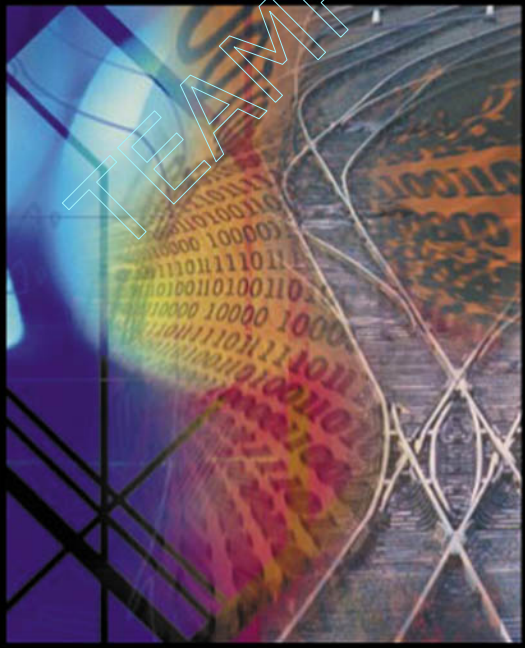
ARTECH HOUSE

COMPUTER SECURITY SERIES

Multicast and Group Security



THOMAS HARDJONO
LAKSHMINATH R. DONDETI

Multicast and Group Security

For quite a long time, computer security was a rather narrow field of study that was populated mainly by theoretical computer scientists, electrical engineers, and applied mathematicians. With the proliferation of open systems in general, and of the Internet and the World Wide Web (WWW) in particular, this situation has changed fundamentally. Today, computer and network practitioners are equally interested in computer security, since they require technologies and solutions that can be used to secure applications related to electronic commerce. Against this background, the field of computer security has become very broad and includes many topics of interest. The aim of this series is to publish state-of-the-art, high standard technical books on topics related to computer security. Further information about the series can be found on the WWW at the following URL:

http://www.esecurity.ch/serieseditor.html

Also, if you'd like to contribute to the series by writing a book about a topic related to computer security, feel free to contact either the Commissioning Editor or the Series Editor at Artech House.

For a listing of recent titles in the *Artech House Computer Security Series*, turn to the back of this book.

Multicast and Group Security

Thomas Hardjono
Lakshminath R. Dondeti

Artech House
Boston • London
www.artechhouse.com

Library of Congress Cataloging-in-Publication Data
Hardjono, Thomas.
Multicast and group security / Thomas Hardjono, Lakshminath R. Dondeti.
p. cm.—(Artech House computer security series)
Includes bibliographical references and index.
ISBN 1-58053-342-6 (alk. paper)
1. Multicasting (Computer networks)—Security measures. 2. Computer networks—Security measures.
I. Dondeti, Lakshminath R. II. Title.
TK5105.887.H37 2003
005.8—dc21 2003048097

British Library Cataloguing in Publication Data
Hardjono, Thomas
Multicast and group security—(Artech House computer security series)
1. Multicasting (Computer networks)—Security measures
I. Title II. Dondeti, Lakshminath R.
005.8

ISBN 1-58053-342-6

Cover design by Christina Stone

685 Canton Street
Norwood, MA 02062

International Standard Book Number: 1-58053-342-6
Library of Congress Catalog Card Number: 2003048097

10 9 8 7 6 5 4 3 2 1

To Joan and Elizabeth

— Thomas

To Sridevi

— Lakshminath

Contents

Foreword . . . . . *xv*

Preface . . . . . *xvii*

Acknowledgments . . . . . *xxi*

1 Introduction 1

1.1 Motivation for multicast security 2
1.2 Multicast content protection 5
1.2.1 Problem area 1: Secure multicast data handling *5*
1.2.2 Problem area 2: Management of keying material *7*
1.2.3 Problem area 3: Multicast security policies *11*
1.3 Infrastructure protection 12
1.4 Applications of secure multicasting 13
1.5 Road map 13
References 14

2 Framework for multicast and group security 17

2.1 The problem scope of multicast security 17
2.2 Fundamental issues 19
2.2.1 Routing infrastructure protection *20*

2.2.2 Controlled access to the multicast distribution tree *20*
2.2.3 Management of keying material *21*
2.3 Transport and applications issues 23
2.3.1 Security of Reliable Multicast protocols *23*
2.3.2 Applications requirements and other issues *24*
2.4 The IETF problem scope for multicast and group security 25
2.4.1 A brief history of multicast security efforts in the IETF *25*
2.4.2 The IETF multicast security Reference Framework *27*
2.4.3 Elements of the Reference Framework *28*
2.5 Three problem areas in the management of keying material 30
2.5.1 Problem area 1: Multicast data handling *31*
2.5.2 Problem area 2: Management of keying material *32*
2.5.3 Problem area 3: Multicast security policies *33*
2.6 The building blocks approach 34
2.6.1 Motivation for building blocks *34*
2.6.2 Functional building blocks *38*
2.7 Summary 42
References 43

3 **Multicast data authentication** **45**

3.1 Issues in multicast data authentication 46
3.1.1 Providing group authentication *48*
3.1.2 Providing source authentication *49*
3.2 Digital signatures for source authentication 50
3.2.1 Block signatures and individual packet authentication *51*
3.3 Hash chaining to authenticate streaming data 55
3.3.1 Graph representation of hash chaining *56*
3.3.2 Efficient multichained stream signature *58*
3.3.3 Augmented chaining *59*
3.3.4 Piggybacking *59*
3.3.5 Discussion on hash chaining for authentication *60*

3.4 MAC-based source authentication of unreliable streams 61
3.4.1 TESLA initialization *63*
3.4.2 MAC-based authentication of packets by the sender *64*
3.4.3 Packet processing at the receivers in TESLA *65*
3.4.4 Enhancements to TESLA *66*
3.4.5 Applicability analysis of TESLA *67*
3.5 IPsec ESP and MESP 68
3.6 Summary 69
References 70

4 Introduction to group key management 73

4.1 A model for group key management 74
4.2 Requirements in group key management 76
4.2.1 Security requirements of unicast key management *76*
4.3 Security requirements of group key management 79
4.4 GSA management 82
4.4.1 The GSA model *83*
4.4.2 Definition of GSA *85*
4.5 Classification of the group key management problem 86
4.6 Summary 88
References 88

5 Architectures and protocols for group key management 91

5.1 Architectural issues and motivations 93
5.2 IKAM 94
5.2.1 Domains, areas, and key distributors *95*
5.2.2 Multicast groups for data and control *96*
5.2.3 Keys: Multicast groups and control multicast groups *98*
5.2.4 Control multicast groups: Address allocation *99*
5.2.5 Arrangement of keys in the domain *100*

5.3 Iolus 103
5.3.1 Hierarchical subgrouping *104*
5.3.2 Subgroup key management *105*
5.3.3 Secure group communication in Iolus *106*
5.3.4 Limitations of Iolus architecture *108*
5.4 Key distribution protocols. 108
5.4.1 GKMP *108*
5.4.2 GSAKMP *112*
5.4.3 GDOI *117*
5.5 Summary 126
References. 126

6 **Group key management algorithms** 129

6.1 Batch and periodic rekeying 131
6.1.1 Trade-offs in batch rekeying *132*
6.2 MARKS 134
6.3 LKH 136
6.3.1 Initializing an LKH *137*
6.3.2 Adding a member to a key tree *137*
6.3.3 Join rekeying in LKH *138*
6.3.4 Efficient join rekeying using LKH+ *140*
6.3.5 Leave rekeying in LKH *140*
6.3.6 Efficient leave rekeying using OFCs *141*
6.4 OFT. 142
6.4.1 Initializing an OFT. *144*
6.4.2 Join rekeying in OFT *145*
6.4.3 Leave rekeying in OFT *146*
6.5 Batch processing of membership changes in key trees 148
6.6 Reliable transport of rekey messages 148
6.6.1 Repeated retransmission of rekey message *148*
6.6.2 FEC for reliability *149*
6.6.3 Weighted key assignment for reliable transport. *149*
6.7 Stateless key revocation algorithms 150
6.7.1 STR for membership revocation *151*
6.7.2 SDR for membership revocation *152*

6.8 Summary 154
References 156

7 Group security policy 159

7.1 Group security policy framework 161
7.2 Classification of group security policy 164
7.2.1 Announcement policy 165
7.2.2 Membership policy 166
7.2.3 Access control or authorization policy 166
7.2.4 Data protection policy 166
7.2.5 Group management delegation policy 167
7.2.6 Key distribution policy 168
7.2.7 Compromise recovery policy 168
7.3 Group security policy specification 169
7.3.1 Ismene policy specification 169
7.3.2 CCNT 170
7.3.3 GSPT 171
7.3.4 Discussion on policy specification languages 173
7.4 Policy negotiation and reconciliation 174
7.4.1 Ismene policy reconciliation 174
7.4.2 Policy negotiation in DCCM 175
7.5 Group security policy enforcement 176
7.5.1 Policy distribution and enforcement in GDOI 176
7.5.2 Antigone policy framework 177
7.5.3 GSAKMP policy distribution and enforcement 178
7.6 Summary 178
References 179

8 Securing multicast routing protocols 181

8.1 The three components of multicast security 182
8.1.1 General types of attacks in multicast routing 184
8.1.2 Multicast routing and security 185

8.2 Overview of multicast routing 186
8.2.1 Classification of multicast routing protocols 188
8.2.2 DVMRP 188
8.2.3 PIM-SM 189
8.2.4 IGMP 191
8.2.5 SSM 193
8.3 Security requirements in unicast and multicast routing 194
8.4 PIM-SM security 197
8.4.1 Background 197
8.4.2 PIM authentication 198
8.4.3 SKMP for PIMv2 199
8.4.4 Revised PIM-SM: Security issues 202
8.4.5 Revised PIM-SM: Possible solutions 204
8.5 MSDP security 205
8.6 IGMP security 207
8.6.1 Membership authorization and authentication issues 209
8.6.2 Membership authorization approaches 210
8.6.3 Message authentication approaches 212
8.6.4 Open issues 213
8.7 Security in other routing protocols 214
8.7.1 Secure CBT multicasting: SMKD 214
8.7.2 KHIP 215
8.8 Summary 216
References 218

9 **Security in Reliable Multicast protocols 223**

9.1 Classification of RM protocols 225
9.1.1 Good throughput strategies 226
9.1.2 Network entity participation and support 228
9.2 Generic security requirements for RM protocols 229
9.3 Security of TRACK protocols 231
9.3.1 Model of TRACK 232

9.3.2 RMTP-II 232
9.3.3 TRAM 237
9.4 Security of NORM protocols 238
9.4.1 Model of NORM 239
9.4.2 PGM 244
9.5 Security of FEC-based protocols 247
9.6 Summary 248
References 249

10 **Applications of multicast and their security** 253

10.1 Stock market data distribution 254
10.1.1 Background 254
10.1.2 Network topology 254
10.1.3 Security requirements and possible approaches 255
10.2 Local area IP Television 257
10.2.1 Background 258
10.2.2 Network topology 259
10.2.3 Security requirements and possible approaches 260
10.3 Nonreal-time multicast distribution 261
10.3.1 MFTP 262
10.3.2 Security requirements of MFTP applications 264
10.3.3 Security solutions for MFTP 264
10.4 SecureGroups project 266
10.4.1 Impact of mobility on group key management 267
10.5 Summary 268
References 268

11 **Conclusion and future work** 271

11.1 IETF multicast security framework 272
11.2 Secure multicast data transmission 272
11.2.1 Group authentication 273
11.2.2 Source authentication 274
11.3 Group key distribution 274
11.3.1 Reliable transport of rekey messages 275
11.3.2 Secure multicast group management 276

11.3.3 Distributed group key management 277
11.3.4 Secure group communication between mobile members in wireless environments 277
11.4 Policy 278
11.5 Infrastructure protection 278
11.6 Future direction and final words 280

Glossary 283

About the Authors 295

Index 297

Foreword

Both multicast and security present interesting technological challenges. Put them together into multicast security, and you have a lot of daunting but interesting problems.

What are some of the challenges of multicast security? The designers of multicast envision its use to distribute content simultaneously to huge numbers of receivers. If only those receivers are allowed to see the content, it must be encrypted. But how can an encryption key be efficiently distributed to so many receivers? Furthermore, a key for encrypting data should be changed periodically, since cryptographers frown on encrypting a lot of data with the same key. And it should perhaps be changed when the membership of the group changes. Suppose group members pay to receive the content (such as premium TV channels). When a member leaves the group, one cannot force the member to forget the key. More subtly, it might be desirable to change the key when a new member joins the group (so they cannot record encrypted content they were not entitled to, and then join for a short time in order to discover the key, enabling them to decrypt content they had not paid for).

Another issue in secure communication is integrity protection and authentication of the data. With a secret key scheme, someone who is verifying the data must know the same secret that was used to create the integrity check. With two-party communication this is not a problem. If Alice is sending something to Bob, with an integrity check created out of a key that only the two of them share, if the integrity check is valid, Bob knows that only Alice or Bob could have created the message. If Bob knows it wasn't him, then it has to be Alice.

However, if the same scheme is used in group communication, where Alice is sending the content to thousands of receivers, then each of those

thousands of receivers would have to know the same secret Alice used in order to generate the integrity check. Which means the content could have originated with any member of the group. One might trust all the group members to ''read'' the data, but you want to cryptographically protect against them being able to generate data and claim it came from Alice.

An alternative is to use public key cryptography, but it would be slow to generate and verify digital signatures on every packet. So the multicast security designers have devised schemes with the per-source authenticity allowed by public key cryptography without the performance penalty.

The authors have spent a significant chunk of their lives nurturing this new field. Thomas Hardjono has been working in the field since 1988—way before the world was ready for it, but a good time for forward thinking. Since then he has been trying to make it a reality. He was cochair of the multicast security group in IETF since it was born in the IRTF and graduated into a working group in IETF. Lakshminath Dondeti chose it, out of all possible topics in computer science, for his Ph.D. dissertation, and has also been active in standardizing it, first in the IRTF group, and now in the IETF multicast security group.

Dr. Radia Perlman
Distinguished Engineer
Sun Microsystems Laboratories
May 2003

Preface

The area of networked group communications is by no means a new field of study. For several years now, researchers and engineers have been studying more efficient ways to harness the potential of Internet protocol (IP)-based networks as the basis for communications in multiparty scenarios. There are many possible approaches to multiparty or group communications, and there are different communications methods and protocols that can be deployed to establish communications within a group. One such method is *IP multicast*—which takes place at the IP network layer within the transmission control protocol/Internet protocol (TCP/IP) model.

Although there have been several books dedicated to IP multicast and other forms of group communications, none has been dedicated to the topic of security in IP multicast networks and the applications that use them. This book attempts to fill that gap, and provide a snapshot of the current state of the art in the network industry.

In many ways, the area of multicast security is still in its infancy. Although the concept of IP multicast can be traced back to the earlier works of Deering in the late 1980s, serious attention was given to IP multicast—and thus to its security issues—only in the late 1990s. At this time, various players in the industry, notably the content industry, saw the potential of IP multicast as a vehicle for delivering data to vast numbers of users.

The industry's interest in IP multicast is reflected in (or resulted in) the establishment of the various multicast-related working groups in the *Internet Engineering Task Force* (IETF). They were seen as a means to speed up the standardization of multicast-related protocols, and therefore the implementation and deployment of IP multicast in the wider community. The promise of broadband access to millions of homes across North America

provided the underlying impetus for maturing these multicast-related protocols, and for getting products out the door.

Much of the material in this book is gathered from efforts being conducted within the IETF—which is the primary standards-setting body for IP-related protocols—and its sister organization, the Internet Research Task Force (IRTF). The first community in the IETF that began addressing multicast security was the Secure Multicast Group (fondly nicknamed *SMuG*), established within the IRTF in early 1998. Since SMuG was established under the IRTF, it functioned as a research group and therefore did not in itself produce standards. However, what SMuG chose to do as a research group was to survey the broader area of group communications security, develop a reference framework, and produce a number of ''near-standards'' documents that could be carried over into a formal working group in the IETF. Indeed, such a working group was established under the IETF in early 2000 in the form of the Multicast Security (MSEC) working group, which was heir to much of the the SMuG research group work.

How to read this book

The contents of this book are grouped according to areas related to multicast and group security. Chapter 1 provides an introduction and outlines three problem areas that will be the focus for the ensuing chapters. These problems areas are defined in the Reference Framework which underlies the work of the SMuG research group in the IRTF and the MSEC working group in the IETF. Chapter 2 delves deeper into this Reference Framework.

Readers interested in the problem of data authentication in multicast will find that Chapter 3 provides introductory material on this topic, as well as discussion on more advanced techniques and algorithms to address the problem.

The problem of key management for groups (group key management) is addressed in Chapters 4, 5, and 6:

- Chapter 4 explains the differences between pair-wise key management and group key management, and explains the security requirements in both cases. It then provides the definition of the Group Security Association (GSA), which extends the Security Association (SA) definition currently understood and deployed in the well-known industry protocols such as Internet key exchange (IKE) and IP security (IPsec).

- Chapter 5 focuses on group key management architectures and protocols, and explains what the terms ''architecture'' and ''protocol'' mean in the context of key management. It goes over two basic group key management architectures; namely, the hierarchic and flat architectures. The chapter then provides an overview of some group key management protocols that have been proposed.
- Chapter 6 focuses on the third aspect of group key management; namely, the algorithms used to manage the cryptographic keys that are used to protect the data (the traffic encryption keys or TEKs), and the keys used to protect the TEK (the key encryption keys or KEKs). The chapter discusses a number of these algorithms, as well as aspects of each.

Security-related policy has always been an interesting as well as contentious topic for many security practitioners. This topic is covered in Chapter 7 with specific reference to multicast and group security. The discussion includes a classification of the various group-oriented policies, and examples of how they are used with specific group key management protocols.

Routing protocol protection is the topic of Chapter 8. In particular, this chapter looks into the issues and requirements for multicast routing, above and beyond the requirements of unicast routing. An overview of a number of popular multicast routing protocols is provided, followed by a discussion on the security issues and possible solutions of two of the most common protocols, namely Protocol Independent Multicast-Sparse Mode (PIM-SM) and IGMP.

Chapter 9 focuses on the issue of security in Reliable Multicast (RM) protocols, which typically execute at the transport layer. A classification of RM protocols is provided to illustrate the differences in approach adopted by the various RM protocols. The chapter then focuses on the tree-based positive acknowledgment (TRACK) and negative acknowledgment–oriented Reliable Multicast (NORM) families of RM protocols, providing a possible security model for each, and some suggested approaches to minimize threats to the protocols.

For readers interested in real-life examples of the use of IP multicast security, Chapter 10 provides a number of applications of multicast, and discusses the security issues relating to each environment.

Finally, Chapter 11 summarizes our discussion on multicast and group security and provides directions for future research.

Acknowledgments

The technologies, ideas, and implementations presented in this book could not have been possible without the hard work and support of the various people active in the area of multicast security, and in the broader IETF community.

We especially thank those whose participation over the years in the multicast security community in the IETF has shaped much of the work presented in this book (in alphabetical order): David Balenson, Mark Baugher, Bob Briscoe, Brad Cain, Ran Canetti, Elisabetta Carrara, Pau-Chen Cheng, Dah Ming Chiu, Andrea Colgrove, Peter Dinsmore, Naganand Doraswamy, Martin Euchner, Eric Harder, Dan Harkins, Hugh Harney, Haixiang He, Paul Judge, Miriam Kadansky, Steve Kent, Amit Kleinmann, Fredrik Lindholm, Doug Maughan, Pat McDaniel, David McGrew, Catherine Meadows, Uri Meth, Inder Monga, Carl Muckenhirn, Mats Näslund, Hilarie Orman, Adrian Perrig, Radha Poovendran, Atul Prakash, Bob Quinn, Pankaj Rohatgi, Debanjan Saha, Gene Tsudik, Brian Weis, and Joe Wesley.

We also thank those in the RMT and routing communities in the IETF who have made significant contributions to the multicast security effort (in alphabetical order): Carsten Borman, Ken Calvert, Steve Deering, Bill Fenner, Brian Haberman, Mark Handley, Roger Kermode, Isidor Kouvelas, Mike Luby, Alison Mankin, Colin Perkins, Radia Perlman, Tom Pusateri, Hal Sandick, Tony Speakman, Lorenzo Vicisano, Liming Wei, Brian Whetten, and Aidan Williams. We apologize to those whose names were inadvertently omitted from these lists. We would like to express our appreciation to Donald Knuth, Leslie Lamport, and countless others who developed the wonderful typesetting system, LaTeX, without which we could not have produced the manuscript in time.

We thank Warwick Ford and Judy Lin (Verisign), and Don Fedyk and Bilel Jamoussi (Nortel Networks) for their support, especially during the latter stages of the manuscript preparation. Special thanks to Rolf Oppliger and Tim Pitts from Artech House for not giving up, and Tiina Ruonamaa, Ruth Harris, Judi Stone, Jessica Nelinder, and Jill Stoodley for their assistance in various stages of the publishing process. We are grateful to the anonymous reviewer(s) for their constructive criticism and suggestions, which helped improve the quality of this book.

Finally, we thank our wives Elizabeth and Sridevi for their love, support, and encouragement during the countless hours spent writing, editing, and reviewing this book, time that otherwise would have been spent with them.

CHAPTER

1

Introduction

Contents

1.1 Motivation for multicast security

1.2 Multicast content protection

1.3 Infrastructure protection

1.4 Applications of secure multicasting

1.5 Road map

Satellite TV distribution, software distribution, stock quote streaming, Web caching, and multimedia conferencing are examples of applications that require one-to-many or many-to-many group communication. Multicast enables efficient group communication by allowing the sender to transmit a single copy of data, with network elements such as routers and switches making copies as necessary for the receivers. Thus multicast reduces the computational load at the sender, as well as the number of copies of data on the network.

Unfortunately, despite the vast amount of research and development of multicast protocols in the past decade, deployment of multicast applications has been slow. While some attribute this to no ''killer applications,'' the major factor is, in fact, that multicast services lack support for traffic management, accounting and billing, reliability, and security.

We identify multicast security as one of the important problems to solve for the successful deployment of group communication applications. For example, investors would like a guarantee that the stock quotes being delivered via multicast are indeed authentic. Similarly, providers would like to limit content distribution to subscribers who paid for the service. Finally, another aspect of security, confidentiality, is a requirement of applications such as conferencing, as well as corporate and military communications via the Internet. In summary, popular applications of multicast require data integrity, access control, and privacy.

IP multicast scales well due its open model. Receivers can join and senders can transmit data to a multicast group, without

any interaction with a centralized entity. However, the same open model makes it difficult to support multicast access control. For privacy, the group members need to have a common key, which may require interaction with a centralized entity. Thus the challenge in front of us is to secure multicast communications without sacrificing scalability.

There are three distinct problem areas to consider in providing multicast security services. First, senders need to encrypt and authenticate multicast data. For encryption, the group members require a common key among themselves. Furthermore, access control can be enforced by distributing a common key to the group membership, without having to change the IP multicast model. When group membership changes, the common key may need to be rekeyed and distributed to the new set of authorized members. Thus scalable *group key distribution* and rekeying schemes are an important part of a secure multicast solution. Next, members must be able to verify that the data received is indeed sent by an authorized sender. Therefore *data origin authentication* and data encryption constitute one of the problem areas.

In addition, the different multicast applications—ranging from many-to-many interactive communications to one-to-many off-line distribution of data—have varying requirements for end systems, communications, and security. *Group policy* allows the group owner or content provider to specify these requirements as well as expected group behavior due to changes in operational environment.

In addition to content protection, we identify multicast infrastructure protection as another important requirement, considering the impact of a denial of service (DoS) attack on the mass distribution service model of multicast. Specifically, multicast routing protocols, Reliable Multicast protocols, and the Internet group management protocol (IGMP) need integrity protection of the control messages for correct operation. Without integrity protection, unauthorized members might flood a multicast tree or illegally pull unnecessary traffic, resulting in denial of service to authorized members. Therefore we need to address control message authentication as well as host or router authorization.

