Linköping Studies in Science and Technology

Thesis No. 1054

Security and Efficiency Tradeoffs in Multicast Group Key Management

by

Claudiu Duma

Submitted to the School of Engineering at Linköping University in partial fulfilment of the requirements for the degree of Licentiate of Engineering

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 2003

Security and Efficiency Tradeoffs in Multicast Group Key Management

by

Claudiu Duma

November 2003
ISBN 91-7373-770-4
Linköpings Studies in Science and Technology
Thesis No. 1054
ISSN 0280-7971
LiU-Tek-Lic-2003:53

ABSTRACT

An ever-increasing number of Internet applications, such as content and software distribution, distance learning, multimedia streaming, teleconferencing, and collaborative workspaces, need efficient and secure multicast communication. However, efficiency and security are competing requirements and balancing them to meet the application needs is still an open issue.

In this thesis we study the efficiency versus security requirements tradeoffs in group key management for multicast communication. The efficiency is in terms of minimizing the group rekeying cost and the key storage cost, while security is in terms of achieving backward secrecy, forward secrecy, and resistance to collusion.

We propose two new group key management schemes that balance the efficiency versus resistance to collusion. The first scheme is a flexible category-based scheme, and addresses applications where a user categorization can be done based on the user accessibility to the multicast channel. As shown by the evaluation, this scheme has a low rekeying cost and a low key storage cost for the controller, but, in certain cases, it requires a high key storage cost for the users. In an extension to the basic scheme we alleviate this latter problem.

For applications where the user categorization is not feasible, we devise a cluster-based group key management. In this scheme the resistance to collusion is measured by an integer parameter. The communication and the storage requirements for the controller depend on this parameter too, and they decrease as the resistance to collusion is relaxed. The results of the analytical evaluation show that our scheme allows a fine-tuning of security versus efficiency requirements at runtime, which is not possible with the previous group key management schemes.

This work has been supported by Vinnova (Swedish Agency for Innovation Systems) and ECSEL (Excellence Center in Computer Science and Systems Engineering in Linköping).

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Abstract

An ever-increasing number of Internet applications, such as content and software distribution, distance learning, multimedia streaming, teleconferencing, and collaborative workspaces, need efficient and secure multicast communication. However, efficiency and security are competing requirements and balancing them to meet the application needs is still an open issue.

In this thesis we study the efficiency versus security requirements tradeoffs in group key management for multicast communication. The efficiency is in terms of minimizing the group rekeying cost and the key storage cost, while security is in terms of achieving backward secrecy, forward secrecy, and resistance to collusion.

We propose two new group key management schemes that balance the efficiency versus resistance to collusion. The first scheme is a flexible category-based scheme, and addresses applications where a user categorization can be done based on the user accessibility to the multicast channel. As shown by the evaluation, this scheme has a low rekeying cost and a low key storage cost for the controller, but, in certain cases, it requires a high key storage cost for the users. In an extension to the basic scheme we alleviate this latter problem.

For applications where the user categorization is not feasible, we devise a cluster-based group key management. In this scheme the resistance to collusion is measured by an integer parameter. The communication and the storage requirements for the controller depend on this parameter too, and they decrease as the resistance to collusion is relaxed. The results of the analytical evaluation show that our scheme allows a fine-tuning of security versus efficiency requirements at runtime, which is not possible with the previous group key management schemes.

Acknowledgements

First of all I would like to express my gratitude to my supervisor Nahid Shahmehri for introducing me to the incredible world of research in security. She provided great support during my studies, and, not rarely, her care crossed the boundaries of work. Without her continuously keeping me on track this thesis would not have been possible.

Many thanks go to Patrick Lambrix who showed interest in my work and provided invaluable support throughout the development of this entire thesis. I also direct my appreciation to Germano Caronni for his inspiring insights.

I am thankful to my colleagues at IISLAB (Laboratory for Intelligent Information Systems) and ADIT (Division for Database and Information Techniques) for their friendship and support. Special thanks are addressed to Lin Han, with whom I have started my journey into the captivating area of multicast security.

I would like to express my endless love to my wife, Aurelia. She was always beside me, offered understanding, patience, and support. She completes me and gives sense to everything.

I am grateful to my parents, Gavril and Stela, for all their love, care, and support.

In addition, I acknowledge the financial support by Vinnova (Swedish Agency for Innovation Systems) and ECSEL (Excellence Center in Computer Science and Systems Engineering in Linköping).

Contents

Chapter 1
Introduction

1.1 Introduction

Here we give the motivation for our research, state the contributions of our work, and set the thesis outline.

1.2 Motivation

The continuous evolution of the Internet amplifies the demand for efficient and secure network technologies capable of responding effectively to the challenges posed by the emerging applications. As such, the latest development of multimedia streaming, collaborative, and push-oriented applications coupled with the advancement in high-speed networking, broadband, and wireless technologies have driven the research and development efforts for enabling the Internet with efficient multicast communication capabilities.

Internet Protocol Multicast (IP Multicast) is an emerging set of technologies and standards that provides efficient delivery of data from a sender to a group of receivers. It reduces the sender transmission overhead, the network bandwidth usage, and the latency observed by the receivers.

Although this concept is very attractive and the multicast technology has existed for several years, multicast was primarily used in the research community, whereas its adoption by the Internet Service Providers (ISPs) was rather moderate. This drift was partially motivated by ISPs waiting for the sophisticated business enabling applications, while application developers were expecting a high deployment and support for multicast in the Internet [MM03]. Also, the moderate adoption of the multicast enabled networks can be explained by the potential problems introduced

by the new technology, such as network flooding, as well as the lack of control and security.

However, this situation is likely to change since the demand for multimedia applications increases and multicast emerges as an enabling technology for a broader variety of newer applications, e.g. pay TV, teleconferencing, distance learning, collaborative workspaces, and control and command applications [WZ01, MM03]. Moreover, a great deal of active research is currently investigating alternative technologies for multicasting in the Internet, such as the overlay multicast [CRS+00, PSV+01]. Work has also been done in reliability, scalability, quality of service, and ease of deployment [Bir99, GHK02, RFC3048, SA01, SM02]. As a result, at this stage the security in multicast applications is a concern. The maturity of multicast security solutions has the potential to enable the use of multicast for confidential and high-value content distribution, as well as to foster the adoption of multicast by newer and sophisticated emerging applications [JA03].

As shown by the previous work, e.g. [Kru98] and [CGI+99], the design of an efficient group key management scheme, capable of scaling to large and dynamic multicast groups, is a critical problem characterizing the multicast security. However, the efficiency and the security are competing requirements and balancing them to meet the application needs is still an open issue [FJA02, DSL03a]. Therefore, in this thesis, we address the problems of data confidentiality and access control in multicast communication, with focus on the security versus efficiency tradeoffs in multicast group key management.

1.3 Problem formulation

The Internet Protocol Multicast (IP Multicast) adopts an open model that allows group membership to be transparent to the sender. Receivers can join and leave an IP Multicast group, identified through a Class D IP address, by sending Internet Group Membership Protocol (IGMP) messages to their local routers [RFC1112]. A sender transmits datagrams to a multicast group by simply directing them to the corresponding multicast group address, without the need to know the identity of each receiver. Instead, it is the responsibility of the multicast capable routers to communicate with each other, using multicast routing protocols, and deliver the datagrams to all members of the group. This model is beneficial because it favors scalability – very little state information is required, and it provides some anonymity for the group members [Shi98]. However, the open model raises also security concerns, such as eavesdropping and theft of services.

Although these issues are not new to multicast as they existed before in the unicast communication, the unicast security solutions can not be directly applied to the multicast case. This is because the unicast security solutions, such as the Secure Socket Layer (SSL) [SSL], are specialized to leverage security in pairs of communicating parties. Naively applying these mechanisms would mean to actually

establish unique unicast secure channels from the sender to each of the receivers, hindering therefore the very wanted property of multicast - the efficient delivery of data from one sender to multiple receivers. Thus, in order to extend the security to multicast communication settings, a *secure group model* is adopted.

In the secure group model, a symmetric cryptographic key, called the *group key*, is established among all the authorized members of the multicast group. The multicast traffic is then encrypted with this key assuring that only the legitimate users, having the group key, can decrypt the data.

The group key needs special handling as it must be distributed only to authorized group members. However, the set of authorized group members could be quite large (e.g. hundreds of thousands) and very dynamic as new members can join or leave the group (e.g. leave due to expiration of their subscription). Whenever the group membership changes, the group key must be updated and redistributed accordingly to reflect the new membership status. This operation is called *group rekeying*. The rekeying triggered by the leaving members can be quite costly from the communication point of view, increasing linearly with the group size in static key management schemes. Thus, the design of an efficient group key management scheme, capable of scaling to large and dynamic multicast groups, is a critical problem characterizing the multicast security. However, the efficiency (to achieve scalability) and the security requirements are competing constraints, and balancing them to meet the application needs is still an open issue.

The efficiency requirements are in terms of minimizing the rekeying communication cost, the key storage cost (for each group member and for the group controller), and the computation complexity [Kru98, BR02].

The security requirements are in terms of backward secrecy, forward secrecy, and resistance to collusion. Backward secrecy (or backward access control) refers to the impossibility of a newly joined user to gain access to previous group keys. Forward secrecy (or forward access control) refers to the impossibility of a former group member, who has been excluded, to gain access to future group keys. The third requirement, the resistance to collusion, refers to the impossibility of a coalition of two or more excluded members to gain access to future group keys by combining their keying material. Although this requirement is considered by the previous work in multicast security, its treatment takes extreme forms; either no resistance to collusion, e.g. [CEK99], or the opposite, perfect resistance to collusion is provided, e.g. [WCS+99]. In any case, the resistance to collusion is achieved at the expense of efficiency. Furthermore, applications may have certain assumptions regarding the users and their accessibility to the multicast channel; implicitly, they may have trust assumptions related to collusion. Based on this, we argue that adequate group key management schemes can be designed to balance the given collusion constraints against efficiency.

1.4 Contributions

The overall contribution of this thesis is in the area of group key management. The motivating applications for our work are the multicast software delivery and the content distribution. However, the work presented here can be applicable to any multicast application with one-to-many communication predominance, where a logical point of control and group "ownership" can be identified.

Our detailed contribution is as follows:

- We identify the collusion resistance requirement as a potential knob for setting the tradeoffs between security and efficiency of group key management. In particular, we observe that resistance to collusion has an impact on the efficiency of any scheme. Also, we note that applications might have certain knowledge regarding the users and their accessibility to the multicast channel; implicitly, they may have trust assumptions related to collusion. Therefore, by taking into consideration such knowledge we can set the security-efficiency tradeoffs to best meet the application needs [DSL03a].
- We extend the definition of collusion resistance requirement and formalize it based on user categorization and accessibility to the multicast channel. Different configurations of categories give different constraints on collusion. We show how the previous work can be reformulated as special cases of our definition [DSL03a].
- We propose a category-based group key management scheme for the general case of collusion resistance requirement that balances the given collusion constraints against efficiency [DSL03a]. For applications that have certain assumptions on user accessibility to the multicast channel, such as the multicast software delivery, this scheme improves the rekeying cost and the storage cost for the controller. However, the storage cost for the user is high. We alleviate this problem and improve the overall scheme using a spanning hash key tree mechanism [DSL02].
- For some applications the user categorization is not always possible. For these cases we propose a cluster-based group key management scheme that can be used to secure any multicast application without assuming specific properties of that application. The cluster-based scheme provides the possibility to fine-tune the security versus efficiency tradeoffs even at system runtime; therefore, the scheme is flexible to adapt to changes in the working environment and in the security assumptions [DSL03b].

1.5 Thesis outline

The remaining of the thesis is structured as follows:

Chapter 2 defines the multicast communication and presents the efficient implementations of multicast in the Internet, namely the traditional IP Multicast and the emerging overlay multicast. Then, we review the existing and emerging

multicast application domains and examine their security requirements. The chapter ends with a brief survey of the multicast security issues and the corresponding security working areas.

Chapter 3 presents the group key management as the core mechanism for achieving confidentiality and access control in multicast communication. We describe the secure group model and give the security and efficiency requirements for the group key management. This leads us to a more detailed description of the problem addressed in this thesis, namely, the study of the tradeoffs between security and efficiency in multicast group key management.

Chapter 4 generalizes the definition of collusion resistance requirement and formalizes it based on user categorization. We show how the previous work can be reformulated as special cases of our more general collusion resistance requirement definition. Also, we present our category-based group key management scheme that can balance the efficiency versus the collusion constraints for applications that have certain assumptions on user accessibility to the multicast channel, such as the multicast software delivery.

Chapter 5 proposes an improvement to our initial category-based group key management scheme. The improvement is based on a spanning hash key tree mechanism which dramatically reduces the storage requirements for the users as well as for the group controller.

Chapter 6 presents our second scheme, the cluster-based group key management, which addresses the applications where the user categorization is not feasible.

Chapter 7 describes, from an architectural point of view, the related work in the group key management area. We focus on the dynamic group key management schemes which constitute an efficient class of secure and scalable multicast schemes.

Chapter 8 concludes the thesis and opens venues for further research.

Chapter 2
Multicast communication

2.1 Introduction

Multicast is a type of communication involving *one sender* and *a group of receivers*, where the sender delivers the same message to the whole group of receivers.

Multicast communication plays an important role in today's computer networks and communication systems as there are many existing and emerging applications based on this type of communication. These applications include bulk data transfer (e.g. the transfer of a software upgrade from the software developer to users needing the upgrade), streaming continuous media (e.g. the transfer of the video, audio, and text of a live lecture to a set of distributed lecture participants), collaborative applications (e.g. whiteboard or teleconferencing), data feeds (e.g. stock quotes), and distributed games.

Multicast is also a communication primitive abstraction defined by the sending of a data packet from one sender to a group of receivers with *a single send operation*, instead of issuing multiple send operations to individual receivers. This amounts to much more than a convenience for the programmer. It also enables the implementation to be efficient in its utilization of network bandwidth, reduces the sender work, and the latency observed by the receivers, as well as it allows the provision of stronger delivery guarantees that would otherwise be impossible [CDK01].

Consider a communication network composed of nodes and communication links interconnecting these nodes. An efficient multicast implementation, used to deliver messages from one source to a set of receivers, would send the message no more than once over any communication link. This greatly reduces both the bandwidth

utilization as well as the total transmission time. Metrix Systems [MS] compares the time in unicast versus multicast data transfer over an Ethernet with a speed of 10 Mbps (Mega bits per second). For instance, it is shown that transmitting 300 MB of data to 1000 receivers requires 67.7 hours in unicast and it takes only 4 minutes using multicast. This makes multicast particularly interesting for both service providers and Internet network providers.

2.2 Types of multiparty communication

Multicast is a type of communication involving more than two communicating parties. Other related types of communication, involving multiple parties are:

- Broadcast. Similar to multicast but still contrasting the concept of group, broadcast refers to sending of messages from one sender to all the hosts in a network. With broadcast, there is no restriction with respect to the group of receivers; everyone is sent the content, whether they want it or not. From this perspective, multicast is more selective as only the hosts who choose to be in the multicast group will get the data.
- Anycast. The anycast communication refers also to sending of messages from one sender to a group of receivers; however, in the case of anycast, the group of receivers is usually a group of servers providing a certain service. The difference is that the anycast group is not used for the actual exchange of data (from the sender to the receivers), as in multicast. Anycast only locates an available server from a group. After server localization, the communication follows a traditional unicast client-server pattern.
- Concast. The concast communication involves multiple senders sending to one receiver. An example of a concast scenario is when simulation results are transmitted from several hosts to one receiver, who then evaluates the results.
- Multipeer or group communication. It is characterized by a group of computers communicating with each other, each sending messages to all others in the group. In multipeer communication any host from the group of receivers can also be sender. From this perspective, multipeer communication can be simulated as a set of superposed multicast communications.

2.3 Multicast group

Multicast communication is about communicating from one sender to a group of receivers. The group of receivers is called the *multicast group* and is a central concept for multicast communication. Typical characteristics of a multicast group include [WZ01]:

- Openness to new members
- Openness to senders
- Dynamics

- Lifetime
- Heterogeneity
- Security

A group can be open or closed with regard to new members. In an open group, any new receiver can receive the multicast traffic without any registration with the sender. In other words, the group of receivers is transparent to the sender, and the sender is not aware of the exact identity of all the receivers. Of course, this does not exclude the possibility that some of the receivers from the group are actually known to the sender. On the other hand, in a closed group all the receivers are known.

A group can also be closed or open with regard to senders. In a closed group, only registered senders can send messages to this closed group. In contrast, data from any sender can be forwarded to open groups.

In static groups, membership of the group is predetermined and does not change during an established communication. In dynamic groups, membership can change during communication.

Regarding the group lifetime, a distinction can be made between permanent groups and transient groups. A permanent group exists even if it currently has no members, whereas a transient group exists only as long as the group has members.

It is also possible to differentiate between heterogeneous and homogeneous groups. In heterogeneous groups, the members have different capabilities, for example, with respect to their network connection (e.g. in terms of available bandwidth or connectivity – continuous versus intermittent). On the other hand, in homogeneous groups all members have the same capabilities.

The multicast communication has certain security requirements, which might be static for the duration of the whole communication, or they can vary during the communication. Moreover, the requirements may differ for the different data streams involved (e.g. video, audio, and text).

Operations on multicast groups are:

- Member joins group. Generally, the group members have the incentive to join the multicast group (e.g. users are interested in watching a certain broadcast event).
- Member leaves group. In contrast to the joining operation, the members do not always have the incentive to leave. Therefore, in many instances, an exclusion operation is needed through which users are rather forced to leave.

Applications can have their own requirements on the multicast group. For instance, a pay TV provider would want to restrict the group membership only to those who subscribed for the service. A more detailed analysis of the application requirements is given later in section 2.5.

2.4 Multicast in the Internet

As we mentioned, multicast implementations can efficiently deliver data from one sender to multiple receivers. However, for this to happen, the communication infrastructure must support and efficiently implement multicast. Therefore, in this section, we describe how multicast support is provided in the Internet, show the open group model implemented by the multicast infrastructure, and review the main security requirements of the multicast infrastructure.

From a network perspective, the multicast abstraction can be implemented in several ways [KR02] including one-to-all unicasts, explicit multicast, and application level multicast.

The one-to-all unicast emulates the multicast communication by establishing unicast communication channels from the sender to each of the receivers. From this point of view the multicast can be seen an N*unicast, where N is the group size. Although possibly attractive by its simplicity, the one-to-all unicast implementation of the multicast does not account for efficiency, and therefore it has value only for low scale settings [WZ01]. Instead, the traditional mechanism to support multicast communication in the Internet is the explicit multicast, as implemented in the Internet Protocol Multicast (IP Multicast). Although very efficient with the network resources, issues such as reliability, manageability, scalability, quality of service, and ease of deployment have prevented IP Multicast from being largely enabled in the Internet. In order to solve these issues, alternative implementations for the multicast have been recently proposed, such as the overlay multicast, which aims at providing multicast capabilities for the Internet at the application layer.

Following, we briefly present the IP Multicast and the overlay multicast.

2.4.1 IP Multicast

IP Multicast implements explicit multicast in the Internet, see Figure 2-1. A single datagram is sent from the sending host. This datagram or a copy of this datagram is then replicated at the network routers whenever it must be forwarded on multiple outgoing links to reach the receivers. This approach poses a number of challenges:

- How to identify the receivers of a multicast datagram?
- How to address a datagram sent to these receivers?
- How to route a datagram from the sender to all the receivers?

The Internet community began to discuss these issues in the mid 1980's using the Internet Engineering Task Force (IETF) Requests for Comments (RFC). The result was the adoption of a host group model [RFC1112] for the multicast in the Internet that has been first defined in [CD85] as: "a host group is a set of network entities sharing a common identifying multicast address, all receiving any data packets addressed to this multicast address by senders (sources) that may or may not be

members of the same group and have no knowledge of the groups membership". That is, a single identifier is used for the group of receivers, and a copy of the datagram that is addressed to the group using this single identifier is delivered to all of the multicast receivers associated with that group. In the Internet, the single identifier that represents a group of receivers is a class D multicast address, which differs from classes A, B, and C that are used for unicast communications. The multicast address space, assigned by the Internet Assigned Number Authority (IANA), covers the range 224.0.0.0 – 239.255.255.255 in IPv4. An initial assignment of IPv6 multicast addresses is defined in [RFC2375].

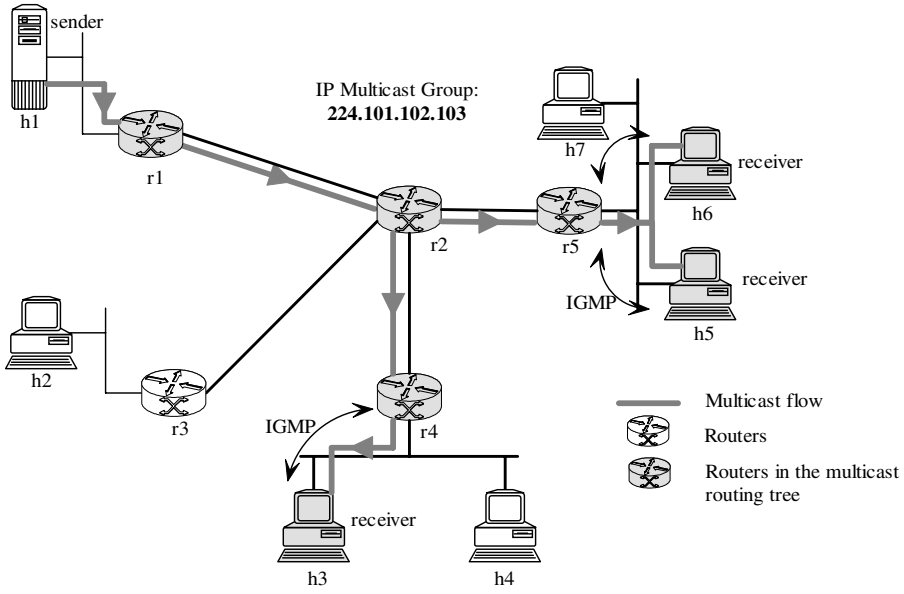


Figure 2-1: IP Multicast

IGMP

The group of receivers associated with a class D address is referred to as an IP Multicast group. An example of an IP Multicast group, identified by the address 224.101.102.103, is depicted in Figure 2-1. Currently, there are three hosts, h3, h5, and h6, in this multicast group. However, any other host can join the group. To join the group a host uses the Internet Group Membership Protocol (IGMP) [RFC2236]. The IGMP works between hosts in a Local Area Network (LAN) and the router(s) serving that LAN. Upon a join request from a host (through IGMP), the router will forward all the multicast traffic with the destination 224.101.102.103 (in our example) to the joined host. For instance, router r4 forwards the multicast traffic

with the address 224.101.102.103 to host h3 while the router r5 forwards the same multicast traffic to hosts h5 and h6.

Note that any host can join any multicast group. Moreover, the IGMP does not restrict the number of members of a multicast group, and there is no restriction on the number of multicast groups a host can join.

Multicast routing

As we saw, the hosts use IGMP to join a multicast group. The routers that have hosts that joined a certain multicast group will forward the multicast traffic to those hosts. The question is how is the multicast traffic efficiently routed from the sender to the routers that need it (i.e. routers that have hosts attached to the multicast group)? The solution is that datagrams addressed to a certain multicast group are routed following a *multicast routing tree*. In our example, the multicast routing tree is (r1 (r2 (r5 r4))). Each leaf in this tree is an "edge" router that has hosts attached to that multicast group, e.g. r4 and r5. However, the routing tree can contain, as inner nodes, routers that have no host attached to the group, e.g. r2. Therefore, for a given multicast group, the problem of the multicast routing is to find the multicast tree optimized according to current cost considerations. In practice, two approaches have been adopted for determining the multicast routing tree [KR02]:

- Multicast routing using group-shared tree: A single multicast routing tree is used for a given multicast group regardless of the sender. That is, all the packets sent to a multicast group are to be routed along the same single multicast tree, independent of the exact sender. The problem of finding a *minimum-cost overall tree* is known as the *Steiner tree problem* [Hak71], and it has been shown that this problem is NP complete. Although good heuristics exist for the Steiner trees, e.g. [WE93], this approach is not actually used for routing in the Internet as it requires every single router in the network to know the state of every link in the network. Instead, a *rendezvous-point* approach has been preferred. For each multicast group a rendezvous (center) point is first identified; this is the root of the multicast routing tree. Then, all the "edge" routers having hosts joined to the multicast group will try to find a path to the identified center. The result is the multicast routing tree with the root in the rendezvous point.
- Multicast routing using source-based tree: A multicast routing tree is constructed for each of the senders. The approach reduces to the problem of finding the *minimum-cost paths spanning tree rooted at the sender*[1]. To solve this problem, Dijkstra proposed an iterative algorithm [Dij59]. Although this approach makes excellent use of network bandwidth, the issue is that each router must have knowledge of some spanning tree for the method to be applicable. Therefore, the actual algorithm used in the Internet, referred to as the *reverse path forwarding (RPF),* is based on flooding and does not require routers to know about spanning trees.

[1] We will return to the shortest paths spanning tree in chapter 5 when we will use it as part of our technique to reduce the key storage cost in group key management.

Each of the multicast routing techniques has advantages and disadvantages. For instance, the RPF requires little state information to be maintained by the routers, which makes it apparently very scalable. However, as RPF uses flooding, it is not very suitable for large networks. On the other hand, the rendezvous point technique is attractive because it completely eliminates flooding. Still, the problem is that center points constitute potential bottlenecks for the traffic and also represent a single point of failure. For a more detailed comparison of the basic techniques for multicast routing we refer the reader to [KR02, WZ01]. Nevertheless, the multicast routing protocols that are standardized and used in the Internet are actually based both on the source-base trees, e.g. Distance Vector Multicast Routing Protocol (DVMRP) [RFC1075], as well as on group-shared trees, e.g. Protocol Independent Multicast (PIM) [RFC2362].

There are two important issues to note:

- First, IP Multicast is not a connectionless service as state information (entries regarding the multicast routing trees) for a multicast connection must be established and maintained in routers, and hence consumes router resources. This raises scalability issues with regard to the multicast group size.
- Second, although different multicast routing algorithms exist, they all use the IGMP as the basis for group management. That is, a host uses IGMP to notify the routing system that it should deliver packets for a particular multicast group to this host. In its turn, the multicast routing system is designed to distribute datagrams to a set of links hosting group members, i.e. to grant access and not to prevent access to information.

2.4.2 Overlay multicast

Although most routers today implement the IP Multicast, the ISPs and network managers are reluctant to actually enable it, partly because of several drawbacks of the IP Multicast [CRS+00]. First, IP Multicast requires routers to maintain per group state (i.e. state information for each multicast group), which introduces high complexity and serious scaling constraints at the IP layer. Second, IP Multicast is a best effort delivery service; i.e. it makes its “best effort” to deliver data from the sender to the receiving hosts, but it makes no guarantees regarding delivery, order of delivery, integrity of the data, etc. However, providing higher level features for multicast, such as reliability, congestion control, or flow control, has been shown to be much harder than in the unicast case [Mil99]. Finally, IP Multicast calls for changes at the infrastructure level, slowing down the pace of deployment.

Responding to these issues, recent research proposes application level multicast as an alternative implementation of the multicast in the Internet. In this approach, the multicast functions, such as group membership, multicast routing, and datagram duplication, are implemented at end systems assuming only IP Unicast services. This is somehow similar to one-to-all unicast, where the sender initiates separate

connections with each of the receivers. Still, the difference is that the receivers are also involved in the process of duplication and forwarding of data to other receivers. Therefore, instead of having the sender itself transmit a copy to each receiver, the sender transmits a copy to a smaller number of receivers, who then make copies themselves and forward these copies further to other receivers. Each of these receivers may then duplicate and forward copies to yet additional receivers, and so on [KR02]. This requires that receivers set up and maintain an application level distribution infrastructure, called the overlay network [CRS+00, PSV+01].

