approximately 100 machines" [Rod97, p. 14]. Also, they claim, that for data confidentiality, "[their] keys are so dynamic and short-lived [changed once a minute] that the approach could be used with a fairly weak cryptographic scheme" [Rod97, p. 1].

Balenson, McGrew, Sherman expires February 25, 2001

[Page 05]

INTERNET-DRAFT

August 25, 2000

Li Gong [Gon96, Keu96] designed and implemented a toolkit called Enclaves for building secure user-level group applications. Enclaves enable users to form virtual private networks on the Internet dynamically. His methods, however, do not scale to large groups. Other recent work in group management includes research by Fenner [Fen97].

2.3 Authentication in Large Groups

There are several ongoing projects to develop infrastructures to support authentication. Among these are the following. The X.509 [Ken81] approach is based on a hierarchical global name space. By contrast, the SDSI/SPKI approaches of Rivest and Lampson [Riv96], and Ellison [Ell97] are based on linked local name spaces. The Secure DNS approach of Eastlake [Eas97] builds on the existing DNS (Domain Name System).

In addition, there is work on batch authentication [Nac94], which provides a way to verify many certificates simultaneously for certificates signed by the same authority under the same signature key.

2.4 Multicast Security

Securing multicast is an active area of research. Some examples of this research are works by Kent [Ken81], Gong and Shachan [Gon94], Ballardie and Crowcroft [Bal95], Deering [Dee98, Dee89], Bagnall [Bag99], Mittra [Mit97], Caronni, et al. [Car98], and Canetti and Pinkas [Can98], which address issues, requirements, architectures, protocols, and techniques. In addition, the works by Ballardie [Bal96] and Harkins [Hark98a] on key establishment discuss a variety of security problems and solutions for multicast. Securing Mbone [Kum95] broadcasts is one driving application.

3. Group Operations and Their Security Requirements

We envision that the management of group communications keys will take place in a setting in which there will be a communications system, a set of possibly overlapping groups of individuals with common purposes, and individual group members. A systems manager will manage the communications system, and a group manager will manage each group. We envision that groups will comprise a hierarchy of subgroups, with subgroup and organizational managers negotiating

on behalf of subgroup members.

Some of the above mentioned works on multicast security (e.g. Canetti and Pinkas [Can98]) and on group management (e.g. Harney and Harder[Harn99a]) also provide relevant background for the requirements of OFT group operations.

Balenson, McGrew, Sherman expires February 25, 2001

[Page 06]

INTERNET-DRAFT

August 25, 2000

3.1 Operations

Associated with each role (system manager, group manager, individual) is a set of operations, minimally including the following operations.

Operations processed by the system manager:

- induct individual into system,
- evict individual from system,
- create group, and
- dissolve group.

Section 5 explains the concept of group induction. In short, the most important results of system (or group) induction are for an individual to establish a base system key known only to the individual and the system (or group) manager, and for the system (or group) manager to check the credentials of the individual.

Operations processed by a group manager:

- add member(s) to group,
- remove member(s) from group,
- evict member(s) from group,
- initiate communications session, and
- terminate communications session.

This document focuses on the crucial operations of adding and deleting members from a group. Note that our OFT method offers some economy of scale when adding or deleting two or more individuals simultaneously. There are two types of membership deletions: a temporary removal (when the individual has not lost his security privileges), and a permanent eviction as the result of loss of security privileges.

Operations requested by individuals:

- request to join group,
- request to leave group,
- request to join session,
- request to leave session, and
- request to return to session.

It is possible that an individual might temporarily lose contact with his group manager -- for example, as might happen if an airplane flies out of radio range of his group. Therefore, there must be a way for such a member to re-synchronize key establishment with the group.

3.2 Security Requirements

The primary security requirements for the group operations of adding and deleting members are forward and backward security, as quantified

Balenson, McGrew, Sherman expires February 25, 2001 [Page 07]

INTERNET-DRAFT

August 25, 2000

by the degree of forward or backward security [Men97, pp. 528-529]. We say that a method has t -forward security if and only if (iff) no t colluding evictees can read any future communications traffic of the group. Similarly, a method has t -backward security iff no group of t colluding new members can read any previous group traffic. A method has perfect forward (backward) security iff the method has t -forward (t -backward) security for all t .

Backward security is optional. When a new member is added, the group manager may choose to create a new group key, thus denying the new member access to the old key and hence previous traffic. We seek methods, such as OFT, that provide perfect forward security, with the option of perfect backward security.

Carefully note our definitions of the phrases "perfect forward security" and "perfect backward security." Our use of these convenient phrases should not be confused with the different requirement, as explained by Menezes et al. [Men97, p. 496], that compromise of long-term keys does not reveal past session keys. Similarly, our use of these phrases does not necessarily imply information-theoretic "perfectly-secure key distribution" in the sense used by Blundo et al. [Blu92].

Two additional valuable security measures are degree of traceability and degree of disclosure amplification. A method is t -traceable iff more than t colluding members are required to leak plaintext or session keys without being identified if leaked material is discovered. Disclosure amplification refers to the extent of unauthorized disclosure of internal state information (e.g. subgroup keys) caused by the unauthorized disclosure of certain other internal state information.

In addition, when specifying security requirements for particular applications, it is important to understand the security needs and assumptions with regard to any underlying cryptographic primitives (e.g. one-way functions) and with regard to fundamental types of cryptographic strength (e.g. information theoretic, computational, quantum uncertainty).

4. One-Way Function Trees

A One-Way Function Tree (OFT) is a binary tree, each node x of which is associated with two cryptographic keys: a node key k_x and a blinded node key $k'_x = g(k_x)$. The blinded node key is computed from the node key using a one-way function g ; it is blinded in the sense that a computationally limited adversary cannot find k_x from k'_x .

Although the concept of a one-way function tree is not new (e.g. in 1979, Merkle [Mer79] proposed an authentication system based on a similar idea), our application of this concept is novel.

Balenson, McGrew, Sherman expires February 25, 2001 [Page 08]

INTERNET-DRAFT

August 25, 2000

4.1 Structure of an OFT

A group manager maintains a one-way function tree. Each leaf is associated with a member of the group. The manager uses a symmetric encryption function E to communicate securely with subsets of group members, using unblinded keys as encryption keys as explained below.

A randomly-chosen key is assigned to each member. This key is shared with the manager (via an external secure channel), and the key is assigned as the node key of the member's leaf. A variety of choices are possible governing who chooses the keys. In particular, the key could be chosen by the manager, member, or a combination thereof (see Section 4.3).

4.1.1 The Original OFT Structure

Each internal node of the tree has exactly two children. The interior node keys are defined by the rule:

$$k_x = f(g(k_{\text{left}(x)}), g(k_{\text{right}(x)})), \quad (1)$$

where $\text{left}(x)$ and $\text{right}(x)$ denotes the left and right child of the node x , respectively. The function g is one-way, and the function f is a "mixing" function (e.g. XOR). The node key associated with the root of the tree is the group key, which the group can use to communicate with privacy among group members and/or authentication of group membership.

The security of the system depends on the fact that each member's knowledge about the current state of the key tree is limited by the following invariant:

OFT Security Invariant - each member knows the unblinded node keys on the path from its node to the root, and the blinded node keys that are siblings to its path to the root, and no other blinded or unblinded keys.