The remainder of this chapter discusses multicast content and infrastructure protection further, and refers to chapters in the book that cover the subtopics therein.

1.1 Motivation for multicast security

Multicasting is an efficient solution for group communication on the Internet. Instead of sending a separate copy of data per receiver, a sender can

send just a single copy, and the multicast routers in the network make copies and forward packets appropriately to all the receivers. Thus multicasting utilizes network resources such as bandwidth and buffer space efficiently, and reduces load at the sender(s) as well as the transit routers.

IP multicast is designed to be massively scalable. Receivers do not directly contact the sender(s) to express their interest in receiving data. Instead, a receiver sends a message to the first hop multicast router that it is interested in receiving data sent to a particular multicast group. Specifically, receivers use the Internet Group Management Protocol [1] to express their interest in receiving data sent to a given group. The multicast forwarding tree itself is established using a routing protocol such as protocol independent multicast (PIM) [2, 3].

While the advantages of multicasting are clear, there are several obstacles for widespread deployment [4]. The popular applications of the Internet are based on unicast, and are dependent on the reliability and sometimes security of the transmission. Most applications use hypertext transfer protocol (HTTP), file transfer protocol (FTP) or telnet, which run over TCP for reliability, and most e-commerce applications run over the secure socket layer (SSL). Reliable unicast transmission has long been taken for granted, and most of us look for the gold lock icon on our Web browsers before entering a password, credit card number, or other sensitive information. End-users and application service providers (ASPs) expect reliability and security for multicast communication as well. Of course not all unicast and multicast applications need reliability or security.

IP multicast communications may need to be encrypted even if data confidentiality is not a requirement. Content providers can charge for unicast data transfers rather easily on the Internet. Charging for software downloads, and monthly subscription to digital libraries and on-line magazines, is commonplace on the Internet. The same cannot be said for multicast applications. This is mainly due to the anonymous receiver model of IP multicast. Any receiver can request to receive data, and the sender has no control over group membership. Similarly, anybody can send data to a multicast group.

One expects access control to be enforced at a higher layer, typically by the applications. Consider the alternative of enforcing access control using IGMP. An edge router could check whether a host is a member, before forwarding a (join) request for multicast data. But once the data flows into a shared medium-based local area network (LAN), all hosts on the LAN get access to the data—whether they are members or not, or whether all members paid for the service or just one. Thus encrypting multicast data and distributing the encryption key to the members is the only way to ensure controlled access to

data. In other words, secure multicast enables content providers to enforce access control, and thus be able to charge for multicast data services.

Access control is only one of the motivating factors for securing multicast communications. Applications in general may need privacy, authentication, integrity, and non-repudiation of multicast data. Moreover, these requirements may have different levels of importance for different applications. For example, some applications may need message source authentication only (e.g., stock quote distribution), whereas others may need privacy as well as authentication. In other words, we must be able to provide a selectable level of security in protecting multicast communications.

Mass distribution of data via multicast is of concern to *Internet service providers* (ISPs). Any sender can start sending data to a multicast group and, similarly, any host can ''pull'' unnecessary multicast traffic, thus wasting network resources such as buffer space on routers and bandwidth on the links. These concerns need to be addressed as well. Multicast routing protocols and Reliable Multicast protocols may need integrity protection for their control messages, to fend off adversaries that may modify, delete, or inject control messages, thus causing denial of service. We discuss these threats to multicast infrastructure in more detail later in this chapter.

Thus, it can be seen that multicast security is motivated by enforcement of group access control, confidentiality and authentication of data transmission, and protection of the network infrastructure. Except for group access control, these requirements have been addressed for unicast communications. Unfortunately, solutions designed for one-to-one communications cannot be used directly for group communications. Specifically, multiparty security policy negotiation may not converge, and distributed group key agreement protocols do not scale to large groups. Note that neither security nor any other service should come at the expense of scalability. In addition to scalability, different applications have different security requirements, and one-size-fits-all solutions will not work for many multicast applications.

The number of senders in a group is an important factor in designing a secure group communication protocol. In many applications there is only one sender. *Source-specific multicast* (SSM) is a routing paradigm[1] specifically designed to support one-sender multicast. Examples of single sender applications include pay-per-view broadcasts, stock quote distribution via multicast on the Internet, and Web cache synchronization.

Traditional multicasting, sometimes referred to as the Internet standard multicast (ISM) or any sender multicast (ASM), supports multisender

1. Multicast routing protocols need to be made to be SSM-compliant, (e.g., PIM-SSM).

communication. The presence of multiple senders introduces several new problems in providing secure multicast services. First, different senders may have different policy requirements, which may be conflicting and thus hard to enforce. Second, mechanisms for replay protection are harder to design for the multisender case. Therefore, we limit our discussion to single sender multicast. However, popular applications such as multimedia conferencing involving multiple senders may also need privacy or message integrity for some sessions. Since these applications typically have only a few senders, one possibility is to use few instances of a secured one-to-many multicast.

1.2 Multicast content protection

The IRTF SMuG Research Group and IETF MSEC Working Group identified three problem areas in providing secure group communications: secure multicast data handling, management of keying material, and multicast security policies. Chapter 2 contains a detailed description of the problem areas and building blocks. In the rest of this section, we define the problem areas, briefly explore the solution space, and discuss application-specific requirements.

1.2.1 Problem area 1: Secure multicast data handling

In this section, we address the problem of secure multicast data transmission. More precisely, we discuss data transforms for multicast data secrecy and integrity protection. For data secrecy, the sender needs to encrypt data with a secret key which is known to the group members: that is, hosts that are authorized to receive multicast data. Scalable distribution and rekeying of a group key is a complex problem and is designated as a problem area in itself.

IPsec encapsulating security payload (ESP) [5] transforms for secrecy are applicable to private multicast communication as well. The only caveat is that the replay protection mechanism of ESP works only for single sender multicast communication. ESP also supports message authentication code (MAC)-based integrity protection for unicast communication. In two party communication, MAC-based authentication is sufficient for a receiver to determine whether a packet originated at the sender. Unfortunately, that is not the case in multiparty communication.

Consider two communicating peers, Alice and Bob, who each hold a secret key for message authentication. Alice uses the key to compute a message authentication code (MAC, e.g., cipher block chaining (CBC) MAC, or hash-based MAC (HMAC) [6]) of the message, and sends the message

along with the MAC to the receiver. Bob repeats the procedure to compute the MAC, and compares it with the received MAC. If the MACs are identical, Bob knows that the message has not been modified en route. He also knows that since he has not sent the message, Alice must have sent it, assuming the authentication key has not been compromised.

We can use MACs for authenticating group communications following a similar procedure as above, but with a reduced level of integrity protection. Consider a group, consisting of Alice, Bob, and Cindy, holding an authentication key. Alice might use a MAC to authenticate a message sent to Bob and Cindy. Bob (or Cindy), however, does not know whether the message has been sent or last modified by Alice or Cindy (or Bob). In general, members of a group can verify only that nonmembers, that is, people who do not hold the group authentication key, have not changed the data in transit.

Group authentication [7] is the property that guarantees only that a message was sent (last modified) by a member of the group. Since a MAC can be used for group authentication, it is rather inexpensive to authenticate even streaming data in real time. For group authentication, the sender needs to establish a common key with the group members. The problem of key distribution is addressed in the next section, and IPsec ESP can be used to carry group authenticated multicast data.

In most applications, receivers must be able to establish the source of the data, at least for themselves. In other words, we need *data source authentication*. A stronger version of the above property, referred to as *non-repudiation*, enables a receiver to prove the origin of data to any impartial third party.

However, source authentication of multicast data is a difficult problem. The simplest solution is to digitally sign each packet. But signing each packet is computationally expensive, and introduces excessive per packet communication overhead. Several solutions have been documented that amortize the cost of digital signatures over multiple packets. Chapter 3 provides a detailed description of the state-of-the-art in stream source authentication techniques.

Unfortunately, ESP cannot be used to carry source authenticated multicast data. A couple of variants of ESP called multicast ESP (MESP) [8] and application layer MESP (AMESP) have been proposed at the IETF to accommodate source authentication schemes for multicasting. MESP, similar to IPsec ESP, operates in the network layer, whereas AMESP is the application layer counterpart. Implementing IPsec ESP or MESP requires kernel-level modifications—which are not possible in some cases, and may result in delayed deployment. AMESP is the transform to use in such cases.

Application requirements

Application requirements greatly influence the solution space for data origin authentication. First, an application may require non-repudiation or source authentication or just group authentication. Next, data transmission may be reliable or lossy. Furthermore, the sender or the receivers may have limited buffer space or computational power (e.g., on mobile devices), or have heterogeneous capacities. Receivers may also be at vastly different distances from the sender. Finally, the application may involve bulk data transfer(s) or streaming. Clearly, a one-size-fits-all solution is not applicable for source authentication of multicast data.

1.2.2 Problem area 2: Management of keying material

Group access control, privacy, and group authentication of multicast data require that a common key be distributed to the current members of the secure group. We use a logical entity called *group controller and key server* (GCKS) to provide access control and key distribution services. A GCKS represents both the entity and functions relating to the issuance and management of cryptographic keys used by a multicast group, and conducts user authentication and authorization checks on each candidate member of the multicast group.

Unicast security negotiation protocols such as IKE [9] result in a separate key per instance, and cannot directly be used for group communications. Instead, a centralized entity such as the GCKS needs to download group key(s) separately to each member via a secure channel.[2]

Member registration

Each member contacts the GCKS to *register* [10] and join the group. After mutual authentication, the GCKS verifies the host's membership, establishes a secure channel, and downloads group keys and policy to the member. Registration is a one-to-one exchange between the GCKS (or one of its authorized representatives) and each member. This one-to-one secure exchange requires similar protections as in IKE or SSL, such as protection from man-in-the-middle, replay and denial of service attacks, connection hijacking, and so forth.

2. Distributed group key agreement schemes have been documented, but they are inefficient for groups of hundreds of members or more.

Scalable registration/initialization of large groups. Since registration is a one-to-one exchange, employing a single point of contact for all members is not efficient. Fortunately, there are several ways to expedite the process. First, the functionality of GCKS registration could be distributed over several entities. Second, members may register at their convenience, thereby avoiding large number of registration requests at the same time. This process is similar to purchasing tickets for sporting events and theater performances at Ticket Master outlets and authorized travel agents.

Group rekeying

Since encryption keys have lifetimes, the GCKS must rekey the group key before it expires. Otherwise, most if not all members may send a request to the GCKS for the new key simultaneously, resulting in rekey request implosion in large groups. Furthermore, consider the problem of enforcing access control in a dynamic group, that is, a group where membership changes frequently. The GCKS may need to rekey the group and distribute the new group key to the current members each time membership changes.

Forward and backward access control. Consider group key distribution to a large and dynamic group. In most applications, while some members join at the beginning of the session and leave at the end, others may join and leave any time during the session. In other words, some members may be allowed to participate in a secure group for only a limited duration. Consider also that a host might be recording multicast data sent before it is authorized to join a group. Therefore, the GCKS needs to change the group key and distribute the new key to all members. Otherwise, the joining host can decrypt data sent before it became a member of the group. We refer to this property as backward access control. Similarly, the GCKS needs to change the group key when a member leaves, so as not to allow the departing member to decrypt future group communications. This property is known as forward access control. In summary, to ensure that only current authorized members can decrypt group data, backward and forward access control must be enforced.

Rekey messages can be sent via unicast or by multicast for efficient distribution. Similar to registration messages, rekey messages must also be protected from replay attacks, and must be signed by the GCKS for authenticity.

Impact of group membership dynamics. The size and dynamics of the group have a significant impact on the enforcement of forward and backward

access control. Consider a naïve approach to group key management. The sender or the GCKS shares a separate secret key with each member. When a member leaves, the manager must send the new group key encrypted separately with each of the remaining members' secret keys. Since the computational overhead at the sender and the communication overhead of this scheme grow with group size, it is inefficient for large and highly dynamic groups.

Group security association

For secure unicast, two communicating peers negotiate security parameters to establish an Internet security association and key management protocol (ISAKMP) SA, which protects the negotiation of an IPsec SA for secure data transmission [11]. Following a similar model, the group security association (GSA) [12] is being standardized at the IETF, and has three parts. The first is the registration SA which protects the key download during the registration protocol. The key download consists of a rekey SA and a data security SA. The rekey SA protects updates to the current rekey SA or a data security SA. The data security SA itself protects multicast data transmission. Examples of a data security SA include IPsec ESP, MESP, and AMESP.

Group key distribution architectures, protocols, and algorithms

We classify the group key distribution literature into architectures, protocols, and algorithms for group key management. Group key distribution *architectures* such as Iolus [13] and Internet keying architecture for multicast (IKAM) [14] use hierarchical subgrouping for efficient group management. They divide members into subgroups, designate members or third-party agents as subgroup managers (SGMs), and delegate group key management tasks to the SGMs. This topic is covered further in Chapter 5.

The term *protocols* is used to describe the set of procedures, message exchanges, and message payloads that govern the behavior of the entities involved in supporting a secure group (e.g., servers), and those participating in a group (e.g., hosts). Group domain of interpretation (GDOI) [15] and group security association key management protocol (GSAKMP) [16] are solutions that fall into this category. This topic is addressed in Chapter 5.

Architectures and protocols benefit from each other. For example, Iolus or IKAM can use GDOI for registration and rekeying within a subgroup. Similarly GSAKMP allows some members to be designated as subgroup managers for scalability. Both architectures and protocols can benefit from the group key management algorithms, described below, for efficient group rekeying.

Group key management *algorithms* typically use logical subgrouping for efficient key distribution and rekeying. (Contrast this with subgrouping by group key management architectures described earlier.) Each logical subgroup has an associated key encryption key (KEK) used to encrypt other KEKs or the group key. We identify two classes of group key management algorithms based on the interdependency of rekey messages. In the first, based on *logical key hierarchies* (LKH) or key trees [17], each member holds a subset of the KEKs, and the GCKS changes those KEKs and the group key when the member joins or leaves. At most, each KEK must be encrypted with as many keys as the degree of the key tree. Thus rekeying using logical key hierarchies requires fewer (logarithmic in number of members) messages, and fewer key computations at the GCKS. In the second class of algorithms, the GCKS divides the authorized members into predefined logical subsets, and sends the group key encrypted with the subset keys. This interesting subject is central to efficient group key management, and is discussed in depth in Chapter 6.

Reliable transport of rekey messages. Rekey messages are typically sent via multicast for efficiency. It is the responsibility of the GCKS to ensure that all members have the current data security and rekey SAs. Otherwise, authorized members may be inadvertently excluded from being able to decrypt group communications. Therefore, the GCKS must use a reliable transport mechanism to send rekey messages. Reliable multicasting is a hard problem, but there are several documented solutions. We discuss reliable transport of rekey messages in this section.

Rekey messages are typically short (for a single membership change in large groups and for small groups in general), which makes it easy to design a reliable delivery protocol. On the other hand, the security requirements add an additional dimension to the problem. There are also some special cases where membership changes are processed in a batch, increasing their size. Finally, among all the KEKs sent in a rekey message, as many as half the members need only a single KEK. We need to take advantage of these properties in designing a rekey message(s) and a protocol for their reliable delivery. Three categories of solutions have been proposed:

1. Because in many cases rekey messages are small (fitting in one or two IP packets), the GCKS may repeatedly retransmit rekey messages.
2. The GCKS may use an existing Reliable Multicast protocol or infrastructure.

3. We may use forward error correction to encode rekey packets, using feedback negative acknowledgements or (NACKs) from members to build the next round of rekey message [18]. Note however that feedback-based reliable delivery may result in implosion of feedback messages at the GCKS.

Application requirements

Application requirements greatly influence the design choices of a solution for group key distribution. In some cases, such as in a pure pay-per-view (PPV) application, all of the SA information needed for the session may be distributed at the time of registration or initialization of a session. Thus there is no rekeying due to membership changes, which obviates the need for an efficient group key management algorithm. Rekey SA may not be necessary for group key management schemes that rely solely on point-to-point communications (e.g., for small groups). Strict enforcement of forward and backward access control may not be necessary in some applications. More precisely, membership changes may be processed periodically or in a batch [19], even if the membership changes happen at different times. Therefore a group key management solution must offer selectable/configurable levels of security.

1.2.3 Problem area 3: Multicast security policies

Multicast security policies provide the rules of operation for the other two problem areas: management of keying material and multicast data handling. We consider two different types of groups, namely, small interactive groups and large groups with one or a few senders. In single-sender groups, the sender is either the content owner or its representative. The content owner typically specifies who can receive the data and what amount of protection is needed, and designates a GCKS and a sender. The GCKS is responsible for both distribution and enforcement of policy. The sender is also partially responsible for enforcing policy. Interactive groups sometimes have a moderator that can be considered as the *group owner*. Policy in small groups may be negotiated [20], but negotiation does not converge in large groups. Alternatively, the group owner may distribute and enforce policy.

Content owners specify only a high-level policy which must then be translated into more precise rules so that the policy can be enforced with available security mechanisms. Policy languages such as Ismene [21], cyptographic context negotiation template (CCNT) [20], and group security policy token (GSPT) [16] have been proposed for unambiguous specification of group policies.

Secure group policy components

Groups bring up several new policy issues compared to peer-to-peer communications. In groups, different entities have different capabilities and roles, and authorization policy defines members, sender(s), and GCKS and its authorized representatives (e.g., subordinate GCKS and rekey server). Access control lists (ACL) and capability certificates are examples of mechanisms for access control enforcement. Policy also dictates which encryption or authentication algorithm to use for rekeying as well as secure data communications. Furthermore, the content owner specifies whether the GCKS should rekey after each membership change, or process such changes periodically. Group policy should also include expected member behavior when a member does not have the latest GSA. Application requirements, the value of content, and sometimes the mechanisms supported by the sender and GCKS all affect the group policy.

We end this section with a note on policy distribution in groups. Recall that policy negotiation does not converge in large groups. However, policy distribution brings up an interesting problem. Since there is no negotiation, members might want to know if they can participate before signing up (i.e., paying money to get access) to receive data. In other words, some of the policy needs to be predistributed. However, it is not a sound practice to advertise the policy of a secure group for all the world to see. Therefore, depending on application requirements, some of the policy is predistributed, while the rest is distributed by the GCKS to authorized members only. Chapter 7 provides a detailed discussion of group policy requirements and solutions.

1.3 Infrastructure protection

Aside from content protection, we need to consider threats to multicast infrastructure. There are three components to this problem, corresponding to sender access control, multicast routing infrastructure protection, and receiver access control.

Recall that in the traditional multicast model, a sender can transmit data to any multicast group; it does not even need to be a member of the group. Therefore, an adversary could inject data onto a multicast tree and waste network resources, such as buffer space on the routers and bandwidth on the links. This problem has begun to be addressed in the recent development of the SSM model, and some traditional protocols have been modified to conform to the SSM model (e.g., PIM-SSM [22]).

Next, in order for a unicast or a multicast routing protocol to behave correctly, all the control packets exchanged among the routers implementing the protocol must be protected against malicious modifications and deletions, and insertions of bogus control packets. Multicast routing protocol security is the topic of Chapter 8.

Similar to multicast routing protocols, Reliable Multicast protocols operate by exchanging control messages among the entities involved in the protocol. For a Reliable Multicast protocol to function properly, its control messages must be (at least) protected from modifications in transit. Reliable Multicast protocol security is addressed in Chapter 9.

Finally, recall that any host can join a multicast group and pull the distribution tree toward itself. An adversary could exploit this feature and pull unnecessary multicast traffic, causing denial of service. Therefore, for infrastructure protection, edge routers must allow only authorized members to pull multicast data.

1.4 Applications of secure multicasting

Data encryption and key distribution to authorized members supports group access control without modifying the IP multicast model. Access control enables content providers to control data distribution, and charge for content. Satellite TV distribution is an application that needs support for group access control.

Several multicast applications require data confidentiality or message integrity. Investors receiving stock quotes need a guarantee that the data is being sent by an authorized sender, and has not been modified en route. Several corporations transmit sensitive information such as software, database, and inventory updates using multicast ftp (MFTP) [23]. We discuss solutions to protect confidentiality and data integrity of MFTP communications in Section 10. Multimedia conferencing over the Internet needs protocols and mechanisms for privacy and data integrity. Finally, multicast *virtual private network* (VPN) is an application that enables forming a private network, say, between all branches of a bank, over the Internet.

1.5 Road map

We expect that the readers understand encryption, data integrity, host authentication, and other basic cryptographic properties. The readers should also be familiar with network security protocol requirements such as protection against man-in-the-middle, replay, connection hijacking, and

denial of service attacks. We also expect the readers to have some knowledge of the IPsec terminology.

The next chapter describes the framework for multicast security developed at the IRTF SMuG Research Group and IETF MSEC Working Group.

Problem area 1, that is, secure multicast data handling, is the topic of Chapter 3. Management of keying material, otherwise known as problem area 2, is introduced in Chapter 4, with further coverage in the following two chapters. Chapter 5 describes group key management architectures and protocols, and Chapter 6 discusses group key management algorithms. Secure group policy, labeled as problem area 3, is the subject of Chapter 7.

Infrastructure protection is the topic of the next two chapters. Routing protocol security is covered in Chapter 8, and Reliable Multicast protocol security is the subject of Chapter 9. Applications of secure multicasting is the topic of the following chapter. Chapter 11 concludes the book with a discussion on future topics.

There are a number of ways to read the material presented here. The chapters on each problem area are more or less independent. The current chapter and Chapter 2 provide an insight into the problem space of multicast security. Chapters 8 and 9 provide a summary of the multicast infrastructure security requirements and solutions. They are independent of the other chapters and could be read separately.

References

[1] Cain, B., et al., ''Internet Group Management Protocol, Version 3,'' draft-ietf-idmr-igmp-v3-09.txt, IETF, January 2002, work in progress.

[2] Deering, S., et al., ''The PIM Architecture for Wide-Area Multicast Routing,'' *IEEE/ACM Trans. on Networking*, Vol. 4, No. 2, 1996, pp. 153–162.

[3] Estrin, D., et al., ''Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification,'' RFC 2362 (experimental), IETF, June 1998.

[4] Diot, C., et al., ''Deployment Issues for the IP Multicast Service and Architecture,'' *IEEE Network, Special Issue on Multicasting*, January/February 2000.

[5] Kent, S., and R. Atkinson, ''IP Encapsulating Security Payload (ESP),'' RFC 2406 (proposed standard), IETF, November 1998.

[6] Krawczyk, H., M. Bellare, and R. Canetti, ''HMAC: Keyed-Hashing for Message Authentication,'' RFC 2104 (informational), IETF, February 1997.

[7] Canetti, R., et al., ''Multicast Security: A Taxonomy and Efficient Constructions,'' in *Proc. of IEEE INFOCOM*, New York, March 1999.

[8] Canetti, R., P. Rohatgi, and P. Cheng, ''Multicast Data Security Transformations: Requirements, Considerations, and Proposed Design,'' draft-irtf-smug-data-transforms-00.txt, IRTF, June 2000, work in progress.

[9] Harkins, D., and D. Carrel, ''The Internet Key Exchange (IKE),'' RFC 2409 (proposed standard), IETF, November 1998.

[10] Baugher, M., et al., ''Group Key Management Architecture,'' draft-ietf-msec-gkmarch-02.txt, IETF, March 2002, work in progress.

[11] Kent, S., and R. Atkinson, ''Security Architecture for the Internet Protocol,'' RFC 2401 (proposed standard), IETF, November 1998.

[12] Hardjono, T., M. Baugher, and H. Harney, ''Group Security Association (GSA) Management in IP Multicast,'' in *Proc. of the 16th International Conference on Information Security (IFIP/SEC)*, Paris, France, June 2001.