In conclusion, the IP Multicast is more efficient than the overlay multicast. However, the overlay multicast has the potential of providing feasible deployment solutions. Therefore, combining the native IP Multicast, where available, with emerging overlay networks might be a viable solution for enabling multicast on a large scale in the Internet.

2.4.3 Multicast infrastructure model

The exact properties of multicast depend partly on the multicast implementation. Thus, we note that we have considered an open model for the multicast infrastructure, as implemented in the host group model underlying the IP Multicast. The properties of such a model are:

- Open group membership: Group membership is transparent to the source as the source can not control which hosts join the multicast group. Multicast group members can join or leave at will. Moreover, there is no restriction on the number of groups a member can join or on the number of members a group can have.
- Open access to senders: Any host can send data to the multicast address; this actually extends the IP Multicast to a "by default" many-to-many communication type.

From the security point of view this is the most challenging model [JA03]. Other multicast models provide more restrictive frameworks that may make it easier to deal with some security aspects. For example, in the overlay multicast, the group membership, datagram duplication, and datagram forwarding are handled at the application level, apparently allowing for more group control. However, this model also introduces new issues, such as the trust in the receiving peers to perform their multicast and security related functions (e.g. duplication and forwarding of packets only to authorized group members). Therefore, the open model is general enough to be relevant to any multicast implementation, be it at the network level (the IP Multicast) or at the application level (the overlay multicast).

2.4.4 Multicast infrastructure security requirements

The multicast infrastructure generates a number of requirements with regard to security [HT00]. Although some of these requirements and issues exist also in the

unicast communication, the multicast scenario opens new dimensions for vulnerabilities. This is mainly due to the adopted open model.

The open model is beneficial because it provides a lightweight join operation, i.e. the source is not required to maintain a state for all group members, and it even allows some anonymity for the group members [Shi98]. This very same property, however, makes the infrastructure (i.e. IP Multicast) vulnerable to security threats, such as denial of service. DoS attacks can be caused by a malicious receiver joining a large number of multicast groups, thereby utilizing large amounts of bandwidth and router resources. A sender could also mount a DoS attack by sending a stream at a very high rate to a well known multicast address, overloading all the networks spanned by the corresponding multicast routing tree. Attacks might be mounted also against the routing mechanisms by injecting altered control data which might corrupt the multicast routing tree.

A detailed analysis of the infrastructure security issues is given in [HT00]. The main requirements include:

- Limit and control the hosts' ability of joining multicast groups
- Limit and control the hosts' ability of sending data to multicast groups
- Assure the integrity of the multicast routing mechanism

2.5 Multicast applications

Multicast has applications in the educational, commercial, and military arenas, spanning a large domain, from push-oriented technologies (e.g. information dissemination and software distribution) to collaborative applications, distributed games, and command and control applications. Multicast is also part of emerging communication infrastructures, distributed middlewares, and distributed databases.

Most of the early multicast application development has been fostered by the MBone. Created in 1992, the MBone [SRL96] is a set of multicast enabled sub-networks connected by IP tunnels[2]. The MBone provided researchers and companies with a test bed for developing and experimenting with multicast applications [MMA].

However, the emergence of multicast enabled commercial networks, such as the WorldCom's very high performance backbone service (vBNS) [JNM+98] has also attracted commercial customers with unique requirements such as high performance IP Multicast. Typical applications include satellite broadcast replacement, audio and video distribution, multimedia conferencing, and distributed simulation [LL03].

[2] Tunneling is a technique that allows multicast traffic to traverse parts of the network that are not multicast enabled, by encapsulating multicast datagrams within unicast datagrams.

2.5.1 Multicast application requirements

The IP Multicast provides only a best effort service to the multicast applications. However, applications have different requirements and many of them need more complex transport protocols. Starting from these requirements, Miller [Mil99] categorizes the multicast applications based on four criteria, as shown in Figure 2-2: multimedia or data-only, and real-time or non real-time applications. These application categories have widely varying requirements for the transport protocol. For example, the set of applications in the left quadrants require low latency or low jitter tolerance. Some of these applications do not have strict error-free requirements; others do not require a high level of scalability. On the other hand, the set of applications in the right quadrants generally lack stringent latency requirements, but they usually have strict reliability requirements and generally call for a higher level of scalability.

	Real-time	**Non real-time**
Multimedia	• Video server • Video conferencing • Internet audio • Multimedia events	• Replication - Video and web servers - Kiosks • Content delivery
Data-only	• Stock quotes • News feeds • Whiteboards • Distributed games	• Data delivery - Server-server - Server-desktop • Database replication • Software distribution

Figure 2-2: Multicast applications

2.5.2 Multicast application security requirements

Many of the multicast applications also require security [RFC3170]. Multimedia streaming applications give an Internet based alternative to the pay (cable or satellite) television. Additionally, there would likely be an equivalent to “pay-per-view”, in which a scheduled event, such as a special sport event, is made available to viewers who paid for it [Mil99]. These examples would need to include security to prevent nonpaying viewers from gaining access to the content and to protect the copyrighted material distributed through the network. Moreover, recent research has been invested in the convergence of high definition television (HDTV) with Internet transport via multicast Real-time Transport Protocol (RTP) over UDP/IP. The focus

is on two tasks: scaling to very high quality and scaling to very large numbers of participants [GPR+02]. In addition, it is also assumed that the target group could change frequently. As we shall see, all these sum up to tighten the efficiency constraints on the group management mechanisms.

Data streaming applications, such as stocks, bonds, commodity feeds, and news feeds, could either be offered publicly or specialized versions could be offered on a paid subscription basis. Moreover, providing anonymity for the users in the multicast group might also be a requirement of these applications.

Bulk data transfer applications can also be offered as a subscription service. For instance, a lot of interest has been shown by companies in software delivery using multicast. Metrix Systems' Vision64 [MS], StarBurst's OmniCast [Sam98], Inria's WebCanal [Inria] are examples of fully fledged multicast software delivery products. In these applications, besides confidentiality, a paramount objective is to assure the integrity of data and the authenticity of the sender. A detailed comparison of software delivery products and their requirements is given in [Han01]. Informative content, such as electronic magazine or newspapers, could be delivered electronically to paying subscribers as well. Again, there is a need to have security controls in place to restrict the content only to valid subscribers.

In command and control applications multicast can efficiently be used to optimize the time of transmission of commands from the decision and control points to soldiers in the battlefield. The commands are strictly confidential and should not be accessible to adversaries.

Collaborative applications, including video, data conferencing, and network based games, may require security as well. For example, different companies may collaborate on technical projects and want to have periodic cyberspace meetings over the Internet [WZ01]. The participants would likely want to keep this technical collaboration secret from the Internet as a whole, yet use the Internet as a common networking medium.

In general, the multicast application requirements may include any combination of the following [RFC3170]:

- Confidentiality and group membership control
- Data integrity and source authentication
- User privacy and anonymity
- Copyright protection
- Availability

Security in multicast applications is therefore a concern, as the maturity of multicast security solutions has the potential to enable the use of multicast for confidential and high-value content distribution, as well as to foster the adoption of multicast by emerging sophisticated applications.

2.6 Multicast security areas

In this section, we first list the main multicast security areas that emerged as response to the specific requirements and constraints in multicast; afterwards, we identify the domain area of this thesis.

Security requirements in multicast are driven by both the infrastructure (section 2.4.4) and the applications (section 2.5.2). Although mature security controls and techniques exist to deal with most of these requirements (e.g. data confidentiality, integrity, and authentication) and provide secure unicast communication, unicast controls can not be directly applied to the multicast communication. The security mechanisms for unicast are not adequate for the multicast scenario since multicast security mechanisms are under tighter scalability and efficiency constraints [WZ01].

Therefore, responding to the security issues in multicast the work has been divided into several areas, including:

- *Multicast data confidentiality and group membership control*: As the data traverses the public Internet, a mechanism is needed to prevent unauthorized access to it. Also, as the authorized group of users changes in time (by users leaving or joining the group), mechanisms are needed to allow group membership control. At the core of such mechanisms are cryptographic primitives and *group key management* schemes.
- *Multicast source authentication*: Typical source authentication schemes, such as digital signatures, exist. However, the computation complexity of producing and verifying digital signatures, as well as the length of the signature, may be significant. Therefore, more efficient solutions have been proposed as response to the requirements of multicast, a survey being given in [CGI+99]. More recent work on this problem is reported also in [IP02].
- *Multicast receiver access control*: Aims at controlling the ability of hosts to join the IP Multicast groups, therefore alleviating the denial of service threats associated with a malicious host joining many multicast groups and taking up network resources. The need for secure IGMP has been first pointed out in [RFC1949]. More recent work in this direction has been reported in [JA02] and [GJV+03].
- *Multicast sender access control*: Similarly to the above, sender access control aims at controlling who can actually send data to the multicast group, avoiding situations where malicious senders would flood the multicast groups. Solutions similar to the secure IGMP are considered. Note that, although multicast receiver access control and sender access control could potentially solve big issues of denial of service, they will need to have support in the routing infrastructure adding therefore to the complexity and, possibly, hindering the scalability.
- *Multicast group policy*: The correct definition, implementation, and maintenance of policies governing the various mechanisms of multicast security are critical factors. Two general categories of policies are

considered - the policies governing group membership and the policies regarding security enforcement [HCD01].

- *Multicast fingerprinting*: Multicast fingerprinting offers solutions to obtain unique fingerprints in a multicast environment while also maintaining the efficiency of multicast [BPC99, CS01].
- *Traitor tracing*: In combating the illegal distribution of copyrighted material, an increasing role is played by the traitor tracing mechanisms, which enable the tracing of illicit users leaking cryptographic keys used by pirated devices [CFN94]. Recent work has proposed the integration of the revocation schemes with the traitor tracing schemes [NNL01].

Group membership control

From the list above, the issue of efficient group membership control, capable of scaling to large and dynamic multicast groups, has been recognized as being a critical problem characterizing the multicast security [Kru98, CGI+99]. This constitutes also the problem addressed in this thesis.

From the point of view of group membership control we can distinguish two main categories of scenarios in multicast communication applications:

- A push-oriented scenario with a single sending party. The sending entity is usually the owner of the transmitted data, and it is directly interested in assuring that only authorized users (e.g. who paid the subscription) have access to the multicasted data. Therefore, in such a scenario a unique point of authority and control over the group membership can be identified.
- A collaborative scenario where any party can assume the sender role. In general, in such a scenario the ownership of the group is shared by all the participants, and so is the authority and control over the group membership. Nevertheless, a centralized control could be adopted in these scenarios too. In this case, trust is invested in a third party responsible for the security of the multicast group.

The focus in this thesis is the push-oriented scenarios where a unique (logical) party has ownership of the multicast group and controls the receivers' access to the transmitted data. Applications in this category include multimedia streaming (e.g. pay TV), data streaming (e.g. distance leaning), bulk data transfer (e.g. software delivery), and command and control. These applications usually involve a provider multicasting confidential or high value data. In the latter case, the multicasted data is a source of revenue for the provider. Therefore, the content is provided on a subscription basis such that only users who pay should have access to the multicast group. To be profitable, such services need to break into the mass market, thus imposing tough requirements for the underlying security mechanisms to scale to very large and dynamic multicast groups.

In the next chapter, we introduce the secure multicast group as a model for achieving confidentiality and group membership control for the multicast group. At

the core of the secure multicast group are the group key and the group key management schemes.

Chapter 3
Group key management

3.1 Introduction

In this thesis we address the issues of data confidentiality and group membership control in multicast communication. The problem is that multicast data traverses public networks, and therefore it could be potentially accessed by any unauthorized party, sniffing the physical links. Moreover, the multicast routing protocols are designed to distribute datagrams to a set of routers hosting group members, i.e. to grant access and not to prevent access to information. Therefore, the main goal of this thesis is to propose security mechanisms, based on cryptographic primitives, to assure that only authorized receivers can get access to the data transmitted through the multicast channel.

Data confidentiality and integrity are known issues in communication networks and there exist mature security technologies to address them, such as the Secure Socket Layer (SSL) [SSL]. However, as these technologies have been created to bring security into a one-to-one unicast communication setting, they can not be directly applied to a one-to-many multicast scenario. Naively applying these mechanisms would mean to actually establish unique unicast secure channels from the sender to each of the receivers (i.e. a one-to-all secure unicasts emulation of the multicast). However, this would hinder the very gain of multicast communication – the efficient transmission of data from one sender to multiple receivers.

Instead, a *secure group model* is adopted for securing multicast communication, see Figure 3-1. At the core of this model there is a cryptographic symmetric key, called the *group key* (GK). All the data transmitted to the multicast group is encrypted by the sender using this group key, whereas the group key is made

available only to authorized receivers of the multicast group. Therefore, only authorized users, having the group key, can decrypt and access the data.

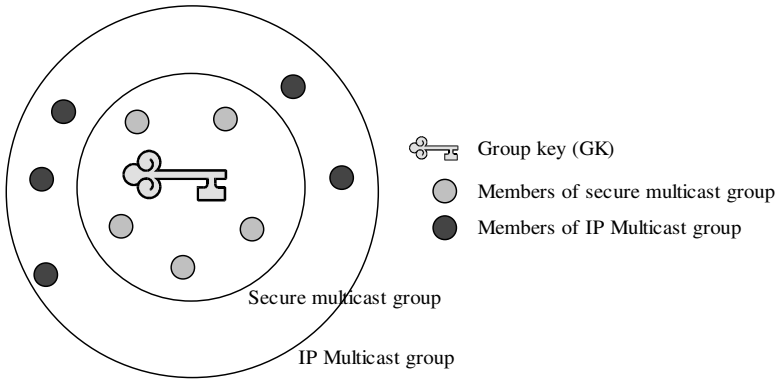


Figure 3-1: Secure multicast group

The receivers' membership to the secure multicast group is defined by whether or not they have the shared group key. Therefore, the management of the multicast group key is a critical factor for the security of the multicast group. Moreover, as the multicast group can involve possibly hundreds of thousands of receivers, the key management is essentially under tough scalability requirements.

3.2 Setting the problem

Consider the scenario in Figure 3-2 where a server pushes, in a multicast fashion, software updates to a group of users. Consider also that the service is offered on a paid subscription basis. Suppose that only two users have paid (i.e. h3 and h6), and therefore only these two hosts should be allowed to have access to the software updates. However, the IP Multicast does not restrict any other host to actually listen to the same multicast group where the updates are transmitted. For instance, in our example, user h5 has also joined the IP Multicast group and has access to the multicast traffic, although it is not an authorized user of the service. Therefore, in order to restrict the access only to authorized users, a group key is established among the authorized users and the sender. The server uses this key to encrypt the data, such that only the authorized receivers who have the group key can decrypt it. All the users who have the group key are members of the secure multicast group. Note that the unauthorized user h5 is not member of the secure multicast group, although she is member of the IP Multicast group and receives (encrypted) multicast traffic. However, as she does not have the group key she can not decrypt and access the actual content.

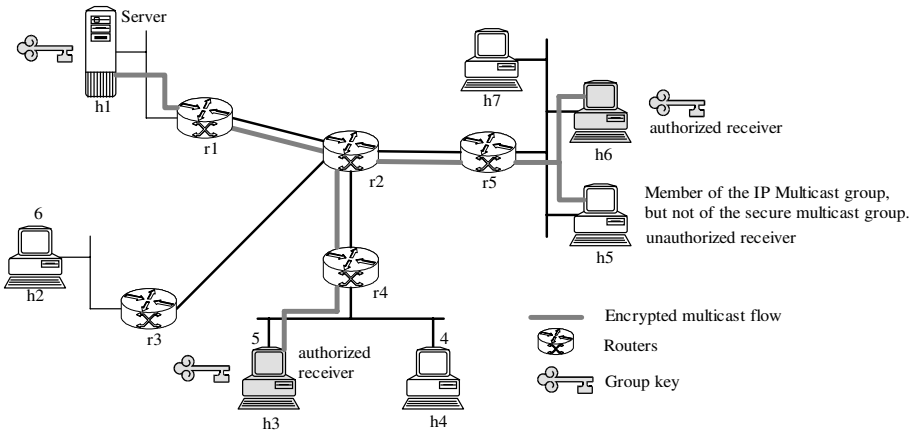


Figure 3-2: Secure multicast group scenario

The subscription could be valid only for a certain number of updates or only for a certain time. Thus, the existing members of the secure multicast group could possibly leave the group upon the expiration of their subscription. Also, new users could subscribe to the service and join the secure multicast group. As users join and leave the secure multicast group, the group key must be updated and distributed according to the current membership of the secure group.

Therefore, the secure group lifetime is marked by join and leave events, as seen in Figure 3-3. We call the time interval between two such events *session*. The group key (sometimes also called session key) does not necessarily change during a session, but it must change from one session to another, when the group membership changes.

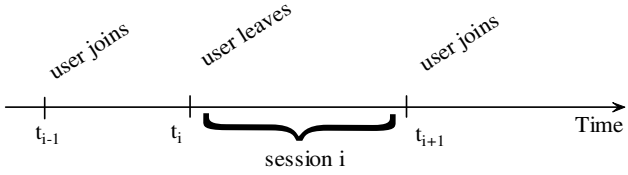


Figure 3-3: Multicast sessions

3.2.1 Group key management

As defined in [MOV97], the key management is the set of processes and mechanisms which support the establishment of a shared secret key and the maintenance of an ongoing keying relationship between parties, including replacing older keys with new keys as necessary.

There are two main problems associated with group key management:

- Group key agreement and establishment: This is the problem of providing a protocol whereby secure agreement can be reached among group members who need to select a mutual key. This problem is particularly important in collaborative scenarios, where the control of the group membership and the properties of the shared group key are under the joint authority of all members of the group.
- Group rekeying: This is the problem of the replacement of the current group key once it is deemed insecure. Rekeying is invoked if the set of group members has changed or if there is a danger that the group key has been leaked to the adversary. Rekeying can also occur periodically to refresh the group key, in order to limit the amount of data encrypted with the same cryptographic key. This problem is common to both push-oriented and collaborative scenarios, although the settings of the push-oriented scenario (involving large and dynamic groups) impose stronger requirements on the efficiency and scalability of the key management.

The focus in this thesis is the push-oriented scenarios. Therefore, we put our effort into the group rekeying problem of the key management, and let the group key agreement and establishment problems be part of our future work where we will focus more on collaborative and peer-to-peer settings.

Architecture

In a push-oriented scenario, the sending party is usually the owner of the multicast group and constitutes a central point of control. Therefore, we consider a logically centralized architecture for the group key management, where a dedicated party, the *group controller*, has total control on the group key and on the secure group membership. The group controller implements the *admission control policy* for the multicast channel. Also, the group controller is fully responsible for generating the appropriate group key that meets the required cryptographic parameters.

Note that the group controller can be either distinct from the sender (as shown in the Figure 3-4), or it can be part of the sender.

The group members, also called users, participate in the process of key management. They are required to store keys and also to locally compute keys.

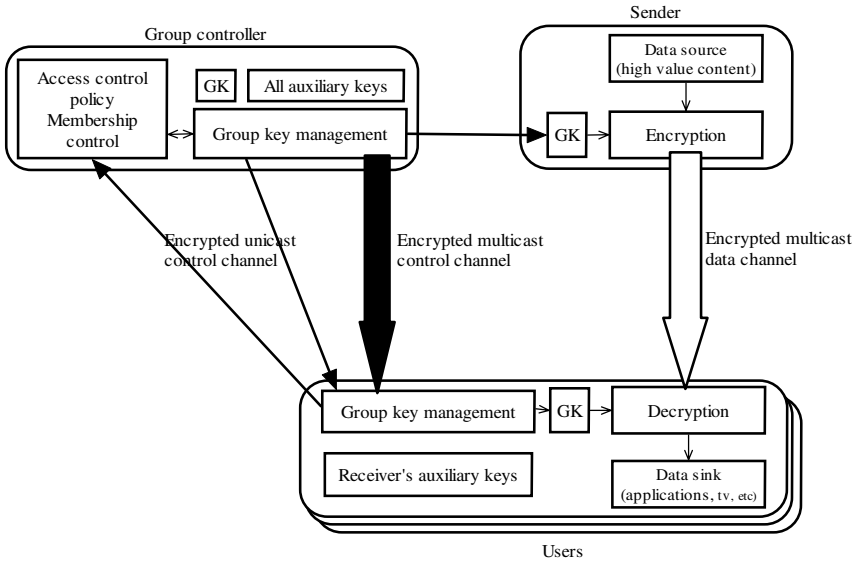


Figure 3-4: Group key management architecture

3.2.2 A security and efficiency matter

The group key management should assure that only authorized users have access to the multicast channel. Whenever the multicast membership changes, the shared group key must be replaced such that the new members can not access old traffic while excluded members can not access future traffic.

When a new member joins the group, the solution is fairly simple. First, a new group key is generated. Then, this key is securely communicated to the newly joined member, as well as to the existing members of the secure multicast group. The new group key is conveyed to the new member through a secure unicast channel. To the existing members, the key is first encrypted with the previous group key, and then it is multicasted to the whole group. Since the existing members were in the possession of the previous group key, they can decrypt the data and read the new group key.

When a member must leave the group, the new group key must be securely transmitted to all the remaining members with the exception of the excluded one. This is a more complicated situation since there is nothing to distinguish all the remaining users, as a whole group, from the one that must be excluded. A naive and simple solution would be for each user to share an individual key with the controller. In this case, the new group key can be individually conveyed to each remaining user by encrypting the new group key with each user's individual key, as shown in

Figure 3-5. However, the communication cost of such a solution is linearly increasing with the number of users in the multicast group. For instance, to exclude a member from a group of 10^4 members, such a scheme requires 9999 transmissions. This is unacceptable because it consumes a lot of network bandwidth and poses delays to the multicast communication.

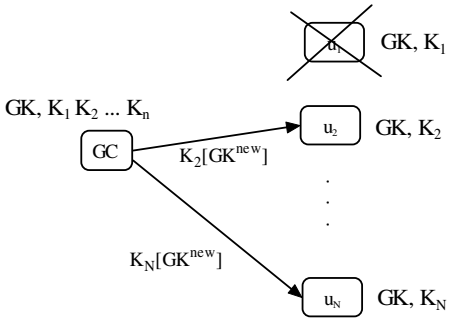


Figure 3-5: One key per user scheme: exclude user u_1

A complementary solution can be adopted such that the communication is only one message. The idea is to give each user in the multicast group the keys assigned to all the other users, but not her key. For instance, user u_1 will receive the keys K_2 to K_N but not K_1, as shown in Figure 3-6. When user u_1 is excluded, the controller will encrypt the new group key using K_1 and multicast the resulting encrypted message to the entire group. Since all the users in the group with the exception of u_1 are in the possession of K_1, the group is rekeyed.

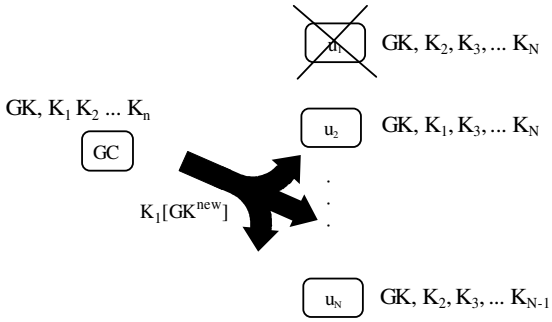


Figure 3-6: Complementary keys scheme: exclude user u_1

The communication cost of the complementary scheme is one message. Note that this low communication cost comes from the fact that the scheme is using (encrypted) multicast as the control channel. However, the key storage cost for each

user is linearly increasing with the number of members in the secure multicast group. Moreover, the scheme is susceptible to collusion.

Collusion is a security attack where two or more malicious users put together their keying material such that they can not be excluded from the secure group, compromising therefore the security of the scheme.

In the complementary scheme, the union of the keys held by any two users covers all the keys in the system. Therefore, any two users, maliciously working together, can decipher all the messages sent by the controller. Consider, for instance, the case where users u_1 and u_2, from Figure 3-6, collude by sharing their keys with each other. If now, user u_1 should be excluded, she will be able to decrypt the rekeying message as she is in possession of K_1 (borrowed from user u_2). Similarly it is for the case when user u_2 should be excluded. Therefore, by colluding, the two users become immune to group rekeying, thus compromising the security of the group key management scheme.

In general, there is a tight relationship between the key storage, rekeying communication cost, and security properties of group key management. Following, we describe in more detail the security and efficiency requirements for the group key management.

3.3 Security and efficiency requirements

The main goal of the group key management is to assure that only authorized users are in the possession of the group key. As the group is dynamic, the group key must be changed accordingly, and this must be done securely. The security requirements for the group management are:

- Forward secrecy (or forward access control): Impossibility of a former group member that has been excluded to gain access to future group keys and implicitly to future multicast group traffic.
- Backward secrecy (or backward access control): Impossibility of a newly joined group member to gain access to past group keys and implicitly to the past multicast traffic.
- Collusion resistance: Impossibility for two or more former group members, who have been excluded, to gain access to future group keys even if they collude and put their keying material together.

The collusion resistance can be regarded as varying within a range of values, i.e. a scheme can be said to be k-resistant iff it is resistant to collusion of at most k users.

By extension, the forward and backward secrecy could also be regarded as varying within a range of values, being actually defined as k-forward secrecy and k-backward secrecy (e.g. in [MS98]). That is, a k-forward secrecy scheme would provide forward secrecy if the number of excluded users is less or equal to k. In this

case, the set of three requirements stated above would reduce to only two, the forward and the backward secrecy, whereas the collusion would be an intrinsic component of these two. The goal would then be to provide perfect forward and perfect backward secrecy. However, as shown by the previous work in group key management, the perfect backward secrecy is not usually a problem since it is relatively easy to provide it, whereas the forward secrecy is a central problem. Therefore, we have chosen to decouple the k-forward secrecy into pure forward secrecy, referring to the problem of one user leaving the group, and the collusion aspect, referring to the case when at least two users are involved. In conclusion, we consider the backward and forward secrecy as having Boolean values, according to whether or not they are achieved, and the collusion resistance as having a range of values, according to the degree of collusion resistance obtained.

A scheme has perfect resistance to collusion when no coalition of users, of no matter what size, could compromise the security of the scheme. Also, a scheme has no resistance to collusion when any two users from the group can collude and become immune to group rekeying.

Since the size of the secure multicast group can be quite big, e.g. hundreds of thousands, the key management is also under tight efficiency constraints. The efficiency requirements are in terms of minimizing the network and computer resources consumed by the key management. Particularly, the resources of interest are:

- Network bandwidth: Minimize the amount of traffic required by the rekeying operation.
- Key storage: Minimize the key storage required by the key management.
- Computation resources: Minimize the computation required by the key management.

We measure the efficiency in terms of communication cost, key storage cost, and computation cost. The lower these costs are, the more efficient the scheme is. Throughout this thesis we pay particular attention to the communication and key storage cost. The communication cost will be measured in number of message units (encrypted keys) needed to be communicated in order to perform a certain management operation, such as group rekeying. The key storage cost is measured in number of keys required to be stored by each user and by the controller, respectively.

3.4 Efficient group key management

The broad and challenging problem of establishing and managing group keys touches upon a large body of previous work, and a detailed overview of the topic can be found in Chapter 7, the related work. Among this work, the logical key hierarchy and the flat key schemes form two main groups of proposals for scalable secure multicasting, which have the efficient group rekeying as their major goal.