This invariant is maintained by all operations that add members to

the group, and by all operations that delete members from the group. See Figure 1.

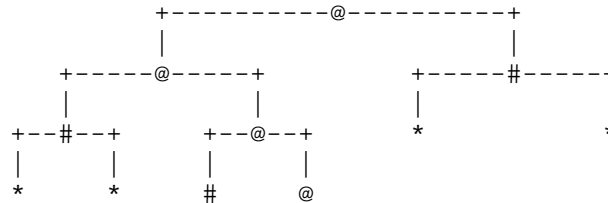


Figure 1: An example of an OFT key tree. The member at the leaf labeled M knows the keys of the nodes labeled @ (including the root key, which is used as the group key), and the blinded keys of the nodes labeled #.

McGrew and Sherman [McG98] discuss the properties of f and g in detail and give some preliminary observations regarding the security of OFT. A rigorous proof of the necessary and sufficient properties of f and g needed to satisfy formal security requirements for OFT remains an open problem.

In comparison with LKH, there is necessarily some loss in entropy of the group key in OFT and OFC as a result of the functional relationships among node keys. We estimate this loss in entropy to be very small (less than one bit for group sizes of 10 million).

4.1.2 An Improved OFT Structure

Given the (claimed) provable security properties of Canetti's [Can99b] OFC algorithm, we believe that the security of OFT (at least in a provability sense) can be improved by adapting some of the detailed functional relationships of OFC to OFT. This Section briefly outlines how this technical modification to OFT would work and introduces new convenient terminology that can be used to describe the three hierarchical methods (LKH, OFT, OFC) in a common framework.

Each node v in the key tree has a node secret x_v and a node key k_v . The group key is the node secret of the root, which is functionally related to the node secrets of the other nodes. The node keys are used only as encryption keys for broadcast communications from the manager.

Let H be any length-doubling pseudorandom function, and let $f || g = H$, where $||$ denotes concatenation. Thus, for any input y , $f(y)$ is the left half of $H(y)$, and $g(y)$ is the right half. (These functions are separate from the f and g from Section 4.1.1.) Now, under suitable assumptions on H , we have that f and g are computationally independent, which fact facilitates security proofs.

Let v be an interior node in the key tree, and let L and R be, respectively, the left- and right- children of v . In both OFT and

Balenson, McGrew, Sherman expires February 25, 2001 [Page 10]

INTERNET-DRAFT

August 25, 2000

OFC, we have that $k_v = g(x_v)$ (similarly, $k_L = g(x_L)$ and $k_R = g(x_R)$). In OFT, the node secret for v is computed as follows:

$$x_v = f(x_L) \text{ XOR } f(x_R), \quad (2)$$

where XOR denotes bitwise exclusive-or. By contrast, in OFC, if the edge (x_L, x_v) lay on the current chain, then it would be true that $x_v = f(x_L)$.

A typical manager broadcast in OFT would include a component such as $E(k_R, f(x_L))$. With the improved OFT structure, in the foregoing broadcast component we have that k_R and $f(x_L)$ are computationally independent.

4.2 Algorithms

The operations of adding and evicting members rely on the communication of new blinded key values, from the manager to all members, after the node key associated with a leaf has changed. To maintain security, each blinded node key must be communicated only to the appropriate subset of members. If the blinded key k'_x changes, then its new value must be communicated to all of the members who store it. These members are associated with the descendants of the sibling of x , and they know the unblinded node key k_s , where s is the sibling of x . To provide the new value of the blinded key to the appropriate set of members, and keeping it from other members, the manager encrypts k'_x with k_s before broadcasting it to the group. (Here and throughout, we shall use the verb "broadcast" in the sense of "group broadcast" -- sending a message from the group manager to all members of the group.)

Because the utility of OFT is not restricted to multicast environments, throughout we sometimes refer to "unicast" transmissions, as for example to implement the add-member operation.

In typical multicast environments, where no unicast is possible, such unicast messages would be sent as multicasts.

4.2.1 Adding a member

As shown in Figure 2, when a new member joins the group, an existing leaf node x is split, creating new nodes $\text{left}(x)$ and $\text{right}(x)$. The member associated with x becomes associated with $\text{left}(x)$, and the new member is associated with $\text{right}(x)$. Both members are given new keys. The old member gets a new key because their former sibling knows the old blinded node key and could use this information in collusion with another group member to find an unblinded key that is not on his path to the root. The new values of the blinded node keys that have changed are broadcast securely to the appropriate subgroups, as described in the previous paragraph. The number of blinded keys that must be broadcast to the group is equal to the distance from x to the root plus two. In addition, in a unicast transmission (see last paragraph of Section 4.2), the new member is

given their set of blinded node keys, using the external secure channel. In order to keep the height h of the tree as low as possible, when a new member is added, the leaf closest to the root is split.

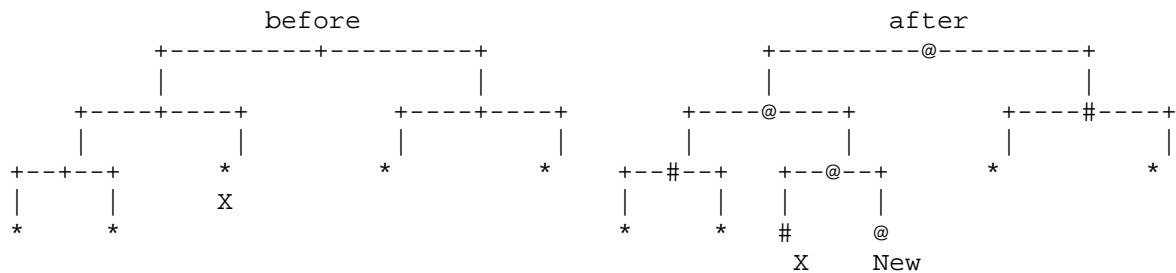


Figure 2: Inserting a new member into an OFT key tree. Via a unicast, the newly added member (New) is informed of the blinded keys of the nodes labeled # and of the unblinded keys of nodes labeled @. Via a multicast, the other members who need to know are informed of the blinded keys at nodes labeled @.

4.2.2 Evicting a member

As shown in Figure 3, when the member associated with a leaf y is evicted from the group, the member assigned to the sibling of y is reassigned to the parent p of y and given a new leaf key value. If the sibling s of y is the root of a subtree, then p becomes s , moving the subtree closer to the root. In this case, one of the leaves of this subtree is given a new key (so that the evictee no longer knows the blinded key associated with the root of the subtree). The new

in such broadcast messages, as we did in our prototype.

4.3 Properties

In this section we comment briefly on the security, resource usage, and salient characteristic features of The OFT method. For more details, see the paper by McGrew and Sherman [McG98].

The security properties of OFT stem from the system invariant stated in Section 4.1, from the strength of the component one-way function, and from the random selection of leaf keys. In short, OFT provides perfect forward security and optional perfect backward security. Thus, arbitrary coalitions of evicted members cannot directly compute the group communications key nor any unblinded node key.