[13] Mittra, S., ''Iolus: A Framework for Scalable Secure Multicasting,'' in *Proc. of ACM SIGCOMM*, Cannes, France, September 1997, pp. 277–288.

[14] Hardjono, T., B. Cain, and I. Monga, ''Intra-Domain Group Key Management Protocol,'' draft-ietf-ipsec-intragkm-02.txt, IETF, February 2000, work in progress.

[15] Baugher, M., et al., ''Group Domain of Interpretation for ISAKMP,'' draft-ietf-msec-gdoi-04.txt, IETF, March 2002, work in progress.

[16] Harney, H., et al., ''Group Secure Association Key Management Protocol,'' draft-ietf-msec-gsakmp-sec-00.txt, IETF, March 2001, work in progress.

[17] Wallner, D., E. Harder, and R. Agee, ''Key Management for Multicast: Issues and Architectures,'' RFC 2627 (informational), IETF, June 1999.

[18] Yang, Y. R., et al., ''Reliable Group Rekeying: Design and Performance Analysis,'' in *Proc. of ACM SIGCOMM*, San Diego, CA, August 2001.

[19] Setia, S., et al., ''Kronos: A Scalable Rekeying Approach for Secure Multicast,'' in *Proc. of IEEE Symposium on Security and Privacy*, Oakland, CA, May 2000.

[20] Dinsmore, P. T., et al., ''Policy-Based Security Management for Large Dynamic Groups: An Overview of the DCCM Project,'' in *Proc. of the DARPA Information Survivability Conference & Exposition, Vol. I of II (DISCEX)*, Hilton Head, SC, January 2000, pp. 64–73.

[21] McDaniel, P., and A. Prakash, *Ismene: Provisioning and Policy Reconciliation in Secure Group Communication*, Technical Report CSE-TR-438-00, Electrical Engineering and Computer Science, University of Michigan, December 2000.

[22] Holbrook, H., and B. Cain, ''Source Specific Multicast for IP,'' draft-ietf-ssm-arch-00.txt, IETF, November 2001, work in progress.

[23] Miller, K., et al., ''Starburst Multicast File Transfer Protocol (MFTP) Specification,'' draft-miller-mftp-spec-03.txt, IRTF, April 1998, work in progress.

CHAPTER

2

Contents

2.1 The problem scope of multicast security

2.2 Fundamental issues

2.3 Transport and applications issues

2.4 The IETF problem scope for multicast and group security

2.5 Three problem areas in the management of keying material

2.6 The building blocks approach

2.7 Summary

Framework for multicast and group security

The problem of security for multicast and group security concerns not only content protection of the data or traffic being delivered to a group through IP multicast, but also concerns the protection of the network infrastructure that implements the multicast-related protocols. Therefore, one of the first tasks in looking at multicast security is to understand the landscape and define a reasonable scope or definition of the problems at hand.

Consequently, the aim of this chapter is to subdivide the complex problem of multicast and group security into manageable pieces. This chapter also reports on the IETF's approach in addressing these pieces. The subdivision also provides a roadmap for subsequent chapters dealing with specific issues.

2.1 The problem scope of multicast security

The problem of multicast and group security is complex because it involves several aspects of the Internet architecture, and covers all layers in the communications stack, above (and including) the network layer.

In order to reduce the complexity of the problems at hand, we have identified two broad categories of problems, denoting them as multicast fundamental issues and multicast transport and applications issues (see Figure 2.1). The first category of

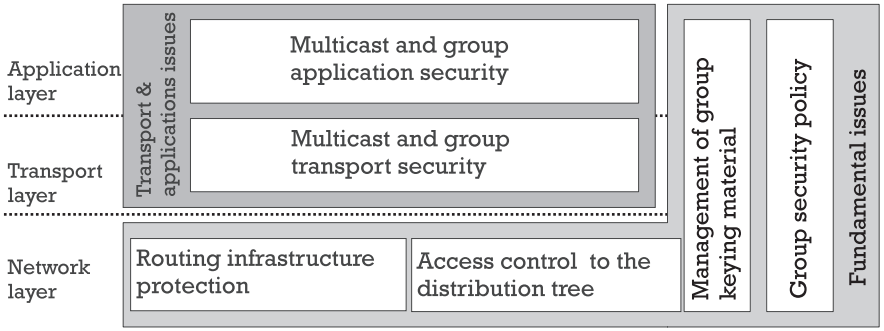


Figure 2.1 Problem scope of multicast security.

issues covers the basic security problems in multicast and group security, whose solutions represent building blocks for solving other problems, including those of the second category:

- Fundamental issues:
  - Routing infrastructure protection;
  - Controlled access to the multicast distribution tree (or group membership management);
  - Management of keying material;
- Transport and applications issues:
  - Security of *Reliable Multicast* (RM) protocols;
  - Applications requirements and other issues.

The thinking is that the fundamental issues need to be solved in order to provide a minimal level of security for IP multicast at the network layer. Without these fundamental issues being addressed, the transport and applications issues would be very difficult—if not impossible—to solve from a practical perspective. Thus, for example, without routing infrastructure protection (as a fundamental issue to be solved), data protection at the application layer would be less effective, as there would no be guarantee that the multicast routing protocol at the network layer would function correctly in the face of attacks (e.g., bogus control IP packets).

2.2 Fundamental issues

The fundamental issues can also be viewed from a topological perspective (see Figure 2.2), where topological boundaries add to the complexity of the issues at hand.

In Figure 2.2, the entities shown include the sender. This sends multicast packets through the multicast distribution tree, represented by the curved tree that spans across several *autonomous systems* (ASs) or domains, including stub ASs, access domains (e.g., access ISP), and transit ISPs. The multicast here is assumed to be a one-to-many multicast with a single sender. The figure does not show multicast tunnels between domains, or tunnels that carry multicast packets. A border router is shown within each domain's ingress and egress points for the distribution tree. Each branch point (fanout) in the distribution tree is shown to occur at either a multicast router (which is assumed to be also a unicast router), or a border router. The hosts are shown to be the receiver members of the multicast group, attached to a multicast router in their respective domains.

Using Figure 2.2 as a guide, we briefly describe the fundamental issues as follows.

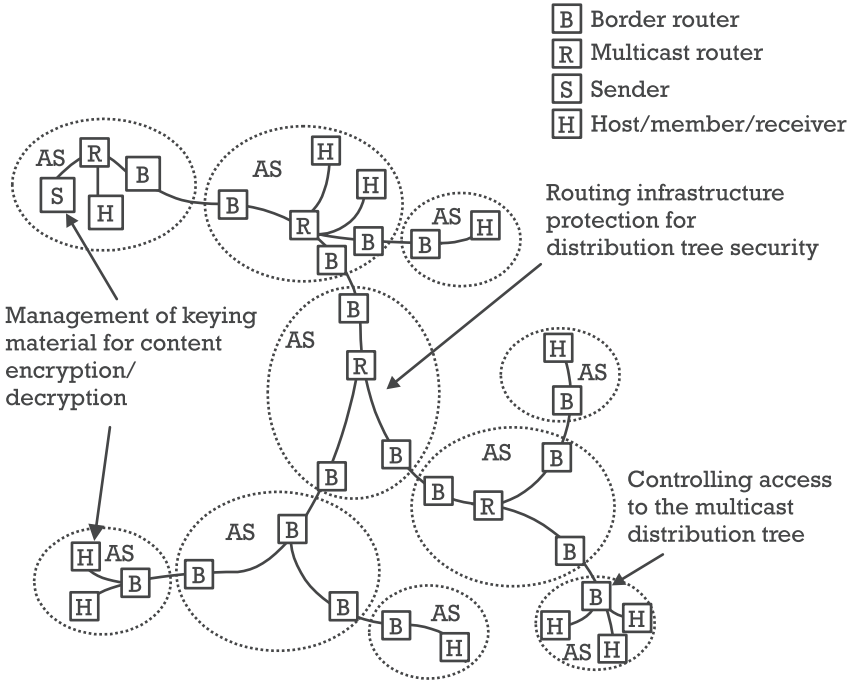


Figure 2.2 Fundamental problems in multicast and group security.

2.2.1 Routing infrastructure protection

A multicast routing protocol (such as PIM [1, 2], distance vector multicast routing protocol (DVMRP) [3] or multicast extensions to open shortest path first (MOSPF) [4]) creates a multicast distribution tree that effectively routes packets from the sender(s) to the receivers who are connected to the tree at its ''edges.'' Here multicast routers in several domains must maintain state information that allows a router to route multicast packets pertaining to a group to the correct outgoing interfaces, to downstream multicast routers, and finally to the receivers. In order for the content to be delivered in a timely fashion to its correct recipients, the multicast distribution tree must behave according to the precise specification as intended by its protocol designers (assuming no errors in the design itself).

In order for a unicast or a multicast routing protocol to behave correctly, all the control packets exchanged among the routers implementing the protocol must be protected against malicious modifications and deletions, and insertions of bogus control packets. This is true for both unicast and multicast routers, since routing tables in routers are often shared between the unicast protocols and the multicast protocols, and the actual shape of the multicast distribution tree is determined to a large extent by the unicast routing table.

Thus routing protection, both unicast and multicast, is a key requirement for the end goal of multicast security. This implies that at the network layer cryptographic protection of control packets is needed (which usually means authentication and integrity protection of control packets), which in turn implies that cryptographic key management and policy management is required for routing entities. This topic will be discussed further in Chapter 8.

2.2.2 Controlled access to the multicast distribution tree

The basic IP multicast model as defined by [5] allows for any host to become a member of the group simply by requesting to join the group. Unless they happen to be on the same subnet, group members in [5] are typically unaware of the existence of other members in the group. Although this IP multicast model may be attractive in its native form from the perspective of scalability, from the perspective of security such uncontrolled behavior may be undesirable. Thus, once a multicast distribution tree has been protected and is correctly functioning according to its specification, a related problem is that of controlled access to the distribution tree by potential hosts/members of a group.

Members (or their host computers) typically connect to a multicast distribution tree at the ''leaves'' (edges) of the tree through a group membership management protocol, such as the IGMP protocol [6, 7]. The IGMP protocol typically runs both at the host and at the closest multicast router upstream to that host. This router is typically referred to as the next hop router from the host.[1]

The use of additional membership management protocols at the network layer, such as IGMP, introduces further security threats to the picture. One potential security threat is resource exhaustion by a malicious nonmember host that pulls (attracts) the multicast distribution tree to its subnet (even when the content is encrypted), effectively wasting state storage on the affected routers in both the host's own domain and other domains upstream. This type of attack can be classified as a DoS or, more appropriately (in multicast) as the denial of quality of service (DQoS) attack. DQoS to multicast applications, such as video streaming in PPV, may render the service unbearable (i.e., slow) to most of the receivers of the service employing multicast.

We refer to this problem of illegal pulling of the multicast distribution tree as the receiver access control problem, where the word ''access'' is intended to mean connections to the distribution tree. A related security threat is that of sender access control, where nonmember hosts send useless content to the multicast distribution tree, effectively jamming the tree with data that members do not wish to receive. This again can lead to a DoS attack and a DQoS attack. Note that the basic model of [5] even allows anyone knowing the multicast group address (i.e., a Class D address in IPv4) to send IP packets to that address, thereby delivering unwanted packets to the rest of the group members (be they legitimate or not). This topic will be discussed further in Chapter 8.

2.2.3 Management of keying material

In the context of security, the underlying assumption is that the content or data being delivered through the multicast distribution tree carries some value, commercial or otherwise, and is thus worthy of being assigned

1. There are differing views as to whether a host is part of the multicast distribution tree. In discussing security issues, we have adopted the view that the host is *not* part of the multicast distribution tree since the host does not execute the multicast routing protocol, but instead runs a separate membership management protocol such as IGMP. This view does not preclude possible future protocols whose instance must be executed by all the routers and the hosts.

additional resources to protect it, with a cost that is proportional to the value of the content itself. Hence, in the context of multicast security, the assumption is that the content must be protected by way of encryption and authentication. Some applications (e.g., stock market data) may be public, but the integrity and timeliness of delivery is paramount. Hence, such data may not need to be encrypted, but must be at the very least authenticated through either source authentication methods (through asymmetric cryptography) or group authenticated methods (through symmetric cryptography).

Since valuable content must be protected (encrypted and/or authenticated) via cryptographic means, this in turn introduces cryptographic keys into picture, which must themselves be protected and managed. Hence the management of keying material is a fundamental issue in multicast security, and is, in fact, a superset of unicast key management. This includes the policies relating to the keying material, policies regarding a member's rights to obtain keying material, and the cryptographic methods and algorithms to apply the keys to the units of data being multicasted.

The work of [8] introduced the principle of the independence of group key management from the underlying multicast routing protocol. That is, regardless of the scope of a group key management protocol, such a protocol must be independent of (or decoupled from) the underlying multicast routing protocol, thereby allowing it to be used in conjunction with various multicast routing protocols. For a group key management protocol to be independent from multicast routing protocols, the group key management protocol must not rely on the structures (e.g., the multicast distribution tree) and mechanisms inherent to any particular routing protocol. A group key management protocol must also be separate from the session advertisement protocol [for example, session description protocol/session announcement protocol (SDP/SAP)]. However, sufficient information about a group and its related security parameters must be advertised in order for a host wishing to become a member to engage in the group key management protocol (assuming that the host implements the protocol).

One of the primary issues in group key management is that of the scalability of the protocols it employs. Often these protocols rely on one (or few) security managing entity (e.g., the key server) that is assumed to be trusted by all other entities in the system. Furthermore, the protocols often require host members to communicate securely with the trusted entity (unicast). Not only does the trusted entity (or entities) become a bottleneck in the scheme, it also becomes the best point of attack by intruders, since it necessarily holds security parameters pertaining to the host members.

Also impacting scalability of group key management protocols is the method used to perform a rekeying of the group key due a host member leaving the group or a new one joining. Rekeying of the group key involves a new group key being delivered to the (affected) members of the group. Ideally, a protocol should strive to minimize the number of affected host members in the case of rekeying, and to minimize the number of messages exchanged during the rekey process, particularly if secure (authentic and confidential) unicast messages must be exchanged. This topic will be discussed further in Chapter 4.

2.3 Transport and applications issues

The second category of problems in multicast and group security concerns security at the transport and applications layers. Since IP multicast is a network layer functionality, for IP multicast to be deployable, some level of reliability must be added atop multicast at the network layer. This is the function of RM protocols, which for simplicity have been placed at the transport layer. This, of course, does not preclude reliability protocols that are implemented across both the network layer and the transport layer.

At the application layer, specific applications often introduce additional security issues. Thus, for example, a conferencing application will have a different set of security needs compared to PPV applications. Similarly, the basic security requirements for these two examples would also be different. Source authentication for conferencing may be more important than in PPV (assuming both use content encryption), since a conference-based meeting might result in binding agreements among its participants. In PPV, as long as the user pays, the service provider may not care about the authentic identity of the user.

2.3.1 Security of Reliable Multicast protocols

A number of RM protocols have been proposed over the years to effect a reliable delivery of IP multicast packets at the network layer. The basic multicast routing protocol typically delivers an IP packet from one router to another, without any feedback as to whether the packet sent by the sender actually arrived at the receivers. Since the main task of a router is to push as many packets through its interfaces in a given time, multicast routing protocols typically rely on an upper layer mechanism to provide some level of guaranteed delivery. The analogy is between unicast delivery of packets at the network layer and TCP at the transport layer.

An RM protocol in itself operates by exchanging control messages among the entities involved in the protocol. For an RM protocol to function properly, its control messages must be (at least) protected from modifications in transit. Thus, in so far as integrity of control messages is concerned, RM protocols share the same need as multicast routing protocols.

However, different RM protocols behave differently, and thus each has its specific needs. For example, in NACK protocols that are based on a Reliable Multicast tree construction, when a host fails to receive a packet, that host issues a NACK message upstream. A branch point in the tree acts as an aggregator for these NACK messages. However, if each NACK message is digitally signed (as it ought to be), then the branch router must be able to verify the signatures (and hence its need to have the public key certificates of these hosts). The next question is whether the router should simply choose one (signed) NACK message and propagate it upstream, or whether it should create a new NACK and sign the NACK message itself. These and similar questions will be addressed in Chapter 9.

2.3.2 Applications requirements and other issues

Security issues in multicast and group security truly come to light only when the specific application that deploys multicast is understood. Some of the issues that typically occur at the application layer and drive the resulting solution include:

- *Many-to-many multicast applications*. Some applications are inherently multisender and multireceiver. Conferencing, chat groups, and video gaming are the most common examples. It is possible to distinguish further between *open* and *closed* many-to-many multicast groups. In the former, the group is open to any person, provided that some authentication is provided before the person is actually allowed to receive and/or send. In the closed case, a predetermined list of members is known in advance, and eligible members must authenticate themselves before being admitted. Typically, some conference or group ''owner'' determines the list. Whether a many-to-many multicast group is open or closed influences the underlying mechanisms and protocols used to implement the group.
- *Trust relationships*. The issue of trust relationships cuts vertically across the communications stack and horizontally across the topological entities involved in the group. Thus, for example, there

is the issue of trust for the certificate authority (CA) that issued a certificate for a given member. Then there is the issue of domain-level trust, where perhaps certain members will only trust entities located in a given set of domains. For example, if a key distributor (KD) is located in a member's domain, then that member would perhaps trust it more compared to one that is located in an ISP. The problem of trust relationship is a difficult one, influenced by several factors; research on this issue and others has been ongoing for a number of years. However, a practical approach embodying trust relationships specifically for multicast security on the Internet has yet to be proposed.

- *Identities and anonymity.* Some applications of multicast may require that a member's participation and the member's location (i.e., IP address) be maintained as private information. This, however, may be contradictory to routing's basic requirements at the network layer. In general, anonymity should be established at the application layer using *pseudonyms* [9, 10] and other cryptographic methods. Routers or network devices (e.g., mail servers) that provide an anonymizer service may also be deployed, where the anonymizer may be the member-by-proxy to the group at the network layer. The problem of anonymity in group communications has been addressed in the context of group-oriented cryptography and financial cryptography, and thus will not be addressed here.

The above problems are by no means the only issues related to transport and applications in multicast. Readers are invited to analyze the security requirements of their own applications.

2.4 The IETF problem scope for multicast and group security

The work on multicast and group security in the IETF originated in the IRTF, which is a sister organization to the IETF, tasked at addressing technologies on the horizon that would soon be developed in the industry, and whose aspects would require standardization. The emphasis in the IRTF is for practical research, conducted in research groups.

2.4.1 A brief history of multicast security efforts in the IETF

An IRTF research group does not produce standards. At the very best, it produces proposals to go into an IETF working group, possibly in the form of

documents that are near standard. Since research groups are looser in their manner of day-to-day operation, over the years each research group has conducted itself differently from others. This is in contrast to working groups, where a specific charter must underlie the working group's existence, and an approximate closure date must be established at the onset, by which all of the working group work items should be completed (although extensions of time are possible, and a recharter is another possible response to an ever-changing Internet technological landscape and industry).

IP multicast, despite its existence for the last decade, came to the forefront of the Internet revolution in the late 1990s as a promising way to deliver content to the growing numbers of users on the Internet. Events such as the IP Multicast Summit[2] attracted not only equipment vendors, but also content providers, content distributors, and content distribution network (CDN) providers seeking to deploy IP multicast for content delivery.

In the IETF, security was one area that was considered lacking, but essential to the deployment of IP multicast. However, since technologies for multicast security were not ready for standardization, an IRTF research group called SMuG was created in early 1998 to begin to address security issues relating to IP multicast and group security. One of SMuG's tasks was to help other research and working groups (related to multicast) in solving their security needs. Thus, SMuG collaborated with (or monitored closely) other groups in the IETF/IRTF such as the Reliable Multicast Research Group (RMRG), the Reliable Multicast Transport (RMT) working group, the PIM working group, and other security-related groups whose developments might impact multicast security (e.g., the IPsec working group). As far as possible, SMuG was to use technologies developed in other IETF and IRTF groups, rather than reinventing them.

SMuG did eventually develop a set of near-standard documents and proposals which were transferred into a new working group created in March 2000 called the MSEC working group. In the meantime, once the work items of SMuG had been carried over into the MSEC working group, SMuG was renamed and rechartered in July 2000 into the Group Security (GSEC) research group which broadened its scope of work.[3]

2. See http://www.ipmulticast.com.
3. Information on SMuG, MSEC, and GSEC can be found at the http://www.securemulticast.org.

2.4.2 The IETF multicast security Reference Framework

As mentioned earlier, one of the fundamental problems in multicast security is the management of keying material, which covers not only cryptographic keys but also the supporting parameters, such as SAs and policies. Key management was the first to be addressed by the SMuG research group and later the MSEC working group in the IETF. This topic was selected since it was independent of other efforts in other multicast-related working groups. The problem of routing infrastructure protection had been addressed briefly in the PIM working group in [11, 12], while the other fundamental problem of controlling access to the distribution tree had also been looked at by the SMuG community, albeit at a lower priority.

One immediate resolution of the SMuG research group was that the ''management of keying material'' needed to be better understood. To that extent, significant progress in SMuG toward a common understanding of the problem scope was the establishment of a Reference Framework for multicast security, which was presented at the 44th IETF in March 1999; also the third meeting of SMuG. This Reference Framework, shown in Figure 2.3, aims broadly at addressing the fundamental problem of the

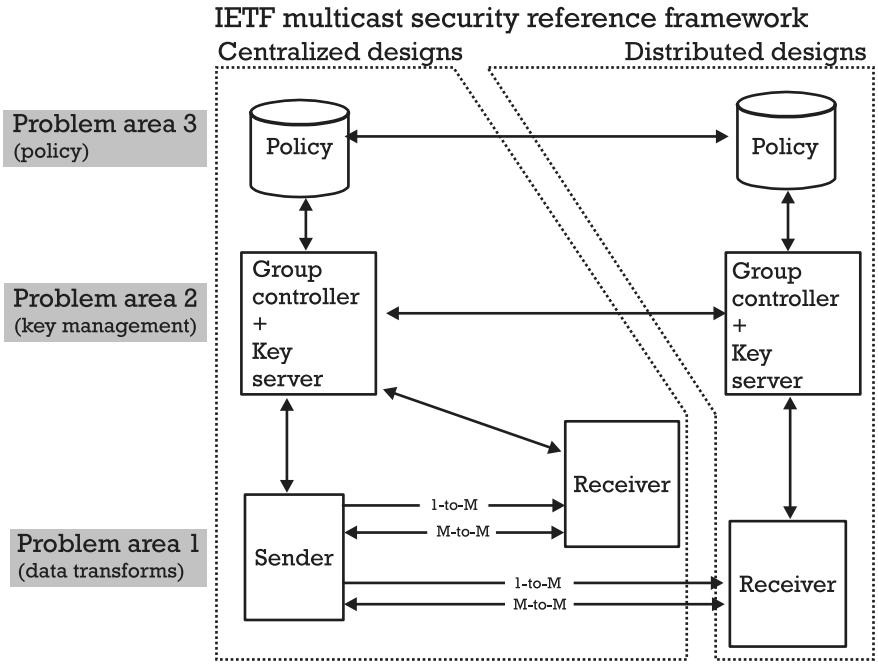


Figure 2.3 The IETF multicast security Reference Framework.

management of keying material. The aim of the Reference Framework is to classify problem areas, functional elements, and interfaces. The Reference Framework defines the building blocks and suggested a program of work for research and standardization in the SMuG research group and later in the MSEC working group.

The Reference Framework attempts to incorporate the main entities and functions relating to multicast security, and to depict the interrelations among them. At the same time it tries to express the complex multicast security question from the perspective of problem classification (i.e., the three problem areas), architectures (centralized and distributed), multicast types (one-to-many or many-to-many), and protocols (the exchanged messages).