The tree key schemes, or the logical key hierarchy (LKH) based schemes, were proposed in [WGL98] by Wong and colleagues, in [CWS+98] by Caronni and colleagues, and in [RFC2627] by Wallner and colleagues. They address the problem of minimizing the communication cost of the rekeying operation by using a key tree arrangement of keys. The idea is to recursively partition the multicast group into subgroups and to assign auxiliary symmetric keys to these subgroups. Whenever a user leaves the multicast group the controller uses these auxiliary keys to rekey the remaining users on a subgroup basis by using a minimum number of subgroups covering all non-excluded users.

The communication cost to exclude one user from a group of N users using LKH is 2*log(N)-1, while each group member is required to store log(N)+1 keys[3]. However, the key storage requirement for the group controller is still high, increasing linearly with the number of users, i.e. it is 2*N-1 keys.

Wong and colleagues generalize the key tree to a key graph [WGL98]. Following, Yang and Lam [YL00] show that, in a key graph, the lower bound complexity for rekeying communication, when the only allowed operation for the GC to distribute one key is to encrypt it by another key, is logarithmic with the group size. This is a property of LKH and its variants, the one-way function trees (OFT) [MS98] and the one-way function chain (OFC) [CGI+99]. These schemes provide perfect resistance to collusion.

Potential ways of further improving the efficiency of group rekeying are to extend the operations allowed for key distribution and to relax the security requirements [YL00]. For instance, more efficient results are achieved by reusing auxiliary keys among different groups of users and encrypting the group key with a combination of these keys. This way, ad-hoc groups can be created which usually cover more users than the groups pre-established by the key tree. The Boolean function minimization (BFM) [CEK99], by Chang and colleagues, and the Flat-VersaKey [WCS+99], by Waldvogel and colleagues, fall in this category called the flat or matrix based schemes.

The communication cost to exclude one user from a group of users using the flat scheme is log(N), as in [CEK99], and the storage requirement for a member of the group is log(N)+1. The big advantage of the flat scheme is that the controller key storage decreases to 2*log(N)+1. Moreover, as shown in [CEK99], these schemes improve the communication cost in the case of cumulative member exclusion. However, the flat scheme is susceptible to collusion attacks.

The security and efficiency parameters of the LKH and flat schemes are summarized in Table 3-1, and a more detailed description of the LKH and flat,

[3] Throughout this thesis the notation log(x) refers to the logarithm in base 2 of a given value x, i.e. $\log_2(x)$, if not specified otherwise.

including the rekeying algorithms, is given in section 7.3.1 and in section 7.3.2, respectively.

Table 3-1: Security and efficiency parameters

	LKH	Flat
Forward secrecy	Yes	Yes
Backward secrecy	Yes	Yes
Collusion resistance	Perfect	No resistance
Communication cost	2*log(N) – 1	log(N)
User key storage	log(N) + 1	log(N) + 1
Controller key storage	2*N – 1	2*log(N) + 1

In general, the resistance to collusion is achieved at the expense of efficiency, and the current key management schemes focus on extreme cases of this requirement – either perfect resistance to collusion is provided, or on the contrary no guarantees on collusion resistance are given. This is schematically presented in Figure 3-7.

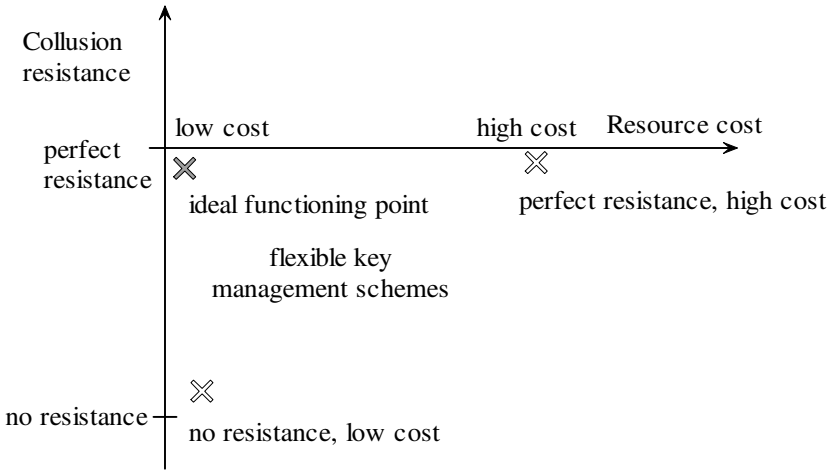


Figure 3-7: Security and efficiency space

However, applications may have certain assumptions regarding members and accessibility to the multicast channel. Based on this we argue that more flexible key management schemes should be designed to balance the given collusion constraints against efficiency, filling the gap between the two extreme positions taken by current key management mechanisms.

Following this idea, the next chapters introduce two flexible group key management schemes capable of best meeting the applications' security and efficiency requirements.

Chapter 4 generalizes the collusion resistance definition and then formalizes it based on the user accessibility to the multicast channel. Then, we introduce a flexible category-based collusion resistance group key management scheme that balances the security-efficiency tradeoffs for multicast applications such as software delivery where a user categorization can be done. Chapter 5 introduces an improvement mechanism that reduces the key storage per user as well as the key storage for the controller.

Finally, Chapter 6 presents our second scheme, the cluster-based group key management, which addresses the applications where the user categorization is not feasible.

Chapter 4

Category-based group key management

4.1 Introduction

In this chapter we generalize the collusion resistance definition and then formalize it based on the user accessibility to the multicast channel. We also show how the previous work can be reformulated as special cases of our definition. Further, we propose and evaluate a flexible key management strategy for the general case where the accessibility relation defines the order of exclusion of categories. The theoretical and experimental results show that our scheme has good performance regarding the communication cost and the key storage cost for the controller. In particular, for a group of N users, the communication cost to exclude a category is lower than or equal to 2*log(N)-1 and the number of keys at the controller is lower than or equal to 2*N-1.

4.2 Collusion resistance requirement

As a step towards a definition of collusion, we introduce the notion of categories. We say that users belonging to the same category have the same accessibility to the multicast group. A binary accessibility relation is defined on the set of categories, reflecting the fact that users from one category have access to at least all sessions that users of another category have access to, and maybe more. The collusion resistance requirement of a certain application can be defined as a specific configuration of the categories and of the accessibility relation.

We introduce the following notions and notation:

- U is the set of users and N is the total number of users.
- C is the set of *accessibility categories* and M is the total number of categories.
- The function *users*: $C \rightarrow \wp(U)$, where users(c) gives all the users of category c. We assume that a user can belong to only one category, therefore: $\forall\ c_i, c_j \in C: c_i \neq c_j \rightarrow users(c_i) \cap users(c_j) = \emptyset$. Users with the same accessibility belong to the same accessibility category. All users need to belong to a category, that is $\bigcup_{i=\overline{1,M}} users(c_i) = U$.
- $\prec \subseteq C \times C$, a reflexive partial order, is the *accessibility relation* on categories reflecting the relative access to the multicast group the users from one category have compared to users from another category. If $c_i \prec c_j$, then users from c_j can access at least all sessions that users from c_i have access to.
- $E \subseteq C$ the set of *excluded categories*.

With the above notation we define the collusion resistance requirement as follows:

> *No coalition of users from a subset I of excluded categories, $I \subseteq E$, can gain access to future group keys unless $\exists\ c_i \in C \setminus E$, $\exists\ c_j \in I: c_i \prec c_j$.*

In other words, a coalition of users from a subset I of excluded categories can gain access to future group keys only if there is a category in I that can access more than a not yet excluded category.

A scenario modeled by our definition of collusion is, for instance, the software delivery [HS00, Han01] where users can be categorized according to how much they buy. Some users could buy only the initial software product, while others could subscribe to a certain number of updates and future versions. This would correspond to a "buy more" total order, as observed in [Han01]. For example, consider the distribution of a software product and two following updates. Using our notation, this setting can be modeled by:

- the set of users U.
- the set of categories $C = \{c_{init}, c_{up1}, c_{up2}\}$, where the three categories correspond to the initial distribution and the two following updates, respectively.
- the accessibility relation $\prec = \{<c_{init}, c_{up1}>, <c_{init}, c_{up2}>, <c_{up1}, c_{up2}>\}$, denoting that the users in category c_{up2} have access to more sessions than the users from categories c_{init} and c_{up1}, as well as that the users in category c_{up1} have access to more sessions than the users from category c_{init}.
- the set E of excluded categories, where initially $E = \emptyset$.

After the initial software and the first update are distributed, the categories c_{init} and c_{up1} will be excluded. At this point $E = \{c_{init}, c_{up1}\}$, as the second update should only be distributed to users from category c_{up2}. The collusion resistance requirement states then that no combination of users from the excluded categories, i.e. c_{init} and c_{up1}, can gain access to the group keys for the second update.

However, software can split in different version branches (e.g. separating features) or merge different tools into one single software product. Also, users could subscribe for more than one product or more than one branch of the same product. Therefore, a more general model of this scenario is the partial order, as proposed above. We observe that this order also intuitively relates to the order in which users are excluded from the authorized group of receivers. Although our accessibility relation generalizes the "buy more" from [Han01], it is not limited only to that scenario, but it can model any situation where the exclusion order is known or can be inferred.

4.3 Discussion

According to our definition, the collusion resistance requirement for an application depends solely on the configuration of the category set C and on the accessibility relation $\prec$.

We discuss a number of possible configurations of C and $\prec$:

1. $C = \{c\}$ and $\prec = \{<c,c>\}$. All users are in one category. This case corresponds to applications that have no constraints with respect to collusion. Solutions for this case, such as BFM [CEK99] and Flat-VersKey [WCS+99], reuse auxiliary keys and achieve very good efficiency. Also, the linear ordering of receivers (LORE) [FJA02] has good performance with regard to communication, as the exclusion operation requires only two messages; however, LORE has no resistance to collusion.

2. $C = \{c_1, c_2, \ldots c_M\}$ and $\prec$ is a total order. Categories are defined. In this case, the controller knows exactly the order in which the categories will be excluded. The solution proposed in [Han01] addresses this case by using BFM with a special arrangement of users, such that users from "buy less" categories can not collude over users from "buy more" categories. Although the number of keys is kept low, the solution imposes too strict constraints regarding the maximum cardinality of each category. Another scheme, the zero side effect multicast key management using arbitrarily revealed key sequences (MARKS) [Bri99], assumes also that applications, such as pay TV, can plan the user exclusion. When joining a multicast session, a new user should exchange only one short set-up message with the key center. Moreover, when excluding users, no additional messaging is needed as the exclusion has been planned already while joining. Thus, the scheme is said

to have zero side effects on other receivers when a single receiver joins or leave a session.

3. $C = \{c_1, c_2, \ldots c_M\}$ and $\prec$ is a reflexive partial order. This is the general case. The controller knows that some order exists among some of the categories, where other categories are incomparable. Solutions for this case do not exist in previous the work.

4. $C = \{c_1, c_2, \ldots c_M\}$ and $\prec = \{<c_i, c_i>\}_{i=1, M}$. In this case, the controller cannot make any assumption regarding the order the categories are going to be excluded. Leaving the multicast group happens in an unpredictable manner. No prior knowledge is available about how much access one category has compared to another. A special scheme addressing this case is the cluster-based group key management presented in Chapter 6, but with the constraint that all categories have the same cardinality. A similar setting is also considered in the hybrid structuring of receivers (HySOR) [FJA02]. This scheme is capable of trading the communication cost versus resistance to collusion by combining the LKH and the LORE.

5. $C = \{c_1, c_2, \ldots c_M\}$, $M = N$, $users(c_i) = \{u_i\}$ and $\prec = \{<c_i, c_i>\}_{i=1, M}$. This is a special configuration of the previous case where each category directly maps to a user. Again, leaving the multicast group happens in an unpredictable manner. The collusion resistance requirement can be rewritten as *no coalition of excluded users can get access to future group keys*. This is the most common requirement in the existing work for securing multicast, e.g. [RFC2094, RFC2627, WGL98, WCS+99]. Although solutions for this case provide perfect resistance to collusion, this comes at an expense on efficiency.

In the next section, we propose a key management strategy for the general case of collusion, which takes into consideration the existence of different categories of users. We show that our solution is more efficient, in terms of transmissions, than the solutions for case 5, which has been the premise for most of the work in securing multicast communication.

4.4 Category-based group key management

We consider a centralized architecture, where one group controller manages the users' access to the group communication channel. The controller initiates all the sessions, generates and distributes the keys, maintains information about the categories, and accordingly adds and removes users and categories to and from the multicast group. Although the central controller can be a potential performance bottleneck, this architecture is more desirable in systems where one party must retain the control such as in multicast software delivery [HS00, Han01]. Moreover, methods of replication can be used to unload the controller.

We assume that categories are excluded based on the accessibility relation and we formulate the requirement that is used in the design of our solution:

- $\forall c_i, c_j \in C : c_j \in E \wedge c_i \prec c_j \rightarrow c_i \in E$;
- No coalition of users from excluded categories can gain access to future group keys.

This requirement implies the collusion resistance requirement but maps better to operations performed by the controller, and it is therefore more helpful when approaching the solution. In particular, the $\prec$ relation can be mapped to the knowledge the controller has regarding the order in which the categories must be excluded. Our new problem is then to find a mechanism to exclude a category when the order of exclusion is known, where the exclusion must have perfect resistance to collusion.

4.4.1 Category accessibility graph

The category accessibility graph (CAG) is a directed acyclic and rooted graph corresponding to the accessibility relation (partial order) $\prec$. The CAG is rooted in the sense that there is a node in the graph that has no incoming arcs.

In CAG the nodes stand for categories and an arc $c_i \leftarrow c_j$ corresponds to the relation $c_i \prec c_j$ between the two categories. In the remainder we talk about categories and nodes in CAG interchangeably.

We note that the partial order $\prec$ is, in general, without a greatest element (refer to [KBR97] for definitions); therefore, it is also without a root. If this is the case, then we add a new category, namely the *rc* (root category), in the set of categories and also add the needed relations such that $\prec$ becomes a reflexive partial order with the greatest element rc.

The CAG is constructed in a similar manner as the Hasse diagram [KBR97] corresponding to the reflexive partial order $\prec$. Thus, the CAG is obtained from the diagraph of relation $\prec$ by eliminating all one length cycles, implied by the reflexive property, and all the arcs implied by the transitive property. The result is a directed, intransitive, acyclic, and non-reflexive graph, as shown in Figure 4-1. Since $\prec$ has a greatest element in rc, it can be shown that CAG has exactly one node, the rc, with in-degree zero. Moreover, it follows from the properties of $\prec$ and the construction of CAG that for any node in CAG, different than rc, there is at least one directed path from rc to that node. That is, CAG is a rooted diagraph with root in rc.

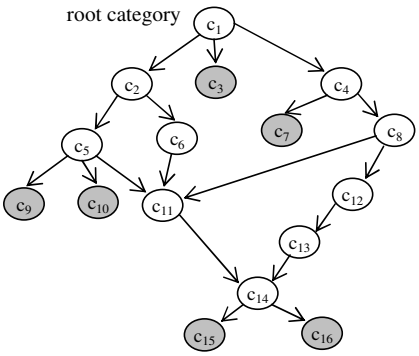


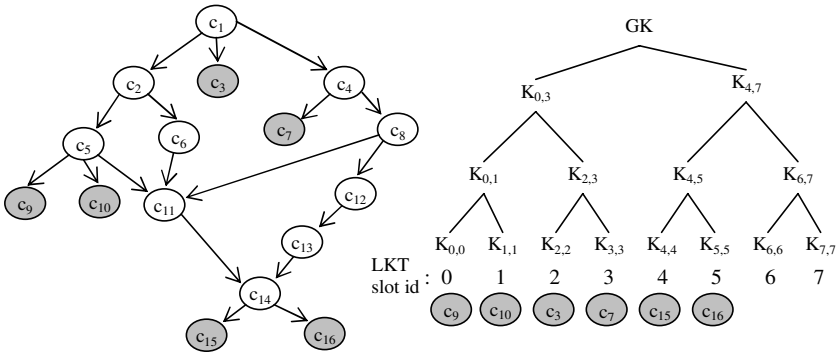
Figure 4-1: Category accessibility graph (CAG)

4.4.2 Key assignment

We start from the following observation. If $c_i \prec c_j$ (or $c_i \leftarrow c_j$ in CAG) then whenever node c_i has to be rekeyed node c_j will also need to be rekeyed. Therefore, we introduce the *propagation principle: if $c_i \prec c_j$ then give node c_j the keys that are used for rekeying c_i.* This will assure that whenever c_i is rekeyed c_j is also rekeyed, without the need to specifically rekey c_j and its ancestors.

However, this will not solve the case when category c_i must leave the group while c_j, together with its ancestors, remain in the group. Therefore, *if $c_i \prec c_j$ then c_j must have at least one key that is not known to c_i.* This key will be used to rekey c_j when c_i leaves. We apply again the first principle for the ancestors of c_j, so that the key of c_j is given to its ancestors too.

We also make the observation that at any time any of the leaf nodes can unpredictably be excluded, independent of the exact branches where they are situated. Therefore, for the leaf nodes we use a perfect resistance to collusion scheme. In particular, we use the LKH key arrangement, and construct the key tree corresponding to the leaf nodes in CAG, naming it the leaf key tree (LKT). A key $K_{a,b}$ from LKT denotes that the key is shared by all nodes with the slot identifier id such that $a \leq id \leq b$. The root of the LKT is the group key (GK).



a) Category accessibility graph (CAG) b) Leaf Key Tree (LKT)

Figure 4-2: Key assignment

Figure 4-2 shows the CAG from our previous example and its corresponding LKT. In this case, node c_9 will be given the group key GK and the auxiliary keys $K_{0,3}$ $K_{0,1}$ and $K_{0,0}$. Following the propagation principle, these keys will also be given to categories c_5, c_2, and c_1 in CAG.

In conclusion the key assignment is as follows:

- Each leaf node c_i in the CAG receives the group key GK and a set of auxiliary keys from LKT, corresponding to the node's slot identifier:

$$(Eq.4\text{-}1)\quad keys_{LKT}(c_i) = GK \bigcup \left\{ K_{a,b} \,\middle|\, a \le id \le b \wedge id = LKT_{slotid}(c_i) \right\}$$

- Each interior node c_j has a key K_j. Further if $c_i \prec c_j$ then all the keys of c_i are given also to c_j.

In general the key assignment is:

$$(Eq.4\text{-}2)\quad keys(c_j) = \begin{cases} K_j \bigcup\limits_{c_i \in descendants_{CAG}(c_j)} keys(c_i), if\ c_j \notin leafs_{CAG} \\ keys_{LKT}(c_j), if\ c_j \in leafs_{CAG} \end{cases}$$

where $leafs_{CAG}$ is the set of leaf nodes in CAG and $keys_{LKT}$ is as defined above.

4.4.3 Group rekeying

The rekeying algorithm to exclude a leaf node c_i from CAG is:

Step 1. Exclude node c_i from the LKT following the LKH rekeying algorithm, see section 7.3.1. This assures that all leafs, except for the excluded leaf will get the new group key GK^{new}. When removing node c_9, for instance, nodes c_{10}, c_3, c_7, c_{15} and c_{16} will be rekeyed. Since their ancestors know the keys of these nodes, the ancestors will also be able to decrypt these messages and learn the new auxiliary keys as well as the new GK^{new}.

Step 2. Further two cases are distinguished:

Step 2.1. No new nodes become leafs in CAG. In this case, all the remaining nodes have been rekeyed. Go to step 3.

Step 2.2. New nodes become leafs in CAG. Since these nodes have no other descendants than the one that is excluded, they cannot be rekeyed. In Figure 4-2.a, when removing node c_{14} (assuming that nodes c_{15} and c_{16} were removed earlier) nodes c_{11} and c_{13} will become leafs. In this case, each new leaf in CAG must be included in the LKT and rekeyed. Two sub-cases follow:

Step 2.2.1. The LKT has enough free slots (unused key leafs) to accommodate the new leafs. Go to step 2.3.

Step 2.2.2. The LKT doesn't have enough free slots. In this case, LKT must be expanded. LKT grows with one or more levels such that it accommodates all the newcomers. To grow the LKT with one level, for instance, the controller generates a new root key GK^{new} for the LKT. The old LKT will be the left sub-tree starting from the new GK^{new}. That is, the GK becomes the $K_{0,2^{d-1}}$, where d is the depth of the initial LKT. The GK^{new} is conveyed to the old members of the tree by encrypting it with the old GK and multicasting to the group. Within the same message the old members are informed that the tree increased with one level and the indexes of their keys changed, such that $K_{2^{d-i}*j,2^{d-i}*(j+1)-1}$ becomes $K_{2^{d-i+1}*j,2^{d-i+1}*(j+1)-1}$. The whole new right sub-tree of the LKT will then be available to newcomers. Observe that LKT can grow more than one level at once making this process more efficient. Go to step 2.3.

Step 2.3. Rekey and include each new leaf c_j into the LKT by sending the new group key GK^{new}, as well as the auxiliary keys associated with LKT, encrypted with its key K_j. Category c_j and all its ancestors will understand this message so the key propagation is re-established.

Step 3. The algorithm ends.

The rekeying communication cost is:

(Eq.4-3) $2 * log(|leafs_{CAG}|) - 1 + newleafs\ (sn) + 1$

The *newleafs* is a function, denoting the number of new leafs in CAG at session number sn. The notation |S| denotes the cardinality (or the size) of a given set S. The last term "1" comes from the worst case assumption that any application of the exclusion algorithm will require expansion of the LKT. Obviously, the formula reduces to:

(Eq.4-4) $2 * log(|leafs_{CAG}|) + newleafs\ (sn)$

The number of keys per user varies from $log(|leafs_{CAG}|) + 1$ for the leafs in the CAG to $M + |leafs_{CAG}| - 1$ for the root of CAG. The total number of keys is $M + |leafs_{CAG}| - 1$ which is also the number of keys at the controller. The solution has good performance for communication, but the number of keys per user, as well as the number of keys at the controller, increases due to the propagation of keys.

For a given number of categories M, the number of transmissions and the number of keys at the controller depends on the number of leafs in CAG. Fewer leafs result in better numbers. On the other hand, the number of keys per user depends not only on the number of nodes M and the number of leafs, i.e. $|leafs_{CAG}|$, but also on the mapping between nodes from $leafs_{CAG}$ and the slots in LKT. According to (Eq.4-2) , the set of keys depends on the union of keys propagating from its descendants, and it has therefore fewer elements in the case where the propagated keys are the same.

4.5 Evaluation

We analytically compare the efficiency of our solution strategy with our reference schemes, namely the LKH and flat, and then present experimental results obtained through simulation.

4.5.1 Comparison

The comparison is given in Table 4-1. We observe that the performance of the exclusion algorithm is dependent on the term newleafs(sn), see (Eq.4-4) . We note that, for a given number of categories M, each category will sooner or later become a leaf in CAG. Also, there will be at least as many sessions as M. Therefore, on average newleafs(sn) ≤ 1. We also note that $|leafs_{CAG}| \leq M$. Moreover, M = N denotes the worst case where every category has exactly one user.

Table 4-1: Comparison with the reference schemes

	LKH	Flat	Category-based GKM
Forward secrecy	Yes	Yes	Yes
Backward secrecy	Yes	Yes	Yes
Collusion resistance	Perfect	No resistance	Collusion resistance is configurable using CAG
Communication cost	2*log(N) – 1	log(N)	2*log($\|leafs_{CAG}\|$) + newleafs(sn)
User key storage	log(N) + 1	log(N) + 1	from log($\|leafs_{CAG}\|$) + 1 to $\|leafs_{CAG}\|$ + M –1
Controller key storage	2*N – 1	2*log(N) + 1	$\|leafs_{CAG}\|$ + M – 1

4.5.2 Experimental results

We evaluate the performance of our proposed solution by simulating different configurations of the CAG in the case where the CAG is a tree. We motivate choosing a tree for CAG by the following facts: the tree is a worst case with respect to memory need; further, it is also a worst case with respect to the number of keys per user; finally, there already exist random tree generators that have been used in benchmarking multicast protocols.

A specific configuration of the CAG models a certain collusion resistance requirement. We randomly generate such configurations by using a random tree generator with three degrees of freedom - the maximum number of nodes, the maximum number of leafs, and the maximum branching factor.

The tree generator is inspired from the random tree generator proposed and used for benchmarking multicast protocols in [RM99]. However, we have modified the original algorithm such that the generator guarantees that the generated tree has exactly the number of nodes and the number of leafs that has been specified as input to the algorithm. This makes our results more accurate.

In the experiments, the number of categories (nodes in CAG) has been kept to 10000, and the number of leafs has been varied from 100 to 7000. Each combination of nodes and leafs has been repeated with a maximum branching factor of 4, 6, 8, 10, and 16.

As shown by the experiments and as expected from (Eq.4-4) , the number of transmissions depends solely on the number of leafs, see Figure 4-3. Fewer leafs result in fewer transmissions, more leafs results in more transmissions. Since the number of leafs is related to the degree of collusion, this result confirms our hypothesis – the more constraints are put on collusion the more communication is needed. Our solution gives the possibility to balance the collusion resistance

requirement and the number of transmissions. The number of keys at the controller, Figure 4-4, is less than in the LKH, but it increases with the number of leafs in the CAG. In fact, the number of transmissions and the number of keys at the controller from the LKH are upper bounds for the number of transmissions and the number of keys at the controller in our scheme.

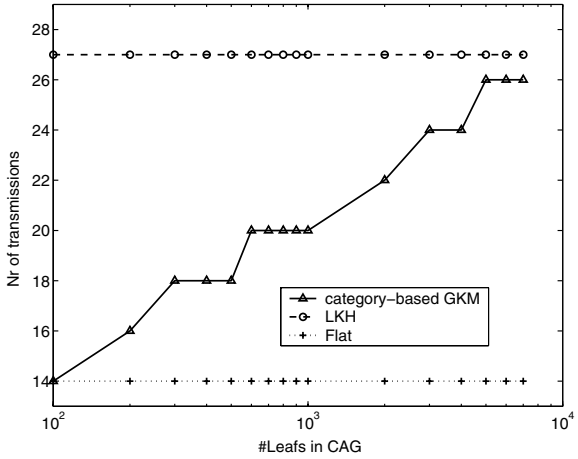


Figure 4-3: Category-based GKM: communication cost

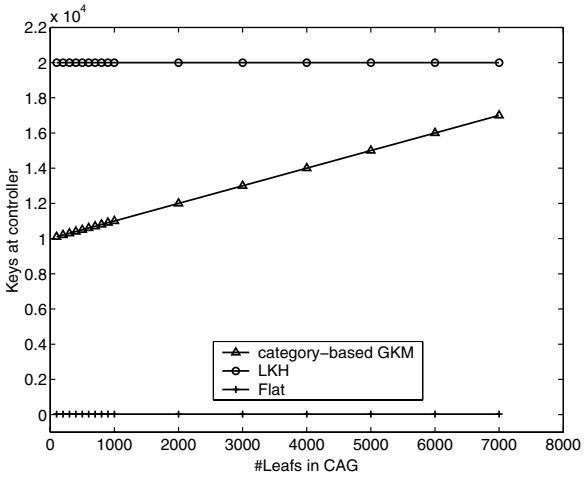


Figure 4-4: Category-based GKM: controller key storage

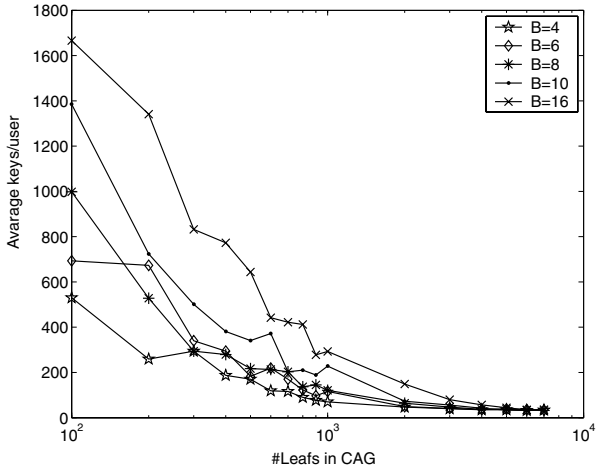


Figure 4-5: Category-based GKM: average user key storage

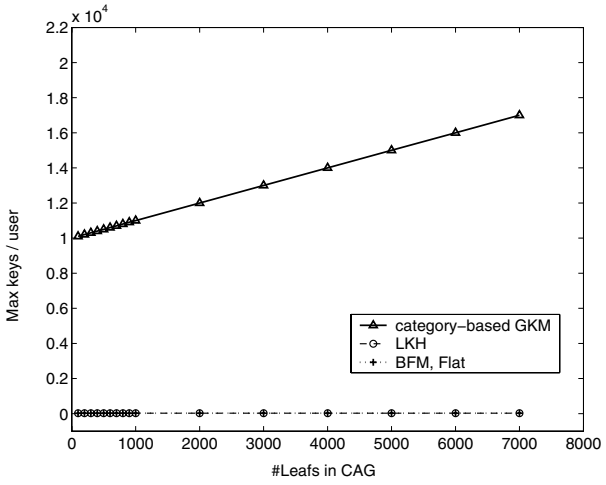


Figure 4-6: Category-based GKM: maximum user key storage

Since the keys from child nodes are given to their parents, the fewer leafs we have the more keys per user we get, as shown in Figure 4-5 (on average) and Figure 4-6 (maximum). Note that we have computed the average number of keys per user in the conditions of only one user per category. If there are more than one user per category the distribution of keys per user can change very much and can greatly influence the average. If categories closer to the root have many users, the average

would become high. On the contrary, if most of the users are concentrated in categories closer to the leafs, the average goes down.