Evicted members have some information about the key tree but not enough to compute directly any unblinded node key. After a member is evicted, the keys along the path from the member's node to the root change. After this change, the evictee knows only the blinded keys of the siblings of the nodes along the path from the evictee to the root. These blinded nodes are insufficient to compute directly any unblinded key.

Balenson, McGrew, Sherman expires February 25, 2001

[Page 13]

INTERNET-DRAFT

August 25, 2000

Interestingly, OFT is a centralized, member-contributory method. OFT is centralized in the sense that the group manager plays a special trusted role. OFT is member contributory in the sense that each leaf can contribute entropy to the group communication key.

The hierarchical nature of OFT distributes the computational costs of re-keying among the entire group, so that the managers computational burden is comparable to that of a group member. Table 1 below summarizes the salient resource usage of adding or deleting a member with OFT in terms of time, memory, number of bits broadcast, and number of random bits needed.

| Resource Measure | Group Member Cost | Group Manager Cost |
|-----------------------|-------------------|--------------------|
| Time | h | h |
| Memory | hK | $2nK$ |
| Bits broadcast | 0 | $hK + h$ |
| Random bits generated | 0 | K |

Table 1: Summary of resource usage of adding or deleting a member with OFT. Here, n is the group size, K is the size of a key in bits, and h is the height of the OFT ($h = \lg n$ when the tree is balanced). Either the member or the manager could generate the random bits needed at the leaves.

4.4 Enhanced OFT+ and LKH+ Algorithms

As suggested in an observation attributed to Radia Perlman, the add-member operation in LKH can be significantly improved to run in constant time. The method is to compute the new group communications key by applying a one-way function to the current group communications key; to unicast the group key to the new member; and to broadcast that this update has been done. We shall refer to this refinement of LKH, which maintains perfect backward security, as LKH+.

Unfortunately, this refinement does not apply to the delete-member operation. Fortunately, in many applications, add-member is performed much more frequently than is evict-member. Independently of Perlman, V. Viswanathan [Vis96, pp. 25-26] also suggested a similar constant-time member-parallel add-member idea.

A similar refinement is also possible for OFT, which we shall call OFT+. For both LKH+ and OFT+, the add-member operation advances the group communications key and marks (i.e. places on a list) the specified member to be added at the next full re-key (e.g. typically at the next evict-member operation). Thus, the work of adding new

Balenson, McGrew, Sherman expires February 25, 2001

[Page 14]

INTERNET-DRAFT

August 25, 2000

members is deferred until the next re-key. Two savings, however, are possible. First, there is an economy of scale when adding multiple members, especially when the new members will be placed compactly in the same section of the key tree. Second, a newly added member might be evicted before the next re-key.

The cost of the deferred add-member operations is given by the size of the so-called CAT (Common Ancestor Tree) of the members (see [McG98, Sections 3.3 and 5.3]). In practice, the cost of adding R new members at once (assuming they will be placed compactly in the same section of the tree) is approximately $2R + \lg(n/R)$, where n is the number of nodes in the key tree.

4.5 Comparisons with Other Leading Key-Establishment Methods

In this section we briefly compare our OFT algorithm against the two leading competing algorithms for establishing keys in large dynamic groups: the Logical Key Hierarchy (LKH)[Harn99b, Wal97], and the Single Key Distribution Center (SKDC). As reviewed in Section 2, these two algorithms and OFT are the main choices available to system engineers who do not wish to trust network routers in large group applications. OFT and LKH are appealing because their computation, broadcast, and memory requirements scale logarithmically with group size. The enhanced algorithms LKH+ and OFT+ (described in Section 4.4) are especially appealing for their constant-time add-member operations. Despite its linear complexity, SKDC is appealing for its

simplicity.

4.5.1 Comparison Criteria

The choice of which key-establishment algorithm should be used can be answered meaningfully only in the context of the requirements of particular applications. To assist engineers in making their choices, we shall summarize some of the major properties of SDKC, LHK, LKH+, OFT, and OFT+ in terms of selected quantitative comparison criteria. These criteria include total delay, number of bits broadcast, number of bits unicast, manager computation, maximum member computation, number of random bits generated, and manager and member memory requirements. Each of these methods provides perfect forward security with the option of perfect backward security.

The five leading candidate methods differ also in terms of required primitives, security semantics, central versus contributory flavor, and resynchronization capabilities (for dealing with group members who temporarily are unable to receive manager broadcasts). For example, SDKC and LKH require encryption functions; LKH+ and OFT require encryption functions and one-way functions. Although none of the authors of any of these methods has provided a formal proof of security, some engineers may have greater confidence in the security of methods with simpler security semantics. Listed in increasing complexity of security semantics, the methods are: SDKC, LKH, LKH+, OFT, OFT+. With regard to the source of entropy of random bits in

Balenson, McGrew, Sherman expires February 25, 2001

[Page 15]

INTERNET-DRAFT

August 25, 2000

common group communication keys, SDKC, LKH, and LKH+ may be viewed as member non-contributory methods because the group manager provides all of the entropy. By contrast, OFT and OFT+ can be used in either a member non-contributory or member-contributory fashion. Most applications have a need to provide a resynchronization capability; some methods may provide some degree of passive member-synchronization.

When asked why she based her LKH method on a keyed encryption function rather than on a faster keyless one-way function (as used in OFT), Wallner [private communication (11/19/97)] explained that her motivating application was a radio communication system. In this application, the available hardware supported a keyed encryption function but not a keyless one-way function. Wallner also remarked (without connection to the choice of encryption function versus one-way function), that in her application, it was very important to conserve battery power. Although these considerations were important to her application, other applications might have different requirements or available primitives; thus the rationale for Wallner's choices do not necessarily apply to other applications.

4.5.2 Quantitative Comparison of SKDC, LKH, LKH+, OFT, and OFT+

Tables 2 and 3 summarize the time, broadcast, and space requirements of each of the five leading candidate algorithms (SKDC, LKH, LKH+, and OFT). Specifically, for each algorithm and for each of the initialize, add-member, evict-member operations, Table 2 summarizes the total delay, number of bits broadcast, number of bits unicast, manager time, maximum member time, and number of random bits generated. Table 3 summarizes the manager and member memory requirements. For more details, see McGrew and Sherman [McG98].

In typical multicast environments, where no unicast is possible, the add-member unicast messages of LKH+, OFT, and OFT+ would be sent as multicasts.

4.5.3 Summary

For moderate size groups, the simple SKDC may often be appealing. For very large groups, however, many applications will likely demand a method that scales logarithmically in total delay and member memory usage. For such applications, especially if the add-member is more frequent than the evict-member operation, the OFT+ and LKH+ methods look very attractive for their constant-time add-member. LKH+ and OFT+ are similar algorithms, with LKH+ offering relatively simpler security semantics, and with OFT+ requiring fewer bits to transmit for re-keying. If for the application it is critical to minimize the number of bits broadcast or the number of random bits generated, or if a member-contributory method is needed, then OFT+ may be the method of choice.