The aim of the Reference Framework is to provide some general context within which problems can be identified and classified (as being within a given problem area), and the relationships among the problems can be recognized. Note that some issues span more than one so-called problem area. In fact, the framework encourages the precise identification and formulation of issues that involve more than one problem area, or those which are difficult to express in terms of a single problem area. An example is the expression of policies concerning group keys, which involves the problem areas of both group key management and multicast policies.

When considering the Reference Framework in Figure 2.3, it is important to realize that the singular boxes in the framework do not necessarily imply a corresponding single entity implementing a given function. Rather, a box in the framework should be interpreted loosely as pertaining to a given function related to a problem area. Whether that function is in reality implemented as one or more physical entities is dependent on the particular solution. As an example, the box labeled ''key server'' must be interpreted in broad terms as referring to the functions of key management. Similarly, the Reference Framework acknowledges that some implementations may, in fact, merge a number of the boxes into a single physical entity.

The Reference Framework can be viewed horizontally and vertically. Horizontally, it displays both the entities and functions as singular boxes, expressing each of the three broad problem areas. Vertically, it expresses the basic architecture designs for solutions; namely, a centralized architecture and a distributed architecture.

2.4.3 Elements of the Reference Framework

The Reference Framework diagram of Figure 2.3 contains boxes and arrows. The boxes are the functional entities and the arrows are the

interfaces between them. Standard protocols are needed for the interfaces, which support the multicast services between the functional entities. There are three sets of functional entities in both centralized and distributed designs as discussed below.

- *GCKS*. The GCKS represents both the entity and functions relating to the issuance and management of cryptographic keys used by a multicast group. They are subject to the user authentication and authorization checks conducted on the candidate member of the multicast group. The GCKS is also taken to mean the functions pertaining to group membership management. The key server is also referred to as key distributor (KD). In a distributed architecture, the GCKS entity also interacts with other GCKS entities to achieve scalability in the key-management-related services. In such a case, each member of a multicast group may interact with one or more GCKS entities (say, the nearest GCKS entity, measured in terms of a well-defined and consistent metric). Similarly, in a distributed architecture, a GCKS entity may interact with one or more policy servers; also arranged in a distributed architecture.
- *Sender and receiver*. The sender is an entity that sends data to the multicast group. In a 1-to-*n* multicast group only a single sender is allowed to transmit data to the group. In an *m*-to-*n* multicast group many (or even all) group members can transmit data to the group. Both sender and receiver must interact with the GCKS entity for the purpose of key management. This includes user authentication, the obtaining of keying material in accordance with key management policies for the group, obtaining new keys during key updates, and obtaining other messages relating to the management of keying material and security parameters. The influence of policies on both senders and receivers is seen as coming indirectly through the GCKS entities, since the event of joining a multicast group is typically coupled with the sender/receiver obtaining keying material from a GCKS entity. This does not preclude direct interaction between the sender/receiver and the policy server. The Reference Framework displays two receiver boxes, corresponding to the situation where both the sender and receiver employ the same GCKS entity (in a centralized architecture), and where the sender and receiver employ different GCKS entities (in a distributed architecture).

- *Policy server*. The policy server represents both the entity and functions used to create and manage security policies specific to a multicast group. The policy server interacts with the GCKS entity in order to install and manage the security policies related to the membership of a given multicast group, and those related to keying material for a multicast group. The interactions between the policy server and other entities in the Reference Framework are dependent to a large extent on the security circumstances being addressed by a given policy.
- *Centralized and distributed designs*. The need for solutions to be scalable to large groups across wide geographic regions of the Internet requires the elements of the framework to also function as a distributed system. This implies that a GCKS entity must be able to interact securely with other GCKS entities in a different location. Similarly, policy servers must interact with each other securely to allow the communication and enforcement of policies across the Internet.

2.5 Three problem areas in the management of keying material

As mentioned previously, Figure 2.3 shows three horizontal labels denoting the classification of the problems into three problem areas, which can be seen as three subproblems of the broader fundamental problem management of keying material (see Figure 2.4). Additionally, the aim of this classification is to allow three subgroups of people to work in parallel to find solutions corresponding to the three areas. A close collaboration is needed, since to a large extent there are dependencies among the three problem areas. For example, group policy might determine key management rules, while rekeying of a group key might be a function of the way the key is used at the end hosts.

The three problem areas are briefly summarized here, and will be expanded further below:

- *Problem area 1: Multicast data handling*. This area covers problems concerning the security-related treatments of multicast data by the sender and the receiver. In particular, algorithms for efficient application of the cryptographic keys in the multicast context need to be studied. This problem area is further discussed in Section 2.5.1.
- *Problem area 2: Management of keying material*. This area is concerned with the secure distribution and refreshment of keying material. This problem area is further discussed in Section 2.5.2.

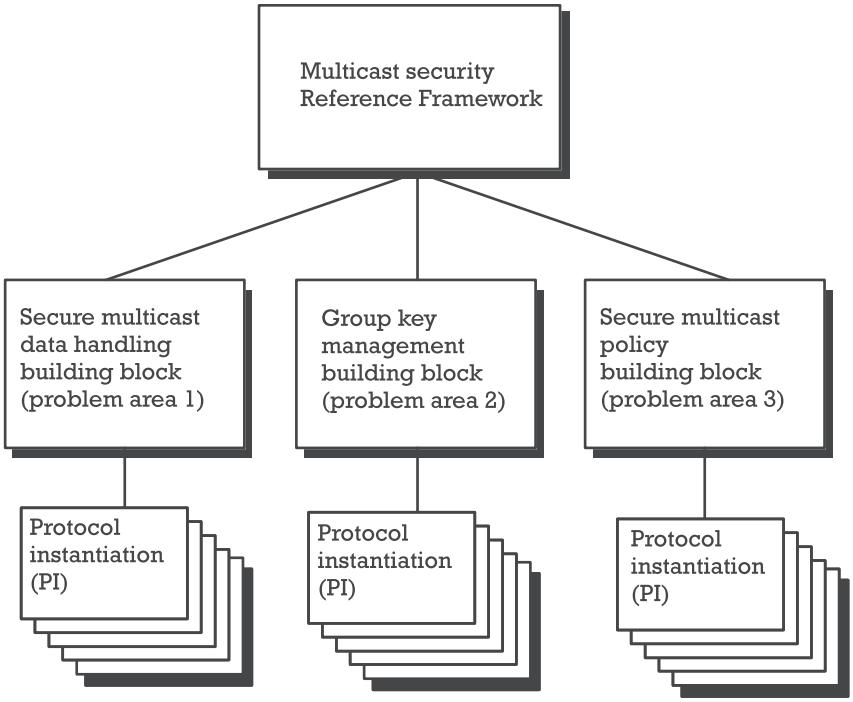


Figure 2.4 Multicast security problem areas, building blocks, and protocol instantiations.

- *Problem area 3: Multicast security policies.* This area covers aspects of policy in the context of multicast security, taking into consideration the fact that policies may be expressed in different ways, that they may exist at different levels in a given multicast security architecture, and that they may be interpreted differently according to the context in which they are specified and implemented. This problem area is further discussed in Section 2.5.3.

2.5.1 Problem area 1: Multicast data handling

In a secure multicast group the data typically needs to be:

- Encrypted using the group key, mainly for access control and possibly also for confidentiality.
- Authenticated, for verifying the source and integrity of the data. Authentication can take two forms:

1. *Source authentication and data integrity*. This functionality guarantees that the data originated from the claimed source and was not modified in transit (either by a group member or an external attacker).
2. *Group authentication*. This type of authentication only guarantees that the data was generated (or last modified) by some group member that possesses the group key. It does not guarantee data integrity unless all group members are trusted.

While multicast encryption and group authentication are fairly standard, and similar to encrypting and authenticating point-to-point communication, source authentication for multicast is considerably more involved. This topic will be discussed at length in Chapter 3.

2.5.2 Problem area 2: Management of keying material

The term ''keying material'' refers to the cryptographic key belonging to a group, the state associated with the keys, and the other security parameters related to the keys. Hence, the management of the cryptographic keys belonging to a group necessarily requires the management of their associated state and parameters. A number of solutions for specific problems must be addressed. These may include the following:

- Methods for member identification and authentication;
- Methods to verify the membership to groups;
- Methods to establish a secure channel between a GCKS entity and the member, for the purpose of delivering shorter term keying material pertaining to a group;
- Methods to establish a long-term secure channel between one GCKS entity and another, for the purpose of distributing shorter term keying material pertaining to a group;
- Methods to effect the changing of keys and keying material;
- Methods to detect and signal failures and perceived compromises to keys and keying material.

The needs related to the management of keying material must be seen in the context of the policies that prevail within the given circumstance.

2.5.3 Problem area 3: Multicast security policies

Multicast security policies must provide the rules for operation of the other elements of the Reference Framework. While much of the work for the multicast security policy area is focused in the policy controller, there are potential areas for work in the application of policy at the group controller element and the member (sender and receiver) elements. While there is already a basis for security policy management in the IETF between the Policy Working Group and the IP Security Policy working group, multicast security policy management should extend the concepts developed for unicast communication in the areas of:

- Policy creation;
- High-level policy translation;
- Policy representation.

Examples of work in multicast security policies include the Dynamic Cryptographic Context Management project [13] the group key management protocol (GKMP) [14], and Antigone [15].

Policy creation for secure multicast has several more dimensions than the single administrator–specified policy assumed in the existing unicast policy frameworks. Secure multicast groups are usually large and by their very nature extend over several administrative domains, if not spanning a different domain for each user. There are several methods that need to be explored for the creation of a single, coherent group security policy. They include a top-down specification of the group policy from the group initiator, and negotiation of the policy between the group members (or prospective members). Negotiation can be as simple as a strict intersection of the policies of the members, or extremely complicated using weighted voting systems.

High-level policy translation is much more difficult in a multicast group environment, especially when group membership spans multiple administrative domains. When policies are specified at a high level with a policy management tool, they must then be translated into more precise rules that the available security mechanisms can both understand and implement. When dealing with multicast communication and its multiple participants, it is essential that the individual translation performed for each participant result in the use of a mechanism that is interoperable with the results of all of the other translations. Typically, the translation from high-level policy to implementation mechanisms must result in the same mechanism, in order to achieve communication between all of the group

members. The requirement that policy translation result in the same mechanism places constraints on the use and representations in the high-level policies. It is also important that policy negotiation and translation be performed as an integral part of joining a group. Adding a member to a group is meaningless if such new members will not be able to participate in the group communications.

Multicast security policies must represent or contain more information than a traditional peer-to-peer policy. In addition to representing the security mechanisms for the group communication, the policy must also represent the rules for the governance of the secure group. Policy must be established for the basic group operations of add and remove, as well as more advanced operations such as leave, rejoin, or resync.

2.6 The building blocks approach

One of the challenges of standardizing any set of technologies is bringing together aspects of different existing protocols and implementations, and deriving common functions that already exist in them.

This challenge was encountered within the Reliable Multicast Transport (RMT) Working Group, where several mature (and already productized) Reliable Multicast protocols were being brought together for standardization. The approach adopted in the RMT working group was to derive building blocks that were common to these existing protocols. Thus, for example, congestion control may be a common feature of many Reliable Multicast protocols, and each existing protocol instantiations (PI) in the RMT working group had some method of performing congestion control. Hence, the thinking was that a common congestion control method could be derived into a common building block that could be standardized, independent of any specific protocol instantiations. Having a congestion control building block allowed for later improvements to scheme, and even possible replacement of the block with a better scheme in the future.

The same building blocks approach, each with one or more PIs, was also adopted in addressing multicast security in both the SMuG research group and MSEC working group, as explained further in Figure 2.4.

2.6.1 Motivation for building blocks

A common approach to solving a complex problem is to subdivide the problem into manageable blocks. Here, each block must serve a well-defined

function, and its relationship with other blocks must be clearly defined. Besides being more manageable, the approach inherently has a number of advantages, including use and reuse of the functional block independently of the whole, and the ability to combine different blocks to satisfy multiple functions.

Although the building blocks approach has advantages, there are a number of risks associated with the approach, particularly in the context of bringing together existing protocols into a standardization body. Some of the risks are [16]:

- Delayed development, which results from the need for additional work to develop ways to combine independent building blocks;
- Increased complexity, which is caused by too many building blocks having too many interfaces;
- Reduced performance, which may be caused by too much modularization;
- Abandonment of prior work, which results from attempts to develop robust, general solutions.

Despite the above-mentioned risks, there are at least four important benefits to multicast security in applying the building blocks approach.

1. *Reuse of publicly reviewed cryptographic protocols*. The reuse of proven technologies is attractive in general, and is more particularly beneficial in cryptographic mechanisms. Thus, existing units or blocks that have been publicly reviewed offer a great advantage in solving complex issues such as multicast security.
2. *Timely delivery of needed technology*. Building blocks allow fast delivery of specific technologies, independent of others. In the context of multicast security, for example, multicast data confidentiality could be standardized independently of other security services that may take longer to specify for a great variety of uses, such as multicast source authentication.
3. *Robust support for a variety of application environments*. Good building block definitions will permit the combination of individual building blocks to flexibly add security services to IP multicast or application-layer multicast traffic.

4. *Simplicity in the proof of correctness.* Verifying the correctness of each building block as a separate block is simpler than verifying an entire homogeneous system. This is inherent in the building blocks approach.

To make the notion of building blocks more concrete, consider the example of the independence of protocols in the IPsec suite. Certain IPsec protocols such as (authentication header) AH [17] and ESP [18] perform their security functions independently of other protocols in the IPsec suite, such as IKE, which provides security association [19] and key management services to AH and ESP. As a result of this independence, a compliant ESP implementation can be used today to provide IP multicast confidentiality, despite the fact that an IKE security association is unique to a pair of communicating endpoints, and is unsuitable for managing multicast group keys. If ESP uses an IPsec SA having a multicast address, however, it effectively supports IP multicast confidentiality, since there is no requirement that an SA used by ESP be established by IKE (although this might be a reasonable policy for some environments). Thus, other applications may be used to establish the keying material needed for an IP multicast ESP service. For this to work, however, an application programming interface (API) might be needed for updates to the host *security association database* (SAD). Taken together, these can provide a useful multicast security service, namely, IP multicast confidentiality. The capacity to provide a useful security service is one important criterion for a multicast security building block, which is realized in an algorithm, protocol, API, or by other means.

A multicast security building block should be able to be combined with other building blocks to provide additional security services. Without this property, the building block is little more than an incomplete solution to the general problem. Thus a second criterion for a good multicast security building block is that it can be combined with other building blocks to provide additional security services. A good building block for IP multicast confidentiality can be combined with other building blocks for IP multicast source authentication, data authentication (integrity), and additional security services.

A good example of the building blocks approach is the work that was done on multicast packet-level source and data authentication within the SMuG community. One output of this effort is a draft specification on multicast packet-level authentication for real-time transport protocol (RTP) applications [20], which is a proposed RTP profile [21]. This work proposes to efficiently authenticate the sender and verify the integrity of multicast

packets, by applying a digital signature over the hash of a sequence of packets [20, 22]. Although practical experience is needed to evaluate this protocol, it illustrates the use of a multicast security building block.

A successful multicast source or data packet authentication building block should be applicable to other applications such as SDP/SAP [23, 24]. Indeed, technology that solves source and data packet authentication for real-time multicast application traffic should be considered for IP multicast traffic as well—at least the algorithm, if not the protocol. Thus, a third criterion for a multicast security building block is its applicability to IP multicast and application-layer multicast security.

The preceding discussion has established a set of three criteria for good multicast security building blocks:

1. A building block provides a flexible security service. A protocol that realizes a building block should be standardizable, independently of other building blocks.
2. Building blocks can be combined in a framework to provide a set of multicast security services that amount to a whole protocol for multicast security.
3. Building blocks can be applied to both IP multicast and application-layer multicast security; good multicast security building blocks can be adapted for both protocol and data security.

As discussed above, useful solutions may not satisfy all three criteria, but the most promising proposals for standardization would probably satisfy more than a single criterion. Thus, the criteria are suggested as good measures for a functional or protocol building block.

In addition to the demands of productive use and standardization, the building blocks approach allows the identification of certain problems that are still poorly understood and thus poorly defined. In the context of the SMuG research group and MSEC working group, the building blocks approach helped focus the research mission and facilitate the standards objectives. By adopting this approach early, SMuG avoided the near-impossible task of extracting building blocks from mature protocols, and the experience can positively influence multicast standards work that may occur in IETF working groups.

The building blocks approach also allows the sharing of standardized technologies with working groups within the IRTF and the IETF. For example, certain blocks developed with the SMuG research group may be useful and deployable by the RMT working group in its efforts to secure the

RMT protocols. The same blocks may also be used to secure other application protocols (e.g., RTP) and multicast routing protocols, and be applied to other areas where both multicast and security services are needed.

2.6.2 Functional building blocks

Having explained the motivations of using the building blocks approach, we discuss the functional building blocks identified by SMuG and MSEC in this section. For example, multicast source authentication, data authentication, and confidentiality occur on the multicast data interface between senders and receivers in Figure 2.3. Authentication and confidentiality services may also be needed between the key server and key clients (i.e., the senders and receivers), but the services that are needed for multicast key management may be unicast as well as multicast. Multicast key management is a separate function and has a separate building block. A functional building block for multicast security therefore identifies a specific function along one or more interfaces of Figure 2.3.

The functional building blocks identified in SMuG and MSEC are:

1. *Multicast data confidentiality.* This functional building block handles the encryption of multicast data at the sender's end, and the decryption at the receiver's end. This building block presumably may apply the keying material that is provided by multicast key management in accordance with multicast policy management, but it is independent of both.

   An important part of the work on the multicast data confidentiality building block is in the identification of and motivation for specific ciphers that should be used for multicast data. Obviously, not all ciphers will be suitable for IP multicast and application-layer multicast traffic. Since this traffic will usually be connectionless user datagram protocol (UDP) flows, stream ciphers may be unsuitable although hybrid stream/block ciphers may have advantages over some block ciphers. In addition, the real-time and other requirements of multicast senders and receivers must be evaluated, and selections must be made for a suitable set of promising ciphers and data protocols for IP multicast and application-layer multicast data confidentiality.

   Regarding application-layer multicast, some consideration is needed for sending encrypted data in a multicast environment lacking admission control, where practically any application program can join a multicast event, independently of its participation

in a multicast security protocol. Thus, this building block is also concerned with the effects of multicast confidentiality services, intended and otherwise, on application programs in all senders and receivers.

With respect to the problem areas (in Figure 2.3), the multicast data confidentiality building block is placed in problem area 1 along the interface between senders and receivers. The algorithms and protocols that are realized from work on this building block may be applied to other interfaces and other problem areas, when multicast data confidentiality is needed.

2. *Multicast source authentication and data integrity.* This building block handles source authentication and integrity verification of multicast data. It includes the transforms to be made both at the sender's end and at the receiver's end. It assumes that the appropriate signature and verification keys are provided via multicast key management in accordance with multicast policy management.

   Work done by members of the SMuG research group suggests that this is one of the harder areas of multicast security, based on the connectionless and real-time requirements of many IP multicast applications. There are classes of application-layer multicast security, however, where off-line source and data authentication will suffice. Not all multicast applications require real-time authentication and data packet integrity. A robust solution to multicast source and data authentication, however, is necessary for a whole protocol solution to multicast security.

   In Figure 2.3, the multicast source and data authentication building block is placed in problem area 1 along the interface between senders and receivers. The algorithms and protocols that are produced for this functional building block may have applicability to building blocks in other problem areas that use multicast services such as multicast key management.

3. *Multicast group authentication.* This building block provides a limited amount of authenticity of the transmitted data. It only guarantees that the data originated from (or was last modified by) an entity that possesses the group key (symmetric key): namely, a group member. It does not guarantee authenticity of the data, in case other group members are not trusted.

   The advantage of group authentication is that it is guaranteed via relatively simple and efficient cryptographic transforms. Therefore, when source authentication is not paramount, group

authentication becomes useful. In addition, performing group authentication is useful even when source authentication is later performed, in that it provides a simple-to-verify weak integrity check that is useful as a measure against DoS attacks.

The multicast group authentication building block is placed in problem area 1 along the interface between senders and receivers.

4. *Multicast group membership management.* This building block describes the functionality of registration and deregistration of members. Registration includes member authentication, notification and negotiation of security parameters, and logging of information according to the policies of the group controller and the would-be member. (Typically, some method for the advertisement of group information would occur before the registration takes place. The registration process will typically be invoked by the would-be member.)

   Deregistration may occur either at the initiative of the member or at the initiative of the group controller. It would result in logging of the deregistration event by the group controller, and an invocation of the appropriate mechanism for terminating the membership of the deregistering member.

   This building block also describes the functionality of the communication related to group membership among different GCKS servers in a distributed group design.

   In Figure 2.3, the multicast group membership building block is placed in problem area 2, and has interfaces to senders and receivers.

5. *Multicast key management.* This building block describes the functionality of distributing and updating the cryptographic keying material throughout the life of the group. Components of this building may include:

   - GCKS to client (sender or receiver) notification regarding current keying material (e.g., group encryption and authentication keys, auxiliary keys used for group management, keys for source authentication, etc.).
   - Updating of current keying material, depending on circumstances and policies.
   - Termination of groups in a secure manner, including the multicast group itself and the associated keying material.

Among the problems to be solved by this building block is the secure management of keys between key servers (GCKS) and clients, the addressing of issues for the multicast distribution of keying material, and the scalability or other performance requirements for multicast key management.

To allow for an interoperable and secure IP multicast security protocol, this building block may need to specify host abstractions such as a *group security association database* (GSAD) and a *group security policy database* (GSPD) for IP multicast security. The degree of overlap between IP multicast and application-layer multicast key management is a consideration. Thus, work on this functional building block must take into account the key management requirements for IP multicast, the key management requirements for application-layer multicast, and to what degree specific realizations of a multicast key management building block can satisfy both.

This building block also describes the functionality of the communication related to key management among different GCKS servers in a distributed group design.

Multicast key management appears in both the centralized and distributed designs as shown in Figure 2.3 and is placed in problem area 2.

6. *Multicast policy management.* This functional building block handles all matters related to multicast group policy including membership policy and multicast key management policy. Indeed, one of the first tasks is to identify the different areas of multicast policy. Multicast policy management includes the design of the policy server for multicast security, the particular policy definitions that will be used for IP multicast and application-layer multicast security, and the communication protocols between the policy server and the key server. This functional building block may be realized using a standard policy infrastructure such as a policy decision point (PDP) and policy enforcement point (PEP) architecture [25]. Thus, it may not be necessary to reinvent a separate architecture for multicast security policy. Rather, products of IETF efforts in the areas of network and security policy could be deployed.

   The multicast policy management building block describes the functionality of the communication between an instance of a GCKS and an instance of the policy server. The information transmitted may include policies concerning groups, memberships, keying

material definition and their permissible uses, and other information. This building block also describes communication between and among policy servers. Thus, the multicast policy management building block is placed in problem area 3, along the interface between key servers and policy servers.

2.7 Summary

The primary aim of this chapter has been to subdivide the complex problem of multicast and group security into manageable pieces; each of which will be addressed or discussed in subsequent chapters.

The broadest division was the categorization into fundamental issues as the first subgroup of problems, and transport and application issues as the second. The first covered problems pertain to routing infrastructure protection, controlled access to the multicast distribution tree, and the management of keying material. The latter include (but are not limited to) the security of RM protocols and other applications requirements.

Alhough these are not the only issues at hand, the first set of problems are referred to as fundamental, since they are a necessary first step toward securing IP multicast, and they lead to building blocks that may be used to solve other needs, such as security in RM protocols.

Efforts in the IETF focused initially on the management of keying material. Within this space, one significant development was the development of a Reference Framework as shown in Figure 2.3. The Reference Framework attempts to incorporate the main entities and functions relating to multicast security, and to depict the interrelations among them. At the same time it also tries to express the complex multicast security question from the perspectives of problem classification (i.e., the three problem areas) architectures (centralized and distributed), multicast types (one-to-many or many-to-many), and protocols (the exchanged messages).