4.6 Discussion and conclusions

This chapter adds the collusion constraints as an explicit requirement for designing group key management systems. We generalize the collusion resistance definition and then formalize it based on the user accessibility to the multicast channel. The collusion resistance requirement for an application can be specified in terms of categories of users and relations among these categories in the form of category accessibility graph (CAG). Further, we proposed a category-based group key management scheme for the general case of collusion resistance requirement. A scalability analysis is given in terms of rekeying communication cost (the number of transmitted keys) and the storage requirement (number of keys). We find that there is a tight dependency between the exact settings of the collusion resistance requirement shown in the form of CAG and the performance of the solution. In general, the solution has good performance for the number of transmissions needed to rekey in case a category must be excluded. However, the number of keys per user may become large due to the propagation principle.

To concentrate on the problem of user categorization, we have discussed in this chapter only the management (inclusion and exclusion) of categories, but not the joins and leaves of individual users. We note that for case 5 from section 4.2 (one user per category) our solution actually deals with users. For the other cases it is possible to use a two level management approach to deal with both users and categories at the same time. Such an approach is adopted in the cluster-based group key management, presented in Chapter 6.

In the next chapter, we propose an improvement mechanism to the category-based key management that alleviates the explosion of keys due to the propagation principle. The mechanism is based on hash dependencies between keys in CAG.

Chapter 5
Spanning hash key tree

5.1 Introduction

The category-based group key management (category-based GKM) proposed in Chapter 4 suffers from the fact that it increases the storage requirement per user. This is the effect of the key propagation in the CAG. In order to alleviate this problem we devise a technique based on a *spanning hash key tree* (SKT) to reduce the propagation and consequently to reduce the storage requirement for users. The SKT adds hash dependencies among the keys assigned to different categories, such that the keys of child categories in the category accessibility graph (CAG) can be derived from their parent keys using a one-way hash function. This way the parent categories should not store all the keys of their child categories, but only a subset of them. The exact keys that need to be stored by a given node are computed using the *maximals* function. In the worst case, the keys that have to be stored in the improved scheme are the same as the keys in the initial category-based GKM. However, the experimental results show that, in general, the improved scheme greatly reduces the key storage for the users and for the controller.

5.2 Spanning hash key tree

We define the spanning hash key tree (SKT) as:

> *The spanning hash key tree (SKT) is the directed shortest paths spanning tree in CAG rooted at rc.*

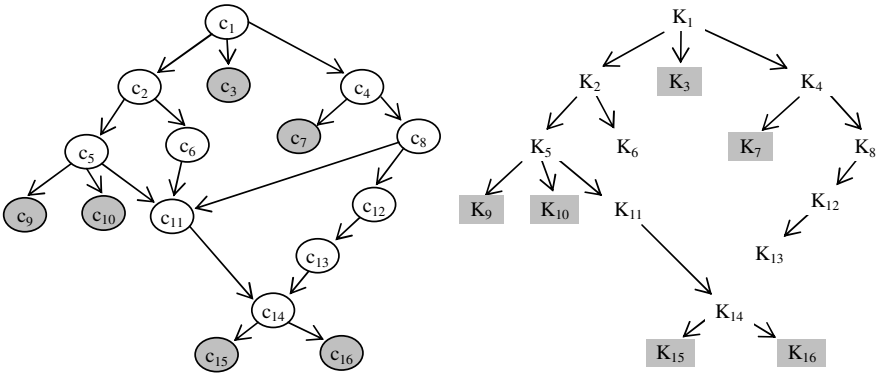
The existence of the spanning tree in CAG is guaranteed by the following theorem:

> *A finite directed rooted graph G with root r has a spanning tree with the root in r [Wan80].*

As CAG is a directed graph rooted at rc, and following the theorem from above, we conclude that the CAG has at least one spanning tree rooted at rc. Among all the possible spanning trees of CAG rooted at rc the SKT has the property of being the *shortest paths spanning tree rooted at rc*. That is, the distance of the paths from the root node rc to each of the nodes in the tree is minimal, where we define the distance of the path to be given by the number of arcs traversed by that path.

We note that efficient algorithms exist [BH89], such as Breadth-First Search algorithm and the Dijkstra's algorithm [Dij59], which find the shortest paths tree in a graph.

A spanning tree of a graph has the same set of nodes as the initial graph. That is, the SKT would have categories as nodes. However, we associate to each category c_i in the spanning tree a unique key K_i and we actually use these keys as nodes in the SKT. In other words, the SKT is obtained by re-labeling the spanning tree corresponding to CAG. Note that re-labeling of a graph or tree does not change in any way its properties, because the re-labeling is an isomorphic operation. Figure 5-1 shows the CAG from Figure 4-1 along with its corresponding SKT.



a) Category accessibility graph (CAG) b) Spanning Hash Key Tree (SKT)

Figure 5-1: CAG and its corresponding spanning hash key tree (SKT)

We use the notation SKT.nodes and SKT.arcs to refer to arcs and nodes from SKT. Also, we use the Boolean function *path* denoting whether or not a directed path exists between two nodes in SKT.

$$path : SKT.nodes \times SKT.nodes \rightarrow \{true,\ false\}$$

(Eq.5-1)

$$path(K_j, K_i) = \begin{cases} true, & if\ a\ path\ exists\ from\ K_j\ to\ K_i; \\ false, & otherwise. \end{cases}$$

We introduce a hash dependency between the keys in SKT, such that the following invariant holds:

(Eq.5-2)

$$\forall K_i, K_j: K_i, K_j \in SKT.nodes \wedge (K_j, K_i) \in SKT.arcs \rightarrow K_i = h\,(K_j /\!/ i)$$

The function *h* is a strong one-way hash function with the following properties:

- Function h maps an input x of arbitrary finite bit length, to an output h(x) of fixed bit length.
- For any given x in the domain of h, it is easy to compute h(x).
- For any given y in the range of h, it is computationally infeasible to find x such that h(x) = y.
- For any given x in the domain of h, it is computationally infeasible to find x' such that h(x') = h(x).
- It is computationally infeasible to find any pair x and x' such that h(x) = h(x').

For an exhaustive discussion of a one-way hash function's properties we direct the reader to [MOV97].

The notation $h(K_j \parallel i)$ denotes that key K_j is concatenated with the index i, corresponding to one of its child nodes, and then the one-way hash function is applied to the resulting concatenation.

Following the invariant from (Eq.5-2) , an internal key in the SKT is obtained by hashing the concatenation of its parent key with the index of the node. For instance, the key tree in Figure 5-1 is obtained, starting from K_1, by applying the one-way hash function h as follows: $K_2=h(K_1 \parallel 2)$, $K_3=h(K_1 \parallel 3)$, $K_4=h(K_1 \parallel 4)$; $K_5=h(K_2 \parallel 5)$, $K_6=h(K_2 \parallel 6)$; $K_9=h(K_5 \parallel 9)$, etc.

5.3 Key assignment

We define the function *keys* that maps a category c_i from CAG to a corresponding set of keys. Since the keys given to categories come from the two disjunctive structures of keys, the LKT and the SKT, we define the function *keys* by defining its two components $keys_{LKT}$ and $keys_{SKT}$.

$$keys(c_j) = keys_{LKT}(c_j) \cup keys_{SKT}(c_j) \quad \textit{(Eq.5-3)}$$

(Eq.5-4)

$$keys_{LKT}(c_j) = \begin{cases} GK \cup \left\{ K_{a,b} \mid a \le id \le b \wedge id = LKT_{slotid}(c_j) \right\}, if\ c_j \in leafs_{CAG} \\ \bigcup_{c_i \in descendants_{CAG}(c_j)} keys_{LKT}(c_i), otherwise \end{cases}$$

(Eq.5-5)

$$keys_{SKT}(c_j) = \begin{cases} \emptyset, if\ c_j \in leafs_{CAG} \\ maximals(K_j \bigcup_{c_i \in descendants_{CAG}(c_j)} keys_{SKT}(c_i)), otherwise \end{cases}$$

The function *maximals* gives the set of maximal keys of a given set of keys, as defined by the order in SKT:

(Eq.5-6)
maximals: $\wp(SKT.nodes) \rightarrow \wp(SKT.nodes)$

$$maximals(Ks) = \{K_j \in Ks \mid \forall K_i: K_i \in Ks \wedge K_i \neq K_j \rightarrow \neg path(K_i, K_j)\}.$$

The $keys_{SKT}$ follows the same principle of propagating up the keys from children to their parents as in previous chapter. Still, we reduce the effect of the propagation by exploiting the hashing dependencies between the keys in SKT. That is, not all the keys are propagated up, but only the keys that cannot be derived following a path in SKT. These are the maximal keys (given by the maximals function) of the propagated set of keys.

For example, we show what keys are given to category c_8 by the $keys_{SKT}$. The descendants of node c_8 are nodes c_{11}, c_{12}, c_{13}, c_{14}, c_{15}, and c_{16}, as in Figure 5-1. The propagation principle says that the keys assigned to them be propagated up to c_8, such that category c_8 will get its assigned key K_8 and the propagated keys K_{11}, K_{12} K_{13}, K_{14}, K_{15}, and K_{16}. This is a total of 7 keys. Applying the maximals function to these 7 keys we obtain only two keys K_8 and K_{11}. Any of the other remaining 5 keys can be derived, through iterative hashing, from these two maximal keys. Therefore, the SKT and the maximals function decrease the number of keys that must be propagated, the worst case being when the application of maximals does not result in any minimization. Therefore, the use of SKT will, at worst, result in the same storage requirements for the keys as in the basic scheme presented in Chapter 4.

Note that $K_i \in keys_{SKT}(c_i)$ iff $c_i \notin leafs_{CAG}$. The leaf keys in SKT are not given to their correspondent categories. These keys are actually not used at all. They appear in the scheme only to keep the one-to-one node correspondence between CAG and SKT.

The reason for using SKT is to decrease the keys per user and keys at the controller. This is achieved from the fact that a user from a category c_i should not store all the keys of its ancestors but only the maximal keys of the union of their

descendants. From this set of maximal keys a user can derive any of the keys of its descendants by hashing (maybe several times) one of its keys. The number of times a key must be hashed to get the result is equal to the length of the path between the maximal key of c_i and the key that must be calculated in the spanning hash key tree. This motivates our choice for SKT to be the shortest paths spanning tree (and not any spanning tree), to reduce, in general, the number of hash iterations.

For instance, we saw that the category c_8 stores only the key K_8 and K_{11}. Now supposing that c_8 wants to decrypt a message encrypted with a key corresponding to a lower category, say c_{13}. Although category c_8 does not have the key K_{13}, used for the encryption, she can still derive it from her key K_8, as follows: $K_{12}=h(K_8\|12)$, and $K_{13}=h(K_{12}\|13)$. That is, c_8 finds the correct key in two hashing iterations. Note that although category c_8 can compute the keys of its descendants, the contrary is not true. This is guaranteed by the one-way properties of function h.

5.4 Evaluation

Similar to the initial category-based GKM from previous chapter we calculate the number of keys per user and number of keys at the controller for the case when CAG is a tree. We compare the results obtained using the spanning hash key tree with our initial category-based GKM scheme, as well as with the flat and the LKH schemes. The SKT gives a big improvement both for the user key storage, Figure 5-2, as well as for the controller key storage, Figure 5-3.

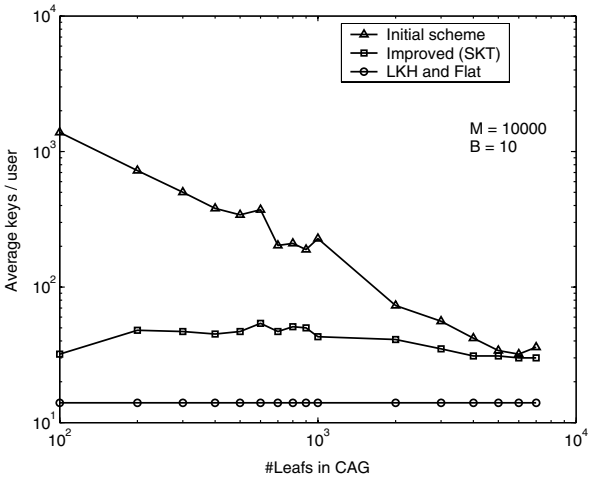


Figure 5-2: SKT: user key storage

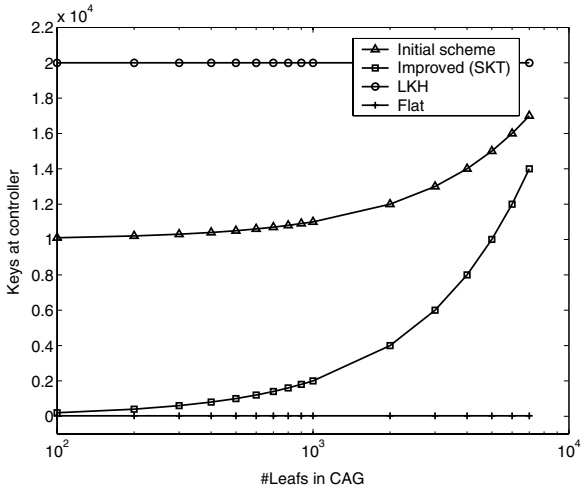


Figure 5-3: SKT: controller key storage

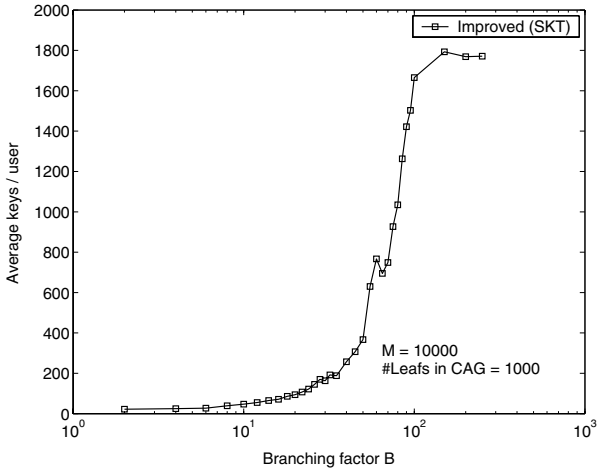


Figure 5-4: Keys per user function of the branching factor

Still, the number of keys per user is quite high when the maximum branching factor increases as depicted in Figure 5-4. For M = 10000 and $|leafs_{CAG}|$ = 1000 the keys per user increase starting from a branching factor of B = 32. For B in the interval [2; 32] the number of keys per user is less than 190.

5.5 Cryptographic primitives

For symmetric encryption, we choose Two-key Triple DES [DES] with key length 112 bit[4]. The function h can be based on cryptographic hash function such as MD5 [RFC1320] or SHA-1[SHA-1]. The SHA-1, for instance, generates a 160 bit length output and processes the input in blocks of 512 bit. The input to the hash function is obtained by concatenating the Triple DES key and the index (or address) of the category. Considering a 32 bit space for addressing categories, giving 2^{32} unique addresses, the input to the hash function will be 144 bit length. The SHA-1 appends to the input the padding bits and the length. The output of the SHA-1 is 160 bit while the Triple DES key length is 112 bit. In this case we discard some of the hash output. The SHA-1 algorithm has the property that every bit of the hash code is a function of every bit of the input. Even more it is unlikely that two different inputs, even exhibiting similar regularities, will have the same hash. That's why hashing the same key, but with different category addresses gives totally different keys, that cannot be correlated. A user, knowing a certain key derived from K, can not learn any other key that is derived from the same key K. For SHA-1, the difficulty of finding the input corresponding to a given hashed output is on the order of 2^{160} operations, while the difficulty of coming up with two messages having the same hash is on the order of 2^{80} operations [MOV97].

A key situated towards the leafs of SKT is obtained by multiple (possible many) iterations of the function h. Since the output is in a finite space of 160 bit, concerns could arise regarding the collision of the h function. This would result in a lower and a higher category, with regard to CAG, being assigned the same hash key, hindering the security of the system. Generically, in a chain x_0, x_1, x_2,...x_i,... x_l with $x_{k+1} = h(x_k)$, where function h: $\{0, 1\}^m \rightarrow \{0, 1\}^m$, it could be possible that there exist distinct indices i and j, such that $x_i = x_j$ [MOV97]. For this to happen with a negligible probability, the length l of the chain must be reasonable short compared to 2^m. For instance, the maximum iteration count for hashing in the S/KEY [RFC1760] is suggested to be something in range of 500-1000. Therefore, we consider 1000 being a safe depth limit of the SKT tree.

5.6 Computation requirements

The CAG is maintained only by the multicast group controller. Based on this structure the controller computes the LKT and the SKT, and distributes the keys to users. LKT is fairly easy to determine if a list of all leafs in CAG is maintained. The time complexity of Dijkstra's algorithm [Dij59] to determine the SKT (the shortest paths spanning tree in a graph) is $O(M^2)$. Faster implementations exist that can run in shorter time, such as implementations based on heap [AMO+90] and topological

[4] We note that Triple DES (112 bit key length) and SHA-1 are provided by the Java Cryptography Extension (JCE) [JCE] included in the J2SDK version 1.4.

ordering [Dia69]. Note that in a diagraph, as CAG is, a topological order on the nodes can always be given. In these conditions the shortest paths tree can be computed more quickly, in linear time.

In addition to computing the SKT, the controller spends computation on generating all the keys corresponding to nodes in SKT by iterative hashing of the root key. The controller computes also the maximal keys for each category.

The user needs only to compute the keys of the categories descending from her category. Depending on the length of the path from her key to the needed key the computation effort varies. Moreover, each user will need a way to determine if she can derive a key from its own set of keys. We discuss below how this aspect influences the user storage requirement.

5.7 Key addressing

When a message is encrypted with a key K_i from SKT and then multicasted, all the receivers who have that key, or can derive that key, should be able to decrypt the message. A problem is how a user can determine whether or not she can compute the K from the set of keys she holds. Second, how can the user determine which is the exact key, say K_j, to be used for computing K_i and how can the user determine the path (the correct indices to apply to the hash function) from the key K_j to key K_i.

One solution would be to actually embed all ancestor keys' indices of K_i into the multicasted message. In such a way each user receiving the message can easily see if she has a key K_j from which the encryption key K_i can be derived. However, this would significantly increase the length of the multicasted messages. In general, if the path from the root key in SKT to K_i is d, then the overload would be $32 \times d$ bits, where d can be any number between 0 and $M - 1$. For $M = 10000$ categories and a very long path the overload can hit 9KB per message. The amount of addressing information would swamp the amount of data actually carried in the message payload.

Another solution could be to put the burden on the user to store enough information to be able to determine if she can derive an encryption key, and if so, how to do it. In this situation each user needs to keep a forest of structure trees from SKT. Note that the user should not store the actual keys from the SKT nodes, but rather their indices and their relationships. The tree structures are sub-trees from SKT rooted at the keys given by the $keys_{SKT}$. This structure allows a user to compute all the needed keys.

The question is how big is such a structure that stores the whole forest a user needs? If we consider a Boolean matrix representation of the forest, the structure stored by a user can vary from 0 to $M \times M$ bits, depending on the user position in the CAG. For $M = 10000$ categories, this gives a maximum of about 10 MB, while

for 100000 categories the maximum is 1GB for the storage requirement per user to keep her forest of descendants. This could be prohibitive too.

Balanced addressing tree

An elegant solution to the problems above is to add more regularity to the SKT, such that most of the information can be inferred and the information to be conveyed and stored is minimized. To do so we consider the submersion of the spanning tree into a balanced tree with a fixed branching factor B and depth d. B and d are respectively the maximum branching factor and the depth of SKT. We call this tree the balanced addressing tree. Figure 5-5 shows a part of the balanced addressing tree of SKT from Figure 5-1. The gray nodes correspond to the real nodes in the SKT, where the other nodes are only for keeping the regularity.

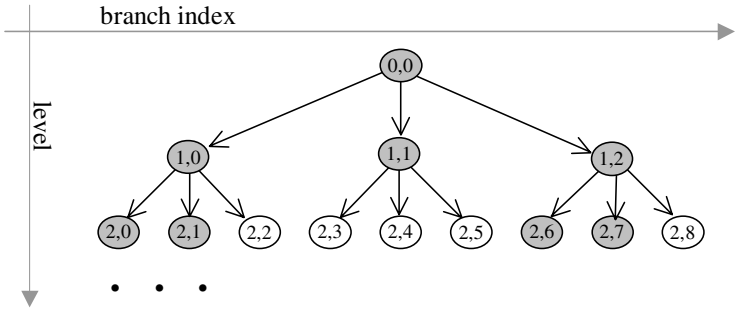


Figure 5-5: Balanced addressing tree

To address a node into this tree we use two axes - the level of the tree and the branch index corresponding to a level. A node in the tree is unambiguously addressed by the pair (i, j), where i is the level of the node and j is the node's branch index at level i, more exactly $i = \overline{0,d}$ and $j = \overline{0,B^i - 1}$, where d is the depth of the tree.

Given two nodes with the addresses (i, j) and (i', j'), i' ≤ i,

- we want to know if there exists a path from (i, j) to (i', j');
- and, if so is, we want to find this path.

Observe that the structure is a tree, and therefore between any two nodes exists at most one path. To answer the first question we just need to perform a simple test. If $(j - j') \le B^{i-i'}$ then there is a path from (i, j) to (i', j'), otherwise there is not.

Suppose that a path exists. We denote this path as the chain of addresses (i_1, j_1), (i_2, j_2), ... (i_m, j_m) such that (i, j) = (i_1, j_1) and (i', j') = (i_m, j_m). Then, starting from (i_m, j_m), such a path is backward recursively given by:

$$(Eq.5\text{-}7)\ i_{k-1} = i_k - 1 \ \text{and} \ j_{k-1} = \left\lfloor \frac{j_k}{B} \right\rfloor, \text{for } k = m, m\text{-}1, \ldots 2.$$

Following the path (i_1, j_1), (i_2, j_2), ... (i_m, j_m), and knowing the hash key $K(i_1, j_1)$, we can derive the key $K(i_m, j_m)$, such that $K(i_{k+1}, j_{k+1}) = h(K(i_k, j_k) \| j_{k+1} \bmod B)$, $1 \le k \le m$.

Therefore, the balanced addressing tree offers a convenient way of addressing the keys such that each user with a set of keys can easily find a path, if it does exist, from one of its keys to the key used for the encryption. And she can do this by storing only a minimum of additional information – the branching factor of the addressing tree. The additional computation of finding indices for the nodes in the indices is negligible compared with the iterative hashing.

The problem with using balanced tree for addressing is that it imposes restrictions on the SKT, respectively CAG. Specifically, in practice we have a limited bit space for addressing; let us say of m bits. Therefore, the maximum depth and the maximum branching factor of the addressing tree are bound by the following inequality, constraining the depth and the maximum branching factor of the SKT, respectively CAG:

$$\text{(Eq.5-8)} \quad \lceil \log(d_{\max}) \rceil + d_{\max} * \lceil \log(B_{\max}) \rceil \le m .$$

For instance, in the example from Figure 5-1 the $d_{max} = 5$ and the $B_{max} = 3$; therefore, we need a minimum of m = 12 addressing bits. However, in a more realistic example, with m = 32 bit long addresses and $B_{max} = 4$ the maximum depth of the SKT supported by this addressing scheme is limited to 14. On the other hand, for $B_{max} = 32$ and $d_{max} = 1000$, which are the limits imposed by the experiments (to avoid increase of keys per user) and by the cryptographic hash function (to avoid collision due to iterative hashing), we would need m = 5010 bits addressing space (≈0.5KB) which becomes again prohibitive.

5.8 Conclusions

The idea of hash trees has been used before. Merkle [Mer89] proposed a digital signature system based on hash tree authentication to improve the storage requirement for one time signature systems. Iterative hashing has been also used in the S/KEY [RFC1760] for a one-time password system. The group key management scheme based on the one-way function tree (OFT) [MS98] uses a bottom-up approach, similar to Merkle, for reducing the storage requirement. Although we employ the same technique of creating hashing dependency among keys so that keys can be computed from other keys, our approach differs in that our hash key tree is top-down and it maps keys to the shortest paths spanning tree of CAG. A similar top-down hash tree construction has been used by Fiat and Naor [FN94] to reduce the storage requirement for their family of broadcast encryption schemes. Their schemes use a balanced binary tree where hashing the key of a node generates exactly two keys corresponding to the node's two children. In our scheme a node can have a variable number of children, and, in general, the tree is not balanced.

The SKT gives a large improvement compared to our initial category-based GKM, both for the number of keys per user and for the number of keys at the controller. The improvement is due to introducing hashing dependencies among the keys assigned to different categories. This way the storage requirements decrease at the price of increasing the requirements on computation. Each user is required now to compute (instead of storing) a certain encryption key from the keys she has. Moreover, to follow a computation, the user must find a path in the SKT, usually by storing additional information regarding key dependencies. But this could hinder the gains in the storage requirements obtained from SKT. We have shown that an elegant solution to this problem is to use a balanced tree with a fixed branching for addressing keys in SKT.

Chapter 6

Cluster-based group key management

6.1 Introduction

In Chapters 4 and 5 we designed a group key management scheme capable to balance the efficiency versus the given collusion resistance requirements. The collusion resistance requirement was based on the user categorization and on the fact that applications have certain assumptions regarding user accessibility to the multicast channel. However, for some applications, the user categorization is not always possible. For these cases we propose a generic cluster-based group key management scheme (cluster-based GKM) that can be used to secure any multicast application without assuming specific properties of that application. Cluster-based GKM provides the possibility to fine-tune the efficiency versus security tradeoffs even at system runtime; therefore, the scheme is flexible to adapt to changes in the working environment and in the security assumptions.

6.2 Hybrid key tree

As in the previous chapters, we consider a centralized architecture where a group controller has the responsibility for the key management. It initializes the creation of the keys, distributes them to the group members, and rekeys the group. In addition, the controller also logically partitions the group of users into clusters of equal sizes.