Initialization

| RESOURCE MEASURE | SKDC | LKH | LKH+ | OFT | OFT+ |
|------------------------------|------|-------|-------|-------|-------|
| Total delay | n | 2n | 2n | 3n | 3n |
| Number of bits broadcast | nK | 2nK+h | 2nK+h | 2nK+h | 2nK+h |
| Number of bits unicast | 0 | 0 | 0 | 0 | 0 |
| Manager computation | n | 2n | 2n | 3n | 3n |
| Max member computation | 1 | h | h | 2h | 2h |
| No. of random bits generated | K | 2nK | 2nK | nK | nK |

Add member

| RESOURCE MEASURE | SKDC | LKH | LKH+ | OFT | OFT+ |
|------------------------------|------|-------|------|------|------|
| Total delay | n | 2h | 1 | 3h | 1 |
| Number of bits broadcast | nK | 2hK+h | h | hK+h | h |
| Number of bits unicast | 0 | 0 | K | hK | K |
| Manager computation | n | 2h | 1 | 3h | 1 |
| Max member computation | 1 | h | 1 | 2h | 1 |
| No. of random bits generated | K | hK | 0 | K | 0 |

Evict member*

| RESOURCE MEASURE | SKDC | LKH | LKH+ | OFT | OFT+ |
|--------------------------|------|-------|-------|------|------|
| Total delay | n | 2h | 2h | 3h | 3h |
| Number of bits broadcast | nK+h | 2hK+h | 2hK+h | hK+h | hK+h |

| | | | | | |
|------------------------------|---|----|----|----|----|
| Number of bits unicast | 0 | 0 | 0 | 0 | 0 |
| Manager computation | n | 2h | 2h | 3h | 3h |
| Max member computation | 1 | h | h | 2h | 2h |
| No. of random bits generated | K | hK | hK | K | K |

(* For LKH+ and OFT+, if R add-member actions have been deferred, then the evict-member operation will additionally broadcast $[2R + \lg(n/R)]K$ bits. See Section 4.4.)

Table 2: Summary of time and communication usage of initialization, add-member, and evict-member with the SKDC, LKH, LKH+, OFT, and OFT+ key-establishment methods. Here, n is the group size, K is the size of a key in bits, and h is the height of the key tree ($h = \lg n$ when the tree is balanced). Note that while LKH and LKH+ have lower magnitudes of total delay, the units of time for OFT are typically much smaller because keyless one-way functions are much faster than are keyed-encryption functions.

Memory Usage

| RESOURCE MEASURE | SKDC | LKH | LKH+ | OFT | OFT+ |
|--------------------|------|-----|------|-----|------|
| Manager storage | nK | 2nK | 2nK | 2nK | 2nK |
| Max member storage | 2K | hK | hK | hK | hK |

Table 3: Summary of manager and member memory usage in the SKDC, LKH, LKH+, OFT, and OFT+ Key-establishment methods. Here, n is the group size, K is the size of a key in bits, and h is the height of the key tree ($h = \lg n$ when the tree is balanced).

Balenson, McGrew, Sherman expires February 25, 2001 [Page 17]

INTERNET-DRAFT August 25, 2000

5. Design Choices, Implementation Issues, and Experience

In this section we identify important engineering decisions that must be addressed in the implementation of the OFT algorithm. We also discuss some of the lessons learned from our prototype implementation.

5.1 Design Choices

Several important engineering decisions must be made in the implementation of the OFT algorithm: the choice of f and g, the format of broadcasts by the manager, the representation of the tree, and time-space tradeoffs by each member involving how many unblinded ancestor keys to store.

The function g can be based on a cryptographic hash function such as MD5 [Riv92] or SHA-1 [Sha-1]. It is possible that the node keys do not need to be as large as the output size of the underlying function. For example, MD5 has a sixteen-byte output, while DES keys are only seven bytes long. The function g can be constructed from

MD5 by discarding some of the output, as is done by S/KEY, so that the node keys (and thus the broadcasts) are smaller.

The function f does not need to be one-way; it needs only to mix its inputs. This fact suggests that $f(x,y) = x \text{ XOR } y$ is a fast, simple, and effective choice (XOR denotes the bitwise exclusive-or function). Of course, the functions f and g must not interact pathologically so as, for example, to undo g .

The representation of the tree and the formats of the messages from the group manager are important engineering decisions. For example, the tree could be represented as a record and pointer structure, or as a linear array. We recommend that message formats for messages broadcast from the manager explicitly include node number information to identify which parts of a message correspond to which subtrees.

5.2 Implementation Issues and Experience

The NAI prototype implementation of the OFT+ algorithm was carried out in 1999, for the DCCM OFT Toolkit. The purpose of this Java implementation is to demonstrate proof of concept and to gain insight into implementation issues.

The prototype uses the following parameter choices: the key size is 128 bits; f is XOR; and g is SHA-1 with 160 bits of output.

For simplicity, and to avoid ever having to change a member's node number in the key tree, the prototype uses the following alternate implementation of OFT operations: the key tree is constructed with all leaves always on the lowest level. Thus, initially, the key tree is allocated for a certain maximum size. To grow beyond this size, the tree could be doubled in size by inexpensively making the current

Balenson, McGrew, Sherman expires February 25, 2001 [Page 18]

INTERNET-DRAFT

August 25, 2000

tree the left subtree of a new tree and broadcasting the new root node number.

The NAI prototype raised three important implementation issues: adapting messaging for unreliable communications, node numbering, and node storage explosion.

To implement the OFT algorithm in the multicast environment for which it was intended requires some messaging adaptations. The finite maximum transmission unit in this connectionless environment requires that messages be sent in packets. UDP packets can arrive out of order or fail to arrive. All messages need to be protected for confidentiality, integrity, and authentication. The prototype protects its internal messages using the Blowfish encryption and with SHA-1-based integrity and authentication checks.

As an example of a messaging implementation detail, note that to encrypt a 160-bit SHA-1 output with a 128-bit block cipher requires

some type of chunking. The prototype applies the Blowfish algorithm in ECB mode with no padding to 128-bit data blocks. These blocks are protected using the standard authentication and integrity mechanisms for all OFT messages; alternatively, one might use some type of block chaining (e.g. CBC mode).

During the course of OFT operations, it is necessary to be able to refer to members by their names and to refer to key tree nodes by their topological positions. Also, it is necessary to be able to map between member names and their associated nodes. To this end it is convenient to maintain node numbers. Moreover, since there is no efficient reliable way to inform members of any changes in their node numbers, it is especially convenient if the node numbers never change. For these reasons, in the prototype, member names are equated with node numbers, and node numbers never change.

Because OFT is intended for very large groups, the amount of storage allocated per node in the key tree significantly affects the total amount of memory used. We refer to this issue as the node storage explosion. Care must be taken not to squander memory on elaborate node structures. For example, on space considerations alone, the tree-balancing strategies of Moyer et al. [Moy99] are unlikely to be wise engineering choices for today's technology.

In addition, the following two lessons were learned. First, it is important but nontrivial for a member to verify when it has the correct group key. A member might compute the wrong group key for two reasons: The member might never receive some key-update messages. Also, in a sequence of key-update messages, the member might not have any way of knowing that additional key-update messages are yet to come. These situations are complicated by the unreliable nature of the multicast environment. We view these key-synchronization issues as orthogonal to OFT; that is, these issues exist in many communications systems for a variety of reasons

Balenson, McGrew, Sherman expires February 25, 2001

[Page 19]

INTERNET-DRAFT

August 25, 2000

and should be solved as part of the standard key management service. For example, one solution is to attach confidential key checksums to keying data, which is what the prototype does. Another solution, which the prototype also uses, is to allow members to send resynchronization request messages to the group manager.