The Reference Framework paved the way for the definition of three problem areas that would be immediately addressed by the SMuG research group (see Section 2.5). The three are multicast data handling (problem area 1), the management of keying material (problem area 2), and multicast security policies (problem area 3).

Using the Reference Framework as the basis, an approach was adopted based on the notion of building blocks, as a way for SMuG to work forward (see Section 2.6). This approach presents several advantages as well as risks. The resulting functional building blocks are: multicast data confidentiality, source authentication and data integrity, group authentication, group

membership management, key management, and multicast policy management. These building blocks have recently been the focus of the MSEC working group in the IETF.

References

[1] Deering, S., et al., ''The PIM Architecture for Wide-Area Multicast Routing,'' *IEEE/ACM Trans. on Networking*, Vol. 4, No. 2, 1996, pp. 153–162.

[2] Estrin, D., et. al., ''Protocol Independent Multicast-Sparse Mode (PIM-SM),'' protocol specification, RFC 2362 (experimental), IETF, June 1998.

[3] Waitzman, D., C. Partridge, and S. E. Deering, ''Distance Vector Multicast Routing Protocol,'' RFC 1075 (experimental), IETF, Nov. 1988.

[4] Moy, J., ''Multicast Extensions to OSPF,'' RFC 1584 (proposed standard), IETF, March 1994.

[5] Deering, S. E., ''Host Extensions for IP Multicasting,'' RFC 1112 (standard), IETF, August 1989.

[6] Cain, B., et al., ''Internet Group Management Protocol, Version 3,'' draft-ietf-idmr-igmp-v3-09.txt, IETF, January 2002, work in progress.

[7] Fenner, W., ''Internet Group Management Protocol,'' RFC 2236 (proposed standard), IETF, November 1997.

[8] Hardjono, T., B. Cain, and N. Doraswamy, ''A Framework for Group Key Management for Multicast Security,'' draft-ietf-ipsec-gkmframework-03.txt, IETF, August 2000, work in progress.

[9] Chaum, D., ''Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms,'' *Communications of the ACM*, Vol. 24, No. 2, Feb. 1981.

[10] Chaum, D., ''Showing Credentials Without Identification: Transferring Signatures Between Unconditionally Unlinkable Pseudonyms,'' In *Advances in Cryptology—AUSCRYPT*, Sydney, Australia: Springer-Verlag Inc., LNCS 453, January 1990.

[11] Hardjono, T., and B. Cain, ''Simple Key Management Protocol for PIM,'' draft-ietf-pim-simplekmp-01.txt, IETF, February 2000, work in progress.

[12] Wei, L., ''Authenticating PIM Version 2 Messages,'' draft-ietf-pim-v2-auth-01.txt, IETF, May 1999, work in progress.

[13] Dinsmore, P. T., et al., ''Policy-Based Security Management for Large Dynamic Groups: An Overview of the DCCM Project,'' in *Proc. of the DARPA Information Survivability Conference & Exposition Volume I of II (DISCEX)*, Hilton Head, SC, January 2000, pp. 64–73.

[14] Harney, H., and C. Muckenhirn, ''Group Key Management Protocol (GKMP) Architecture,'' RFC 2094 (experimental), July 1997.

[15] McDaniel, P., A. Prakash, and P. Honeyman, ''Antigone: A Flexible Framework for Secure Group Communication,'' in *Proc. of the 8th USENIX Security Symposium*, Washington, D.C., August 1999, pp. 99–114.

[16] Whetten, B., et al., ''Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer,'' RFC 3048 (informational), IETF, January 2001.

[17] Kent, S., and R. Atkinson, ''IP Authentication Header (AH),'' RFC 2402 (proposed standard), IETF, November 1998.

[18] Kent, S., and R. Atkinson, ''IP Encapsulating Security Payload (ESP),'' RFC 2406 (proposed standard), IETF, November 1998.

[19] Kent, S., and R. Atkinson, ''Security Architecture for the Internet Protocol,'' RFC 2401 (proposed standard), IETF, November 1998.

[20] McCarthy, L., ''RTP Profile for Source Authentication and Non-Repudiation of Audio and Video Conferences,'' draft-mccarthy-smug-rtp-profile-src-auth-00.txt, IETF, May 1999, work in progress.

[21] Schulzrinne, H., et al., ''RTP: A Transport Protocol for Real-Time Applications,'' RFC 1889 (proposed standard), IETF, January 1996.

[22] Wong, C. K., and S. S. Lam, ''Digital Signatures for Flows and Multicasts,'' *IEEE/ACM Trans. on Networking*, Vol. 7, No. 4, August 1999, pp. 502–513.

[23] Handley, M., and V. Jacobson, ''SDP: Session Description Protocol,'' RFC 2327 (proposed standard), IETF, April 1998.

[24] Handley, M., C. Perkins, and E. Whelan, ''Session Announcement Protocol,'' RFC 2974 (experimental), IETF, October 2000.

[25] Yavatkar, R., D. Pendarakis, and R. Guerin, ''A Framework for Policy-Based Admission Control,'' RFC 2753 (informational), IETF, January 2000.

CHAPTER

3

Multicast data authentication

Contents

3.1 Issues in multicast data authentication

3.2 Digital signatures for source authentication

3.3 Hash chaining to authenticate streaming data

3.4 MAC-based source authentication of unreliable streams

3.5 IPsec ESP and MESP

3.6 Summary

Secure multicast data handling is one of the three problem areas (problem area 1; see Section 2.5) identified by the SMuG Research group and adopted by the IETF MSEC working group. Different applications have different requirements for secure data transmission. For example, in some cases, data source authentication and integrity protection may be sufficient (e.g., in news distribution), whereas other applications might need confidentiality (e.g., news distribution with access control) as well. Confidentiality of group communications is discussed in detail in Chapters 4, 5, and 6. Data source authentication of group communications is a challenging problem, and is the focus of this chapter.

Different applications may need different levels of authentication for secure group communication. For example, in some applications it may be sufficient to know that a data packet originated within the secure group and has not been modified in transit by nonmembers. This property is known as group authentication. In other applications, members (or receivers) might need to verify to themselves that a data packet was sent by the claimed (or the authorized) source. This property is known as source authentication. Finally, in some cases a neutral third party should also be able to independently determine the data source. In other words, a sender should not be able to repudiate having ever sent the data.

There is another dimension to the challenges in source authentication of multicast data, presented by the nature of the applications. While there are some applications that involve

bulk transfer of data to a group of receivers, most send streaming data. Furthermore, data may need to be transmitted in real time, with implications on buffer space requirements at both the sender and the receivers. Finally, packets may be lost in transit or arrive out of order. Thus, source authentication and non-repudiation of group communications are difficult problems.

There are several documented solutions that provide various levels of authentication with trade-offs in computation and communication overhead, buffer space requirements, authentication delay, and verification probability. Some of these schemes are loss tolerant, while others require reliable delivery. Among the loss tolerant schemes, some can tolerate any kind of losses, whereas others are optimized to tolerate bursty or random losses, with limitations such as verification probability.

Digitally signing each packet addresses the security needs of most, if not all, applications. But it is computationally expensive to sign or verify digital signatures. The per-packet communication overhead is also excessive. Thus, most schemes use amortization techniques to reduce both these costs. Block hashing, described in Section 3.2.1, amortizes the computational cost of a digital signature over a block of packets. In hash chaining–based schemes, each packet's hash is sent as part of one or more packets. Signature packets hold the hashes of end points (packets) of the hash chains. Section 3.3 describes a variety of schemes based on hash chaining. Another class of protocols uses MAC-(symmetric key–based) based authentication, with keys derived from one-way function chains. These protocols get around the limitation of MAC keys for source authentication using delayed key disclosures.

Note that irrespective of the authentication scheme, IPsec ESP [1], MESP [2] or AMESP protocols may be used to send authentication information or keys.

3.1 Issues in multicast data authentication

The problem of secure group communication can be divided into several building blocks for better understanding of the requirements, and simpler analysis of solutions. In that spirit, Section 2.6 lists several building blocks, two of which, multicast source authentication and data integrity, and group authentication, are addressed here. Recall that these two building blocks belong to problem area 1 of the Reference Framework described in Chapter 2.

The problem of multicast data authentication has three components: data integrity, authentication, and non-repudiation.

- Receivers must be able to determine that data has not been modified either by other members of the multicast group or by external adversaries. This property is referred to as data integrity protection.
- Receivers need to be able to establish the source of the data, at least for themselves. In other words, we need data origin authentication.
- A stronger version of the above property, referred to as non-repudiation, allows impartial third-party verification of the data source.

Data integrity and authentication go hand in hand. Notice that if data has been modified in transit, the source is no longer the legitimate origin of data. Similarly, if a receiver can establish the source of data (at least to itself), data has not been modified en route. Therefore data integrity and authentication are dependent on each other. Non-repudiation is essentially a stronger version of data authentication. In other words, a protocol or mechanism that ensures nonrepudiation also guarantees authentication. Additional security services such as confidentiality and access control are addressed in Chapters 4, 5, and 6.

There are two distinct types of applications, with varying requirements, to consider in authenticating multicast data. In the first, a sender transmits a bulk of data to receivers. In other words, receivers can wait until they receive all the data sent, before verifying the authenticity and integrity of that data. Examples of such applications include multicast ftp [3] and Web cache synchronization. The second category of applications streams multicast data. Receivers might want to verify the integrity and authenticity of each data packet as it arrives, and use it immediately. We also need to handle lossy communication channels, as well as out-of-order packet delivery. In other words, authentication information must be associated with each packet. Video-on-demand and multimedia conferencing are examples of applications that need multicast streaming.

Two different mechanisms are generally used for source authentication. The first is to use digital signatures for non-repudiation, and the second is to use MACs for authentication only. Recall that non-repudiation is a stronger form of authentication. However, while MACs cannot provide non-repudiation, they are more efficient compared to digital signatures. Digitally signing each packet of streaming data is prohibitively expensive (both computationally and with respect to communication overhead per packet).

In unicast communication, MACs support data authentication as follows. Consider two communicating peers, Alice and Bob, holding a secret key for authentication. Alice uses the key and a one-way function to

compute the *keyed hash* (e.g., HMAC [4]) of the message, and sends the message along with the MAC to the receiver. Bob repeats the procedure to compute the MAC, and compares it with the received MAC. If the MACs are identical, Bob knows that the message has not been modified en route. He also knows that he has not sent the message and therefore Alice must have sent it, assuming the authentication key has not been compromised. However, a third party cannot verify whether the message has been sent by Alice or Bob. Therefore, MACs cannot provide non-repudiation.

We can use MACs for authenticating group communications following a similar procedure as above, but with a reduced level of security. Consider a group, consisting of Alice, Bob and Cindy, holding an authentication key. Alice might use a MAC to authenticate a message sent to Bob and Cindy. Bob (or Cindy), however, does not know whether the message has been sent or last modified by Alice or Cindy (or Bob). In general, members of a group can verify only that nonmembers, that is, people who do not hold the group authentication key, have not changed the data in transit. This property that guarantees only that a message was sent (last modified) by a member of the group is referred to as *group authentication* [5].

In contrast, if a member can establish whether the data sender is legitimate or not, we refer to that property as *source authentication*. With source authentication, a member can verify the data source and know that data has not been modified en route. Solutions for source authentication in general are either expensive or complex, and often application dependent.

3.1.1 Providing group authentication

Group authentication of a message implies that the message originated within the group, and has not been modified by entities outside the group. A MAC is used for group authentication, and thus it is rather inexpensive to authenticate even streaming data in real-time. Group authentication has some important applications. Consider, for example, secure communication between entities (e.g., gateways) that trust each other, over the public Internet. Group authentication is sufficient in this case, since members holding the group keys are assumed to be not interested in modifying data sent by other members. Group authentication only serves a limited purpose however, and may not be sufficient for most applications.

Members of the group need a common a key for group authentication (for MAC computation). Thus, we need to be able to establish and update the authentication keys among the members securely. Group key distribution protocols and algorithms described in Chapters 4, 5, and 6, provide ways to establish a common key among members of a group. Along with the

encryption keys, a group manager may also distribute authentication keys [6, 7]. The registration protocol is used to send the keys initially, and the rekey protocol is used to send key updates [6]. A sender can use those keys in a data security protocol (e.g., IPsec ESP, MESP [2], or AMESP) for authenticated group communication.

3.1.2 Providing source authentication

It is often not sufficient to be able to verify that a message originated within the group. Members would like to establish, at least to themselves, the sender of multicast data. Recall that a stronger property is for any third party to be able to independently verify the sender of the data. This, as introduced earlier, is known as non-repudiation.

Application requirements greatly influence the solution space for source authentication. First, an application may require non-repudiation or only source authentication. Next, data transmission may be reliable or lossy. Furthermore, the sender or the receivers may have limited buffer space. Moreover, receivers could have limited computational power (e.g., mobile devices), and, in some cases, receivers' computational capacity may be heterogeneous. Receivers may be at different distances from the sender. Finally, the application may involve bulk data transfer(s) or streaming. In the following, we discuss the source authentication requirements of several different types of applications.

Reliable bulk data transmission. We assume that the data transmission is reliable and the sender has the data available in advance. The receivers can use the data only after all of it has been received. Buffer space is not of concern either at the sender or at the receivers. These flows are referred to as *all or nothing flows* [8]. Multicast ftp and Web cache synchronization are examples of all or nothing flows.

A simple solution may be for the sender to compute the hash of the data and sign it. Notice, however, that an adversary can disrupt this authentication process by changing just a single bit in the flow. Receivers cannot detect this attack until all the data has been received. Furthermore, they cannot identify the portion of the data that has been changed. Thus, they have to request retransmission of the entire flow.

Reliable streaming of stored data. Consider the reliable transmission of data that the sender knows in advance. The sender needs a large buffer to store the data. Since the sender knows the data in advance, it can perform authentication transforms off-line. Receivers are expected to authenticate

and use the data packets as they arrive. Thus, receiver-side buffering requirements are relatively modest.

Lossy streaming of (partially) stored data. This is similar to the above category, except that packets may be lost in transmission. Considering packet losses, receivers should be able to verify the authenticity of portions of data that are received. Video-on-demand is an application that falls into this category. These applications may be able to tolerate delayed (with fixed delay) verification.

Real-time streaming with packet loss. Real-time applications require the sender to transmit as soon as the data becomes available. Thus the sender needs to apply the authentication transforms in real-time. Considering lossy transmission, each packet's authenticity must be independently verifiable. Multimedia conferencing is an application that requires real-time streaming in the presence of packet losses.

3.2 Digital signatures for source authentication

Source authentication can be achieved using digital signatures. The sender divides the data into blocks. For each block, it computes a hash, signs the hash, and sends the signature along with the data block. There are several issues to address. As the block size increases, the sender needs to perform fewer digital signatures and members need to perform fewer signature verifications. However, a member needs to receive an entire block before verifying its authenticity. For smaller block sizes, the number of signatures and verifications increases. The advantage is that members need not wait long before verifying and using a block.

Note that this procedure has several useful properties. Each block is individually authenticated and thus independently verifiable. Furthermore, this technique provides block-level non-repudiation. But all this comes at a cost. Signing and verifying each block is computationally expensive. Moreover, each block needs to carry its own signature, which results in excessive communication overhead. Independent packet authentication makes signing each packet look attractive. However, in practice, signing each packet in a high data rate real-time stream may not be feasible. Using 1-time signatures [9] is a slightly efficient alternative to signing each packet. But, 1-time signatures require a large number (60–80 [9]) of hash computations, and cannot handle packet losses.

Given that digital signatures can provide individual packet authentication, several solutions have been proposed that amortize their cost over a stream or a block. Some of the solutions are applicable to specific application scenarios, while others make assumptions about the relationship between the sender(s) and receivers (e.g., that they are synchronized). A few solutions simply trade the excessive computational cost of digital signatures with communication cost.

In the next section, we describe schemes that amortize a digital signature over a block of data. These mechanisms reduce the number of digital signatures at the expense of increased communication overhead per packet.

3.2.1 Block signatures and individual packet authentication

For delay-sensitive flows, signing each packet is too expensive. However, we still need authentication mechanisms so receivers can verify packets as they are received. In other words, each packet must be independently verifiable. In the rest of this section, we describe a couple of techniques called star hashing and the more efficient tree hashing [8]. These schemes amortize the cost of a signature over a block of packets, and they require a sender-side buffer that can hold the block of packets. Thus, star and tree hashing are sometimes referred to as block hashing.

Star hashing

The sender divides a block of data into m packets. It signs a hash of the block (*block hash*), and thus amortizes the cost of the signature operation over m packets. For individual packet authentication, it computes the hashes h_1, h_2, …, h_m, of the m packets. The block hash, h, is a hash of the concatenation of all the individual packet hashes. Thus, $h = \text{hash}(h_1, h_2, \ldots, h_m)$, where $h_i = \text{hash}(P_i)$. Note that P_i represents packet i. With each packet, the sender includes the block hash and the hashes of all the packets in the block. It also sends the relative position of the packet in the block.

Figure 3.1 illustrates star hashing. The edges (or leaves) represent packet hashes, and the root represents the block hash. The hash dependency graph is a star and hence the name star hashing. The figure also illustrates the relationship between the packet hashes and the block hash; that is, a block hash is dependent on all of the packet hashes.

Upon reception of P_i, the receiver computes its hash, h_i' (the prime indicates receiver-side computation). It repeats the block hash computation procedure as described earlier, but using h_i' instead of h_i. If the signed block

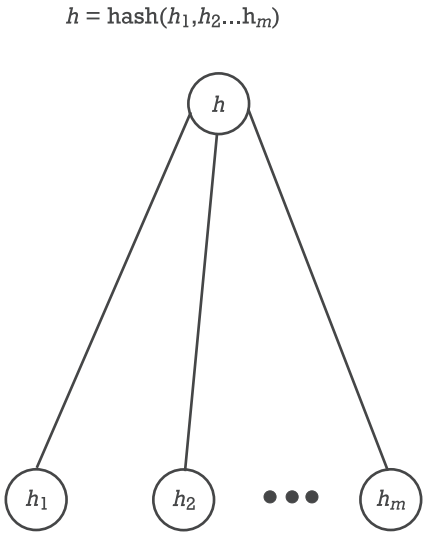


Figure 3.1 Star hashing.

hash is identical to the computed block hash, the receiver knows that P_i is authentic. Furthermore, it also knows that the rest of the hashes are also authentic. Otherwise, the block hash comparison would have failed. For other packets in the same block, the receiver needs only to compute and verify whether the computed packet hash is equal to the received hash. In other words, there is only a single signature verification operation per block, at the receivers.

Receivers perform one signature verification operation and two hash computations to verify the authenticity of the first received packet of a block. For the other packets of that block, a single hash computation and comparison suffices. While the computational overhead in star hashing is minimal, the same cannot be said about the communication overhead. Recall that each packet needs to carry the hashes of all the packets (m) in a block, as well as a digital signature. A hash is typically 20 bytes (*secure hash algorithm*, SHA-1) [10] in length, whereas a digital signature is about 128 bytes in size (e.g., 1,024-bit RSA [11]).

Tree hashing

Tree hashing [12] employs a different block hash computation mechanism than in star hashing. While the hash computation mechanism is itself slightly complicated and inefficient, this scheme reduces the communication overhead associated with hashing.

The sender divides a block of data into m packets and computes the individual packet hashes. For block hash computation, it associates each individual packet hash with a leaf node of the *hash tree* (see Figure 3.2). Each internal node's hash is the hash of the concatenation of the children's hashes. Thus $h_{12} = \text{hash}(h_1, h_2)$. Using this function, the sender recursively computes the root node's hash. With each packet, the sender includes the signed block hash, the packet ID, and the hashes of siblings of all the nodes in the current packet's path to the root.

Receivers follow a similar procedure to that in star hashing to verify the authenticity of each packet. A receiver first computes the hash of the received packet. It uses the computed hash and the received hashes to compute the root hash. If the computed root hash is identical to the signed block hash, the received packet is authentic.

We use Figure 3.3 to illustrate the computation process at a receiver. Let us say P_2 is received first. The receiver computes the hash h'_2 of the received packet. With P_2, the sender includes the hashes of the siblings of all nodes in P_2's path to the root. In our example, that implies hashes h_1, h_{34}, and $h_{5.8}$. The receiver computes $h'_{12} = \text{hash}(h_1, h'_2)$, $h'_{1.4} = \text{hash}(h'_{12}, h_{34})$, and $h'_{1.8} = \text{hash}(h'_{1.4}, h_{5.8})$. If $h'_{1.8}$ and the signed block hash, $h_{1.8}$ are identical, P_2 is authentic. Furthermore, the received and the computed hashes, that is, h_1, h_{34}, and $h_{5.8}$, and h_2, h_{12}, and $h_{1.4}$, are authentic as well. The receiver caches the verified nodes of the hash tree for efficient verification of the other packets in the same block.

Figure 3.4 illustrates the advantage of caching verified hash nodes. For example, if P_4 is received next, the receiver needs only to compute h'_4 followed by h'_{34}. Notice that h_{34} is among the verified nodes; therefore it is sufficient to compare h'_{34} to h_{34}. If they are identical, P_4 is authentic.

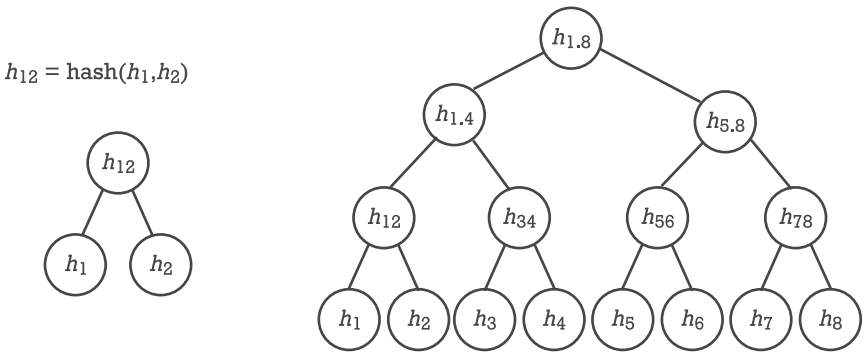


Figure 3.2 Tree hashing.

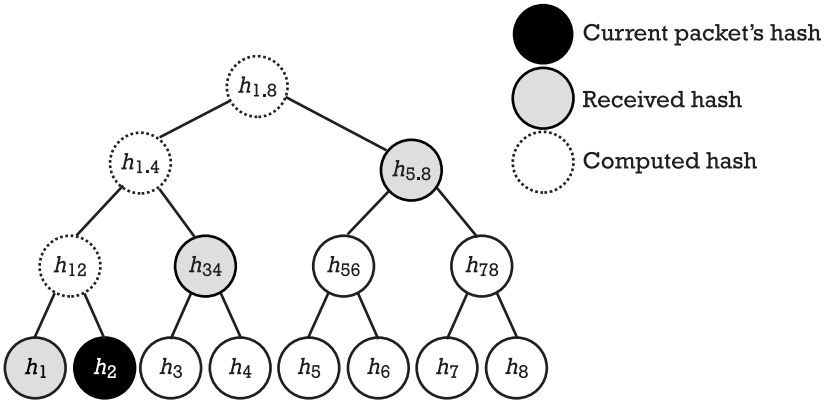


Figure 3.3 Block hash computation in tree hashing.

Authenticity verification of the first received packet of a block consists of a digital signature verification operation, and computation of all hashes in the path from the packet's position in the tree to the root. In all, the receiver needs to compute $O(\log m)$ hashes. Future packet verifications require fewer hash computations and no digital signature verification operations.