The controller stores all the keys into a *hybrid key tree* (HKT). Two types of keys are distinguished in HKT, the group key (GK), used to encrypt the multicast channel, and the auxiliary key encryption keys (the $K_{a,b}$ and $K^{i}_{l,q}$), used to rekey the group. The auxiliary keys are further partitioned into *global* and *cluster level*, each leaf key in the global level being the *root of a cluster* (e.g. $K_{2,2}$ is the root of the

cluster c_2, in figure Figure 6-1). The keys in the cluster level are used to logically control the membership of users to clusters, i.e. they are used to distribute the cluster root key among the members of that cluster. There is a bijection between each cluster of users and a set of keys from the HKT's cluster level, as shown in Figure 6-1.

The global level logically controls the membership of clusters to the multicast group, i.e. they are used to distribute the GK to all users within cluster members in the multicast group. By coupling the global and local levels we control the membership of users to the whole multicast group.

The HKT is hybrid in that the mapping of the encryption keys to nodes in the key tree structure differs in the global level from the cluster level. At the global level, each node receives a unique key. On the other hand, at the cluster level, the same key can be mapped to more than one node, as it can be seen in the second level of the cluster c_2 in Figure 6-1. The rationale for having clusters and a hybrid key structure is to achieve configurability of the efficiency versus security in a granular way. The cluster level uses fewer keys than the global, and has better rekeying performance. However, users within a cluster could collude. On the other hand, users from different clusters cannot collude. Therefore, by tuning the cluster cardinality, applications using HKT can balance their efficiency and security requirements.

Key assignment

The group controller uniquely assigns each new joining user to a cluster and implicitly to a leaf node in the HKT. Currently, this assignment is done arbitrarily to any of the clusters that can accommodate a new user, such that the HKT remains balanced, although being balanced is not mandatory for the HKT to work.

Based on this assignment, the user will receive the keys in the path from the assigned key leaf to the root of the tree. This set of keys includes the GK as well as the auxiliary keys from both global and cluster levels.

Let N be *the total number of users* and N_c *the number of users per cluster*. Then, there are $M = \lceil N/N_c \rceil$ clusters, each cluster c_i having an identifier from 0 to M−1. For the remainder of this chapter, we consider N to be a multiple of N_c and the $\lceil N/N_c \rceil$ to equal N/N_c.

A key $K_{a,b}$ in the global level denotes that the key is logically shared by all clusters with identifier i such that $a \leq i \leq b$. Implicitly, the $K_{a,b}$ is owned by all users in these clusters. All the keys in the global level are unique.

Each cluster c_i is assigned the group key GK and a set of auxiliary keys from the global level corresponding to its identifier:

$$(Eq.6\text{-}1)\ keys(c_i) = GK \cup \{K_{a,b} \mid a \leq i \leq b\}$$

Each user u_j^i from the cluster c_i borrows its cluster's identifier, and also receives a user identifier j within its cluster, $0 \le j < N_c$.

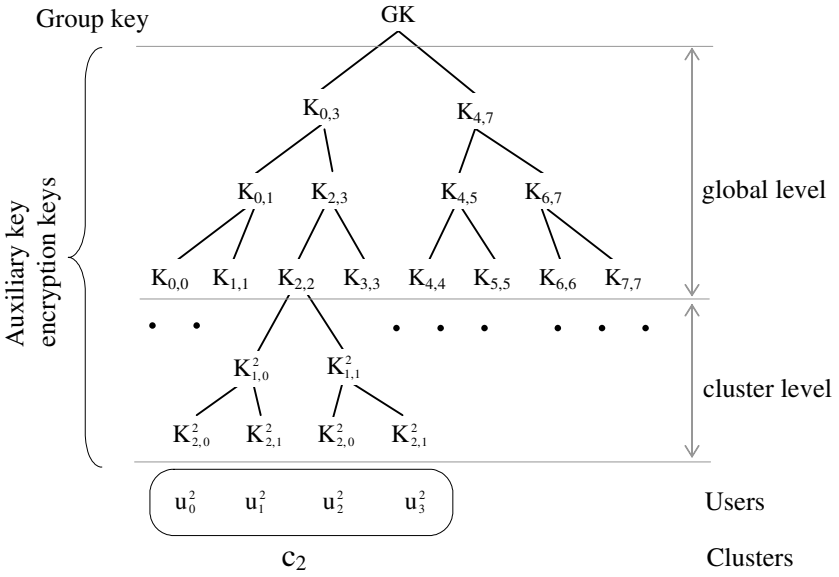


Figure 6-1: The hybrid key tree

A key $K_{l,q}^i$ denotes that the key is corresponding to cluster c_i, level l in the cluster level, where q can be 0 or 1. The level is local to the cluster. The keys in the cluster level are reused (only within a cluster but not among clusters), such that each new level in a cluster introduces only two new keys, $K_{1,0}^i$ and $K_{1,1}^i$ to that cluster. The depth of the cluster level is $d_c = \log(N_c)$.

Each user u_j^i receives the keys of its cluster along with a set of keys from the cluster level:

$$(Eq.6\text{-}2)\ \ keys(u_j^i) = keys(c_i) \cup \left\{ K^i_{1,b_1} \;\middle|\; (b_1 b_2 ... b_1 ... b_{d_c})_2 = j \wedge l \in \{1..d_c\} \right\}$$

where $(b_1 b_2 ... b_1 ... b_{d_c})_2$ is the binary representation of j.

As an example, user u_2^2 from Figure 6-1 will receive the following keys: the group key GK; the auxiliary key at the global level $K_{0,3}$ and $K_{2,3}$; the cluster root key $K_{2,2}$; and the auxiliary keys at the cluster level $K_{1,1}^2$ and $K_{2,0}^2$.

6.3 Management operations

This section describes the management operations for the multicast group. Two sets of operations are considered: the operations of changing the multicast group membership (i.e. users *joining* and *leaving* the group), and the operations of changing the characteristics of the key structure (i.e. *increasing* and *decreasing the cluster cardinality*). The scheme's resistance to collusion is dependent on the cluster cardinality (as shown in section 6.4); in particular, the bigger the cluster size the more relaxed the collusion resistance. Thus, the latter set of operations provides the means of actually configuring the scheme's resistance to collusion.

6.3.1 Membership change operations

We assume the existence of unicast protocols for authenticated and confidential one-to-one communication, used to securely distribute the keying material (GK and corresponding auxiliary keys) to new users joining the group. The keys and certificates involved in these protocols are not counted as part of the multicast key management scheme.

Single join

In order to join the multicast group, a user sends a unicast request to the group controller. The user and the controller use their certificates to authenticate each other. The controller checks the admission policy for the multicast group, and accordingly, grants or rejects user requests. If granted, the user is assigned to a cluster and is given an identifier within the cluster. To assure backward secrecy the group controller updates all the keys in the path from the root group key to the joining user. The keys are refreshed through a one-way function and then they are securely unicasted to that joining user. The previous members of the multicast group are informed about the update with the first multicasted data packet encrypted with the new GK. Each member will update its keys locally by performing the same one-way function operation, as the controller did. Therefore, to join a new user to the group the controller must transmit only log(N) keys to that new user.

Multiple join

If the controller receives many requests for admission in a short period of time, treating all the joins in a batch can optimize the operations. This is possible when the joining users are assigned slots that share paths into the key tree. The keys within these paths are then updated only once, rather than many times if a sequential single join approach is taken.

Single leave

The leaving operation can be initiated by the user or by the controller. When initiated by the user, the leaving can happen silently, such that the user just stops listening to the multicast channel. The controller can initiate the leaving operation

due to different reasons (end of subscription, breach of policy by user, etc.). In this case the user is actually forced to leave, and the controller does an active exclusion operation by rekeying the secure multicast channel. It is this case that we consider further.

Suppose user u_2^2 must leave the multicast group. In this case, the GK and all the keys in the branch from GK to that user must be updated. First, the controller generates new keys for the nodes that need to change. Then, the controller sends these keys to all the users except the user who is excluded. The rekeying is done in two steps.

We illustrate the algorithm:

Step 1. The controller updates the keys from cluster c_2. First, the controller generates a new root key $K_{2,2}^{new}$ for the cluster c_2. The controller encrypts this key with $K_{2,1}^2$ and $K_{1,0}^2$ and sends it to the group. The users generate new keys $K_{2,0}^2$ and $K_{1,1}^2$ by hashing the previous ones with the newly received $K_{2,2}^{new}$, i.e. $K_{2,0}^{2\ new} = h(K_{2,0}^2 \| K_{2,2}^{new})$ and $K_{1,1}^{2\ new} = h(K_{1,1}^2 \| K_{2,2}^{new})$.
(To complete this step $\log(N_c)$ keys are sent.)

Step 2. The controller updates the keys for the global level. It first updates $K_{2,3}$ by multicasting the new $K_{2,3}^{new}$ key encrypted with $K_{2,2}^{new}$ and $K_{3,3}$. Then, the new $K_{0,3}^{new}$ is encrypted with $K_{2,3}^{new}$ and $K_{0,1}$ and multicasted. Finally the new group key GK^{new} is encrypted with $K_{0,3}^{new}$ and $K_{4,7}$. Now all the users, except the excluded u_2^2 are rekeyed.
(To complete this step $2*\log(N/N_c)$ keys are transmitted.)

Multiple leave

It can be that the controller must exclude a certain number of users in a short period of time. These exclusions can be done sequentially, following many times the rekeying algorithm for single leave, or it can be improved by exploiting the fact that users that must be excluded could share common paths into the key tree. For instance, if all users from a cluster must be excluded, it is easier to do this in batch, such that the cluster itself is actually excluded. The Boolean function minimization (BFM) [CEK99] has techniques that primarily target the cumulative user exclusion. These techniques can also be used here when excluding users from the same cluster. They cannot directly apply to cumulative exclusion of users from different clusters. However, in the case of users from different clusters we can still apply a technique to minimize the rekeying when clusters are rekeyed in a batch. More precisely, consider that users u_2^2 and u_3^3 must be excluded in a short period of time, and it is

decided that they will be excluded in batch. First, the controller excludes individually each user from its cluster by applying the first step of the algorithm for clusters c_2 and c_3. Further, the controller updates the keys from global level, but it does it simultaneously for both clusters, such that it uses only 6 messages instead of the 2∗5 messages used for sequential updates. The cumulative exclusion in a batch is a main principle for achieving efficiency in [CEK99] and [SZJ02].

6.3.2 Operations on cluster cardinality

The operations for changing the cluster cardinality are *increase* N_c and *decrease* N_c. We treat the case when N_c is a power of 2. In this case we observe that the overall structure of the key tree is not changed (no nodes are added or deleted) but the border between the global level and cluster level moves up respectively down the overall hybrid tree. The scheme's resistance to collusion and its efficiency depend on the cluster cardinality. Therefore, the increase N_c and decrease N_c operations are performed by the controller, at runtime, to accommodate the possible changes in the environment, such as a considerable increase of the group size, decrease in bandwidth, or an increase of the probability of malicious users trying to collude.

Decrease N_c

When decreasing the cluster cardinality, the resistance to collusion increases. The cluster border moves down the hybrid tree. We assume that the border moves only one level down. The number of clusters doubles while the cluster cardinality is halved. Each cluster c_i generates two new clusters, with the indices 2∗i, and 2∗i+1. We need to assign new root keys for each cluster as well as to update the auxiliary keys within a cluster (otherwise each two clusters will share the same keys). For instance, for the previous cluster c_2 in Figure 6-1, instead of key $K_{2,2}$ we need two new and unique keys. We observe that the keys $K^2_{1,0}$ and $K^2_{1,1}$ are unique and that they can be used as the new root keys for the new clusters c^{new}_{2*i}, and c^{new}_{2*i+1}. Therefore, the two keys will be referred from now on as being the new $K^{new}_{4,4}$ and $K^{new}_{5,5}$. Note that this is the way each user computes its new cluster identifier starting from the previous identifier i and the previous keys $K^i_{1,0}$ or $K^i_{1,1}$. The previous keys $K_{i,j}$ will be renamed to $K^{new}_{2*i,\ 2*j+1}$.

The new auxiliary keys from the two clusters generated by the splitting of cluster i will be renamed from $K^i_{1,q}$ to $K^{2*i\ new}_{1-1,q}$ and $K^{2*i+1\ new}_{1-1,q}$. The keys must be updated such that they become different. The update is done locally by each user through applying a one-way hash function h (defined in Chapter 5), such that the new $K^{2*i\ new}_{1-1,q} = h(K^{2*i}_{1-1,q} \| K^{new}_{2*i,2*i})$ and the new $K^{2*i+1\ new}_{1-1,q} = h(K^{2*i+1}_{1-1,q} \| K^{new}_{2*i+1,2*i+1})$. Now, the two new clusters generated from cluster c_i have different auxiliary keys.

To perform the decrease N_c operation, the group controller needs to send only one message announcing the increment of the global level depth. All other update operations are performed locally by the users and by the controller.

Increase N_c

When increasing the cluster cardinality, the resistance to collusion decreases. The cluster border moves up the overall key tree and every two sibling clusters will be merged into one cluster. Therefore, cluster c_{2*i} and c_{2*i+1} are merged into the new cluster c_i^{new}. The previous root cluster keys $K_{2*i,\,2*i}$ and $K_{2*i+1,\,2*i+1}$ will be renamed to $K_{1,0}^{i\,new}$ and $K_{1,1}^{i\,new}$. The key $K_{2*i,\,2*i+1}$ becomes $K_{i,i}^{new}$, the new root of the new cluster.

To unify the auxiliary keys of the two merging clusters, the group controller takes the keys of one of the clusters, say cluster c_{2*i}, and passes them through a one-way function. That is $K_{1,q}^{2*i\;new} = h(K_{1,q}^{2*i})$. Each key is encrypted with the corresponding key from the other cluster, i.e. the key $K_{1,q}^{2*i\;new}$ is encrypted with $K_{1,q}^{2*i+1}$ and then it is multicasted to the group. Note that only the users from cluster c_{2*i+1} who had the $K_{1,q}^{2*i+1}$ can decrypt this message. Also note that users from cluster c_{2*i} can easily compute the new keys $K_{1,q}^{2*i\;new}$, by hashing their own keys. Now the keys $K_{1,q}^{2*i\;new}$ are renamed to $K_{1+1,q}^{i\;new}$ and these are the new auxiliary keys to be used into the new cluster. The increase N_c operation requires $2*\log(N_c)$ messages to be transmitted by the group controller.

6.4 Collusion properties

Colluding users belonging to the same cluster cannot be sequentially excluded. Since the keys in the cluster level are reused, the users could collude by putting their keying material together and cover an innocent user not in the coalition.

Assume users u_2^2 and u_3^2 must be excluded and assume they are colluding. Sequential exclusion of the two users will fail. This is because after the exclusion of the first user, e.g. u_2^2, user u_3^2 can "lend" her new keys to u_2^2. Using the new root cluster key, u_2^2 can update (by hashing) her auxiliary keys corresponding to the cluster level. This allows u_2^2 functioning as if no exclusion has taken place while the controller thinks the user u_2^2 is not in the group any longer. When u_3^2 is excluded, the u_2^2 pays back u_3^2's service and lends her keys to u_3^2. Similarly, u_3^2 will update her keys and continue to function.

The only way to securely exclude colluding users is to exclude the whole collusion at once. In our case to exclude u_2^2 and u_3^2 at once the controller has to send the new root cluster key encrypted with $K_{1,0}^2$. This message will be decrypted only by u_0^2 and u_1^2. With this new key users u_0^2 and u_1^2 can refresh their auxiliary keys and the rekeying is over at the cluster level. The problem is that excluding at once could also exclude innocent users that happen to have the same keys as the colluding users. In the worst case, the whole cluster is excluded, and therefore the whole cluster needs to be regenerated and distributed from scratch. For instance, if users u_0^2 and u_3^2 would collude they would have together all the auxiliary keys of all the other users in the cluster. In this case, the only way to exclude them is to reinitialize the cluster and join to this fresh cluster the remaining authorized users.

By partitioning the whole group of users in clusters, our scheme is able to contain the effects of users colluding within a cluster. In our scheme the worst case is to regenerate the keys for a whole cluster.

Colluding users belonging to different clusters can be sequentially excluded. Assume now that users u_2^2 and u_3^3 must be excluded. We show that they can be sequentially excluded from the group even if they maliciously collude. To exclude u_2^2 we first rekey within cluster c_2. Since the new root cluster key $K_{2,2}^{new}$ is encrypted with $K_{2,1}^2$ and $K_{2,0}^2$ it can not be decrypted by u_2^2 or u_3^3 even if they put their keys together (there is no dependency among their keys and the keys used for encrypting $K_{2,2}^{new}$). Users from c_2 use $K_{2,2}^{new}$ to update the auxiliary compromised keys. Then, the rekeying at the global level is done. Note that $K_{2,2}^{new}$ is not transmitted to any other cluster, therefore, u_3^3 will not know this key and cannot "help" u_2^2 to update its auxiliary keys. However, u_3^3 can give the new GK, $K_{0,3}$ and $K_{2,3}$ to u_2^2. But this does not allow u_2^2 to function any longer as before, but it is rather bound to u_3^3 such that when u_3^3 is excluded u_2^2 will be excluded too. Therefore, colluding users belonging to different clusters can be sequentially excluded.

In conclusion, the scheme's resistance to collusion depends on the cluster cardinality; the smaller the cluster the bigger the resistance to collusion. Therefore, for a given cluster cardinality N_c, we use $1/(N_c - 1)$ as a measure for the strength of the resistance to collusion. For clusters of size 1 and 2, the scheme's resistance is perfect. Therefore, we do not distinguish between these two cases, and we chose the domain for N_c to be 2 to N.

6.5 Evaluation

6.5.1 Efficiency analysis

We compute the storage cost for the controller and for the users, and the cost for rekeying in the single leave operation.

The user's key storage is (nr of keys):

$$\log\left(\frac{N}{N_c}\right)+\log(N_c)+1=\log(N)+1 \qquad \textit{(Eq.6-3)}$$

The controller's key storage is (nr of keys):

$$\left(2*\frac{N}{N_c}-1\right)+\frac{N}{N_c}*\left(2*\log(N_c)\right)=2*\frac{N}{N_c}*\left(\log(N_c)+1\right)-1 \qquad \textit{(Eq.6-4)}$$

The communication cost for rekeying is (nr of rekey messages):

$$\textit{(Eq.6-5)}\quad 2*\log\left(\frac{N}{N_c}\right)+\log(N_c)=2*\log(N)-\log(N_c)$$

Figure 6-2 depicts the controller's key storage in function of the number of rekey messages when N_c varies from 2 to 256, for a total of $N = 10^6$ users. The figure captures the tradeoff between efficiency (storage and communication cost) and security (resistance to collusion). The higher the value of N_c is, that is the more relaxed resistance to collusion, the lower the values are for both rekeying cost and controller key storage.

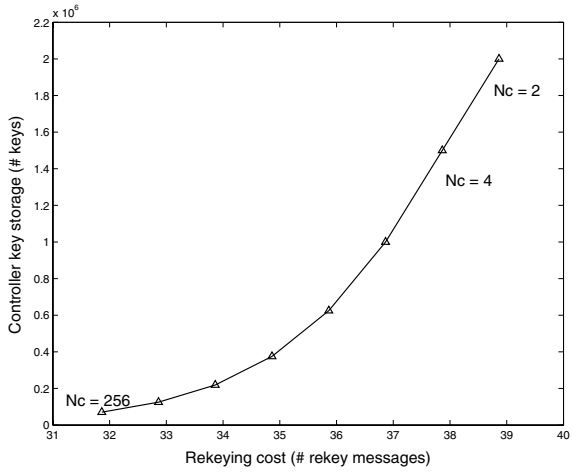


Figure 6-2: Security versus efficiency tradeoff in the HKT

6.5.2 Comparison

We summarize the comparison with our reference schemes, the LKH and flat, in Table 6-1. Then, we compare our solution with Canetti's scheme [CMN99].

Table 6-1: Comparison with the reference schemes

	LKH	Flat	Cluster-based GKM
Forward secrecy	Yes	Yes	Yes
Backward secrecy	Yes	Yes	Yes
Collusion resistance	Perfect	No resistance	Collusion resistance is configurable by cluster cardinality N_c
Communication cost	2*log(N) – 1	log(N)	2*log(N) – log(N_c)
User key storage	log(N) + 1	log(N) + 1	log(N) + 1
Controller key storage	2*N – 1	2*log(N) + 1	$2 * \frac{N}{N_c} * (\log(N_c) + 1) - 1$

One of the important properties of our cluster-based GKM scheme is that it can greatly reduce the controller key storage, see Figure 6-2. Another scheme having as major objective the decrease of the controller key storage is the tradeoff scheme proposed by Canetti and colleagues [CMN99]. In particular, Canetti's scheme trades the controller key storage with the communication cost while the security properties remain constant (i.e. the scheme has always perfect resistance to collusion). However, the scheme substantially increases the communication cost for rekeying. As a comparison, our scheme reduces both the controller key storage and the communication cost at the same time; the tradeoff is the resistance to collusion.

Figure 6-3 shows the different kind of tradeoff the two schemes achieve. Canetti's scheme balances communication and storage, whereas our scheme balances communication plus storage against resistance to collusion. This explains why the storage-communication curve of our scheme is always below the curve from Canetti's scheme, i.e. it is always more efficient. The price paid is security.

For a given number of users, in our graphic visualization $N = 10^6$, the LKH and the flat schemes have constant values for rekeying communication cost and controller key storage. They are depicted by the two fixed points in Figure 6-3. Our scheme is actually connecting these two points, allowing for fine-tuning of security requirements versus efficiency requirements. The increase and decrease N_c operations make it possible to move the working point of the HKT scheme along the communication-storage curve at runtime.

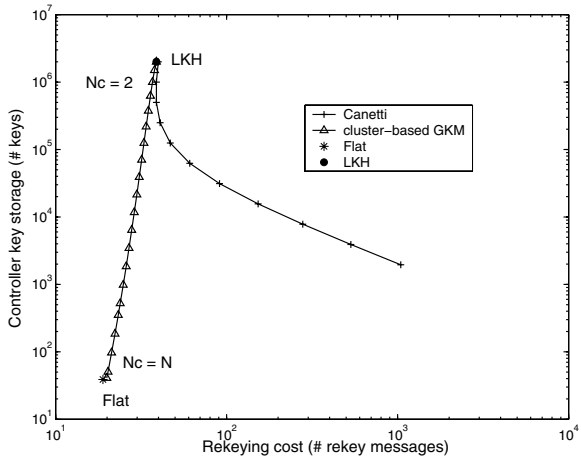


Figure 6-3: Comparison of the cluster-based GKM with Canetti's scheme, LKH, and Flat

6.6 Conclusions

This chapter proposes a cluster-based group key management based on a two level hybrid key tree (HKT). In the HKT the whole group of users participating in the multicast group is partitioned into clusters of equal size. The global level of HKT controls the membership of clusters to the multicast group, and the cluster level controls the membership of users to clusters. The different mapping of encryption keys to nodes in the key tree and the different algorithms used for controlling at global and cluster levels allows the two levels in HKT to have different security and performance properties. In particular, the cluster level is very efficient, but users within a cluster could collude. On the other hand, the global level, even if less efficient, assures that users from different clusters cannot collude. Therefore, by tuning the cluster cardinality, applications using cluster-based GKM can balance their efficiency and security requirements even at system runtime.

Chapter 7
Related work

7.1 Introduction

Here we present the related work in the area of secure multicast with the focus on efficient group key management mechanisms.

7.2 Group key management architectures

The broad and challenging problem of establishing and managing group keys touches upon a large body of previous work.

The existing literature classifies the group key management schemes upon various criteria. According to [KPT00], group key management protocols come in two different flavors: *centralized, server-based key distribution protocols* for large groups, specific to push-oriented scenarios such as the multicast software delivery, and *contributory key agreement protocols* for small groups, specific to collaborative environments. However, from an architectural point of view, the group key management can be classified into three main categories [Esk02], as shown in Figure 7-1. This will be our departure point in discussing the previous work in group key management. The three architectural categories are:

- Centralized group control: A single entity, the group controller, controls all the members in the group. It is responsible for the generation, distribution and replacement of the group key.
- Decentralized subgroup control: The secure group is divided into smaller groups, and each subgroup is assigned a different subgroup controller. The coordination of the subgroups can be either centralized through a unique super group controller, or it can be distributed among the sub-controllers.

- Decentralized member control: With no group or subgroup controllers, each member of the group is trusted with the access control and contributes to the generation of the keys.

As shown in the figure, the left side of the category hierarchy tends to be more appropriate for push-oriented applications, whereas the right side, specifically the member control category architecture, is devoted to collaborative settings.

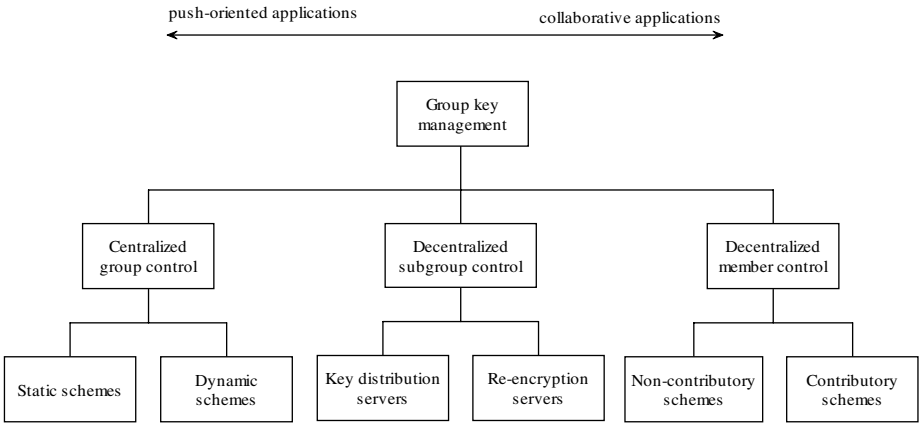


Figure 7-1: Group key management architectures

7.2.1 Centralized group control

In centralized group control architecture, a single entity, the group controller (GC), is fully responsible for the security of the multicast group. The group controller generates all the keying material and distributes it to the authorized group members. Whenever a group change should occur, it is the responsibility of the GC to perform the rekeying operation. The centralized architecture gives full control to the GC without spending trust into additional trusted parties. This makes it desirable for push-oriented applications distributing high value or confidential data.

However, the scalability of such a solution is a concern, since the group membership problem in large and dynamic groups must be efficiently handled. Therefore, this category can be subdivided into two classes of schemes: first, a static class of schemes which have simple, unicast based, and therefore non-scalable group key change mechanisms; and second, a dynamic class which provides special mechanisms for group key change within the context of large and dynamic groups.

Static schemes

The static schemes do not provide a solution for group key change upon group membership change. Whenever the group membership should be changed a new

group must be created. That is, a new group key is generated and distributed to all the remaining users by secure unicast means. Therefore, the key establishment protocols for one-to-one communication, such as the Simple Key-Management for Internet Protocol (SKIP) [CLA+96], by Caronni and colleagues, and the Internet Key Exchange (IKE) [RFC2409], by Harkins and Carrel, can be considered to be a part of this class.

The Group Key Management Protocol (GKMP) [RFC2094], by Harney and Muckenhirn, is a specialized protocol for group key management within the context of IP Multicast. However, GKMP uses a static key distribution mechanism too, such that the new GK is conveyed through secure unicasts to each group member whenever a group membership should occur. Although possibly attractive for its simplicity, the GKMP has linear communication cost for group rekeying, and therefore does not scale well to large and dynamic groups.