Second, it is important that the components be appropriately matched and coordinated. For example, inefficiencies can arise if the keysize is not a multiple of the blocksize of the encryption function used for confidentiality.

Currently, the prototype has handled groups of up to 100,000 members on a Pentium with 64 MB of RAM, with adjustments to the swap space. The maximum possible group is limited by two factors: the size of the initial message, and the node storage explosion.

6. Amortized Group Induction

Before a group can establish an initial group communications key, the members must be inducted (enrolled) into the group. For centralized key establishment methods including OFT, an important objective of this induction step is for each member to establish an individual base key known only to the member and the group manager. The induction process also ensures that each member has the necessary certificates to take advantage of the supporting authentication infrastructure. The main computational goal of induction is to minimize its total delay.

For each group member to establish a separate base key with the Group Manager, the simplest approach is for the manager to engage each member in a separate pairwise authenticated key exchange protocol, such as the Internet Key Exchange (IKE) protocol [Hark98b, Mau98, Orm]. Unfortunately, this simple approach scales linearly in group size.

In this section we describe a new approach to group induction due to Sherman [She98, Bal98b] that amortizes the relatively high cost of a pairwise key exchange over multiple entries into groups. This amortization saves time when many users become members of two or more groups. We call this new approach "amortized induction." An IBM team [Blu97] independently discovered a similar idea in a network context.

6.1 Induction Model

Assume there is a universe of N individuals from which G groups are formed, each group having at most n members. Assume further that N is much smaller than Gn ; that is, many individuals belong to several groups. It is for this reasonable assumption that amortized induction offers significant savings.

Balenson, McGrew, Sherman expires February 25, 2001

[Page 20]

INTERNET-DRAFT

August 25, 2000

We assume that there is a system that administers multiple groups, each of whose group communications key is established by a key-establishment algorithm such as OFT. Each participant may join one or more group. Each group has a group manager, and there is a system manager who controls induction into the system. For simplicity we shall describe the induction process in terms of a single system manager, though the concept of system manager can be extended to a more general system management function which might be distributed. For each group, each member must establish an individual base key known only to the member and the group manager.

Amortized induction reduces the number of expensive pairwise key exchanges from Gn to N , which can be a significant savings. Amortized induction comes at the cost of Gn one-way function

applications by the system manager, but these function evaluations are much faster than pairwise key exchanges. For example, a single application of the MD5 hash function is approximately 1000 times faster than a single ISAKMP/OAKLEY key exchange.

6.2 Induction Algorithm

Whenever an individual first enters the system, the member establishes an individual system base key known only to the individual and the system manager. For example, this step could be carried out using the IKE Key-Exchange Protocol. Thereafter, whenever the individual joins a group, the individual can establish an individual group base key with the group manager as a one-way function of the individual's system base key, the individual's name, and the group name.

More specifically, let A be the group name. For each member M , the group base key KA_M for M is computed as a one-way function F of M 's system base key K_M , the name of M , and the group name. For example,

$$KA_M = F(K_M, M, A), \quad (2)$$

where F is a publicly known one-way function such as the hash function MD5. The total member delay in this computation is constant.

The function F must satisfy the following properties: it must be easy to compute given its three inputs, and it must be infeasible for anyone to compute the output given only the last two inputs (even under an adaptive chosen plaintext/ciphertext attack).

Whenever a group manager is appointed, the group manager establishes a manager key k_A known only to the group manager and the system manager. For added security this key establishment could be carried out using the same pairwise key-exchange protocol used in the induction process; alternatively, it would be possible to compute manager keys along the lines of Equation 2.

Whenever members of a group need to be inducted, the system manager computes the group base keys and unicasts them to the group manager, encrypted under the manager key k_A . The length of this unicast is linear in the group size. Note that, unlike the SKDC method, no group broadcast is required. Members of different groups can be inducted in parallel.

6.3 An Example of Induction

To illustrate the amortized induction process, consider a toy example in which there are three individuals Q , R , S and two groups $A = \{Q, R\}$ and $B = \{S\}$.

When Member Q is inducted into the system, he establishes a base system key K_Q known only to Q and the system manager. Similarly, Members R and S establish base system keys K_R and K_S , respectively. At the appointment of Group Manager A, Manager A establishes a manager key k_A known only to Manager A and to the system manager. Similarly, Group Manager B shares a manager key k_B with the system manager.

To induct members of Group A, the system manager computes the group base keys KA_Q , and KA_R for Members Q and R, respectively, and sends them to Manager A encrypted under manager key k_A . For example, $KA_Q = f(K_Q, Q, A)$ and $KA_R = f(K_R, R, A)$. In parallel, Members Q and R each computes his own group base key. Members of Group B are similarly inducted.

7. Conclusion and Open Problems

We have presented and analyzed a new practical hierarchical algorithm for establishing shared cryptographic keys for large, dynamically-changing groups. Our algorithm is based on a novel application of One-way Function Trees (OFTs).

Unlike previously proposed solutions based on information theory, public-key cryptography, hybrid approaches, or a single key distribution center, our OFT algorithm has communication, computation, and storage requirements, which scale logarithmically with group size, for the add or evict operation. Each of the aforementioned methods typically scales linearly or worse (see Section 2.1).

In comparison with the first proposed hierarchical method that does not depend on trusted routers -- the Logical Key Hierarchy (LKH) -- our OFT algorithm reduces by approximately half the number of bits broadcast by the manager per add or evict operation. By contrast, LKH has relatively simpler security semantics than does OFT. The user time and space requirements of OFT and LKH are roughly comparable. For many applications, including multicasts, minimizing broadcast size is especially important.

The One-Way Function Chain (OFC) variation of OFT recently due to Canetti, et al. [Can99b] is a very attractive proposal for its simplicity, broadcast size (similar to OFT), and provable security.

The exact savings in broadcast size of OFT over LKH depends on the parameter choices in the OFT and LKH implementations, including the key lengths and the output sizes of the one-way function, encryption function, and mixing function. If the one-way function expands its input more than does the encryption function, then the savings in broadcast size will be less than a factor of two. For example, using

128-bit group keys and the 160-bit SHA-1 one-way function, our prototype OFT achieves a savings in broadcast size over LKH of $(2 \times 128) / 160 = 1.6$, which is slightly less than two.

Although a reduction in broadcast size of one half may seem relatively small, we conjecture that the performance of the LKH algorithm may be approaching theoretical limits and therefore only relatively small improvements may be possible. Canetti et al. [Can99] give some evidence supporting this conjecture by proving upper and lower bounds with gap of at most $O(b) = O(\lg n)$, where b is a parameter (see Section 2.1).

Refinements to the OFT and LKH methods which we call OFT+ and LKH+ (see Section 4.4), offer significant performance improvements. Specifically, in OFT+ and LKH+, the cost of the add-member operation one one-way function application, a unicast of one key, and one short broadcast. There is also a deferred cost for the next re-key operation. These improvement to OFT and LKH make OFT+ and LKH+ attractive choices for many applications.