Tree hashing trades additional computational overhead (of multiple hash computations) with the linear (in block size) communication overhead in star hashing. Each packet carries only $O(\log m)$ hashes along with the signed block hash. Authenticity verification may require as many as $O(\log m)$ hash computations. Caching verified nodes decreases the number of hash computations for subsequent packet verifications of the same block.

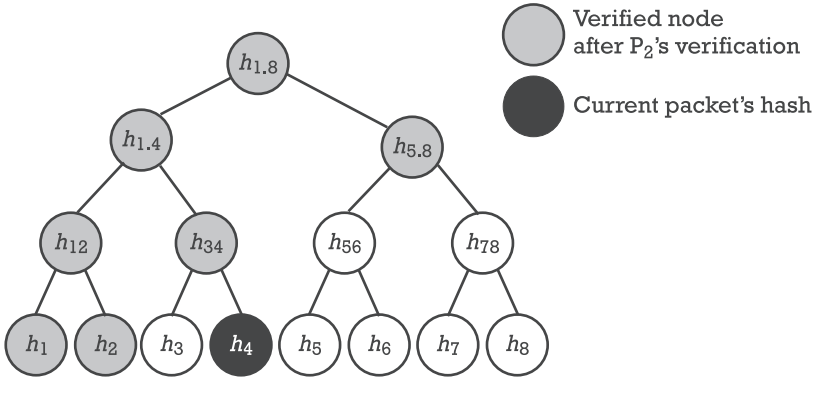


Figure 3.4 Efficient hash verification in tree hashing.

Block signatures used by both star and tree hashing have some additional disadvantages as well. The sender needs to have a block of data available in advance to compute the hashes and the signature. Thus, these schemes delay the flow transmission. Furthermore, the sender needs a buffer to hold a block of data. Thus, if m is large, we need a larger buffer at the sender, and incur increased communication overhead. Finally, the sender may transmit the block of packets at once after authentication, which may result in bursts that are prone to losses [13].

However, the number of signatures per flow decreases with increasing m. Also, no receiver-side buffering is required, which is a plus. Receivers can verify each packet as it is received, and supply it to the application. Block hashing also supports non-repudiation (since the hash is signed by the sender).

3.3 Hash chaining to authenticate streaming data

Chaining is a common solution for amortizing the cost of digital signatures for authenticating streaming data. Consider the reliable transmission of streaming data (e.g., a PPV movie). We assume that the sender has the whole data stream available in advance.

Hash chaining works as follows [9]. First, the sender divides the data into n blocks. Next, it computes the hash, [using for example, message digest 5 (MD5) [14] or SHA-1] of the first block, signs the hash payload, and sends the signature to the receivers. It then sends each block except the last block, appended with the hash of the next block. Figure 3.5 illustrates hash chaining. Data is divided into n blocks named B1, B2, . . ., Bn. In the figure, the first block's hash B1, is signed. The first block consists of B2's hash along with the data. This continues until the block B$n-1$, which contains the hash of Bn. The final block does not contain a hash.

Each member verifies the sender's signature, extracts the hash of the first block, and stores the hash. When the first block arrives, the receiver computes the hash and compares it to the stored hash. If the hashes are

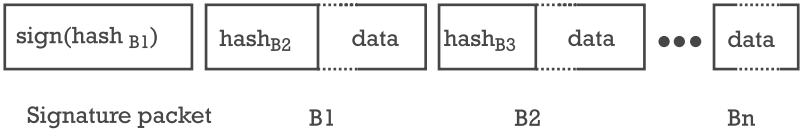


Figure 3.5 Hash chaining.

identical, the first block's integrity and authenticity is established. The member then extracts the second block's hash and stores it. It repeats this verification procedure until the last block. If the packets are received in sequence, the receivers only need a buffer to hold a block and a hash.

Note that due to the authentication chain that ends up in a signature packet, hash chaining provides non-repudiation. Any third party can repeat the verification procedure independently, and determine the data source.

Hash chaining is very efficient. There is only one public key operation at the sender (a signature) and each member (a verification). In addition, all parties compute n hashes. However, hash computation is typically 1,000 times faster than public key computations. The communication overhead is also minimal. Overall, this scheme adds a signature (e.g., 128 bytes) and n hashes (n * 20 bytes, with SHA-1 for hashing) worth of overhead, in authenticating the entire stream.

Unfortunately, hash chaining has some severe limitations for practical use. First, this scheme works only when the sender has the entire data stream available in advance. More importantly, authentication can be done only in the strict sending sequence of the blocks. This scheme cannot tolerate packet loss. Receivers cannot authenticate any future blocks, once any portion of a block is lost in transit. Out-of-order block reception is also troublesome. An out-of-order block must be buffered until all the blocks leading up to it are received and verified.

3.3.1 Graph representation of hash chaining

Hash chaining is better understood with a graph representation [13, 15–17] of authentication. Each packet containing authentication information represents a node in the graph. Thus a packet containing a hash or a digital signature is a node in the graph. A directed edge represents an authentication relationship (dependency) between two packets. If a packet P_i contains a hash of another packet P_j, then there is directed edge from P_j to P_i. Similarly, there is a directed edge from a data packet P_i to a signature packet that directly authenticates P_i. Finally, for source authentication of a packet P_j, there must be a path between P_j and a signature packet.

Figure 3.6 illustrates a hash chain represented as a directed graph. The graph illustrates several points noted earlier in the description of the hash chaining algorithm. Most importantly, there is only a single path from any node to the origin. Note also that from a receiver's perspective, if a packet is lost in transit, the corresponding node from the graph is deleted. Consequently, there cannot be a path from any future packets to the

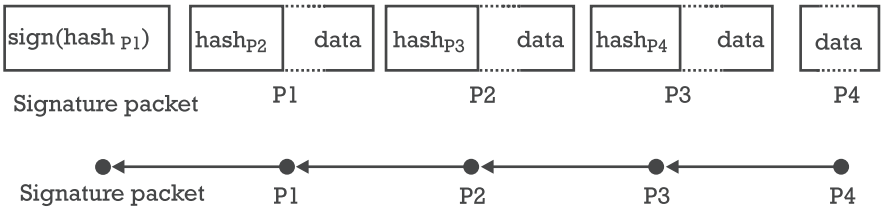


Figure 3.6 A hash chain represented as a graph.

signature packet. Thus the graph representation captures hash chaining's inability to handle packet losses.

Simple hash chaining has a couple of shortcomings that need to be addressed. First, the sender needs to know the entire stream in advance. Second, this authentication scheme cannot tolerate packet loss. Recall that we would like to support a wider suite of applications that need authentication of real-time streaming in the presence of packet losses.

Several variations [13, 15, 16] of the simple hash chaining presented in Section 3.3 have been proposed to address these shortcomings. The basic ideas are to use: (a) forward authentication (or a combination of forward and backward authentication) to support real-time streaming, and (b) multiple hash chains for loss tolerance.

Notice that hash chaining as explained earlier uses backward chaining, that is, P_i contains authentication information of P_j, where $i < j$. In the simplest form of chaining, $j = i + 1$. Backward chaining requires the whole stream to be available before transmission, but has the advantage of immediate authentication at the receivers (assuming in-order delivery of packets). Forward chaining is the opposite of backward chaining in that the signature packet follows the data packets. Thus the sender can start sending real-time data as it is available, whereas the receivers cannot authenticate data until the signature packet arrives. Thus forward chaining results in delayed authentication.

The key to loss tolerance is to send the hash of a packet in multiple packets, and include multiple hashes (of different packets) in the signature packets. Reliable transmission of signature packets is crucial in these schemes. Notice that in supporting loss tolerance, the per-packet communication overhead increases. Moreover, as the number of signature packets increases, the signing and verification costs at the sender and the receivers, respectively, also increase. In the rest of this section, we discuss a few different approaches that address authentication of flows with varying loss tolerance.

3.3.2 Efficient multichained stream signature

Efficient multichained stream signature (EMSS [13]) is an authentication scheme for providing non-repudiation service for lossy transmission of a real-time flow. EMSS uses forward chaining and thus there is no data packet buffering required at the sender. EMSS designers experimented with several hash sequences to understand the effectiveness of multiple hash chains.

The basic scheme defines a chaining sequence $a - b - c$, with static (i.e., the values of a, b, and c do not change) edges, as follows: each packet P_i contains the hashes of P_{i-a}, P_{i-b}, and P_{i-c}, and P_i's hash is sent with packets P_{i+a}, P_{i+b}, and P_{i+c}. In the graph terminology introduced earlier, the node P_i has incoming edges from packets P_{i-a}, P_{i-b}, and P_{i-c}, and outgoing edges to P_{i+a}, P_{i+b}, and P_{i+c} [13].

The chaining sequence $a - b - c$ has a (maximum) degree of three. Thus each packet holds (at most) three hashes, and three packets carry its hash. The hash chains eventually end up in a signature packet. If there is a path from a packet P_i to a signature packet, it is non-repudiable. However, in the absence of such a path, the packet cannot be authenticated. Packets whose authentication cannot be verified are dropped. Thus, EMSS only supports probabilistic authentication verification. Even authentic packets may be dropped by a receiver simply because it cannot verify them. Finally, note that the distance to a signature packet, (the number of edges in the path from a packet to a signature packet) may have an impact on verification probability.

Simulations with various combinations of schemes determined that most of the combinations of degree *six* have a verification probability over 90% [13]. Thus EMSS designers suggest the use of random chaining sequences. Further simulations showed that random chaining sequences had a higher verification probability compared to static sequences, as the distance to a signature packet increases.

Recall that in a degree k hash chain, each packet's hash is sent in k different packets. In other words, each hash is repeated k times in a stream. Alternatively, the sender could encode [using *forward error correction* (FEC), for example] a packet's hash, and split it into m chunks such that any $m' < m$ chunks are sufficient to rebuild the hash upon reception. If the size of the encoded hash is limited to the number of bits in k hashes, such encoding does not increase the communication overhead. Simulations show that this technique further improves the verification probability without having to increase the communication overhead [13].

3.3.3 Augmented chaining

Augmented chaining is an authentication scheme for transmission of (partially) stored data in the presence of bursty losses. First, we describe a *base chain* to which additional nodes are added to form an *augmented authentication chain* [15]. Each node in the chain represents a data packet, and the chain terminates with a signature packet.

In the base scheme, each packet P_i's hash is included in packets P_{i+1} and P_{i+a}, where a is an arbitrary constant. Thus this scheme tolerates loss of $a-1$ successive packets. Each packet carries two hash packets, and the last packet in the chain, P_n, is signed. The base scheme does not require any sender-side packet buffering, but requires a buffer to hold a hashes. Receivers cannot verify the authenticity of any packet until the corresponding signature packet arrives. Thus this scheme requires the receiver to be able to buffer at least n packets.

Augmented chaining improves upon the base scheme by adding $p-1$ additional nodes between each pair of nodes in the base chain, with each node representing a data packet. This scheme can tolerate loss of $p(a-1)$ successive packets. Figure 3.7 illustrates base chaining and the construction of an augmented chain for $p=2$. Augmented chains of arbitrary size (any value p) are also defined in the literature [15]. Augmented chaining uses a combination of backward and forward chaining (with no loops in the resulting directed graph [15]). Thus, unlike in base chaining, the sender cannot transmit data in real time. Furthermore, we need a finite buffer to hold packets at the sender. Specifically, the sender needs a buffer to hold p packets and $a+p-1$ hashes. Augmented chaining also results in delayed authentication at the receivers. The receivers also need a buffer to hold n packets.

3.3.4 Piggybacking

Piggybacking [16] is applicable to streams that have groups of packets of varying importance. Moving picture experts group, MPEG-2, streams, for example, can benefit from this scheme. The sender needs to have the entire stream available before transmission. It divides the packets in the stream into several priority classes. Higher priority packets are evenly spaced in the data stream for loss tolerance.

Edges in the authentication graph always start from a packet of lower priority and point to a packet of a higher priority. Thus, lower priority packets ''piggyback'' on higher priority packets. These general principles can

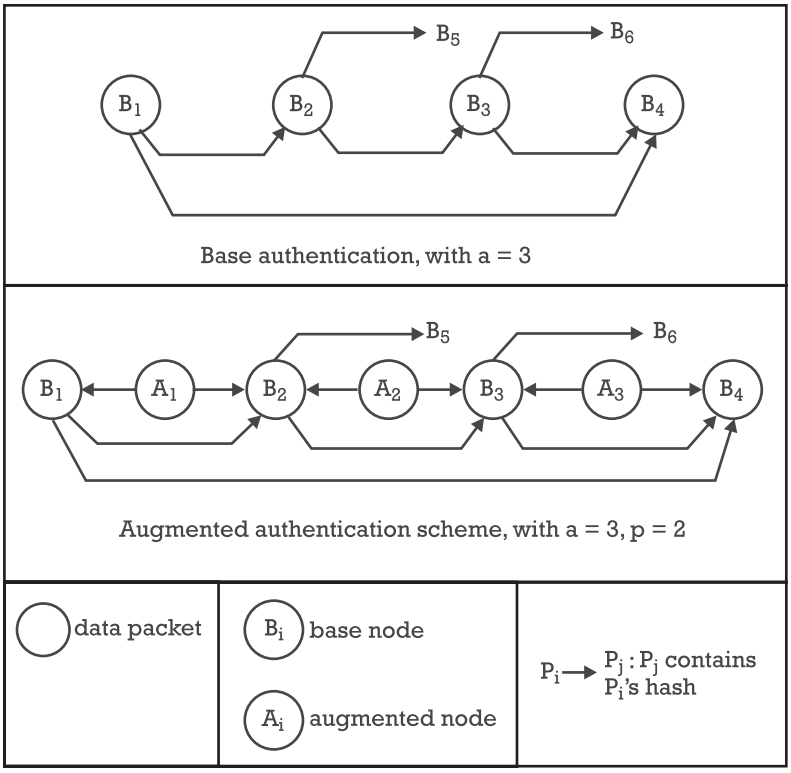


Figure 3.7 Illustration of augmented chaining.

be applied to construct hash chain–based authentication schemes that tolerate single and multiple bursts of packet losses [16].

3.3.5 Discussion on hash chaining for authentication

Hash chaining schemes for unreliable stream authentication typically include more than one hash per data packet. Similarly, a signature packet contains multiple hashes for higher verification probability. In most schemes, the sender transmits more than one signature packet. Whether there is one signature packet or more, all the schemes in the literature pay special attention to the transmission of signature packets.

Both EMSS and augmented chaining send signature packets at the end. This introduces delayed authentication at the receivers, and the receivers also need to buffer packets waiting to authenticate them. Thus, these schemes are prone to DoS attacks. In general, hash chaining schemes can be reversed in direction to include signature packets at either end of a stream

(or a portion thereof) [16]. Schemes with signature packets at the beginning of a stream support immediate authentication at the receivers. Protocols that include signature packets at the end support real-time streaming and require no (or a small amount of) buffering at the sender.

The interdependency between packets for authentication results in packets being dropped due to the inadequacy of the authentication protocols. More precisely, packet losses during transmission may result in several other packets being dropped, since they cannot be authenticated. The advantage of chaining, however, is that when we do authenticate a packet, we have support for non-repudiation as well.

The design of a signature packet, as noted earlier in this discussion, is crucial to the verification probability of schemes that use multiple hash chains. Some of the factors that influence the verification probability are number of hashes per signature packet, and frequency and reliable transmission of signature packets. Simple solutions such as multiple transmissions of a single signature packet or techniques such as FEC encoding [13, 17] might be beneficial to reliable transmission of signature packets. Some designers [16] suggest feedback-based reliable transmission of signature packets, but that might result in feedback implosion in large groups.

3.4 MAC-based source authentication of unreliable streams

One-way key chains for MAC-based authentication have been used for sending authenticated link state routing updates [18]. The sender generates a hash chain [19] of length l, using a random secret r and a one-way function h. The elements of the hash chain $h(r), h^2(r), \ldots, h^l(r)$, where $h^2(r) = h(h(r))$ are used in the reverse order as MAC keys to authenticate link state updates, as time progresses. The routers are assumed to be loosely synchronized, and the MAC keys are revealed by the sender after a known delay. In this section, we describe a protocol that uses these techniques and others for efficient authentication of lossy real-time streams.

The timed efficient stream loss-tolerant authentication (TESLA) [20] protocol provides comprehensive support for source authentication of a multicast stream using MACs. Recall that a MAC is based on a symmetric key, and all members of the group must know the key for authentication. However, as we discussed in Section 3.1, symmetric keys only support group authentication. TESLA gets around this limitation by committing to a MAC key first, and revealing it after a preadvertised delay. Next, we introduce some terminology and concepts to help describe TESLA.

Time intervals. Each packet P_i is authenticated separately, with a MAC. Time is divided into t intervals of duration T_{int} each. The sender may transmit zero or more packets within each interval I_j, $1 \leq j \leq t$. This flexibility of being able to send any number of packets within an interval allows the sender to adapt to dynamic transmission rates of applications. Corresponding to each interval I_j, there is an authentication key k'_j. In other words, all packets sent within the interval I_j are authenticated with the MAC key k'_j.

MAC keys. The sender generates a key chain [19], $k_1, k_2, \ldots, k_t$, using a one-way function f. To do so, it first randomly generates the last key k_t of the chain, and uses the equation $k_{j-1} = f(k_j)$, to generate the rest of the keys. Note that by the definition of a one-way function, given k_{j-1}, it is computationally infeasible to compute k_j. The sender generates the MAC keys[1] $k'_j = g(k_j)$, $1 \leq j \leq t$, where g is another one-way function, from the key chain. Figure 3.8 illustrates the key chain and MAC key derivation in TESLA.

One of the properties of the key chains serves an important purpose in TESLA. Given a key, k_a in the chain, a receiver can compute all the keys $k_i, i < a$, by repeatedly applying the one-way function f. Thus, even if some keys are lost in transmission, a receiver can compute them as long as it receives a future key in the chain. In other words, the protocol is loss tolerant.

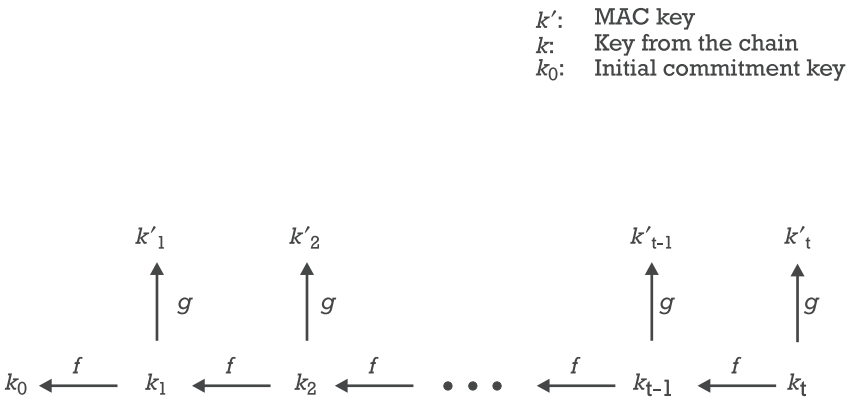


Figure 3.8 Illustration of MAC key chaining for TESLA.

1. This is to avoid using the same keys for different purposes, that is, key generation as well as authentication [21].

Commitment to a key chain. A sender can commit to a key chain by either signing a packet containing a key from the chain, or including the key in an authenticated packet. The *commitment* is to the key chain terminating with the included key. For example, to commit to the key chain, $k_1, k_2, \ldots, k_t$, the sender would transmit an authenticated packet containing $k_0 = f(k_1)$.

Loose time synchronization between sender and receivers. Receivers need to be in loose synchronization with the sender, that is, they need to know an upper bound on the sender's time. Thus, if the time difference between the sender and a receiver is δt, we assume that the receiver knows a Δt such that $\Delta t \geq \delta t$ [20].

Disclosure delay. The disclosure delay indicates the time (in number of intervals) that a receiver needs to wait before being able to authenticate a packet in the interval I_j. Disclosure delay has implications on buffer space requirements at the receivers, as well as on authentication delay. A short disclosure delay may result in a receiver not being able to successfully authenticate many packets. TESLA designers suggest a disclosure delay of $\lceil RTT/T_{int} \rceil + 1$, where RTT is an upper bound on the round trip time between the sender and the receivers.

3.4.1 TESLA initialization

To initialize a receiver, the sender transmits a digitally signed packet that contains: information about time intervals, time synchronization information, a commitment to the key chain starting at the current interval, and the disclosure delay, d. A receiver's subscription to the secure group may start at the beginning of the transmission or midstream. For receivers joining midstream, the sender may have to send the bootstrapping information via unicast.

The group manager may initialize TESLA during the registration protocol defined by the group key management architecture [6]. Recall that one of the purposes of registration is to initialize the data security SA (see Section 4.4), of which source authentication is a component. Hosts that are members of the group at the beginning may also be initialized via unicast. The sender may also choose to initialize them en masse. Group initialization is more efficient, since the sender needs to sign only one packet and can send it via multicast. The problem here, however, is Reliable Multicast delivery.

In general, the signed initialization packet from the sender contains:

- The current interval index j;
- The beginning time T_j of interval I_j;
- The interval duration T_{int};
- A commitment to the key chain starting at k_i, $i < j - d$ (if $j < d$, then $i = 0$);
- Key disclosure delay d (in number of intervals).

Receivers use the above information in determining the authenticity of packets received. Recall that TESLA gets around the limitations of MAC-based authentication for groups by using loose time synchronization between the sender and the receivers. More specifically, the sender delays revealing a MAC key used in authenticating a packet until all (in practice, most of) the receivers have received the packet. Thus, upon reception of a packet, receivers in TESLA need to be able to determine whether the sender has already revealed the MAC key used in authenticating that packet. If, according to a receiver's calculations, the sender has revealed the MAC key already, the receiver discards the data packet. Otherwise, it caches the packet, pending authenticity verification using a MAC key to be received.

3.4.2 MAC-based authentication of packets by the sender

In TESLA, a sender can start sending packets as soon as they are available. In other words there is no necessity for sender side buffering.[2] Senders do need to follow the time intervals, use the corresponding MAC keys to authenticate packets, and, finally, wait before revealing the MAC key.

Thus a sender authenticates each data packet with the current interval's key, and includes the authentication information with the data. It also includes the MAC key (more precisely the key from the key chain that is used to compute the MAC key) used to authenticate packets sent d intervals ago. In other words, the sender authenticates the packet P_i sent in interval I_j with k'_j, and includes the MAC with the packet. In addition, it sends the key k_{j-d}, which the receivers use to verify the authenticity of packets sent in the

2. TESLA does require receiver-side buffering. Later in this section, we describe an alternative scheme that uses sender-side buffering to avoid DoS attacks on receivers' buffers.

interval $I_{j-d}, j > d$. Figure 3.9 illustrates the authentication information sent along with a data packet in TESLA.[3]

3.4.3 Packet processing at the receivers in TESLA

Receivers verify authenticity of a packet in two phases. First, for each packet received, they need to verify TESLA's security condition. This is to verify that at the time of reception, the sender has not yet revealed the corresponding authentication key to other members. For this verification, upon reception of packets in the interval I_j, receivers check to ensure if the sender could be in interval I_{j+d}. Recall that due to the loose time synchronization established during initialization, a receiver knows within Δt the time at the sender. Using this information, along with the interval duration, the sender's time during initialization, and the current packet's interval index, the receiver computes the time at the sender, at the time of reception of the packet.

For example, consider the reception of P_i sent in interval I_j. If the current time at a receiver is t_c^r, it knows that the time at the sender was $t_c^r \pm \Delta t$ and the current interval at the sender, $j_c^s = (t_c^r \pm \Delta t - T_0)/T_{int}$. The receiver then needs to verify that $j_c^s < j + d$.