Dynamic schemes

Opposing the static schemes from above, the dynamic schemes have special mechanisms for handling large and dynamic groups. As these mechanisms constitute the main focus in this thesis, we will discuss this class in more detailed later in this chapter.

7.2.2 Decentralized subgroup control

The dynamic schemes provide efficient and scalable group key management in the context of centralized architectures. However, an alternative solution to achieve scalability is to actually adopt a decentralized architecture, by splitting the large multicast group in smaller (and therefore more manageable) subgroups, and delegate part of the security activities to sub-controllers administrating these subgroups.

Moreover, such a solution increases the fault tolerance as it removes the one point of failure introduced by the centralized group controller. However, the disadvantage is the reliance on trusted intermediary nodes.

The sub-controllers have different functions, depending on whether or not a unique group key is used to secure the whole multicast group, or contrary, each subgroup has its own subgroup key. In the former case, the subgroup controllers behave as key distribution servers, where the coordination of the servers can be either centralized or decentralized. In the latter case, the group controller has the role of re-encrypting the multicast traffic from one key to another key. We discuss below both the distributed key servers and the distributed re-encryption servers approaches.

Key distribution servers

The Scalable Multicast Key Distribution (SMKD) [RFC1949] by Ballardie proposes a distributed architecture for securing multicast that is highly interconnected with the multicast routing infrastructure. More exactly, the approach rests on the Core Based Tree (CBT) routing protocol; in CBT each router knows the identity of its

neighbor routers in the multicast routing tree. SMKD extends this idea by investing the routers with security functions, such that the routers will also authenticate newly joining members and will provide them with a group key, making the distribution of the group key and the security control apparently scalable. However, the scheme suffers from several drawbacks. First, the main disadvantages are the dependency of the security on the communication infrastructure and the need to actually deploy and manage new functionality within the core routers. Second, the SMKD has only a static solution to the problem of forward secrecy – whenever a group member is excluded the whole group must be recreated. This is a very costly operation that renders the solution non-scalable. Third, the SMKD requires a high level of trust in all routers in the CBT, as they all are in the possession of the group key.

The proposal from Intra-domain Group Key Management Protocol (IGKMP) [HCM00], by Hardjono and colleagues, adopts also a distributed architecture. In IGKMP, the group is divided into administratively scoped areas [RFC2365]; there is one Domain Key Distributor (DKD) and several Area Key Distributors (AKDs). The whole group has a unique group key which is generated by the DKD and it is distributed to the group members by the corresponding AKDs.

The dual encryption protocol (DEP) [DMS99], by Dondeti and colleagues, uses also hierarchical sub-grouping to achieve scalability. Third parties, hosts or members, are designated as subgroup managers (SGMs). They are responsible for secret key distribution and group membership management at the subgroup level. Two types of keys are defined in DEP: the key encryption keys (KEKs), used to hide data encryption keys (DEKs), and the local subgroup keys, used by SGMs to distribute encrypted DEKs to their subgroup members. When rekeying, a SGM signs the new subgroup key and encrypts it with the public keys of all the subgroup members. Then, it multicasts the protected key to its subgroup members. The protocol doesn't need to trust the subgroup managers with the distribution of the encryption keys.

The Hydra system [RH02], by Rafaeli and Hutchison, is a scalable decentralized architecture to create and distribute group keys to large multicast groups. In Hydra, the group is divided into a number of time-to-leave (TTL) scoped regions, each such a region being controlled by a dedicated Hydra server (HS). However, Hydra departs from the previous approaches as it does not employ a central manager for the HSs, thus being resilient to the failure of a single entity. Instead, the coordination of the HSs is done in a fully decentralized manner. To assure the synchronization of all HSs, the authors propose a Synchronized Group Key Distribution Protocol (SGKDP) which builds upon the Spread group communication service [AS98]. Spread provides totally ordered message delivery, which in turn guarantees the needed synchronization of the Hydra servers. A unique group key is used for the security of the multicast group. Whenever the group membership changes a new group key is agreed upon by the HSs using the SGKDP protocol. The actual rekeying is done on a subgroup basis and uses efficient key tree mechanisms.

However, the overall scalability and effectiveness of the scheme for Internet applications is highly dependent on the performance of Spread over a very large network area.

Re-encryption servers

Iolus by Mittra [Mit97] is a high-level infrastructure for securing multicast, designed to overcome the scalability problem of GKMP, by decentralizing the control to a hierarchy of trusted servers. In Iolus, a given group is partitioned into subgroups. Each subgroup is assigned a subgroup controller, called group security intermediary (GSI). GSIs are specialized trusted servers that are authorized to act like proxies of a centralized group security controller (GSC). Each GSI is responsible for the security of its subgroup, as it establishes the encryption key for that subgroup; i.e. each subgroup has its own local subgroup key. This positively influences the scalability; each change of the subgroup membership is locally handled and does not affect the other subgroups. However, the subgroup rekeying is based on simple unicast techniques; therefore, the communication complexity increases linearly with the size of the subgroup, imposing a limitation on the maximum group size. The GSC coordinates the top-level GSIs and it is ultimately responsible for the security of the overall group. Each GSI communicates with the GSC using a unique key shared by that GSI and the GSC. The GSC only multicast data traffic and keys to the GSI, and therefore it stores only one encryption key for each GSI. Most of the work, however, is done by the GSI which has to relay the multicast traffic from the GSC to the subgroup it controls after a re-encryption phase. This leads to substantial resource consumption for each GSI and to considerable latency for the overall multicast communication. Iolus provides fault-tolerance, to certain extent, but at the price of investing full trust in each subgroup controller, since each GSI has access to the multicasted traffic.

As noted above, a strong drawback of Iolus is that it fully trusts the intermediary nodes with the security of the group. A solution to this problem was provided by Molva and Pannetrat in their Cipher Sequences [MP99]. In this scheme the intermediate elements perform security transformations but without being trusted, as they do not have access to the deciphered multicast content. However, a main drawback of Cipher Sequences is that it relays on asymmetric cryptographic transformation which limits its use to typically short encrypted messages. To overcome this problem, the authors propose a scalable multicast access control framework targeted for large and dynamic groups [PM02]. The new scheme uses intermediary elements in the network that contribute individually to the encryption, providing confidentiality, backward and forward secrecy, and also containment – a property that limits the impact of compromised member access keys.

7.2.3 Decentralized member control

With no group or subgroup controllers, each member of the group is trusted with the access control and, in general, contributes to the generation of the keys. Regarding the latter aspect, two classes can be distinguished. The first class of schemes focuses on the efficiency of the membership control. However, it does not provide the means by which group members could equally contribute to the generation of the keying material. This makes some of the parties more trusted than others. The second class, a contributory class of schemes, focuses on assuring that all members input into the group key generation. We note that most of the non-contributory schemes are based on extensions of dynamic schemes from the group control category, such as the flat or the OFC, where the contributory schemes are mostly based on extensions to the two party Diffie-Hellman key agreement protocol [DH76].

Non-contributory schemes

The Distributed Flat scheme from VersaKey [WCS+99], by Waldvogel and colleagues, adopts a flat scheme for group key management in a totally decentralized manner. There is no dedicated group manager. Instead, every participant may perform admission control and other administrative functions. Moreover, as these administrative functions could be easily transferred to another participant, the scheme is highly resilient to network or node failure having an inherent self-healing capability. However, the scheme does not provide a fully fledged key agreement protocol, because it lacks the contributory key agreement feature. Furthermore, the flat scheme is highly susceptible to collusion attacks. Also, the scheme might experience important delays in synchronizing keys in conditions of intermittent connectivity and relatively many group members scattered on a wide area network.

Contributory schemes

Cliques [STW98], by Steiner and colleagues, addresses dynamic membership change in group key agreement and proposes a family of Group Diffie-Hellman (Group DH) protocols based on a natural extension of the two party Diffie-Hellman key exchange. Group DH provides contributory authenticated key agreement, key independence, and integrity. While this protocol provides a way to distribute a group key in highly dynamic groups, the solution does not scale well to large groups because it requires as many rounds (exponentiations) as the number of new members to complete the key agreement. Consequently, the rekeying messages become also prohibitively large.

The work by Kim and colleagues [KPT00] proposes a new approach to group key agreement by blending the binary key trees with Diffie-Hellman key exchange. The resultant protocol is simple, secure, and fault-tolerant, targeting mostly collaborative applications involving groups of rather small sizes.

7.3 Dynamic group key management schemes

The dynamic group key management schemes play an important role for the scalability of centralized group key management. However, the dynamic schemes have also an important impact to the scalability of sub-group based decentralized key management, since the dynamic schemes can be used as base for sub-group key management, e.g. in Hydra [RH02]. Moreover, extensions to the dynamic schemes are also used in the context of decentralized member group control, e.g. [WCS+99] and [Per99].

7.3.1 Logical key hierarchy schemes

The tree key schemes, or the logical key hierarchy (LKH) based schemes, were proposed in [WGL98] by Wong and colleagues, in [CWS+98] by Caronni and colleagues, and in [RFC2627] by Wallner and colleagues. They address the problem of minimizing the communication complexity of the rekeying operation without relaying on secure third parties (e.g. as in Iolus). Therefore, LKH schemes are usually used within centralized architectures. However, the key tree can be split and distributed to trusted parties, each managing a part of the overall tree. Also, the LKH can be used as an efficient scheme for sub-group key management in the context of sub-group based decentralized architectures. For instance, LKH is the adopted scheme in Hydra's subgroup management [RH02].

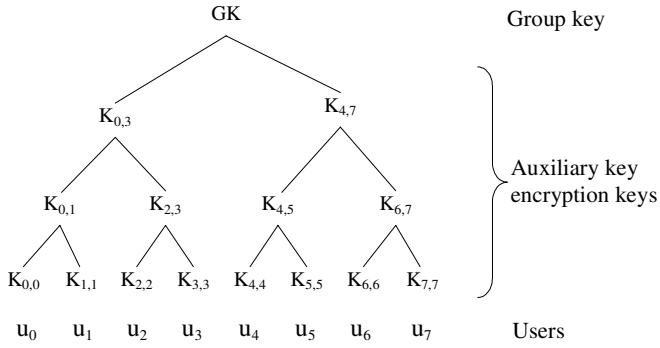


Figure 7-2: Logical key hierarchy (LKH) scheme

In LKH all the keying material is organized in a tree which is usually presented as being binary and balanced (see Figure 7-2), although this is not necessarily a requirement. The key leafs in LKH are the group members' individual keys, the key at the root of the tree is the group key (GK), and the internal keys are auxiliary key encryption keys. The group controller (GC) stores all the key structure, where each user stores only the keys along the path from itself to the root. This includes the GK and a small number of auxiliary keys. For instance, user u_3 from the figure receives

the keys GK, $K_{0,3}$, $K_{2,3}$, and $K_{3,3}$. In general, for a group of N members, the GC stores 2*N-1 keys where each group member stores log(N)+1 keys.

When rekeying the group, due to, for instance, user exclusion, the GC uses auxiliary keys from the LKH to encrypt the control traffic. During rekeying all the compromised keys, i.e. the auxiliary and the GK held by the excluded user, are replaced.

To describe the rekeying algorithm, we consider that user u_3 must be excluded. To do this, the GC generates a new group key GK^{new} that must be securely conveyed to all users but u_3. Also, the auxiliary keys $K_{0,3}$ and $K_{2,3}$, in u_3's possession, must be renewed. The algorithm starts with renewing the auxiliary keys bottom up. This is done by sending to user u_2 the new $K_{2,3}^{new}$ encrypted with $K_{2,2}$; sending to users u_0 and u_1 the new $K_{0,3}^{new}$ encrypted with $K_{0,1}$; sending to user u_2 the new $K_{0,3}^{new}$ encrypted with $K_{2,3}^{new}$. The new session key SK^{new} is now sent to users u_0, u_1, and u_2 encrypted with $K_{0,3}^{new}$, and to users u_4, u_5, u_6, and u_7 encrypted with $K_{4,7}$. The scheme uses only 5 messages to rekey all 7 remaining users. In general, the communication cost for rekeying is 2*log(N)-1.

A number of improvements have been proposed to the initial LKH. The one-way function tree (OFT) [MS98], by McGrew and Sherman, halves the communication cost for rekeying. The improvement is based on a special arrangement of the auxiliary keys in the key tree such that a parent key is obtained by hashing its child nodes. The one-way function chain (OFC) [CGI+99], by Canetti and colleagues, applies a similar principle to LKH, but only to the compromised keys that need to be updated when a rekey is done, and again, halves its communication. It has been shown that these schemes provide backward secrecy, forward secrecy, and perfect resistance to collusion.

Di Pietro and colleagues [PMM03] propose a methodology to establish the minimal key length that guarantees a certain level of confidentiality for the multicast traffic when securing with LKH scheme. The authors base their work on an extended threat model for LKH which takes into consideration the required lifetime of the information confidentiality, the level of a certain key in the key tree structure, and the frequency of user exclusion from the group. From these, the authors deduce the relation between the length of a key at a certain level in the tree and the achieved confidentiality level. The findings show that keys towards the root can have smaller length than the once at the leaves. The proposed methodology makes consequential savings in communication, as well as in user and controller key storage.

7.3.2 Flat schemes

Wong and colleagues generalize the key tree to a key graph [WGL98]. Following, Yang and Lam [YL00] show that, in a key graph, the lower bound complexity for rekeying communication, when the only allowed operation for the GC to distribute one key is to encrypt it by another key, is logarithmic with the group size. This is a

property of LKH and its variants. However, more efficient results are achieved by reusing auxiliary keys among different groups of users and encrypting the group key with a combination of these keys. This way ad-hoc groups can be created which usually cover more users than the groups pre-established by the key tree. The Boolean function minimization (BFM) [CEK99], by Chang and colleagues, and the Flat-VersaKey, by Waldvogel and colleagues [WCS+99], fall in this category called the flat or matrix based schemes. Chang and colleagues [CEK99] describe the flat scheme similarly to the LKH; i.e. all the keying material is arranged on a binary tree, with the GK at the root of this tree, as shown in Figure 7-3.

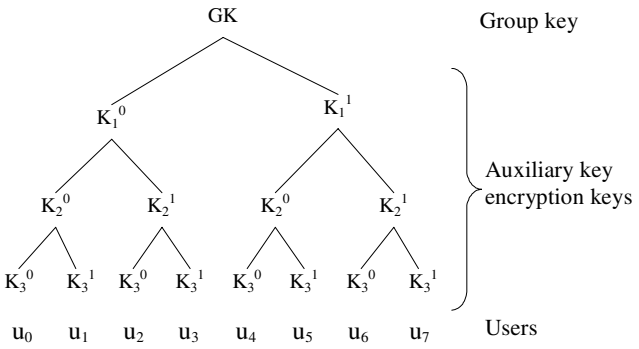


Figure 7-3: Flat scheme

Similarly to LKH, the GC stores the entire key tree, whereas each user is uniquely assigned the keys from a branch of this tree. The difference is that the auxiliary keys at each level in the tree are not unique as in the LKH. Instead, the auxiliary keys are reused, such that the child keys of any two parent keys, at a certain level in the tree, are the same. That is, all parents at a certain level in the tree have the same set of sibling child keys. Only the siblings differ among them. Therefore, for a binary key tree, each level in the key tree adds only one pair of siblings. As the GC has to store all the keying material, this leads to a drastic reduction of the storage cost for GC from 2*N-1, as in LKH, to 2*log(N)+1.

We illustrate the rekeying algorithm for the flat scheme as proposed in [CEK99]. For instance, to exclude user u_3 from the group (see Figure 7-3), the following steps are taken: the GC generates a new group key (GK^{new}) and multicast it encrypted with the keys K_3^0, K_2^0, and K_1^1. Each user from the group with the exception of u_3 has at least one of these three auxiliary keys, and therefore can decrypt and learn the new group key. To renew the compromised keys K_1^0 and K_2^1, the GC and the users pass them through a one-way hash function, such as $K^{new} = h(K^{old} \| GK^{new})$ for any compromised key K^{old} that needs to be updated.

The schema uses only 3 messages to rekey all 7 remaining users. For N users this means log(N) transmissions. That is, the communication cost is halved compared to LKH, and all this without any additional cost in group member storage.

However, the tradeoff is security; the flat scheme is highly prone to collusion attacks. In flat scheme any two users could actually maliciously collude and become resistant to the member exclusion operation. Therefore, although very attractive from an efficiency point of view, the flat scheme has no resistance to collusion.

7.3.3 Tradeoff schemes

The work in this thesis aims at providing scalable group key management schemes that fill the gap between the not so efficient schemes, but providing perfect resistance to collusion, and the very efficient schemes, but with no resistance to collusion. Our proposed schemes have a configurable working point, which can be set to best meet the application efficiency and security requirements. Therefore, we categorize our schemes as tradeoff schemes.

The tradeoff schemes strive at balancing certain parameters, such as storage versus communication, or security. These schemes are flexible and adapt to the unique requirements of each application. In many cases the tradeoff schemes combine different basic techniques having different properties.

Canetti and colleagues [CMN99] describe a scheme that reduces the storage cost for the controller at the price of increasing communication. The scheme has a variable working point (shown in section 6.5.2) that could be configured to meet the given requirements. To this last aspect, the algorithms proposed by Li and colleagues [LPB01] find the optimal configuration when constraints on communication and storage are given. However, the proposed scheme decreases the storage requirement only at the price of increasing communication. Instead, our cluster-based GKM scheme (Chapter 6) can reduce both the controller key storage and the communication cost at the same time, by relaxing the requirement on resistance to collusion.

A similar tradeoff, security versus efficiency, is made in the efficient large-group key distribution protocol (ELK) [PST01], by Perrig and colleagues. The scheme proposed there allows trading communication cost with security and computation. The idea is to avoid the transmission of the whole key when rekeying, by sending only smaller hints of the updates that would enable an authorized receiver to compute the key, partially by brute search.

The family of broadcast encryption schemes by Fiat and Naor [FN94] reduces the group members' storage cost and the communication cost for dynamically changing privileged subset of users. The schemes allow a controller to broadcast a secret to any set of privileged users so that coalitions of k users, not in the privileged set, cannot learn the secret. The parameter k specifies the maximum size of a corrupt

coalition to which the scheme is still resistant. However, their work has mostly a theoretical value since the communication cost for rekeying is prohibitively high, increasing squarely with the resistance to collusion. For a group of N members and perfect resistance, the complexity of the rekeying cost is $O(N^2*\log^3(N))$. Moreover, they use a static distribution scheme with predefined slots.

The hybrid structuring of receivers (HySOR) [FJA02], by Fan and colleagues, is a hybrid scheme combining the LKH and the linear ordering of receivers (LORE), which is a new protocol developed by the authors. LORE has good performance with regard to communication, but has no resistance to collusion. Combining the two mechanisms, the authors achieve a scheme capable of trading the communication versus resistance to collusion. At the extremes, the scheme functions either as pure LKH or as pure LORE. From this point of view the HySOR is similar to the cluster-based GKM proposed in Chapter 6. However, HySOR reduces only the communication cost, whereas our scheme is capable of reducing both the communication cost and the controller key storage cost, at the same time. We also note that the ordering principle at the base of LORE is similar to our user categorization presented in Chapter 4. Nevertheless, our user categorization is more general since we use a partial order, which makes the user ordering from [FJA02] a special case in which the ordering relation is confined to a total order.

7.3.4 Other schemes

Batching schemes

Some authors, e.g. in [CEK99] and [YLZ+01], propose that groups are rekeyed periodically instead of on every membership change. Periodic or batched rekeying can reduce both the processing and communication overhead at the key server, as well as improve the scalability and performance of the key management protocols. For instance, Kronos [SKJ02], by Setia and colleagues, is a group-based decentralized key management system where the group management is driven by periodic group rekeying. A new key is generated after a certain period of time; during this time period all the group membership changes are collected and processed at the end of the period, when the new group key is distributed. The coordination among the Kronos subgroup controllers is based on synchronized clocks, so that the servers could change the key at the same time. If desynchronizations should occur then the integrity of the group status is lost. Moreover, each new group key is generated from the previous group key; a disclosed key compromises all the following keys.

The flat scheme from [CEK99] proposes Boolean minimization techniques (BFM) to reduce the number of messages required to rekey multiple membership leaves into one single operation. The resulted number of messages to exclude k users at once is generally less than k**messages to individually exclude a user*. We

note that these techniques can be also applied to our cluster-based GKM and to the category-based GKM when a flat scheme is chosen to rekey within a category.

Self-healing schemes

The Keystone [WL00], by Wong and Lam, and the ELK [PST01], by Perrig and colleagues, address the problem of group key establishment for large and dynamic groups over an unreliable network. The problem is that in an unreliable network, the messages carrying the group key might never reach a user. In this case the user should request a retransmission from the group controller. However, the retransmissions would add to the network traffic, probably already heavily burdened, and could even overwhelm the GC with requests. Keystone and ELK address these problems through non-interactive means; users shouldn't contact the GC for a lost key, but rather they can recover these keys on their own, without requiring additional retransmissions. These schemes achieve resistance to packet loss by appending additional key update information to the packets following a key distribution; Keystone uses error correcting codes to generate information about past group keys, whereas ELK uses "hints" for the updated keys. However, care must be taken in these schemes to assure that new members do not receive information letting them recover past group keys, and compromise the backward secrecy. Moreover, these schemes are statefull – an off-line member becomes excluded if she doesn't receive the updates in time. To overcome these issues, Staddon and colleagues [SMF+02] introduce a self-healing key distribution mechanism that extends the subset difference rekeying techniques, proposed in [NP00] and [NNL01]. Moreover, the described approach is stateless, as inherited from its underlying techniques [NP00]; an authorized group member who has been off-line for some time is able to recover new group keys immediately after coming back on-line.

The work by Zhu and colleagues [ZSJ03] is also based on the subset difference rekeying method [NNL01] and aims at actually enhancing the latter with reliability and self healing capabilities.

Subset cover

Motivated by the copyright protection problem, the subset cover [NNL01], by Naor and colleagues, is a framework targeting systems with stateless receivers, such as CD or DVD players. In such a scenario, the receiver is not capable of recording the history of all the transmissions and changing accordingly the state. Instead, the receiver's operations should be based on the current transmission (e.g. the CD it currently plays), and the set of keys from the initial configuration. Only compliant devices should be able to decrypt the content. Moreover, illegal and compromised devices should be revoked, and ideally, the traitors should be traced and made responsible for their actions. To accomplish the revocation goal, the authors propose two explicit schemes within the context of their framework – the complete sub-tree, and the subset difference schemes. The two schemes' performance is as follows: the receivers storage requirement is $\log(N)$ and $\frac{1}{2}*\log^2(N)$ respectively, where the

message length to revoke r users is r*log(N) and 2*r keys respectively. In both schemes the center storage is linearly increasing with N. Also, the authors describe a general tracing mechanism that can be seamless integrated with any of the revocation schemes. The second scheme is substantially better than any previously known algorithm for the revocation problem, since it uses only 2*r keys to establish a new group encryption key. However, we note that the number of revoked users is constantly increasing, because the users revoked at a certain session must be revoked also at all further sessions. Although this fits the stateless scenario, where the receivers are actually media players (e.g. DVD player) and the transmissions consist on actually playing certain media (e.g. a DVD), the mechanisms do not accommodate the conditions of very dynamic and transitive groups encountered in the multicast Internet applications.

Chapter 8
Conclusions and future work

8.1 Introduction

This chapter presents the conclusions of our research and gives pointers to open issues and future research.

8.2 Conclusions

Multicasting is an enabling technology that plays an important role in the design, development, and operation of many current and next generation applications and services that rely on the efficient delivery of data from one sender to multiple receivers. However, at this stage in the development of multicast technologies, security is a concern as the maturity of multicast security solutions has the potential to enable the use of multicast for confidential and high-value content distribution.

A central problem of multicast security is to provide mechanisms for data confidentiality and group membership control. To deal with these security issues we adopt a secure group model. A unique symmetric cryptographic key is shared by all authorized members of the multicast group. The sender uses this key to encrypt the multicast content such that only authorized users, members of the secure group and being in the possession of the group key, can decrypt the content. As the secure group can be quite large and dynamic, the efficiency of the group key management is a core problem for the scalability of the multicast security.

However, tradeoffs can be made between security and efficiency of group key management in order to fit the application requirements. Therefore, we have identified the collusion resistance requirement as a potential knob for setting the tradeoffs between security and efficiency of group key management schemes. In

particular, we observe that resistance to collusion has an impact on the efficiency of any scheme. Also, we note that applications might have certain knowledge regarding users, their accessibility to the multicast channel, and implicitly, they have trust assumptions regarding users' collusion. Therefore, by taking into consideration such knowledge we set the security-efficiency tradeoff to best meet the application needs.

We generalize the collusion resistance definition and formalize it based on the user accessibility to the multicast channel. We show how different configurations of categories give different constraints on collusion resistance, and we point out that the previous work can be reformulated as special cases of our collusion resistance requirement definition.

We propose a category-based scheme for the general case of collusion resistance requirement that balances the given collusion constraints against efficiency. For applications such as software delivery, having certain assumptions on user accessibility to the multicast channel, this scheme improves the rekeying cost and the storage cost for the controller. The analysis is given in terms of the number of message units required to be transmitted when rekeying the group and the number of keys required being stored by each group member and by the controller. We find that, in general, there is a tight dependency between the exact settings of the collusion resistance requirement, represented by the category accessibility graph (CAG), and the performance of the proposed solution. In general, the solution has good performance for the rekeying cost. However, the user key storage may become large due to the propagation principle.

To alleviate the explosion of keys due to the propagation principle, we propose a mechanism based on the spanning hash key tree (SKT). The usage of SKT brings a big performance improvement compared to our initial category-based scheme, both for the number of keys per user and for the number of keys at the controller. The improvement is due to introducing hashing dependencies among the keys assigned to different categories. This way the storage requirements decrease at the price of increasing the requirements on computation. Each user is required now to compute (instead of storing) a certain encryption key from the set of keys she has. Moreover, to follow a computation, the user must find a path in the SKT, usually by storing additional information regarding key dependencies. Nevertheless, this could hinder the gains in the storage requirements obtained with SKT. Thus, we have shown that an elegant solution to this problem is to use a balanced tree with a fixed branching for addressing keys in SKT. Still, this would impose constraints on the maximum branching factor, maximum depth, and the addressing space in the messages.

For some applications the user categorization is not always possible. For these cases, we propose a cluster-based group key management scheme that can be used to secure any multicast application, without assuming specific properties of that application. The cluster-based scheme provides the possibility to fine-tune the efficiency-security tradeoffs even at system runtime; therefore, the scheme is

flexible to adapt to changes in the working environment and in the security assumptions.

8.3 Future work

The research conducted in this thesis leads us to three main directions for our future work. We present these directions in the remainder of this section.

8.3.1 Efficient group key management

In general, the security and efficiency requirements have complex interdependences, and finding the best scheme, to balance all the parameters, is still an open issue. As such, future work remains to be invested in designing smarter key management schemes that bring the best balancing between security and efficiency.

In order to improve the efficiency of the group key management, a possible open track is to explore the knowledge on the heterogeneity of the multicast group. For instance, the user network connectivity to the Internet could vary much, from phone lines with modems supporting several Kbps, to cable broadband of several Mbps or even Gbps. Also, users could have different behaviors with regard to the groups. Studies done on the MBone have shown that there is a large spectrum of user behaviors [AA97], ranging from users who are always connected and listening to users who just sporadically enter the group, listen for a short while, and then leave the group.

Having the possibility to cluster the users with similar connectivity properties and behavior with regard to the group gives provisions for specialized mechanisms best fitting the conditions of each cluster. This could lead to substantial improvement of the efficiency and overall quality of the security mechanisms. However, the question will be how to actually infer the user connectivity and behavior and how to model it. A possibility we see here is to collect statistical data and propose probabilistic metrics. Still, it is questionable how feasible it is to collect such data and what are the privacy implications of doing so.