It is possible to implement the OFT algorithm with k -ary trees, as studied in LKH by Canetti et al. [Can99]. The optimal choice of k depends on what cost metric is to be minimized. The choice $k = 2$ minimizes broadcast size for single member eviction. By contrast, to minimize total manager computation, Wong et al. [Won98] show experimentally that $k = 4$ is best in LKH. We suspect that a similar result may also hold for OFT.

McGrew and Sherman [McG98] discuss and analyze savings that are possible by separately batching multiple add or multiple evict operations. Similar and possibly even greater savings may be possible by batching adds and evicts together, as also suggested by Yang (personal correspondence 6/22/99).

The OFT method is a centralized algorithm with the option of member contributions to the entropy of the common communications key. Its main advantages are that, in comparison with LKH+, it reduces by a factor of two the number of bits required to be broadcast for each evict-member operation. It is also the only proposed scalable member-contributory method.

Our preliminary security analysis of OFT [McG98] raises some

interesting questions about the security of function iterates, and that of bottom-up one-way function trees. Formally proving the security of key-establishment algorithms is a difficult problem, even in the two-party case (e.g. see Bellare and Rogaway [Bel93]).

It is important to realize that there are significant fundamental limitations to achieving security in large groups -- one might even say that a secure large group is an oxymoron. In most large groups,

it is very likely that at least one member is unreliable, untrustworthy, malicious, or careless. Each member knows the common communications key and the plaintext, which is the main commodity being protected. Using multiple communications keys for different subgroups would not enhance security since each member would still have the plaintext. Any member could disclose the plaintext. Consequently, in large groups, it becomes especially important to detect traitors (e.g. through fingerprints and watermarks [Kur98, Sta97, Cho94]) and to limit the loss caused by disclosures (e.g. by rapid evictions and re-keying). Special-purpose, physically secure hardware may play a role in these objectives, by restricting access to communication keys, complicating effective use of compromised keys, and providing unique fingerprints.

Our OFT algorithm offers a practical approach with low broadcast size to manage the demanding key establishment requirements of secure applications for large, dynamic groups.

8. Security Considerations

This document proposes a new algorithm for securely establishing a shared, secret group communications key in a large, dynamic group. Sections 3.2, 4.1, and 4.3 describe the security properties of this algorithm.

9. References

[Ate98] Ateniese, Giuseppe, Michael Steiner, and Gene Tsudik, "Authentication group key agreement and friends" in Proceedings of the 5th Conference on Computer & Communications Security, ACM Press (1998), 17-26.

[Bag99] Bagnall, P., R. Briscoe, and A. Poppitt, "Taxonomy of communication requirements for large-scale multicast applications," Internet Draft (work in progress), draft-ietf-lsma-requirements-03.txt, Internet Engineering Task Force (May 14, 1999). 25 pages.

[Bal95] Ballardie, Tony, and Jon Crowcroft, "Multicast-specific threats and counter-measures," Proceedings of the Internet Society 1995 Symposium on Network and Distributed System Security, February 16-17, 1995, San Diego, California, IEEE Computer Society (1995), 2-14.

Balenson, McGrew, Sherman expires February 25, 2001

[Page 24]

INTERNET-DRAFT

August 25, 2000

[Bal96] Ballardie, A., "Scalable multicast key distribution," Request for Comments (RFC) 1949, Internet Engineering Task Force (May 1996), 18 pages.

[Bal97] Ballardie, A. "Core based tree (CBT) multicast routing architecture," Request for Comments (RFC) 2201, Internet Engineering

Task Force (September 1997), 14 pages.

[Bal98] Balenson, David M., Dennis K. Branstad, David A. McGrew, and Alan T. Sherman, "Dynamic cryptographic context management (DCCM): Report #1: Architecture and system design," TIS Report No. 0709, TIS Labs at Network Associates, Inc., Glenwood, MD (June 2, 1998). 121 pages.

[Bal98b] Balenson, David M., David A. McGrew, and Alan T. Sherman, "Key management for large dynamic groups: One-way function trees and amortized initialization," *Advanced Security Research Journal - NAI Labs*, Vol. 1, No. 1 (fall 1998), 27-46.

[Bel93] Bellare, Mihir, and Phillip Rogaway, "Entity authentication and key distribution," *Advances in Cryptology: Proceedings of Crypto 93*, Douglas R. Stinson, ed., LNCS 773, Springer-Verlag (1993), 232-249.

[Ber91] Berkovitz, S., "How to broadcast a secret," *Advances in Cryptology: Proceedings of Crypto 91*, Feigenbaum, ed., LNCS 576, Springer-Verlag (1991), 535-541.

[Blo90] Bloom, Joel, ed., X9.24-1990, "Financial services retail key management," ANSI X9A3 (March 1990). 84 pages.

[Blu92] Blundo, Carlo, Alfred de Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung, "Perfectly-secure key distribution for dynamic conferences," *Advances in Cryptology: Proceedings of Crypto92*, E. F. Brickell, ed., LNCS 740, Springer-Verlag (1992), 471-486.

[Blu97] Blumenthal, Uri, Nguyen C. Hien, and Bert Wijnen, "Key derivation for network management applications," *IEEE Network* (May/June 1997), 26-29.

[Bur94] Burmester, Mike, and Yvo Desmedt, "A secure and efficient conference key distribution system," *Advances in Cryptology: Proceedings of Eurocrypt 94*, A. De Santis, ed., LNCS 950, Springer-Verlag (1994), 275-286.

[Bur97] Burmester, Mike, and Yvo G. Desmedt, "Efficient and secure conference key distribution," *Secure Protocols*, M. Lomas, Ed., LNCS 1189, Springer-Verlag (1997), 119-130. [Revised and expanded version of the corresponding Eurocrypt 94 paper by the same authors.]

Balenson, McGrew, Sherman expires February 25, 2001

[Page 25]

INTERNET-DRAFT

August 25, 2000

[Can98] Canetti, R. and B. Pinkas, "A taxonomy of multicast security issues," *Internet Draft (work in progress)*, draft-canetti-secure-multicast-taxonomy-00.txt, Internet Engineering Task Force (May 1998). 13 pages.

[Can99] Canetti, Ran, Tal Malkin, and Kobbi Nissim, "Efficient communication-storage tradeoffs for multicast encryption" in *Advances in Cryptology: Proceedings of Eurocrypt 99*, Springer-Verlag (1999), 459-474.

[Can99b] Canetti, R., Juan Garey, Gene Itkis, Daniele Micciancio, Moni Naor, and B. Pinkas, "Multicast security: A taxonomy and efficient constructions," *Infocom99* (1999), 20 pages.

[Car98] Caronni, Germano, Marcel Waldvogel, Dan Sun, and Bernhard Plattner, "Efficient security for large and dynamic multicast groups" in *Proceedings of the Seventh Workshop on Enabling Technologies (WET ICE 98)*, IEEE Computer Society Press (1998). 20 pages.
<http://skip.incog.com/wetice98/HacknSlash.htm>

[Chi89] Chiou, Guang-Huei, and Wen-Tsuen Chen, "Secure broadcasting using the secure lock," *IEEE Transactions on Software Engineering*, 15:8 (August 1989), 929-934.