If a packet fails the security condition, the receiver discards it. Otherwise, the receiver caches the packet for authenticity verification in the future. Whether the security condition fails or not, a receiver can use the key revealed (from the key chain) in the current packet to authenticate already cached packets. Note that a receiver can independently verify the authenticity of keys from the key chain. For example, if k_{i-d} is the revealed key, the sender computes $k_{i-d-1} = f(k_{i-d})$ and compares it with its own copy of that key. Note that even if a receiver does not have an authenticated copy

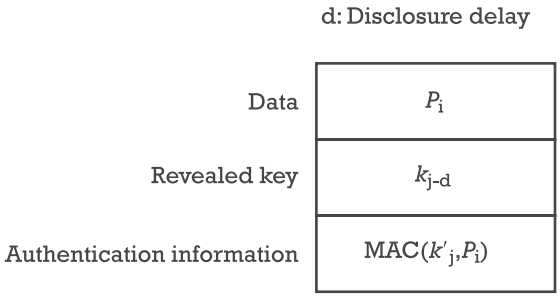


Figure 3.9 Contents of a TESLA packet.

3. For a complete specification of a TESLA packet, see the corresponding Internet draft [21].

of k_{i-d-1}, it will have at least one authenticated key from the chain, received during the initialization phase. Figure 3.10 illustrates intervals, authenticated packets, and key disclosures in TESLA.

We close this section by reiterating that TESLA authentication is loss tolerant. For example, even if all the packets sent in an interval are lost (and therefore the key revealed during that interval), a receiver can still authenticate packets from future intervals. This property is due to the definition of the key chain. For example, packets sent in interval I_j can be authenticated even if all the packets sent in interval I_{j+d}, where k_j is revealed, are lost. A receiver can still compute k_j from any $k_{j+m}, m \geq d$.

3.4.4 Enhancements to TESLA

The disclosure delay is very crucial to the effectiveness of TESLA. If d is too low, receivers might end up not being able to authenticate many packets. In particular, receivers too far from the sender may not be able to successfully verify the security condition in most cases. Alternatively, if the disclose delay is very high, receivers may have to cache several packets. Thus, the receiver side buffer needs to be very large. Furthermore, there is a long delay before a receiver can provide the received (and authenticated) data to the application.

Immediate authentication. TESLA designers propose a mechanism that allows immediate authentication of packets by the receivers at the expense of sender-side buffering. Thus an application might make the choice of either sender or receiver-side buffering.

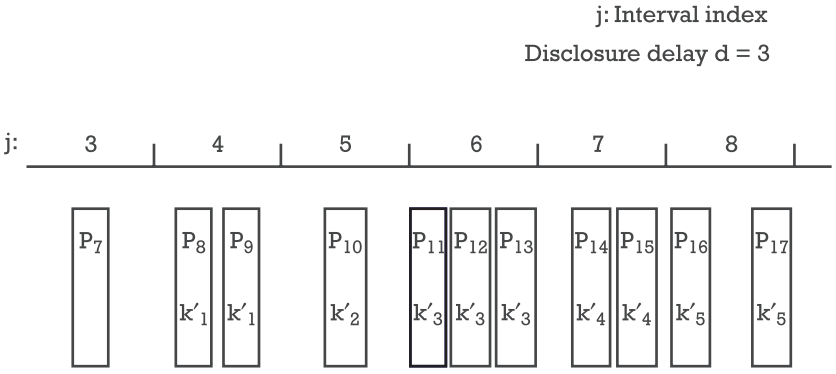


Figure 3.10 Illustration of TESLA intervals.

We provide a sketch of the scheme in the following. With each packet, the sender includes a hash of a future packet. When the receiver authenticates a current packet, it also would have authenticated the packet whose hash is included with the current packet. Unfortunately, this scheme is not efficient. First, it increases per-packet overhead. More importantly, it is not robust to packet loss. For example, if the packet containing the hash of the current packet were lost, the sender could not immediately verify the authenticity of the current packet.

Handling heterogeneous receivers. It is possible that a single disclosure delay is not suitable for all receivers. In other words, the sender may need to define different key chains for different sets of receivers. For example, receivers near the sender might need a smaller disclosure delay (due to smaller round trip time) whereas receivers farther away might need a larger delay. Note that the stream rate does not change. In other words, the sender transmits only one stream, but it does need to compute and include multiple MACs with each packet. Without getting into the details [20], it can be seen that the security condition will not be violated. Although, some receivers might be able to authenticate packets earlier than others, they cannot change the contents of the stream. This is because keys on which the ''distant'' receivers base their trust, are not revealed any earlier. Thus receivers that verify authenticity early cannot illegally inject authenticated packets into the stream.

3.4.5 Applicability analysis of TESLA

TESLA amortizes the cost of a single digital signature operation over several packets (perhaps the entire stream). As long as the sender knows the time that a stream is going to last, it can use a single key chain. Alternatively, the sender can efficiently start a new key chain, without any need for digital signature operations [13]. The sender performs a single MAC computation per packet. With each data packet, it only needs to send a key from the key chain and a MAC. It may optionally send the current interval index as well.

Each receiver needs only to perform a MAC computation to verify a packet. The security condition verification is not computationally significant, but it may result in packets being dropped. Thus packets may be dropped even if they are authentic and have not been compromised. This drop probability is dependent on the timing constraints (d and Δt) of the protocol.

Moreover, time synchronization may be difficult to implement in several application scenarios.

The large receiver-side buffering requirement is also of concern. Since there is no limit on the packet sending rate, the number of packets received in d intervals may be very large. This may result in buffer overflows. While there is a way to trade receiver-side buffering with sender-side buffering, the latter scheme is relatively less robust.

3.5 IPsec ESP and MESP

For group authentication, the sender only needs to include the cryptographic checksum computed using the group authentication key, with each packet. Thus IPsec ESP as defined in RFC 2406 can be used for this purpose. ESP also supports transport of per-packet digital signatures for source authentication [22].

Source authentication techniques that amortize the cost of digital signatures for efficiency (described earlier in this chapter), require additional protocol support for interoperable implementation and successful deployment. More specifically, notice that hash chaining schemes send a variable number of hashes with each packet, and signatures in some packets. Similarly, block hashing schemes send multiple hashes as well as a digital signature with each data packet. Finally, MAC-based schemes include a MAC and an authentication key with each packet. All these schemes may also send additional information to identify the hashes or MACs. Thus, we need a secure multicast data transport protocol that supports inclusion of a variable amount of source authentication information with each packet.

Efforts are under way at the IETF MSEC working group to standardize a new protocol known as multicast ESP (MESP) [23]. This protocol has several goals, including:

- Providing a combination of data encryption, source authentication, and group authentication of multicast data;
- Support for identifying multicast groups with $<$ *source address*, *destination address* $>$ pair in the security parameter index (SPI) of an SA;
- Protection against replay and DoS attacks.

The order of the three transforms, that is, group secrecy (GS), group authentication (GA), and source authentication, is a point of contention.

It is a cryptographically sound practice to authenticate encrypted data [24].[4] Furthermore, GS followed by GA allows us to verify the authenticity of the message (albeit using a weaker form of authentication) before performing the expensive decryption operation on data. On the other hand, source authentication of unencrypted data is required for non-repudiation. Thus, MESP may use the order: source authentication, GS, and GA on the sender side and GA, GS, followed by source authentication on the receiver side.

3.6 Summary

Source authentication is a challenging problem in group communications. In the context of groups, applications may seek three levels of protection: group authentication, source authentication, or non-repudiation.

Group authentication is relatively straight forward to provide. A group manager can distribute the authentication key to be used in a MAC computation, following a procedure similar to that of key distribution for confidentiality. However, a majority of applications may need source authentication, while some may need non-repudiation guarantees. Unfortunately, source authentication of real-time streams in the presence of packet losses is a challenging problem.

Digitally signing each packet is the simplest solution for providing source authentication and non-repudiation. However the computational and communication overhead involved in doing so makes this solution impractical. The alternative is to amortize the cost of digital signatures over multiple packets.

An ideal solution would support immediate authentication of real-time streams with a reasonable overhead. Unfortunately, there are no ideal solutions for source authentication of group communications. For example, hash chaining works only for reliable transmission of stored data. Schemes based on multiple hash chains require either sender-side or receiver-side packet buffering. They have further limitations such as delayed or probabilistic packet authenticity verification. Block hashing supports immediate authentication of partially stored streams, but has more per-packet communication overhead than digitally signing each packet. While MAC-based chaining has the least computational and communication

4. Note that encrypting after authentication is also safe if the encryption mode is cipher block chaining (CBC) or if a stream cipher is used [24].

overhead, it requires time synchronization between the sender and the receivers. Furthermore, it only supports delayed verification, and may result in dropping perfectly legitimate packets.

Thus, selecting a source authentication mechanism for group communication is not easy. A thorough understanding of the nature of the application, buffer space availability at the sender and receivers, the level of authentication required, and the application's tolerance to losses and delays would be very useful in the selection process.

References

[1] Kent, S., and R. Atkinson, ''IP Encapsulating Security Payload (ESP),'' RFC 2406 (proposed standard), IETF, November 1998.

[2] Canetti, R., P. Rohatgi, and P. Cheng, ''Multicast Data Security Transformations: Requirements, Considerations, and Proposed Design,'' draft-irtf-smug-data-transforms-00.txt, IRTF, June 2000, work in progress.

[3] Miller, K., et al., Starburst Multicast File Transfer Protocol (MFTP) specification, draft-miller-mftp-spec-03.txt, IRTF, April 1998, work in progress.

[4] Krawczyk, H., M. Bellare, and R. Canetti, ''HMAC: Keyed-Hashing for Message Authentication,'' RFC 2104 (informational), IETF, February 1997.

[5] Canetti, R., et al., ''Multicast Security: A Taxonomy and Efficient Constructions,'' in *Proc. of IEEE INFOCOM*, New York, March 1999.

[6] Baugher, M., et al., ''Group Key Management Architecture,'' draft-ietf-msec-gkmarch-02.txt, IETF, March 2002, work in progress.

[7] Baugher, M., et al., ''Group Domain of Interpretation for ISAKMP,'' draft-ietf-msec-gdoi-04.txt, IETF, March 2002, work in progress.

[8] Wong, C.K., and S. S. Lam, ''Digital Signatures for Flows and Multicasts,'' *IEEE/ACM Trans. on Networking*, Vol. 7, No. 4, August 1999, pp. 502–513.

[9] Gennaro, R., and P. Rohatgi, ''How to Sign Digital Streams,'' in *Advances in Cryptology—CRYPTO*, Santa Barbara, CA, Springer-Verlag, LNCS 1294, August 1997, pp. 180–197.

[10] *Secure Hash Algorithm SHA-1*, NIST FIPS PUB 180-1, April 1995.

[11] Rivest, R. L., A. Shamir, and L. M. Adleman, ''A Method for Obtaining Digital Signatures and Public-Key Crypto Systems,'' *Communications of the ACM*, Vol. 21, 1978, pp. 120–126.

[12] Merkle, R. C., ''A Certified Digital Signature,'' *Advances in Cryptology—Crypto*, Berlin, Springer-Verlag, LNCS 435, August 1989, pp. 218–238.

[13] Perrig, A., et al., ''Efficient Authentication and Signing of Multicast Streams over Lossy Channels,'' in *Proc. of IEEE Symposium on Security and Privacy*, Oakland, CA, May 2000, pp. 56–73.

[14] Rivest, R., ''The MD5 Message-Digest Algorithm,'' RFC 1321 (informational), IETF, April 1992.

[15] Golle, P., and N. Modadugu, ''Authenticating Streamed Data in the Presence of Random Packet Loss,'' in *Proc. of Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2001, pp. 13–22.

[16] Miner, S., and J. Staddon, ''Graph-Based Authentication of Digital Streams,'' in *Proc. of IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001, pp. 232–246.

[17] Park, J., E. Chong, and H. Siegel, ''Efficient Multicast Packet Authentication Using Signature Amortization,'' in *Proc. of IEEE Symposium on Security and Privacy*, Oakland, CA, May 2002.

[18] Cheung, S., ''An Efficient Message Authentication Scheme for Link State Routing,'' in *Proc. of 13th Annual Computer Security Applications Conference*, San Diego, CA, December 1997.

[19] Lamport, L., ''Password Authentication with Insecure Communication,'' *Communications of the ACM*, Vol. 24, No. 11, November 1981.

[20] Perrig, A., et al., ''Efficient and Secure Source Authentication for Multicast,'' in *Proc. of Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2001, pp. 35–46.

[21] Perrig, A., et al., ''TESLA: Multicast Source Authentication Transform, draft-irtf-smug-tesla-00.txt, IRTF, November 2000, work in progress.

[22] Weis, B., ''The Use of RSA Signatures Within ESP and AH,'' draft-bew-ipsec-signatures-00.txt, IETF, October 2002, work in progress.

[23] Baugher, M., et al., ''MESP: Multicast Encapsulating Security Payload,'' draft-ietf-msec-mesp-00.txt, IETF, October 2002, work in progress.

[24] Krawczyk, H., ''The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?),'' in *Advances in Cryptology—Crypto*, Santa Barbara, CA, Springer-Verlag, LNCS 2139, August 2001, pp. 310–381.

CHAPTER

4

Contents

4.1 A model for group key management

4.2 Requirements in group key management

4.3 Security requirements of group key management

4.4 GSA management

4.5 Classification of the group key management problem

4.6 Summary

Introduction to group key management

Group key management is the field of study of the management of cryptographic keying material for groups. The aim of group key management is generally to provide a common symmetric key or *group key* to all members of a group. The group key is often called the TEK, to signify the fact that the given key is to be used to encrypt the data traffic being transmitted to the group from one or more senders.

In delivering the group key or TEK to all members of a group, a trusted KD is typically assumed to exist. This model, centered around the notion of a trusted entity that distributes keys to participants, is a natural one, which has evolved since the early 1980s in the form of the conference key distributor [1]. The same notion of a trusted key distribution center (KDC) is also found in kerberos [2, 3], although kerberos itself was not designed to perform group key management.

The efforts focusing on conference key distribution schemes are usually related to group-oriented cryptography, and have mainly focused on ways to distribute a common key in such a way that it satisfies a number of cryptographic conditions or requirements. In that view of the world, the ideal conference key distribution scheme should not, in fact, require a single trusted entity, but should function in a ''democratic'' fashion. Members are mutually distrustful of each other. In the computation of the common conference key through the accumulation of pieces of the key contributed from each member, each member should verify for itself the correctness of

the process (or key computation) by verifying the accumulated result so far [1]. Some examples of the requirements include proof of participation within a conference, equal and provable contribution of member private keys toward the establishment of the conference key, non-repudiation of departure from a conference, cheater detection and identification, and other more stringent requirements. Examples of group-oriented cryptographic schemes include [1, 4–8].

As a result of the complex requirements, conference key distribution schemes have typically been computationally intensive and impractical to implement. In addition, these schemes have not taken into consideration other parameters that are needed to support the ongoing management of the conference key, particularly in the face of possible changes in the membership of the conference.

What distinguishes the area of group key management from that of conference key distribution or computation is precisely the management of the common key in a dynamic environment in a practical manner. To achieve this practical goal requires the introduction of additional support keys or KEKs to protect the delivery of the group key or TEK. The KEK itself will change over time due to membership changes or expiration in the lifetime of the KEK. Furthermore, the area of group key management makes use of the fact that multicast communication itself is available to be used for key management, either to the entire group or to a targeted subset of the group.

This chapter provides an introduction and context setting for group key management for the following chapters. In particular, this chapter lays out the security requirements for group key management as a superset of the requirements for unicast key management. Central to this chapter is the notion and precise definition of GSA as the multicast counterpart of the unicast SA now commonly used in IKE and IPsec.

4.1 A model for group key management

As discussed in Chapter 2, a Reference Framework was developed in the IETF as a method to subdivide the problem of multicast and group security into manageable portions. In the following discussions, the model used for group key management will be based on the IETF framework.

Figure 4.1 shows the entities involved in group key management, both in a centralized fashion and in a distributed environment. The model assumes that a one-to-many multicast is deployed. Hence, it assumes that there is only one source (sender) for the multicast data. The entities involved

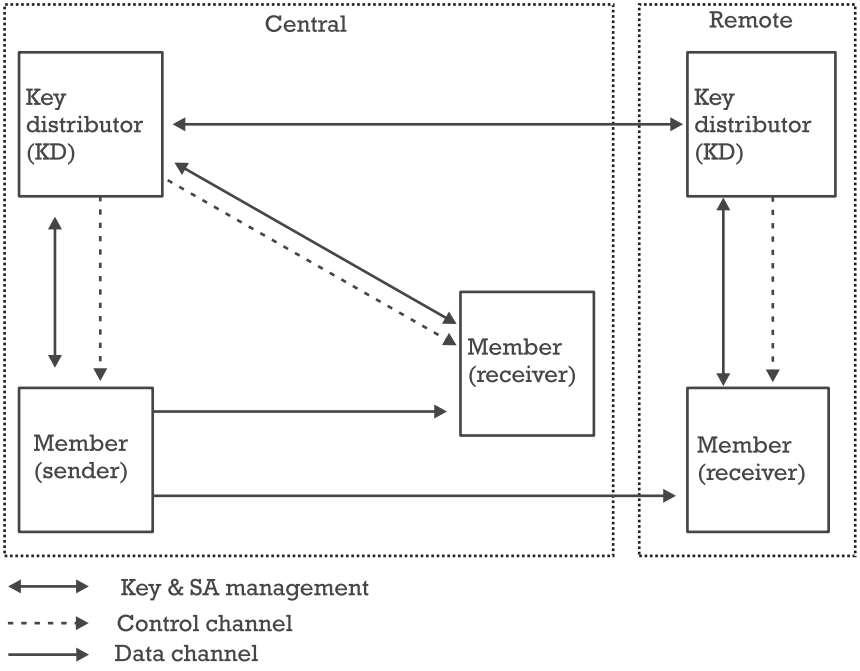


Figure 4.1 Model for group key management.

in the model are the KDs, the member sender, and the member receivers. Figure 4.1 distinguishes the notion of a central KD from a remote KD; each with its respective members that interact with it. The motivation behind this distinction is to introduce the notion of key management scalability, where multiple KDs are deployed, and where each member is associated with one ''local'' or preferred KD.

In Figure 4.1, key management and SA management functions are denoted by the double-arrow lines. As shown, each member performs key management and SA management with one KD. The member sender is shown to be transmitting (encrypted) data packets on the multicast group (denoted by thick full lines), which is received by both the local receivers and the remote receivers through the multicast distribution tree. The figure does not make a distinction about the method of routing the data packets to the receivers.

From the perspective of data protection, the data packets are assumed to be encrypted under the group key, which is delivered and managed through the key management and SA management functions (double-arrow lines). Figure 4.1 also shows the KDs sending control packets (dotted lines) to their respective member sets. A control channel is required for the KD to

issue messages (e.g., commands, reminders, and emergency notifications) to the members. In addition, the control channel is used for the management of both keys (TEKs and KEKs), keying material, and SAs.

The control channel is assumed to be reliable. Thus, although not shown in Figure 4.1, an RM protocol could be used to deliver control packets to members, although this may imply the existence of a back-channel from the members (both sender and receivers) to the KDs. Since reliability of transport is not an inherent core function of key management, the RM transport is not depicted in Figure 4.1, although the understanding is clear.[1]

4.2 Requirements in group key management

A crucial component of group key management is that of the management of SAs. In the unicast world, SA management for pair-wise communications has been addressed by the ISAKMP framework [9], and has been embodied in the IKE protocol [10]. As understood by IKE, ISAKMP, and, more broadly by the IETF, the term ''key management'' incorporates the broader aspects of keying material, including cryptographic keys, key identities, and other parameters that support the establishment of common (symmetric) keys at both ends of a unicast connection.

In the context of IP multicast, and the wider field of group communications, the two-party SA management model underlying ISAKMP/IKE is insufficient, due to the fact that a multiparty group has many members (senders and receivers). In ISAKMP/IKE the responding party (responder) in the unicast exchange chooses some parameters that it returns to the initiating party (initiator). In multicast, where there are multiple responding parties, the unicast SA negotiation model is simply not mappable to groups. Furthermore, a group negotiation procedure for SA parameters would simply be impractical and resource consuming for many multicast applications.

4.2.1 Security requirements of unicast key management

Since we assume that group key management must operate across diverse internetworks, particularly IP multicast networks, at least some of the

1. The need for the control channel (namely, a security protocol) to be reliable, and the need for RM protocols to be secure may be perceived as a chicken-and-egg problem. However, this circular reasoning may be broken by assuming that the control channel deploys some primitive method to achieve reliability, such as multiple transmission of important packets containing keying material.

properties of Internet key management are required for group key management. These properties, broadly stated, are summarized as follows [11, 12]:

1. *Protection against attacks*, such as man-in-the-middle, connection hijacking, replay/reflection, and DoS attacks. The authenticated key exchange (AKE) notion is basic to Internet key management and key determination protocols, which seek to thwart attacks that may occur on an unsecured network. The types of attacks include man-in-the-middle, connection hijacking, and reflection/replay attacks, many of which can be combated by mechanisms such as direct authentication, which integrate authentication into the key exchange, as described in the station-to-station (STS) protocol [13].

   Messages that are exchanged as part of a ''run'' should be chained with authenticable information, including random data that is contributed by each party in a two-party key exchange. This technique helps ensure that messages received by a peer match what the other peer sent. Work has been done, moreover, to formally prove AKE properties, based upon the matching of messages sent and received by peers in the exchange [14]. When session keys are used to protect exchanges that determine other session keys, perfect forward secrecy (PFS) can ensure that ''...disclosure of long-term secret keying material does not compromise the secrecy of the exchanged keys from earlier runs,'' [13] as long as authentication is linked to the key exchange. The PFS requirement, however, entails the performance penalty of a Diffie-Hellman exchange, which may not be appropriate for all applications.

2. *Selectable level of security protection in key establishment*, such as alternative transforms, optional PFS, and identity protection to support heterogeneous Internet applications and computers.

   The notion of a ''selectable level of security'' is basic to key management on internetworks, which are composed of diverse communications networks and host computers. In this environment, some applications may trade-off better security for reduced communications and computing costs. The security choices depend upon application needs, as well as the capabilities of the hosts and network devices. In order to support heterogeneous network and host devices, Internet key management supports multiple types of exchanges that can be composed in various ways. Some exchanges may support identity protection and provide PFS, for example, while others may not [15]. To accommodate diversity, a versatile

approach supports a variety of transforms and Diffie-Hellman groups, all of which can be negotiated among communicating entities [10, 16].

3. *Alternative authentication mechanisms*, such as shared key, *public key infrastructure* (PKI), and public key to support diverse trust models. Over time, a key establishment procedure may need to be replaced. The Internet Security Architecture [17] has a key management framework, the ISAKMP, which defines an abstract set of exchanges, organized by modes and phases, to provide a selectable level of protection [9, 15]. To provide a versatile solution for Internet key management, ISAKMP permits alternative authentication mechanisms in its exchanges, and is parameterized by a domain of interpretation (DOI), in which specific key determination mechanisms are defined through the specification of the name space, policy, specific payloads, and, optionally, new exchanges. In this way, ISAKMP is designed to be extended for alternative uses, and to allow a forward migration of key exchange protocols and cryptographic transforms. Although the flexibility of their approach may arguably result in more complexity, which may in turn lead to weaker security, the ISAKMP authors recommend the use of ISAKMP as a single key management framework for new uses such as group key management, as well as transport and application key management [9]. New uses can be realized through the specification of a DOI.

4. *Forward migration path* for new security mechanisms, such as new cryptographic transforms and even new exchanges.

   Internet key management, as formulated by ISAKMP, supports a forward migration path, so that new algorithms can be introduced as older methods need to be replaced [9, 10, 15, 16].

5. *A single key management framework* to support the establishment of SAs according to the local policies of Internet host and intermediate systems.