In this thesis we have considered a reliable control channel, such that all the transmitted (encrypted) keys arrive to their destinations, even if transmitted in a multicast fashion. This assumption is indeed supported by the multitude of reliable transport protocols [LG98, Bir99]. However, these protocols usually add delays to the overall communication. To address this issue the self-healing group key management proposes solutions based on adding redundant data into the control packets. However, applications have different reliability and real-time requirements, as shown in 2.5.1. A pay TV application may, for instance, very well support the loss of a number of packets, but require considerable speed instead [LL03]. Therefore, a more in-depth study of the application requirements with regard to

reliability and real-time issues is needed, and adequate group key management should be consequently designed.

The communication and key storage costs, as well as their tradeoffs with collusion, were in the focus of our work. However, an open venue remains to investigate the balance between security, communication, key storage, and computation complexity. The initial results show that a 'one key per controller' scheme is possible. However, the computation increases while the group membership changes in time. Therefore, the feasibility of such a solution is still to be proven.

In our future work we will also look at decentralizing our schemes. We see this task as naturally following the structure of our two proposed solutions. The category-based group key management schemes could be mapped to a decentralized re-encryption tree, whereas for the cluster-based approach we are looking for a decentralized key distribution service with both central cluster coordination and distributed cluster coordination. The goal of this future work is to achieve fault tolerance and containment for the group key management.

8.3.2 Secure multicast in wireless networks

A wide spectrum of wireless computing devices are becoming increasingly popular, ranging from powerful laptops to handheld Personal Digital Assistants (PDAs) and mobile phones with no negligible computing capabilities and interconnectivity. Since wireless networks are inherently broadcast technologies, the security solutions developed for multicast group communication are becoming relevant here as well. Moreover, the latest advances in incorporating multicast capabilities within the wireless networks enable the wireless environment to become an efficient multicast distribution environment [Var02]. However, the emerging wireless landscape adds a new dimension to the complexity of securing multicast – the mobility of the group members [Esk02]. The adopted group key management solutions must allow members to move to other networks without leaving the secure group and without disturbing the seamless connectivity. As members move from network to network, the trustworthiness of these networks could be an important factor in the design of the group key management. Therefore, we see the study of the intrinsic relation between the trust and the overall efficiency of the group key management in wireless and mobile networks as a follow-up of our current research in security versus efficiency tradeoffs in multicast group key management. Moreover, although not a high concern for the schemes developed in this thesis, the problem of minimizing the computation complexity for the group members becomes a stringent issue in the context of mobile devices with limited computation facilities powered by limited life time batteries.

8.3.3 Secure groups in peer-to-peer networks

The focus in this thesis was the push-oriented scenarios. However, in our future work we would like to approach also the secure group communication within collaborative and peer-to-peer settings.

The concepts of group and group communication are fundamental for the peer-to-peer (P2P) systems. Virtual organizations and virtual communities are built based on the group concept. Therefore, the emerging P2P programming frameworks, such as JXTA [JXTA], extensively support the creation and maintenance of groups, as well as the inter-peer communication within groups. However, the security in peer-to-peer systems is still a research issue.

An essential aspect in P2P settings is the lack of a central authority. In contrast to the push-oriented scenario, where a central point of authority could be identified (i.e. the group controller), in P2P settings the ownership and authority within the group is shared by all members in the group. Therefore, the core problem of the group key management shifts from the efficient rekeying mechanisms to the agreement and establishment of the group key. Moreover, these issues transcend to an even larger problem – the aspect of policy and trust in collaborative P2P settings.

Since there is no unique point of control in a P2P system, a very important factor in the overall security of the group will be the trust that group members afford to invest in each other, as well as the security and management policies governing the group.

Although a great deal of research has recently been published in the area of trust management, the issues of negotiating trust and security policies have been only partially studied before, and mostly only in the context of one-to-one interaction [SWY01, WL02]. According to [GKK+01] there are two aspects of a policy that need to be negotiated: a set of access properties, i.e. a common interpretation of a policy model, and a common state, i.e. the configuration of authorizations. These problems have been addressed to some extent but usually only in the case where a common interpretation was assumed. Therefore, adequate specification languages and protocols are needed to support policy and trust negotiation within peer-to-peer secure groups. Moreover, in order for the policies and negotiated trust to be effective, adequate enforcement is needed.

The significance of trust and security for the widespread adoption of the emerging overlay network technologies has been recognized by both P2P and Grid communities. Still, their views differ. The P2P community adopts a peer centric view where individual personal opinions are collected, exchanged, and evaluated, creating a highly decentralized and personalized web of recommendations, such as Poblano [CY01] or XRep [DCP+02]. On the other hand, Grid practitioners have had traditionally in their focus organizations and inter-organization trust and security. Their approach is towards centralized Public Key Infrastructure (PKI) certification

and extending the existing organizational trust models, such as the hierarchical centralized X.509 [X509]. However, more effort is needed to provide flexible and reliable trust and security models in overlay networks, especially within the context of converging P2P and Grid protocols and technologies [FI03].

As we have seen, the existing PKI infrastructures take extreme views to the logical topology of trust, which is either centralized (the hierarchical X.509 PKI) or totally decentralized (the Pretty Good Privacy (PGP) mesh PKI [Zim95]). The centralized approach has good efficiency with regard to verifying the trustworthiness of a certain certificate, but it also has the one point of failure problem that can hinder the whole structure. If the public-private key pair of the root certificate is compromised, then all the certificates used within that PKI must be revoked. Moreover, although the hierarchical PKI maps well within organizations with well-defined structures, it does not map naturally to the large scale of end user communities, and it might even introduce conflicts of interest when bridging trust between different organizations. On the other hand, the PGP adopts a total mesh trust topology, which seems more suitable to the large user communities. Also, the PGP is more resilient to the compromised key problem. However, the mesh trust topology introduces great penalties to the efficiency of algorithms for finding trust paths and for eventually deciding on the trustworthiness of a certificate. Therefore, we envisage an overlay PKI converging the existing pure centralized and pure decentralized trust models into a scalable, efficient, and resilient PKI framework, capable of naturally mapping trust from both the structured organizations and chaotic end user communities.

References

[AA97] K. Almeroth, M. Amar, "Multicast Group Behavior in the Internet's Multicast Backbone (MBone)", *IEEE Communications*, vol. 35, no. 6, pp 124-129, June 1997.

[AMO+90] R. Ahuja, K. Mehlhorn, J.B. Orlin, and R.E. Tarjan, "Faster algorithms for the shortest path problem", *Journal of the ACM*, vol. 3, no. 2, pp 203-213, 1990.

[AS98] Y. Amir, J. Stanton, "The Spread wide are group communication system", TR 98-4, Department of Computer Science, Johns Hopkins University, 1998.

[BH89] F. Buckley, F. Harary, *Distance in Graphs*, Addison-Wesley, 1989.

[Bir99] K. P. Birman, "A review of experiences with reliable multicast", *Software Practice and Experience*, vol. 29, no. 9, pp 741-774, July 1999.

[BPC99] I. Brown, C. Perkins, J. Crowcroft, "Watercasting: Distributed watermarking of Multicast Media", *Networked Group Communication (NGC'99)*, pp 286-300, November 1999.

[BR02] D. Bruschi, E. Rosti, "Secure multicast in wireless networks of mobile hosts: protocols and issues", *Mobile Networks and Applications*, vol. 7, no. 6, pp 503-511, 2002.

[Bri99] B. Briscoe, "MARKS: Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences", *First Int'l Workshop on Networked Group Communication (NGC'99)*, pp 301-320, November 1999.

[CD85] D. Cheriton, S. Deering, "Host groups: A multicast extension for datagram inter-networks", *9th Data Communication Symposium*, September 1985.

[CDK01] G. Colulouris, J. Dollimore, T. Kindberg, *Distributed Systems: Concepts and Design*, Third edition, Addison Wesley, ISBN 0-201-61918-0, 2001.

[CEK99] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, D. Saha, "Key Management for Secure Internet Multicast using Boolean Function Minimization Techniques", *IEEE Infocom '99*, vol. 2, pp 689-698, March 1999.

[CFN94] B. Chor, A. Fiat, M. Naor, "Tracing traitors", Advances in Cryptography (Crypto'94), LNCS, vol. 839, pp 257-270, 1994.

[CGI+99] R. Canetti, J. Garey, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, "Multicast security: A taxonomy and some efficient constructions", *Infocom '99*, vol. 2, pp 708-716, March 1999.

[CLA+96] G. Caronni, H. Lubich, A. Aziz, T. Markson, R. Skrenta, "SKIP: Securing the Internet", *5th IEEE Int'l Workshop on Enterprise Security (WETICE'06)*, pp 62-67, 1996.

[CMN99] R. Canetti, T. Malkin, K. Nissim, "Efficient Communication-Storage Tradeoffs for Multicast Encryption", *Eurocrypt'99*, LNCS 1592, Springer-verlag, pp 459-474, 1999.

[CRS+00] Y. Chu, S. G. Rao, S. Seshan, H. Zhang, "A Case for End System Multicast", *ACM SIGMETRICS 2000*, pp 1-12, Santa Clara, August 2000.

[CS01] G. Caronni, C. Schuba, "Enabling Hierarchical and Bulk-distribution for Watermarked Content", *The 17th Annual Computer Security Applications Conference*, pp 277-285, New Orleans, December 2001.

[CWS+98] G. Caronni, M. Waldvogel, D. Sun, B. Plattner, "Efficient security for large and dynamic multicast groups", *7th IEEE Int'l Workshop on Enterprise Security (WETICE'98)*, pp 376-383, 1998.

[CY01] R. Chen, W. Yeager, "Poblano: A Distributed Trust Model for Peer-to-Peer Networks", JXTA Security Project White Paper, 2001, http://www.jxta.org/docs/trust.pdf. Accessed: October 24, 2003.

[DCP+02] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks", *9th ACM Conference on Computer and Communications Security (CCS'02)*, pp 207-216, November 2002.

[DES] ANSI, "Triple Data Encryption Algorithm Modes of Operation", Standard ANSI X9.52-1998, 1998.

[DH76] W. Diffie, M.E. Hellman, "New directions in cryptography", *IEEE Transactions on Information Theory*, vol. 22, pp 644-654, 1976.

[Dia69] R. Dial, "Algorithm 360: Shortest path forest with topological ordering", *Communications of ACM*, vol. 12, pp 632-633, 1969.

[Dij59] E. W. Dijkstra, ”A note on two problems in connection with graphs”, *Numerische Mathematik 1*, pp 269-271, 1959.

[DMS99] R. Dondeti, S. Mukherjee, A. Samal, “A Dual Encryption Protocol for Scalable Secure Multicasting”, *4th IEEE Symposium on Computers and Communications*, pp 2-8, 1999.

[DSL02] C. Duma, N. Shahmehri, P. Lambrix, “Efficient Storage Requirement for Category-Based Key Management for Multicast”, Internal Technical Report, Linköpings universitet, Sweden, November 2002.

[DSL03a] C. Duma, N. Shahmehri, P. Lambrix, “A Flexible Collusion-Resistant Category-Based Key Management for Multicast”, *18th IFIP Int’l Information Security Conference*, pp 133-144, Athens, Greece, 2003.

[DSL03b] C. Duma, N. Shahmehri, P. Lambrix, “A Hybrid Key Tree Scheme for Multicast to Balance Security and Efficiency Requirements”, *12th IEEE Int’l Workshop on Enterprise Security (WETICE’03)*, pp 208-213, Austria, June 2003.

[Esk02] A. M. Eskicioglu, “Multimedia Security in Group Communications: Recent Progress in Wired and Wireless Networks”, *IASTED Int’l Conference on Communications and Computer Networks*, pp 125-133, November 2002.

[FI03] I. Foster, A. Iamnitchi. ”On death, taxes, and the convergence of peer-to-peer and grid computing”, *2nd Int’l Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, February 2003.

[FJA02] J. Fan, P. Judge, M. H. Ammar, “HySOR: Group Key Management with Collusion-Scalability Tradeoffs Using a Hybrid Structuring of Receivers”, *IEEE 11th Int’l Conference on Computer Communications and Networks (ICCCN 2002)*, pp 196 -201, October 2002.

[FN94] A. Fiat, M. Naor, “Broadcast encryption”, *Crypto ’92,* Springer-Verlag, LNCS 839, pp 257-270, 1994.

[GHK02] R. Gau, *Z.* J. Haas, B. Krishnamachari, “On Multicast Flow Control for Heterogeneous Receivers”, *IEEE Transactions on Networking*, vol. 10, no. 1, February 2002.

[GKK+01] Virgil D. Gligor, H. Khurana, R. K. Koleva, V. G. Bharadwaj, J. S. Baras, “On the Negotiation of Access Control Policies”, *IEEE 2nd Int’l Workshop on Policies for Distributed Systems and Networks*, pp 188-201, 2001.

[GJV+03] S. Gorinsky, S. Jain, H. Vin, Y. Zhang, “Robustness to Inflated Subscription in Multicast Congestion Control”, *SIGCOMM’03*, Germany, August 2003, http://www.acm.org/sigcomm/sigcomm2003/papers/p87-gorinsky.pdf. Accessed: October 24, 2003.

[GPR+02] L. Gharai, C. Perkins, R. Riley, A. Mankin, "Large Scale Video Conferencing: A Digital Amphitheater", *8th Int'l Conference on Distributed Multimedia Systems*, pp 573-580, San Francisco, CA, USA, September 2002.

[Hak71] S. L. Hakimi, "Steiner's problem in graphs and its implications", *Networks*, vol. 1, pp 113-133, 1971.

[Han01] L. Han, *Secure and Scalable E-Service Software Delivery*, Licentiate Thesis no. 906, Linköpings universitet, Sweden, September 2001.

[HCD01] H. Harney, A. Colgrove, P. McDaniel, "Principles of Policy in Secure Groups", *Network and Distributed System Security Symposium*, San Diego, February 2001, http://www.isoc.org/isoc/conferences/ndss/01/2001/papers/mcdaniel.pdf. Accessed: October 24, 2003.

[HCM00] T. Hardjono, B. Cain, I. Monga, "Intra-domain Group Key Management Protocol". Internet Draft, September 2000.

[HS00] L. Han, N. Shahmehri, "Secure Multicast Software Delivery", *9th IEEE Int'l Workshop on Enterprise Security (WET-ICE)*, pp 207-212, June 2000.

[HT00] T. Hardjono and G. Tsudik, "IP Multicast Security: Issues and Directions", *Annales de Telecom*, pg 324-340, July-August 2000.

[Inria] The French Institute for Research in Computer Science and Control, http://www.inria.fr. Accessed: October 24, 2003.

[IP02] M. Al-Ibrahim, J. Pieprzyk, "Authentication of transit flows and k-siblings one-time signature", *6th IFIP Conference on Communication and Multimedia Security*, pp 41-55, September 2002.

[JA02] P. Judge, M. Ammar, "GOTHIC: A Group Access Control Architecture for Secure Multicast and Anycast", *IEEE INFOCOM'02*, June 2002, http://www.ieee-infocom.org/2002/papers/734.pdf. Accessed: October 24, 2003.

[JA03] P. Judge, M. Ammar, "Security Issues and Solutions in Multicast Content Distribution: A Survey", *IEEE Network*, vol. 17, no. 1, pp 24-29, January/February 2003.

[JCE] Java Cryptography Extension (JCE) Reference Guide, http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/JCERefGuide.html. Accessed: October 24, 2003.

[JNM+98] J. Jamison, R. Nicklas, G. Miller, K. Thompson, R. Wilder, L. Cunningham, C. Song, "vBNS: Not Your Father's Internet", *IEEE Spectrum*, vol. 35, no. 7, pp 38-46, July 1998.

[JXTA] Project JXTA, http://www.jxta.org. Accessed: October 24, 2003.

[KBR97] B. Kolman, R. Busby, S. Ross, *Discrete Mathematical Structures*, Third Edition, Prentice Hall of India, 1997.

[KPT00] Y. Kim, A. Perrig, G. Tsudik, "Simple and Fault Tolerant Key Agreement for Dynamic Collaborative Groups", *7th ACM Conference on Computer and Communications Security*, pp 235-244, November 2000.

[KR02] J. Kurose, K. Ross, *Computer Networking: A Top-Down approach Featuring the Internet*, Second Edition, Addison-Wesley, ISBN 0-201-97699-4, 2002.

[Kru98] P.S. Kruus, "A Survey of Multicast Security Issues and Architectures", *21st National Information Systems Security Conference*, Arlington, VA, October 1998.

[LG98] B. N. Levine, JJ Garcia-Luna-Aceves, "A comparison of reliable multicast protocols", *Multimedia Systems*, vol. 6, pp 334–348, Springer-Verlag, 1998.

[LL03] B. Li, J. Liu, "Multirate Video Multicast over the Internet: An Overview", *IEEE Network*, vol. 17, no. 1, pp 24-29, January/February 2003.

[LPB01] M. Y. Li, R. Poovendran, C. Bernstain, "Optimization of Key Storage for Secure Multicast", *Conference on Information Science and Systems 2001*, pp 771-774, March 2001.

[Mer89] R. C. Merkle. "A Certified Digital Signature", *Advances in Cryptography*, pp 218-238, July 1989.

[Mil99] C. K. Miller, *Multimedia Networking and Applications*, Addison-Wesley, ISBN 0-201-30979-3, 1999.

[Mit97] S. Mittra, "Iolus: A Framework for Scalable Secure Multicasting", pp 277-288, *ACM SIGCOMM'97*, 1997.

[MM03] G. Manimaran, P. Mohapatra, "Multicasting: An Enabling Technology", *IEEE Network*, vol. 17, no.1, pp 6-7, January/February 2003.

[MMA] MBone and Multicast Applications, http://multimedia.asti.dost.gov.ph/multicast/applications. Accessed: October 24, 2003.

[MOV97] A. Menezes, P. Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[MP99] R. Molva, A. Pannetrat, "Scalable multicast security in dynamic groups", *6th ACM Conference on Computer and Communication Security (CCS'99)*, pp 101-112, November 1999.

[MS] Metrix Systems, http://www.vision64.com. Accessed: October 24, 2003.

[MS98] D. A. McGrew, A. T. Sherman, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees", Technical Report no. 0755, TIS Labs at Network Associates, May 1998.

[NNL01] D. Naor, M. Naor, J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers", *Advances in Cryptology (CRYPTO 2001)*, Springer-Verlag, LNCS 2139, pp 41-62, 2001.

[NP00] M. Naor, B. Pinkas, "Efficient Trace and Revoke Schemes", *4th Int'l Financial Cryptography*, pp 1-20, 2000.

[Per99] A. Perrig, "Efficient collaborative key management protocols for secure autonomous group communication", *International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC'99)*, pp 192-202, 1999.

[PM02] A. Pannetrat, R. Molva, "Multiple Layer Encryption for Multicast Groups", *6th IFIP Conference on Communication and Multimedia Security*, pp 137-153, September 2002.

[PMM03] R. Di Pietro, L. V. Mancini, A. Mei, "A Time Driven Methodology for Key Dimensioning in Multicast Communication", *IFIP 18th Int'l Conference on Information Security (Sec2003)*, pp 121-132, Greece, May 2003.

[PST01] A. Perrig, D. Song, J. D. Tygar, "ELK, a New Protocol for Efficient Large-Group Key Distribution", *IEEE Security and Privacy Symposium 2001*, pp 247-263, May 2001.

[PSV+01] D. Pendarakis, S. Shi, D. Verma, M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure", *3rd USENIX Symposyum on Internet Technologies*, pp 49-60, San Francisco, March 2001.

[RFC1075] D. Waitzman, C. Partridge, S. Deering, "Distance Vector Multicast Routing Protocol", RFC 1075, 1988.

[RFC1112] S. Deering, "Host Extensions for IP Multicasting", RFC 1112, 1989.

[RFC1320] R. Rivest, "The MD5 Message-Digest Algorithm", RFC 1320, 1992.

[RFC1760] N. Haller, "The S/KEY One-Time Password System", RFC1760, February 1995.

[RFC1949] A. Ballardie, "Scalable Multicast Key Distribution", RFC 1949, May 1996.

[RFC2094] H. Harney, C. Muckenhirn, Group Key Management Protocol (GKMP), RFC 2093 and RFC 2094, July 1997.

[RFC2236] R. Fenner, "Internet Group Management Protocol, Version 2", RFC 2236, November 1997.

[RFC2362] D. Estrin, D. Farinacci, A. Helmy, and collegues, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification", RFC 2362, 1998.

[RFC2365] D. Meyer, "Administratively Scoped IP Multicast", RFC 2365, July 1998.

[RFC2375] R. Hinden, S. Deering, "IPv6 Multicast Address Assignments", RFC 2375, July 1998.

[RFC2409] D. Harkins, D.Carrel, "The internet key exchange (IKE)", RFC 2409, 1998.

[RFC2627] D. Wallner, E. Harder, R. Agee, "Key Management for Multicast: Issues and Architectures", RFC 2627, June 1999.

[RFC3048] B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, M. Luby, "Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer", RFC 3048, January 2001.

[RFC3170] B. Quinn, K. Almeroth, "IP Multicast Applications: Challenges and Solutions", RFC 3170, September 2001.

[RH02] S. Rafaeli, D. Hutchison, "Hydra: A decentralized group key management", *11th IEEE Int'l Workshop on Enterprise Security (WETICE'02)*, pp 62-67, June 2002.

[RM99] S. Ratnasamy, S. McCanne, "Inference of Multicast Routing Trees and Bottleneck Bandwidths using End-to-End Measurements", *Infocom '99*, vol. 1, pp 353-360, March 1999.

[SA01] K. Sarac, K. C. Almeroth, "Supporting Multicast Deployment Efforts: A Survey of Tools for Multicast Monitoring", *Journal of High Speed Networking - Special Issue on Management of Multimedia Networking*, vol. 9, no. 3/4, pp 191-211, March 2001.

[Sam98] M. Sampson, "OmniCast leads the distribution troops", *Network Computing*, vol. 9, no. 15, pp 36-38, August 1998.

[SHA-1] FIPS PUB 180-1, "Secure Hash Standard", *National Institute of Standards and Technology*, 1995.

[Shi98] C. Shields, *Secure Hierarchical Multicast Routing and Multicast Internet Anonymity*, PhD thesis, University of California, Santa Cruz, June 1998.

[SKJ02] S. Setia, S. Koussih, S. Jajodia, "Kronos: A Scalable Group Re-Keying Approach for Secure Multicast", *IEEE Symposium on Security and Privacy*, pp 215-228, Oakland CA, May 2000.

[SM02] A. Striegel, G. Manimaran, "A Survey of QoS Multicasting Issues", *IEEE Communications*, vol. 40, no. 6, pp 82-87, 2002.

[SMF+02] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, D. Dean, "Self-Healing Key Distribution with Revocation", *2002 IEEE Symposium on Security and Privacy*, pp 241-257, May 2002.

[SRL96] K. Savetz, N. Randall, Y. Lepage, *MBONE: Multicasting Tomorrow's Internet*, IDG Books Worldwide, Inc., ISBN 1-56884-723-8, 1996.

[SSL] SSL 3.0 Specification, Netscape, http://wp.netscape.com/eng/ssl3. Accessed: October 24, 2003.

[STW98] M. Steiner, G. Tsudik, M. Waidner, "Cliques: A New Approach to Group Key Agreement", *18th Int'l Conference on Distributed Computing Systems (ICDCS'98)*, pp 380-387, 1998.

[SZJ02] S. Setia, S. Zhu, S. Jajodia, "A Scalable and Reliable Key Distribution Protocol for Multicast Group Rekeying", Technical Report, ISE-TR-02-02, 2002.

[SWY01] K. E. Seamons, M. Winslett, T. Yu, "Limiting the disclosure of access control policies during automated trust negotiation", *Network and Distributed Systems Security Symposium*, 2001, http://www.isoc.org/isoc/conferences/ndss/01/2001/papers/seamons.pdf. Accessed: October 24, 2003.

[Var02] U. Varshney, "Multicast over Wireless Networks", *Communications of the ACM*, vol. 45, no. 12, pp 31-37, December 2002.

[Wan80] M. Wand, *Induction, Recursion, and Programming*, Elsevier North Holland, 1980.

[WCS+99] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, B Plattner, "The VersaKey Framework: Versatile Group Key Management", *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp 1614-1631, August 1999.

[WE93] L. Wei, D. Estrin, "A comparison of multicast trees and algorithms", TR-USC-CD-93-560, University of California, September 1993.

[WGL98] C. K. Wong, M. Gouda, S. S. Lam, "Secure Group Communications Using Key Graphs", *ACM SIGCOMM '98*, pp 16-30, 1998.

[WL00] C. Wong, S. Lam, "Keystone: A group Key Management Service", *Int'l Conference on Telecommunications (ICT)*, May 2000.

[WL02] W. H. Winsborough, N. Li, "Towards practical automated trust negotiation", *IEEE 3rd Int'l Workshop on Policies for Distributed Systems and Networks*, pp 92-103, 2002.

[WZ01] R. Wittmann, M. Zitterbart, *Multicast Communication: Protocols and Applications*, Morgan Kaufmann, ISBN 1-55860-645-9, 2001.

[X509] Comite Consultatif Internationale de Telegraphie et Telephonie (CCITT), "X.509 The Directory – Authentication Framework", *Blue Book*, VIII.8, Geneva, 1988, https://ecs.itu.ch/itudoc/itu-t/rec/x/x500up/x509_27505.html. Accessed: October 24, 2003.

[YL00] Y. R. Yang, S. S. Lam, "A Secure Group Key Management Protocol Communication Lower Bound", the University of Texas at Austin, Department of Computer Sciences, Technical Report TR-00-24, September 2000.

[YLZ+01] Y. Yang, X. Li, X. Zhang, S. Lam, "Reliable group rekeying: Design and Performance Analysis". *ACM SIGCOMM 2001*, pp 27-38, San Diego, CA, USA, August 2001.

[Zim95] P. R. Zimmermann, *The Official PGP Users Guide*, MIT Press, 1995.

[ZSJ03] S. Zhu, S. Setia, S. Jajodia, "Adding Reliable and Self-Healing Key Distribution to the Subset Difference Group Rekeying Method for Secure Multicast", George Mason University Technical Report ISE-TR-03-02, April 2003.

List of Tables

List of Figures

Department of Computer and Information Science
Linköpings universitet

Linköping Studies in Science and Technology
Faculty of Arts and Sciences - Licentiate Theses

No 17 **Vojin Plavsic:** Interleaved Processing of Non-Numerical Data Stored on a Cyclic Memory. (Available at: FOA, Box 1165, S-581 11 Linköping, Sweden. FOA Report B30062E)

No 28 **Arne Jönsson, Mikael Patel:** An Interactive Flowcharting Technique for Communicating and Realizing Algorithms, 1984.

No 29 **Johnny Eckerland:** Retargeting of an Incremental Code Generator, 1984.

No 48 **Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching, 1985.

No 52 **Zebo Peng:** Steps Towards the Formalization of Designing VLSI Systems, 1985.

No 60 **Johan Fagerström:** Simulation and Evaluation of Architecture based on Asynchronous Processes, 1985.

No 71 **Jalal Maleki:** ICONStraint, A Dependency Directed Constraint Maintenance System, 1987.

No 72 **Tony Larsson:** On the Specification and Verification of VLSI Systems, 1986.

No 73 **Ola Strömfors:** A Structure Editor for Documents and Programs, 1986.

No 74 **Christos Levcopoulos:** New Results about the Approximation Behavior of the Greedy Triangulation, 1986.