[Cho94] Chor, B., A. Fiat, and M. Naor, "Tracing traitors," *Advances in Cryptology: Proceedings of Crypto 94*, Y. G. Desmedt, Ed., LNCS 839, Springer Verlag (1994), 257-270.

[Dee89] Deering, S., "Host Extensions for IP Multicasting," *Request for Comments (RFC) 1112*, Internet Engineering Task Force (August 1989). 17 pages.

[Dee98] Deering, S., D. Estrin, D. Farinacci, M. Handley, A. Helmy, V. Jacobson, C. Liu, P. Sharma, D. Thaler, and L. Wei, "Protocol independent multicast-sparse mode (PIM-SM): Motivation and architecture," *Internet Draft (work in progress)*, [draft-ietf-idmr-pim-arch-05.txt](#), Internet Engineering Task Force (August 4, 1998). 26 pages.

[Eas97] Eastlake, Donald E., 3rd, and Charles W. Kaufman, "Domain name system security extensions," *Request for comments (RFC) 2065*, Internet Engineering Task Force (January 1997).

[Ell97] Ellison, C. M., "SPKI Requirements" (February 1997). [For latest version, see <http://www.clark.net/pub/cme/html/spki.html>]

[Fen97] Fenner, W., "Internet group management protocol, version 2," *Request for Comments (RFC) 2236*, Internet Engineering Task Force (November 1997). 24 pages.

[Fia93] Fiat, Amos, and Moni Naor, "Broadcast encryption," *Advances in Cryptology: Proceedings of Crypto93*, D. R. Stinson, ed., LNCS 773,

Springer-Verlag (1993), 481-491.

[Gon89] Gong, Li, and David J. Wheeler F. R. S., "A matrix key distribution scheme," *Journal of Cryptology*, 2:1 (1990), 51-59.

[Gon94] Gong, Li, and N. Shacham, "Elements of trusted multicasting," *Proceedings of the IEEE International Conference on Network Protocols*, Boston, Massachusetts (October 1994).

[Gon96] Gong, Li, "Enclaves: Enabling secure collaboration over the internet," *Proceedings of the Sixth USENIX Unix and Network Security Symposium*, San Jose, California (July 1996), 149-159.

[Hard97] Harding, Michael, David A. McGrew, and Alan T. Sherman, "A new key-management algorithm for large dynamic groups," transparencies from talk given by Alan Sherman at NSA (November 19, 1997). 8 pages.

[Hark98a] Harkins, Dan, and Naganand Doraswamy, "A secure, scalable multicast key management protocol (MKMP)," *Draft (work in progress)*, Cisco Systems and Bay Networks (March 1998). 20 pages.

[Hark98b] Harkins, D. and D. Carrel, "The Internet key exchange (IKE)," *Internet Draft (work in progress)*, draft-ietf-ipsec-isakmp-oakley-08.txt, Internet Engineering Task Force (June 1998).

[Harn97a] Harney, Hugh, Carl Muckenhirn, and Thomas Rivers, "Group key management protocol (GKMP) architecture," *Request for Comments (RFC) 2094*, Internet Engineering Task Force (July 1997).

[Harn97b] Harney, Hugh, Carl Muckenhirn, and Thomas Rivers, "Group key management protocol (GKMP) specification," *Request for Comments (RFC) 2093*, Internet Engineering Task Force (July 1997).

[Harn99a] Harney, Hugh, and Eric Harder, "Multicast security management protocol (MSMP): Requirements and policy," *Draft (work in progress)*, draft-harney-sparta-msmp-sec-00.txt, SPARTA, Inc. (March 1999). 26 pages.

[Harn99b] Harney, Hugh, and Eric Harder, "Logical Key Hierarchy Protocol," *Internet Draft (work in progress)*, draft-harney-sparta-lkhp-sec-00.txt, Internet Engineering Task Force (March 1999). 22 pages.

[Harn99c] Harney, Hugh, and Eric Harder, "Group Secure Association Key Management Protocol" *Draft (work in progress)*, draft-harney-sparta-gsakmp-sec-00, SPARTA, Inc. (April 1999). 35 pages.

[Hay98] Hayden, Mark Garland, "The Ensemble system," Ph.D. Dissertation, Department of Computer Science, Cornell University (January 1998). 106 pages.

[Hor] The Horus Project,
<http://simon.cs.cornell.edu/Info/Projects/HORUS/>.

[Jus94] Just, Michael K., "Methods of multi-party cryptographic key establishment," MS Thesis, School of Computer Science, Carleton University (August 9, 1994). 77 pages.

[Ken81] Kent, Stephen T., "Security requirements and protocols for a broadcast scenario," IEEE Transactions on Communications (June 1981).

[Keu96] Keung, S., and L. Gong, "Enclaves in Java: APIs and implementations," Technical Report SRI-CSL-96-07, SRI International, Menlo Park, California (July 1996).

[Kru98] Kruus, Peter, "A survey of multicast security issues and architectures," Proceedings 21st National Information Systems Security Conference, October 5-8, 1998, Arlington, VA, 408-420.

[Kum95] Kumar, Vinay, Mbone Interactive Multimedia on the Internet, New Riders Publishing (Indianapolis, IN, 1995).

[Kur98] Kurosawa, Kaoru, and Yvo Desmedt, "Optimum traitor tracing and asymmetric schemes with arbiter," Draft (work in progress), spring 98). 13 pages.

[Mau98] Maughan, Douglas, Mark Schertler, Mark Schneider, and Jeff Turner, "Internet security association and key management protocol (ISAKMP)," Internet Draft (work in progress), draft-ietf-ipsec-isakmp-10.txt, Internet Engineering Task Force (July 3, 1998). 86 pages.

[McG98] McGrew, David A., and Alan T. Sherman, "Key establishment in large dynamic groups using one-way function trees," TIS Report No. 0755, TIS Labs at Network Associates, Inc., Glenwood, MD (May 1998). 13 pages. [A revised version of this paper has been conditionally accepted to appear in the IEEE Transactions on Software Engineering.]

[Men97] Menezes, Alfred J., Paul C. van Oorschot, and Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, Boca Raton, Florida (1997).

[Mer79] Merkle, Raph C., "Secrecy, authentication, and public-key cryptosystems," Technical Report No. 1979-1, Information Systems Laboratory, Stanford University, Palo Alto, CA (1979).

[Mit97] Mittra, Suvo, "Iolus: A framework for scalable secure multicasting," Proceedings of the ACM SIGCOMM '97, September 14- 18, 1997, Cannes, France. 11 pages.

[Moy99] Moyer, M. J., J. R. Rao, and P. Rohatgi, "Maintaining Balanced Key Trees for Secure Multicast," Internet Draft (work in progress), draft-irtf-smug-key-tree-balance-00.txt, Internet Engineering Task Force (June 25, 1999). 16 pages.

[Nac94] Naccache, David, David M'Raithi, Serge Vaudenay, and Dan Raphaelli, "Can D.S.A. be improved? Complexity trade-offs with the digital signature standard," Advances in Cryptology: Eurocrypt '94, Alfredo De Santis, Ed., LNCS 950, Springer-Verlag (1994), 77-85.