No 104 **Shamsul I. Chowdhury: S**tatistical Expert Systems - a Special Application Area for Knowledge-Based Computer Methodology, 1987.

No 108 **Rober Bilos:** Incremental Scanning and Token-Based Editing, 1987.

No 111 **Hans Block:** SPORT-SORT Sorting Algorithms and Sport Tournaments, 1987.

No 113 **Ralph Rönnquist:** Network and Lattice Based Approaches to the Representation of Knowledge, 1987.

No 118 **Mariam Kamkar, Nahid Shahmehri:** Affect-Chaining in Program Flow Analysis Applied to Queries of Programs, 1987.

No 126 **Dan Strömberg:** Transfer and Distribution of Application Programs, 1987.

No 127 **Kristian Sandahl:** Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems, 1987.

No 139 **Christer Bäckström:** Reasoning about Interdependent Actions, 1988.

No 140 **Mats Wirén:** On Control Strategies and Incrementality in Unification-Based Chart Parsing, 1988.

No 146 **Johan Hultman:** A Software System for Defining and Controlling Actions in a Mechanical System, 1988.

No 150 **Tim Hansen:** Diagnosing Faults using Knowledge about Malfunctioning Behavior, 1988.

No 165 **Jonas Löwgren:** Supporting Design and Management of Expert System User Interfaces, 1989.

No 166 **Ola Petersson:** On Adaptive Sorting in Sequential and Parallel Models, 1989.

No 174 **Yngve Larsson:** Dynamic Configuration in a Distributed Environment, 1989.

No 177 **Peter Åberg:** Design of a Multiple View Presentation and Interaction Manager, 1989.

No 181 **Henrik Eriksson:** A Study in Domain-Oriented Tool Support for Knowledge Acquisition, 1989.

No 184 **Ivan Rankin:** The Deep Generation of Text in Expert Critiquing Systems, 1989.

No 187 **Simin Nadjm-Tehrani:** Contributions to the Declarative Approach to Debugging Prolog Programs, 1989.

No 189 **Magnus Merkel:** Temporal Information in Natural Language, 1989.

No 196 **Ulf Nilsson:** A Systematic Approach to Abstract Interpretation of Logic Programs, 1989.

No 197 **Staffan Bonnier:** Horn Clause Logic with External Procedures: Towards a Theoretical Framework, 1989.

No 203 **Christer Hansson:** A Prototype System for Logical Reasoning about Time and Action, 1990.

No 212 **Björn Fjellborg:** An Approach to Extraction of Pipeline Structures for VLSI High-Level Synthesis, 1990.

No 230 **Patrick Doherty:** A Three-Valued Approach to Non-Monotonic Reasoning, 1990.

No 237 **Tomas Sokolnicki:** Coaching Partial Plans: An Approach to Knowledge-Based Tutoring, 1990.

No 250 **Lars Strömberg:** Postmortem Debugging of Distributed Systems, 1990.

No 253 **Torbjörn Näslund:** SLDFA-Resolution - Computing Answers for Negative Queries, 1990.

No 260 **Peter D. Holmes:** Using Connectivity Graphs to Support Map-Related Reasoning, 1991.

No 283 **Olof Johansson:** Improving Implementation of Graphical User Interfaces for Object-Oriented Knowledge-Bases, 1991.

No 298 **Rolf G Larsson:** Aktivitetsbaserad kalkylering i ett nytt ekonomisystem, 1991.

No 318 **Lena Strömbäck:** Studies in Extended Unification-Based Formalism for Linguistic Description: An Algorithm for Feature Structures with Disjunction and a Proposal for Flexible Systems, 1992.

No 319 **Mikael Pettersson:** DML-A Language and System for the Generation of Efficient Compilers from Denotational Specification, 1992.

No 326 **Andreas Kågedal:** Logic Programming with External Procedures: an Implementation, 1992.

No 328 **Patrick Lambrix:** Aspects of Version Management of Composite Objects, 1992.

No 333 **Xinli Gu:** Testability Analysis and Improvement in High-Level Synthesis Systems, 1992.

No 335 **Torbjörn Näslund:** On the Role of Evaluations in Iterative Development of Managerial Support Sytems, 1992.

No 348 **Ulf Cederling:** Industrial Software Development - a Case Study, 1992.

No 352 **Magnus Morin:** Predictable Cyclic Computations in Autonomous Systems: A Computational Model and Implementation, 1992.

No 371 **Mehran Noghabai:** Evaluation of Strategic Investments in Information Technology, 1993.

No 378 **Mats Larsson:** A Transformational Approach to Formal Digital System Design, 1993.

No 380 **Johan Ringström:** Compiler Generation for Parallel Languages from Denotational Specifications, 1993.

No 381 **Michael Jansson:** Propagation of Change in an Intelligent Information System, 1993.

No 383 **Jonni Harrius:** An Architecture and a Knowledge Representation Model for Expert Critiquing Systems, 1993.

No 386 **Per Österling:** Symbolic Modelling of the Dynamic Environments of Autonomous Agents, 1993.

No 398 **Johan Boye:** Dependency-based Groudness Analysis of Functional Logic Programs, 1993.

No 402	**Lars Degerstedt:** Tabulated Resolution for Well Founded Semantics, 1993.
No 406	**Anna Moberg:** Satellitkontor - en studie av kommunikationsmönster vid arbete på distans, 1993.
No 414	**Peter Carlsson:** Separation av företagsledning och finansiering - fallstudier av företagsledarutköp ur ett agentteoretiskt perspektiv, 1994.
No 417	**Camilla Sjöström:** Revision och lagreglering - ett historiskt perspektiv, 1994.
No 436	**Cecilia Sjöberg:** Voices in Design: Argumentation in Participatory Development, 1994.
No 437	**Lars Viklund:** Contributions to a High-level Programming Environment for a Scientific Computing, 1994.
No 440	**Peter Loborg:** Error Recovery Support in Manufacturing Control Systems, 1994.
FHS 3/94	**Owen Eriksson:** Informationssystem med verksamhetskvalitet - utvärdering baserat på ett verksamhetsinriktat och samskapande perspektiv, 1994.
FHS 4/94	**Karin Pettersson:** Informationssystemstrukturering, ansvarsfördelning och användarinflytande - En komparativ studie med utgångspunkt i två informationssystemstrategier, 1994.
No 441	**Lars Poignant:** Informationsteknologi och företagsetablering - Effekter på produktivitet och region, 1994.
No 446	**Gustav Fahl:** Object Views of Relational Data in Multidatabase Systems, 1994.
No 450	**Henrik Nilsson:** A Declarative Approach to Debugging for Lazy Functional Languages, 1994.
No 451	**Jonas Lind:** Creditor - Firm Relations: an Interdisciplinary Analysis, 1994.
No 452	**Martin Sköld:** Active Rules based on Object Relational Queries - Efficient Change Monitoring Techniques, 1994.
No 455	**Pär Carlshamre:** A Collaborative Approach to Usability Engineering: Technical Communicators and System Developers in Usability-Oriented Systems Development, 1994.
FHS 5/94	**Stefan Cronholm:** Varför CASE-verktyg i systemutveckling? - En motiv- och konsekvensstudie avseende arbetssätt och arbetsformer, 1994.
No 462	**Mikael Lindvall:** A Study of Traceability in Object-Oriented Systems Development, 1994.
No 463	**Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av Sandviks förvärv av Bahco Verktyg, 1994.
No 464	**Hans Olsén:** Collage Induction: Proving Properties of Logic Programs by Program Synthesis, 1994.
No 469	**Lars Karlsson:** Specification and Synthesis of Plans Using the Features and Fluents Framework, 1995.
No 473	**Ulf Söderman:** On Conceptual Modelling of Mode Switching Systems, 1995.
No 475	**Choong-ho Yi:** Reasoning about Concurrent Actions in the Trajectory Semantics, 1995.
No 476	**Bo Lagerström:** Successiv resultatavräkning av pågående arbeten. - Fallstudier i tre byggföretag, 1995.
No 478	**Peter Jonsson:** Complexity of State-Variable Planning under Structural Restrictions, 1995.
FHS 7/95	**Anders Avdic:** Arbetsintegrerad systemutveckling med kalkylkprogram, 1995.
No 482	**Eva L Ragnemalm:** Towards Student Modelling through Collaborative Dialogue with a Learning Companion, 1995.
No 488	**Eva Toller:** Contributions to Parallel Multiparadigm Languages: Combining Object-Oriented and Rule-Based Programming, 1995.
No 489	**Erik Stoy:** A Petri Net Based Unified Representation for Hardware/Software Co-Design, 1995.
No 497	**Johan Herber:** Environment Support for Building Structured Mathematical Models, 1995.
No 498	**Stefan Svenberg:** Structure-Driven Derivation of Inter-Lingual Functor-Argument Trees for Multi-Lingual Generation, 1995.
No 503	**Hee-Cheol Kim:** Prediction and Postdiction under Uncertainty, 1995.
FHS 8/95	**Dan Fristedt:** Metoder i användning - mot förbättring av systemutveckling genom situationell metodkunskap och metodanalys, 1995.
FHS 9/95	**Malin Bergvall:** Systemförvaltning i praktiken - en kvalitativ studie avseende centrala begrepp, aktiviteter och ansvarsroller, 1995.
No 513	**Joachim Karlsson:** Towards a Strategy for Software Requirements Selection, 1995.
No 517	**Jakob Axelsson:** Schedulability-Driven Partitioning of Heterogeneous Real-Time Systems, 1995.
No 518	**Göran Forslund:** Toward Cooperative Advice-Giving Systems: The Expert Systems Experience, 1995.
No 522	**Jörgen Andersson:** Bilder av småföretagares ekonomistyrning, 1995.
No 538	**Staffan Flodin:** Efficient Management of Object-Oriented Queries with Late Binding, 1996.
No 545	**Vadim Engelson:** An Approach to Automatic Construction of Graphical User Interfaces for Applications in Scientific Computing, 1996.
No 546	**Magnus Werner :** Multidatabase Integration using Polymorphic Queries and Views, 1996.
FiF-a 1/96	**Mikael Lind:** Affärsprocessinriktad förändringsanalys - utveckling och tillämpning av synsätt och metod, 1996.
No 549	**Jonas Hallberg:** High-Level Synthesis under Local Timing Constraints, 1996.
No 550	**Kristina Larsen:** Förutsättningar och begränsningar för arbete på distans - erfarenheter från fyra svenska företag. 1996.
No 557	**Mikael Johansson:** Quality Functions for Requirements Engineering Methods, 1996.
No 558	**Patrik Nordling:** The Simulation of Rolling Bearing Dynamics on Parallel Computers, 1996.
No 561	**Anders Ekman:** Exploration of Polygonal Environments, 1996.
No 563	**Niclas Andersson:** Compilation of Mathematical Models to Parallel Code, 1996.
No 567	**Johan Jenvald:** Simulation and Data Collection in Battle Training, 1996.
No 575	**Niclas Ohlsson:** Software Quality Engineering by Early Identification of Fault-Prone Modules, 1996.
No 576	**Mikael Ericsson:** Commenting Systems as Design Support—A Wizard-of-Oz Study, 1996.
No 587	**Jörgen Lindström:** Chefers användning av kommunikationsteknik, 1996.
No 589	**Esa Falkenroth:** Data Management in Control Applications - A Proposal Based on Active Database Systems, 1996.
No 591	**Niclas Wahllöf:** A Default Extension to Description Logics and its Applications, 1996.
No 595	**Annika Larsson:** Ekonomisk Styrning och Organisatorisk Passion - ett interaktivt perspektiv, 1997.
No 597	**Ling Lin:** A Value-based Indexing Technique for Time Sequences, 1997.

No 598	**Rego Granlund:** C^3Fire - A Microworld Supporting Emergency Management Training, 1997.
No 599	**Peter Ingels:** A Robust Text Processing Technique Applied to Lexical Error Recovery, 1997.
No 607	**Per-Arne Persson:** Toward a Grounded Theory for Support of Command and Control in Military Coalitions, 1997.
No 609	**Jonas S Karlsson:** A Scalable Data Structure for a Parallel Data Server, 1997.
FiF-a 4	**Carita Åbom:** Videomötesteknik i olika affärssituationer - möjligheter och hinder, 1997.
FiF-a 6	**Tommy Wedlund**: Att skapa en företagsanpassad systemutvecklingsmodell - genom rekonstruktion, värdering och vidareutveckling i T50-bolag inom ABB, 1997.
No 615	**Silvia Coradeschi**: A Decision-Mechanism for Reactive and Coordinated Agents, 1997.
No 623	**Jan Ollinen:** Det flexibla kontorets utveckling på Digital - Ett stöd för multiflex? 1997.
No 626	**David Byers:** Towards Estimating Software Testability Using Static Analysis, 1997.
No 627	**Fredrik Eklund:** Declarative Error Diagnosis of GAPLog Programs, 1997.
No 629	**Gunilla Ivefors:** Krigsspel coh Informationsteknik inför en oförutsägbar framtid, 1997**.**
No 631	**Jens-Olof Lindh:** Analysing Traffic Safety from a Case-Based Reasoning Perspective, 1997
No 639	**Jukka Mäki-Turja:**. Smalltalk - a suitable Real-Time Language, 1997.
No 640	**Juha Takkinen**: CAFE: Towards a Conceptual Model for Information Management in Electronic Mail, 1997.
No 643	**Man Lin**: Formal Analysis of Reactive Rule-based Programs, 1997.
No 653	**Mats Gustafsson**: Bringing Role-Based Access Control to Distributed Systems, 1997.
FiF-a 13	**Boris Karlsson:** Metodanalys för förståelse och utveckling av systemutvecklingsverksamhet. Analys och värdering av systemutvecklingsmodeller och dess användning, 1997.
No 674	**Marcus Bjäreland:** Two Aspects of Automating Logics of Action and Change - Regression and Tractability, 1998.
No 676	**Jan Håkegård**: Hiera rchical Test Architecture and Board-Level Test Controller Synthesis, 1998.
No 668	**Per-Ove Zetterlund**: Normering av svensk redovisning - En studie av tillkomsten av Redovisningsrådets rekommendation om koncernredovisning (RR01:91), 1998.
No 675	**Jimmy Tjäder**: Projektledaren & planen - en studie av projektledning i tre installations- och systemutvecklingsprojekt, 1998.
FiF-a 14	**Ulf Melin**: Informationssystem vid ökad affärs- och processorientering - egenskaper, strategier och utveckling, 1998.
No 695	**Tim Heyer**: COMPASS: Introduction of Formal Methods in Code Development and Inspection, 1998.
No 700	**Patrik Hägglund:** Programming Languages for Computer Algebra, 1998.
FiF-a 16	**Marie-Therese Christiansson:** Inter-organistorisk verksamhetsutveckling - metoder som stöd vid utveckling av partnerskap och informationssystem, 1998.
No 712	**Christina Wennestam:** Information om immateriella resurser. Investeringar i forskning och utveckling samt i personal inom skogsindustrin, 1998.
No 719	**Joakim Gustafsson:** Extending Temporal Action Logic for Ramification and Concurrency, 1998.
No 723	**Henrik André-Jönsson:** Indexing time-series data using text indexing methods, 1999.
No 725	**Erik Larsson:** High-Level Testability Analysis and Enhancement Techniques, 1998.
No 730	**Carl-Johan Westin:** Informationsförsörjning: en fråga om ansvar - aktiviteter och uppdrag i fem stora svenska organisationers operativa informationsförsörjning, 1998.
No 731	**Åse Jansson:** Miljöhänsyn - en del i företags styrning, 1998.
No 733	**Thomas Padron-McCarthy:** Performance-Polymorphic Declarative Queries, 1998.
No 734	**Anders Bäckström:** Värdeskapande kreditgivning - Kreditriskhantering ur ett agentteoretiskt perspektiv, 1998.
FiF-a 21	**Ulf Seigerroth:** Integration av förändringsmetoder - en modell för välgrundad metodintegration, 1999.
FiF-a 22	**Fredrik Öberg:** Object-Oriented Frameworks - A New Strategy for Case Tool Development, 1998.
No 737	**Jonas Mellin:** Predictable Event Monitoring, 1998.
No 738	**Joakim Eriksson:** Specifying and Managing Rules in an Active Real-Time Database System, 1998.
FiF-a 25	**Bengt E W Andersson:** Samverkande informationssystem mellan aktörer i offentliga åtaganden - En teori om aktörsarenor i samverkan om utbyte av information, 1998.
No 742	**Pawel Pietrzak:** Static Incorrectness Diagnosis of CLP (FD), 1999.
No 748	**Tobias Ritzau:** Real-Time Reference Counting in RT-Java, 1999.
No 751	**Anders Ferntoft:** Elektronisk affärskommunikation - kontaktkostnader och kontaktprocesser mellan kunder och leverantörer på producentmarknader,1999.
No 752	**Jo Skåmedal:** Arbete på distans och arbetsformens påverkan på resor och resmönster, 1999.
No 753	**Johan Alvehus:** Mötets metaforer. En studie av berättelser om möten, 1999.
No 754	**Magnus Lindahl:** Bankens villkor i låneavtal vid kreditgivning till högt belånade företagsförvärv: En studie ur ett agentteoretiskt perspektiv, 2000.
No 766	**Martin V. Howard:** Designing dynamic visualizations of temporal data, 1999.
No 769	**Jesper Andersson:** Towards Reactive Software Architectures, 1999.
No 775	**Anders Henriksson:** Unique kernel diagnosis, 1999.
FiF-a 30	**Pär J. Ågerfalk:** Pragmatization of Information Systems - A Theoretical and Methodological Outline, 1999.
No 787	**Charlotte Björkegren:** Learning for the next project - Bearers and barriers in knowledge transfer within an organisation, 1999.
No 788	**Håkan Nilsson:** Informationsteknik som drivkraft i granskningsprocessen - En studie av fyra revisionsbyråer, 2000.
No 790	**Erik Berglund:** Use-Oriented Documentation in Software Development, 1999.
No 791	**Klas Gäre:** Verksamhetsförändringar i samband med IS-införande, 1999.
No 800	**Anders Subotic:** Software Quality Inspection, 1999.
No 807	**Svein Bergum**: Managerial communication in telework, 2000.

No 809 **Flavius Gruian:** Energy-Aware Design of Digital Systems, 2000.

FiF-a 32 **Karin Hedström:** Kunskapsanvändning och kunskapsutveckling hos verksamhetskonsulter - Erfarenheter från ett FOU-samarbete, 2000.

No 808 **Linda Askenäs:** Affärssystemet - En studie om teknikens aktiva och passiva roll i en organisation, 2000.

No 820 **Jean Paul Meynard:** Control of industrial robots through high-level task programming, 2000.

No 823 **Lars Hult:** Publika Gränsytor - ett designexempel, 2000.

No 832 **Paul Pop:** Scheduling and Communication Synthesis for Distributed Real-Time Systems, 2000.

FiF-a 34 **Göran Hultgren:** Nätverksinriktad Förändringsanalys - perspektiv och metoder som stöd för förståelse och utveckling av affärsrelationer och informationssystem, 2000.

No 842 **Magnus Kald:** The role of management control systems in strategic business units, 2000.

No 844 **Mikael Cäker:** Vad kostar kunden? Modeller för intern redovisning, 2000.

FiF-a 37 **Ewa Braf**: Organisationers kunskapsverksamheter - en kritisk studie av ”knowledge management”, 2000.

FiF-a 40 **Henrik Lindberg:** Webbaserade affärsprocesser - Möjligheter och begränsningar, 2000.

FiF-a 41 **Benneth Christiansson:** Att komponentbasera informationssystem - Vad säger teori och praktik?, 2000.

No. 854 **Ola Pettersson:** Deliberation in a Mobile Robot, 2000.

No 863 **Dan Lawesson**: Towards Behavioral Model Fault Isolation for Object Oriented Control Systems, 2000.

No 881 **Johan Moe:** Execution Tracing of Large Distributed Systems, 2001.

No 882 **Yuxiao Zhao:** XML-based Frameworks for Internet Commerce and an Implementation of B2B e-procurement, 2001.

No 890 **Annika Flycht-Eriksson:** Domain Knowledge Management inInformation-providing Dialogue systems, 2001.

Fif-a 47 **Per-Arne Segerkvis**t: Webbaserade imaginära organisationers samverkansformer, 2001.

No 894 **Stefan Svarén:** Styrning av investeringar i divisionaliserade företag - Ett koncernperspektiv, 2001.

No 906 **Lin Han**: Secure and Scalable E-Service Software Delivery, 2001.

No 917 **Emma Hansson:** Optionsprogram för anställda - en studie av svenska börsföretag, 2001.

No 916 **Susanne Odar:** IT som stöd för strategiska beslut, en studie av datorimplementerade modeller av verksamhet som stöd för beslut om anskaffning av JAS 1982, 2002.

Fif-a-49 **Stefan Holgersson:** IT-system och filtrering av verksamhetskunskap - kvalitetsproblem vid analyser och beslutsfattande som bygger på uppgifter hämtade från polisens IT-system, 2001.

Fif-a-51 **Per Oscarsson:**Informationssäkerhet i verksamheter - begrepp och modeller som stöd för förståelse av informationssäkerhet och dess hantering, 2001.

No 919 **Luis Alejandro Cortes:** A Petri Net Based Modeling and Verification Technique for Real-Time Embedded Systems, 2001.

No 915 **Niklas Sandell:** Redovisning i skuggan av en bankkris - Värdering av fastigheter. 2001.

No 931 **Fredrik Elg:** Ett dynamiskt perspektiv på individuella skillnader av heuristisk kompetens, intelligens, mentala modeller, mål och konfidens i kontroll av mikrovärlden Moro, 2002.

No 933 **Peter Aronsson:** Automatic Parallelization of Simulation Code from Equation Based Simulation Languages, 2002.

No 938 **Bourhane Kadmiry**: Fuzzy Control of Unmanned Helicopter, 2002.

No 942 **Patrik Haslum**: Prediction as a Knowledge Representation Problem: A Case Study in Model Design, 2002.

No 956 **Robert Sevenius:** On the instruments of governance - A law & economics study of capital instruments in limited liability companies, 2002.

FiF-a 58 **Johan Petersson:** Lokala elektroniska marknadsplatser - informationssystem för platsbundna affärer, 2002.

No 964 **Peter Bunus:** Debugging and Structural Analysis of Declarative Equation-Based Languages, 2002.

No 973 **Gert Jervan:** High-Level Test Generation and Built-In Self-Test Techniques for Digital Systems, 2002.

No 958 **Fredrika Berglund:** Management Control and Strategy - a Case Study of Pharmaceutical Drug Development, 2002.

Fif-a 61 **Fredrik Karlsson:** Meta-Method for Method Configuration - A Rational Unified Process Case, 2002.

No 985 **Sorin Manolache:** Schedulability Analysis of Real-Time Systems with Stochastic Task Execution Times, 2002.

No 982 **Diana Szentiványi:** Performance and Availability Trade-offs in Fault-Tolerant Middleware, 2002.

No 989 **Iakov Nakhimovski:** Modeling and Simulation of Contacting Flexible Bodies in Multibody Systems, 2002.

No 990 **Levon Saldamli:** PDEModelica - Towards a High-Level Language for Modeling with Partial Differential Equations, 2002.

No 991 **Almut Herzog:** Secure Execution Environment for Java Electronic Services, 2002.

No 999 **Jon Edvardsson:** Contributions to Program- and Specification-based Test Data Generation, 2002

No 1000 **Anders Arpteg:** Adaptive Semi-structured Information Extraction, 2002.

No 1001 **Andrzej Bednarski:** A Dynamic Programming Approach to Optimal Retargetable Code Generation for Irregular Architectures, 2002.

No 988 **Mattias Arvola:** Good to use! : Use quality of multi-user applications in the home, 2003.

FiF-a 62 **Lennart Ljung:** Utveckling av en projektivitetsmodell - om organisationers förmåga att tillämpa projektarbetsformen, 2003.

No 1003 **Pernilla Qvarfordt:** User experience of spoken feedback in multimodal interaction, 2003.

No 1005 **Alexander Siemers:** Visualization of Dynamic Multibody Simulation With Special Reference to Contacts, 2003.

No 1008 **Jens Gustavsson:** Towards Unanticipated Runtime Software Evolution, 2003.

No 1010 **Calin Curescu:** Adaptive QoS-aware Resource Allocation for Wireless Networks, 2003.

No 1015 **Anna Andersson:** Management Information Systems in Process-oriented Healthcare Organisations, 2003.

No 1018 **Björn Johansson:** Feedforward Control in Dynamic Situations, 2003.

No 1022 **Traian Pop:** Scheduling and Optimisation of Heterogeneous Time/Event-Triggered Distributed Embedded Systems, 2003.

FiF-a 65 **Britt-Marie Johansson:** Kundkommunikation på distans - en studie om kommunikationsmediets betydelse i affärstransaktioner, 2003.

No 1024 **Aleksandra Tesanovic:** Towards Aspectual Component-Based Real-Time System Development, 2003.

No 1034 **Arja Vainio-Larsson**: Designing for Use in a Future Context - Five Case Studies in Retrospect, 2003.
No 1033 **Peter Nilsson:** Svenska bankers redovisningsval vid reservering för befarade kreditförluster - En studie vid införandet av nya redovisningsregler, 2003.
Fif-a 69 **Fredrik Ericsson:** Information Technology for Learning and Acquiring of Work Knowledge, 2003.
No 1049 **Marcus Comstedt:** Towards Fine-Grained Binary Composition through Link Time Weaving, 2003.
No 1052 **Åsa Hedenskog:** Increasing the Automation of Radio Network Control, 2003.
No 1054 **Claudiu Duma:** Security and Efficiency Tradeoffs in Multicast Group Key Management, 2003.
.

LINKÖPINGS UNIVERSITET

Avdelning, institution
Division, department

Institutionen för datavetenskap

Department of Computer and Information Science

Datum
Date

2003-10-08

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English
☐ _____

Rapporttyp
Report category

☒ Licentiatavhandling
☐ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport

_____

URL för elektronisk version

ISBN 91-7373-770-4

ISRN LiU-Tek-Lic-2003:53

Serietitel och serienummer
Title of series, numbering

ISSN 0280-7971

Linköping Studies in Science and Technology

Thesis No. 1054

Titel
Title

Security and Efficiency Tradeoffs in Multicast Group Key Management

Författare
Author

Claudiu Duma

Sammanfattning
Abstract

An ever-increasing number of Internet applications, such as content and software distribution, distance learning, multimedia streaming, teleconferencing, and collaborative workspaces, need efficient and secure multicast communication. However, efficiency and security are competing requirements and balancing them to meet the application needs is still an open issue.

In this thesis we study the efficiency versus security requirements tradeoffs in group key management for multicast communication. The efficiency is in terms of minimizing the group rekeying cost and the key storage cost, while security is in terms of achieving backward secrecy, forward secrecy, and resistance to collusion.

We propose two new group key management schemes that balance the efficiency versus resistance to collusion. The first scheme is a flexible category-based scheme, and addresses applications where a user categorization can be done based on the user accessibility to the multicast channel. As shown by the evaluation, this scheme has a low rekeying cost and a low key storage cost for the controller, but, in certain cases, it requires a high key storage cost for the users. In an extension to the basic scheme we alleviate this latter problem.

For applications where the user categorization is not feasible, we devise a cluster-based group key management. In this scheme the resistance to collusion is measured by an integer parameter. The communication and the storage requirements for the controller depend on this parameter too, and they decrease as the resistance to collusion is relaxed. The results of the analytical evaluation show that our scheme allows a fine-tuning of security versus efficiency requirements at runtime, which is not possible with the previous group key management schemes.

Nyckelord
Keywords

Security, Key management, Multicast communication, Security-Efficiency tradeoffs, Collusion resistance, Push-oriented applications