[Orm] Orman, Hilarie K., "The OAKLEY key determination protocol," Internet Draft (work in progress), draft-ietf-ipsec-oakley- 02.txt, Internet Engineering Task Force. 48 pages.

[Poo99] Poovendran, R., and J. S. Baras, "An information theoretic analysis of rooted-tree based secure multicast key distribution schemes" in Advances in Cryptology: Proceedings of Crypto 99, M. Wiener, ed., LNCS 1666, Springer-Verlag (1999), 624638.

[Rei93] Reiter, Michael, Kenneth Birman, and Robert van Renesse, "A security architecture for fault-tolerant systems," Technical Report TR93-1354, Department of Computer Science, Cornell University (June 1993). 29 pages.

[Rei94] Reiter, Michael K, "A secure group membership protocol," Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, California, May 14-16, 1994, IEEE Press (1994).

[Riv92] Rivest, Ronald L., The MD5 Message-Digest Algorithm, Request for Comments (RFC) 1321, Internet Engineering Task Force (1992).

[Riv96] Rivest, Ronald L., and Butler Lampson, "SDSI: A simple distributed security infrastructure," Version 1.1 (October 2, 1996).

[Rod97] Rodeh, Ohad, Ken Birman, and Mark Hayden, "Dynamic virtual private networks," Technical Report TR97-1654, Department of Computer Science, Cornell University (November 26, 1997). 16 pages.

[Sch96] Schneier, Bruce, Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons (New York, 1996).

[Sel00] Selcuk, A., C. McCubbin, and D. Sidhu, Probabilistic Optimization of LKH-Based Multicast Key Distribution Schemes, Internet Draft (work in progress), (January 2000), 10 pages. <draft-selcuk-probabilistic-lkh-00.txt>

[Sha-1] FIPS Publication 180-1, Secure hash standard, NIST, U.S. Department of Commerce, Washington, D.C. (April 1995).

[She98] Sherman, Alan T., "A new amortized approach to group initialization: Refinements and analysis," TIS Report No. 0754, Trusted Information Systems, Inc. Glenwood, MD (March 26, 1998). 12 pages.

[Sta97] Staddon, Jessica Nicola, "A combinatorial study of communication, storage and traceability in broadcast encryption systems," PhD Dissertation, Dept. of Mathematics, Univ. of California, Berkeley, CA (September 1997). 43 pages.

[Ste96] Steiner, Michael, Gene Tsudik, and Michael Waidner, "Diffie-Hellman key distribution extended to group communication," Proceedings of the 3rd ACM Conference on Computer and Communications Security, March 14-16, 1996. 7 pages.

[Ste97] Steiner, M., G. Tsudik, and M. Waidner, "CLIQUES: A new approach to group key agreement," IBM Research Report RZ 2984 (#93030) (December 12, 1997). 17 pages.

[Sti96] Stinson, D. R., "On some methods for unconditionally secure distribution and broadcast encryption," unpublished document (November 21, 1996). 35 pages.

[Vis96] Viswanathan, Vaidhyanathan, "Unconditionally secure dynamic conference key distribution," MS Thesis, University of Wisconsin-Milwaukee (December 1996). 28 pages.

[Wal97] Wallner, Debby M., Eric J. Harder, and Ryan C. Agee, "Key management for multicast: Issues and architectures," Internet Draft (work in progress), draft-wallner-key-arch-01.txt, Internet Engineering Task Force (September 15, 1998). 18 pages.

[Won97] Wong, Chung Kei, Mohamed G. Gouda, and Simon S. Lam "Secure group communications using key graphs," Technical Report TR-97-23, Dept. of Computer Science, Univ. of Texas at Austin (July 28, 1997). 24 pages.

[Won98] Wong, Chung Kei, Mohamed Gouda, and Simon S. Lam "Secure group communications using key graphs" in Proceedings of SIGCOMM 98, ACM Press (1998), 68-79.

10. Acronyms and Abbreviations

| | |
|------|---|
| DCCM | Dynamic Cryptographic Context Management (a DARPA project [Bal98]) |
| DNS | Domain Name System |
| GDH | Group Diffie-Hellman (a Group key exchange protocol) |
| LKH | Logical Key Hierarchy |
| LKH+ | Logical Key Hierarchy, with improved constant-time add-member operation |
| MSMP | Multicast Security Management Protocol (an NSA/Sparta effort [Harn99]) |
| NAI | Network Associates, Inc. |
| OFC | One-Way Function Chain |
| OFT | One-Way Function Tree |
| OFT+ | One-Way Function Tree, with improved constant-time add-member operation |
| SKDC | Single Key Distribution Center |
| SMG | Secure Multicast Group |
| TIS | Trusted Information Systems, Inc. |
| XOR | Bit-wise exclusive-or |

11. Authors' Addresses

Note: All correspondence relating to this document should be sent to David Balenson.

David Balenson
NAI Labs, Network Associates, Inc.
3060 Washington Road (Rt. 97)
Glenwood, MD 21738
USA

Phone: +1 443 259 2358
Fax: +1 301 854 4731
Email: david_balenson@nai.com
URL: www.nailabs.com

Dr. David A. McGrew
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA

Phone: +1 408 525 8651
Fax: +1 408 527 7099
Email: mcgrew@cisco.com
URL: www.cisco.com

INTERNET-DRAFT

August 25, 2000

Dr. Alan T. Sherman
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County (UMBC)
1000 Hilltop Circle
Baltimore, MD 21250
USA

Phone: +1 410 455 2666
Fax: +1 410 455 3969
Email: sherman@umbc.edu
URL: www.csee.umbc.edu/~sherman

12. Acknowledgments

This work was done as part of the Dynamic Cryptographic Context Management (DCCM) project at NAI Labs, Network Associates, Inc. (formerly the Advanced Research & Engineering (AR&E) Division of Trusted Information Systems (TIS)). Support for the DCCM project was provided by the Defense Advanced Research Project Agency (DARPA) through the Air Force Research Laboratory under Contract No. F30602-97-C-0277.

At the time of the work, David McGrew was a cryptography researcher at NAI Labs and Alan Sherman was a contractor on sabbatical leave from The University of Maryland, Baltimore County (UMBC). McGrew and Sherman designed and analyzed the OFT algorithm; Sherman devised the amortized induction procedure; and Balenson assisted with project management and final document preparation. In October 1998, McGrew became the head of the Cryptographic Software Development Group at Cisco Systems; now he is in the research division.

The prototype OFT implementation was carried out by Pete Dinsmore, Michael Ferguson, Mike Heyman, Peter Kruus, Matt Mundy, and Caroline Scace.

We would like to acknowledge contributions by several colleagues at what was TIS Labs. We are very grateful to Michael V. Harding for significant contributions to the original algorithm and for discussing implementation strategies with us. Dennis K. Branstad, Principal Investigator (former) of the DCCM Project, suggested the problem and provided constructive recommendations and technical guidance. Thanks also to Jay Turner and Michael Ferguson for helpful comments, and to Sharon Osuna and Caroline Scace for editorial suggestions.

Balenson, McGrew, Sherman expires February 25, 2001

[Page 32]