   ISAKMP achieves its versatility by being more abstract than a key determination protocol, since it manages SAs and not just keys. The SA abstraction [9, 17–19] encapsulates keys and information about keys, such as key lifetimes and cryptographic policies, so as to allow all significant aspects of the security to be modified to the needs of the application and environment. In the current Internet Security Architecture, however, SA management is peer to peer as depicted in Figure 4.2.

Figure 4.2 Unicast SA as defined in ISAKMP.

The SA is defined to be simplex in the Internet Security Architecture [17], and is identified by a security parameter index (SPI) [17, 18]. SAs are established according to local policy [17, 19], using exchanges that are designed to protect against basic key establishment attacks, such as man-in-the-middle, connection hijacking, replay/reflection, and DoS [9]. Although the first three types of attack are the subject of authenticated key exchange mechanisms, protection against the DoS attack uses a pair-wise cookie mechanism [18] between peer entities, which appears in the ISAKMP header for all exchanges [9, 10].

We assume that these properties should be properties of group key management as well. As discussed next, group key management has additional needs beyond the five points summarized here.

4.3 Security requirements of group key management

From the previous section, it is clear that many of the requirements and design features of Internet key management are needed by group key management. In fact, many of the payloads, exchanges, and transforms found in ISAKMP and IKE may be suitable for group key management. Many group key management protocols and algorithms, moreover, such as GKMP [20, 21], LKH [22], one-way function tree (OFT) [23], GSAKMP [24], NARK [25], and multicast key management with arbitrarily revealed key sequences (MARKS) [26] assume a unique key for a member, which is established using point-to-point procedures with a key server. For the purposes of authenticating a potential group member and initializing it with keys, group keying material must be ''pulled'' by an individual client from the server. Group members whose computers are off-line

during key updates must also pull keying material to be reinitialized (or to request reinitialization by the KD) in a secure, probably point-to-point protocol.

The use of IKE unchanged (with the IPsec DOI), however, is out of the question owing to the need to support group key distribution (i.e., where an external key is given to the member by the KD), in addition to the need for policy distribution rather than policy negotiation, and the use of multicast communications to push key updates to promulgate key changes. These are needed to refresh keys that reach the end of their cryptographic lifetime, and to replace keys resulting from changes in group membership. Several algorithms have been proposed to efficiently accomplish group rekey and maintenance [22–24, 26]. A versatile group key management functional building block should support a variety of alternative algorithms, to offer a forward migration path when new algorithms are developed, or flaws in existing algorithms are uncovered.

The use of a multicast service to ''push'' key updates and other control messages from the KD to members relieves the KD of the burden of contacting each member individually to change the key or the configuration of the group. In this way, group key management can scale to very large numbers of members. This ability to deploy multicast itself for group key management is attractive for a variety of applications. This property may be superfluous for pure PPV sessions, where the member is keyed once and never again for the duration of the session. Except for subscription sessions or sessions where keys must be changed, good multicast application design principles will protect the KD from being the target of periodic, and possibly synchronized, requests from large numbers of members attempting to pull keys.

Unlike large-scale subscription groups, short-lived, dynamic groups, which are characterized by relatively small numbers of members, may need group key management to minimize the time it takes to create and add members to a group. Thus, group key management must be able to efficiently maintain very large, secure groups, to support large numbers of members, while not precluding fast initialization, maintenance, and destruction for smaller groups that engage in impromptu group communications. The need to support a range of performance and scalability needs for diverse applications is very much a goal of Internet key management that is shared by group key management.

Group key management, therefore, uses a different set of abstractions than ISAKMP and IKE. The abstractions used, however, may be built from the ISAKMP abstractions, where the GSA includes the attributes of the

Internet Security Architecture SA, which is succinctly defined as the encapsulation of keys and policies [17] as follows:

- A unicast SA has selectors, such as source and destination transport addresses.
- A unicast SA has properties, such as an SPI or cookie pair, and identities.
- A unicast SA has cryptographic policy, such as the algorithms, modes, key lifetimes, and key lengths used for authentication or confidentiality.
- A unicast SA has keys, such as authentication, encryption, and signing keys.

As discussed in the next section, a GSA contains the SA attributes plus some additional ones:

- A GSA has group policy attributes. Examples would be policy on the kind of signed credential needed for group membership, and on whether the group will be given new keys when a member is added (called ''backward rekey'' below) or whether group members will be given new keys when a member is removed from the group (called ''forward rekey'' below).
- A GSA has SAs as attributes.

The final point, that a GSA includes multiple SAs, is discussed more fully in the next section.

The following list summarizes the desired properties of Internet group key management:

1. The five properties of Internet key management as described previously.
2. Support for the IETF Secure Multicast Reference Framework [27], having a KD that controls access to the group of sending and receiving members, according to the group policy it distributes. This is the Reference Framework that has been adopted by the MSEC working group[2] in the IETF.

2. See http://www.securemulticast.org.

3. Support for IP multicast applications where there may be one or more senders to the group, who may each have a unique SA to the group, or who may each share a common SA with the group.
4. Support for both receiver-initiated pull of policy and keying material, and KD-initiated push, using a variety of rekey algorithms.
5. Selectable level of performance for group key management, which permits trade-offs in startup latency, reinitialization complexity, message overhead, join latency, leave latency, and other security-related performance such as transforms.

A further general requirement is backward rekey and forward rekey. The rekey operation is needed to ensure that messages sent to the group cannot be accessed by a former member whose membership has been revoked by the KD. Some applications may also require that a member who joins a group be denied access to messages that were sent to the group prior to its membership. We call the first case forward rekey, when a key change is prompted by a member leaving the group. The latter is called backward rekey, when a rekey is caused by a new member joining the group. The rekeying is to maintain ''perfect forward confidentiality'' and ''perfect backward confidentiality,'' respectively. These properties are also referred to as ''forward/backward secrecy,'' ''forward/backward security'' and ''forward/backward access control'' in the literature [23, 24, 28].

4.4 GSA management

It is clear that the security associations for group key management are more complex, or at least more numerous, than for unicast key management. Whereas the latter establishes a key management SA to protect application SAs (which in the case of IKE is most commonly the IPsec protocol), where a minimum of two SAs are needed to key an Internet application process, group key management requires at least three. There is a pull SA between the group member and the KD, a push SA between the KD and all the group members, and an SA to protect application data from sender-members to receiver-members. In fact, each sender to the group may use a unique key for its data and use a separate SA. Thus, there may be more SAs than there are group senders.[3]

3. Some authors also refer to the pull SA as the ''registration'' SA, and the push SA as the ''rekey'' SA.

In the following sections we discuss the GSA model and describe each of its components SAs that make up a GSA. One of the primary realizations of the GSA model is the necessity of a ''bootstrap'' stage between a member and the KD. That is, in order for a member to begin to join a multicast group, that member must obtain some private parameters from a trusted entity, either out-of-band or inband. The point, therefore, is that a unicast SA is unavoidably part and parcel of the definition of a GSA.

4.4.1 The GSA model

The work of [11] defines the GSA to be an aggregate of three categories or types of SAs. This structure was chosen by its authors to better realize a GSA in a multicast environment defined by the IETF Reference Framework [27]. There is a need to maintain SAs between a KD and a group member (either a sender, a receiver or both), and among members. In the Reference Framework, the KD is charged with access control to the group keys, with policy distribution to client members or prospective members, and with group key dissemination to sender and receiver members. This structure is common in many group key management environments (e.g., [22–24, 26, 29]).

There are two SAs established between the KD and the members. The first is referred to as the Category 1 SA (or SA1), while the second as Category 2 SA (or SA2). In addition, there is a third SA which is established among the sending and receiving members. This is referred to as the Category 3 SA (or SA3). These three SAs are shown in Figure 4.3. The terms SA1, SA2, and SA3 are used to simplify the following discussion. (The terms pull/registration SA, push/rekey SA, and data security SA may also be used.)

The first category of SAs (namely, SA1 in Figure 4.3) is initiated by a member to pull GSA information from the KD. This is how the member requests to join the secure group, or has its GSA keys reinitialized after being disconnected from the group (e.g., when its host computer has been turned off during rekey operations, as described below). The GSA information that is pulled down from the KD includes the SA, keys, and policy used to secure the data transmission between sending and receiving members (this is the Category 3 SA or SA3 in Figure 4.3).

Note that SA3 is a category of SA, which implies that there may be multiple SAs established between member senders and member receivers—at least as an option. There may exist, for example, a single SA of Category 3 in which all senders share common keys and associated information. Alternatively, there may be one or more SAs of Category 3 that are unique to the particular sender. An SA3 may be reestablished or have its keys modified

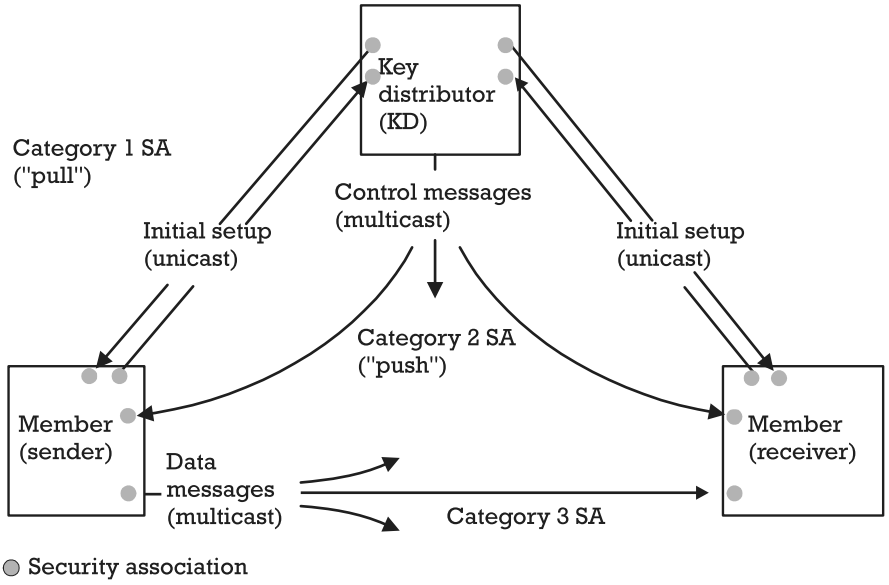


Figure 4.3 GSA definition.

through rekey operations, which occur over an SA of Category 2. Keys may be pushed to members from the KD through an SA of Category 2 (e.g., to support subscription groups).

Thus, the aim is to initially use SA1 to securely download SA2 and SA3 from the KD to the members. The SA2 is then used for control messages sent by the KD, while SA3 is used for data messages (i.e., traffic or content) from the sender to the receivers. Included in the set of control messages is the update or replacement of SA3. Thus we say that SA2 is used to update SA3, since it is anticipated that there will be far fewer uses of SA2 compared to SA3 (e.g., SA3 is used for voluminous streaming media data). Naturally, the cryptographic policy for SA2 must specify strengths equal to or stronger than SA3. Two options (at least) are available for updating the SA2 in turn. The SA2 can be updated through SA1 again (unicast), or the ''old'' SA2 can be used to update to a ''new'' SA2. This has been left as an implementation option by [11], since the definition of a GSA must cater to a wide variety of applications.

Note that for applications where key updates occur within the data stream (protected using SA3), the GSA definition requires that SA2 be declared as null (which is different from saying it is nonexistent). In some cases, such as in a pure PPV application, all of the SA information needed for the session may be distributed at the time of registration or selection of a

session (i.e., over an SA1). The rekey and reinitialization may not be necessary, so SA2 is null. Most applications combine unicast exchanges for initialization with multicast distribution for rekey. For subscription groups where keying material is changed as membership changes, an SA2 is needed to reinitialize an SA3. Hence, in summary, the GSA concept sees the three categories of SAs as being inseparable.

4.4.2 Definition of GSA

A GSA is defined to include an aggregate of three categories of SAs. The three categories of SAs correspond to the three kinds of communications, best seen from the point of view of the receiver (member). Figure 4.3 depicts this concept:

- *Category 1 SA: SA1*. An SA is required for (bidirectional) unicast communications between the KD and a group member (be it a sender or receiver). This SA is established only between the KD and a member. In the IETF Reference Framework, the KD entity is charged with access control to the group keys, with policy distribution to members (or prospective members), and with group key dissemination to sender and receiver members. This use of a unicast SA as a starting point for key management is common in a number of group key management environments.

  Note that this unicast SA is used to protect the other elements of the GSA (such as the other two categories of SAs), either in a push or pull model. As such, this SA is crucial and is inseparable from the other two SAs as the definition of a GSA.

  From the perspective of a given KD, there are as many unique Category 1 SAs as there are members (senders and/or receivers) in the group. Thus there may be a scalability concern for some applications, and so a Category 1 SA may be used on demand, whereas Category 2 and Category 3 SAs are established at least for the life of the sessions that they support. Note also that in a distributed architecture several KDs may be deployed for scalability; thus spreading the number of SAs across these KDs.

- *Category 2 SA: SA2*. An SA is required for the multicast transmission of key management/control messages (unidirectional) from the KD to all group members. As such, this SA is known by the KD and by all members of the group.

This SA is not negotiated, since all the group members must share it. Thus, the KD must be the authentic source, and act as the sole point of contact for the group members to obtain this SA.

From the perspective of each participant in a group (consisting of the KD and all members), there is at least one Category 2 SA for the group. Note that this allows for the possibility of the KD deploying multiple Category 2 SAs for other security management purposes. (For example, there might be one for critical/urgent control messages, and another for regular/periodic control messages.)

- *Category 3 SA: SA3*. One or more SAs are required for the multicast transmission of unidirectional data messages from the sender to other group members. This SA is known by the KD and by all members of the group.

  Similar to a Category 2 SA, regardless of the number of instances of this third category of SA, this SA is not negotiated. Rather, all group members obtain it from the KD. The KD itself does not use this category of SA since it is assumed that the KD does not transmit data (content) messages.

  From the perspective of the receivers, there is at least one Category 3 SA for the member sender (one or more) in the group. This allows for the possibility of including group IDs (GID) in transmission of data packets from the senders in the group.

  There are a number of possibilities with respect to the number of Category 3 SAs and the use of GIDs:

  1. Each sender in the group could be assigned a unique Category 3 SA, thereby resulting in each receiver having to maintain as many Category 3 SAs as there are senders in the group.
  2. The entire group deploys a single Category 3 SA for all senders, together with the use of GIDs. Receivers would then be able to filter based on the GIDs, while maintaining only one Category 3 SA.
  3. There could be a combination of the two choices above.

4.5 Classification of the group key management problem

Prior to concluding this chapter, we present a classification of group key management problem areas. As mentioned previously, group key

management pertains to the management of the group key or TEK in a dynamic environment in a practical manner, possibly employing multicast itself to aid in the management of the group key. Several interrelated components present themselves when one looks at the problem more closely:

- *Architecture*. The term architecture refers to the relationship and placement of group keys (or TEKs) and the other keys (or KEKs) supporting the safe delivery of the group keys. This arrangement is in turn influenced by other aspects such as the spread and density of membership, dynamicity of memberships, topology of data flows, and others. This topic is covered further in Chapter 5.
- *Protocols*. The term protocols is used to describe the set of procedures, message exchanges, and message payloads that govern the behavior of the entities involved in supporting a group (e.g., servers) and those participating in a group (e.g., hosts). This topic is also addressed in Chapter 5.
- *Algorithm*. The term algorithm is used to describe the method to arrange and update the supporting keys used to manage the group key. A more precise term would be a group key determination or management algorithm. The supporting keys are typically arranged into an LKH (or key tree), where each member holds certain keys from the logical key tree. Changes in the membership of the group usually requires that a new group key be delivered to the remaining members of the group. This is accomplished by encrypting copies of the group key under different sets of keys obtained from the key tree. Since a member has only a subset of the keys within a key tree, that member will only be able to decrypt those messages sent by the KD intended for that member (namely, those encrypted by the KD using the appropriate keys within the key tree). When the membership changes, the key tree itself must be reconfigured, following the specific algorithm to rearrange the tree. This interesting subject is central to group key management and is discussed in depth in Chapter 6.
- *Policies.* Rules are needed to govern the behavior of entities during the group initialization, the group key distribution, membership changes, emergency situations, and others. The topic of group security policies is the subject of Chapter 7.

4.6 Summary

This chapter has served as an introduction to the next two chapters which deal specifically with group key management. Deriving from the IETF Reference Framework, the chapter described a model specifically for group key management where the interaction among differing entities was discussed. The group key management model matches the definition of GSA, which was discussed later in the chapter.

As the basis for describing the requirements for group key management, a review of the requirements for unicast key management was given. Five requirements of unicast key management were discussed, as the basis for further requirements in group key management. These are: protection against various possible attacks in key exchange, a selectable level of security protection, alternative authentication mechanisms, a forward migration path, and a unifying key management framework.

The core of this chapter has been the definition of GSA for multicast, which consists of an aggregate of three categories of SAs. The definition of the GSA in [11] allows for a flexible use of the component SAs that make up a GSA.

This chapter also introduced the distinctions within the problem area of group key management, namely, of architectures, protocols, algorithms and policies. In the following chapters, we discuss these components of group key management in detail.

References

[1] Ohta, K., and K. Koyama, ''Identity-Based Conference Key Distribution Systems,'' in Pomerance C., G. Goos and J. Hartmanis (eds.), *Advances in Cryptology—CRYPTO*, Springer-Verlag, LNCS 293, Santa Barbara, CA, August 1987, pp. 175–184.

[2] Kohl, J., and C. Neuman, ''The Kerberos Network Authentication Service (V5),'' RFC 1510 (proposed standard), IETF, September 1993.

[3] Steiner, J. G., C. Neuman, and J. I. Schiller, ''Kerberos: An Authentication Service for Open Network Systems,'' in *Proc. of USENIX*, March 1988.

[4] Burmester, M., and Y. Desmedt, ''A Secure and Efficient Conference Key Distribution System,'' in *Proc. of EUROCRYPT*, Springer-Verlag, LNCS 950, Perugia, Italy, May 1994, pp. 275–286.

[5] Ingemarsson, I., D. T. Tang, and C. K. Wong, ''A Conference Key Distribution System,'' *IEEE Trans. on Information Theory*, Vol. 28, No. 5, 1982, pp. 714–720.

[6] Koyama, K., and K. Ohta, ''Security of Improved Identity-Based Conference Key Distribution Systems,'' in Gunther C. G., (ed.), *Proc. of EUROCRYPT*, Springer-Verlag, LNCS 330, Davos, Switzerland, May 1988, pp. 11–19.

[7] Simmons, G. J., ''An Introduction to Shared Secret and/or Shared Control Schemes and Their Application,'' in Simmons, Gustavus J. (ed.), *Contemporary Cryptology: The Science of Information Integrity*, IEEE Press, 1992, pp. 441–497.

[8] Steiner, M., G. Tsudik, and M. Waidner, ''Diffie-Hellman Key Distribution Extended to Group Communications,'' in *Proc. of the Third ACM Conference on Computer and Communications Security*, New Delhi, India, March 1996.

[9] Maughan, D., et al., ''Internet Security Association and Key Management Protocol (ISAKMP),'' RFC 2408 (proposed standard), IETF, November 1998.

[10] Harkins, D., and D. Carrel, ''The Internet Key Exchange (IKE),'' RFC 2409 (proposed standard), IETF, November 1998.

[11] Hardjono, T., M. Baugher, and H. Harney, ''Group Security Association (GSA) Management in IP Multicast,'' in *Proc. of the 16th International Conference on Information Security (IFIP/SEC)*, Paris, France, June 2001.

[12] Harney, H., M. Baugher, and T. Hardjono, ''GKM Building Block: Group Security Association (GSA) Definition,'' draft-irtf-smug-gkmbb-gsadef-01.txt, IRTF, September 2000, work in progress.

[13] Diffie, W., P. van Oorschot, and M. Wiener, ''Authentication and Authenticated Key Exchanges,'' *Designs, Codes and Cryptography*, Vol. 2, No. 2, June 1992, pp. 107–125.

[14] Bellare, M., and P. Rogaway, ''Entity Authentication and Key Distribution,'' in *Advances in Cryptology: Proc. of Crypto*, Springer-Verlag, LNCS 773, Santa Barbara, CA, August 1993, pp. 232–249.

[15] Krawczyk, H., ''SKEME: A Versatile Secure Key Exchange Mechanism for the Internet,'' in *Proc. of Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, 1996.

[16] Orman, H., ''The OAKLEY Key Determination Protocol,'' RFC 2412 (informational), IETF, November 1998.

[17] Kent, S., and R. Atkinson, ''Security Architecture for the Internet Protocol,'' RFC 2401 (proposed standard), IETF, November 1998.

[18] Karn, P., and W. Simpson, ''Photuris: Session-Key Management Protocol,'' RFC 2522 (experimental), IETF, March 1999.

[19] Rogers, H. L., ''An Overview of the CANEWARE Program,'' in *Tenth National Security Conference*, National Security Agency, 1988.

[20] Harney, H., and C. Muckenhirn, ''Group Key Management Protocol (GKMP) Architecture,'' RFC 2094 (experimental), July 1997.

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

[22] Wallner, D., E. Harder, and R. Agee, ''Key Management for Multicast: Issues and Architectures,'' RFC 2627(informational), IETF, June 1999.

[23] Balenson, D., D. McGrew, and A. Sherman, ''Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization,'' draft-irtf-smug-groupkeymgmt-oft-00.txt, IRTF, August 2000, work in progress.

[24] Harney, H., et al., ''Group Secure Association Key Management Protocol,'' draft-ietf-msec-gsakmp-sec-00.txt, IETF, March 2001, work in progress.

[25] Briscoe, B., and I. Fairman, ''NARK: Receiver-Based Multicast Non-Repudiation and Key Management,'' in *Proc. of the First ACM Conference on E-commerce (EC)*, Denver, CO, November 1999.

[26] Briscoe, B., ''MARKS: Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences,'' in *Proc. of First International Workshop on Networked Group Communication (NGC)*, Pisa, Italy, November 1999.

[27] Hardjono, T., et al., Secure IP Multicast: Problem Areas, Framework and Building Blocks, draft-irtf-smug-framework-01.txt,, IRTF, Sept. 2000, work in progress.

[28] Harkins, D., and N. Doraswamy, ''A Secure Scalable Multicast Key Management Protocol (MKMP),'' draft-ieft-ipsec-mkmp-00.txt, IETF, November 1997, work in progress.

[29] Harney, H., and E. Harder, ''Multicast Security Management Protocol (MSMP) Requirements and Policy,'' draft-harney-msmp-sec-00.txt, IETF, March 1999, work in progress.

CHAPTER

5

Architectures and protocols for group key management

Contents

5.1 Architectural issues and motivations

5.2 IKAM

5.3 Iolus

5.4 Key distribution protocols

5.5 Summary

The area of group key management architectures holds an important role in multicast and group security, since it impacts the network entities involved in the group, the group-specific entities, and their roles and behaviors. The term *architecture* in this context is used to express the concept or notion that entities that are involved in group key management are purposely arranged or configured in relation to one another to achieve an intended effect. This intended effect, such as optimal rekeying when a member leaves, is in turn dependent on the specifics of the group key management protocol and the algorithm used to maintain the logical arrangement of keys held by the members and other entities. From a key management perspective, the term addresses the relationship and placement of group keys (or traffic keys) and the other keys (or KEKs) supporting the safe delivery of the group keys.

The term *protocol* is used here to refer to the procedures, message exchanges, and message payloads that govern the behavior of the entities involved in supporting a group (e.g., servers) and those participating in a group (e.g., hosts). Thus, there is a close relationship between a group key management architecture and the protocol(s) used to implement that architecture. For example, if an architecture specifies that a key management entity shares one key with a subset of members, and a different key with a different subset of members, the protocol, when implemented, must address how those keys are

distributed to the respective subset of recipients, and how those keys are subsequently managed throughout the lifetime of the group. Many proposals for group key management protocols have in fact an underlying architecture in mind, although it is usually less than explicit in their presentation.

It is worth noting up front that no single group key management architecture will satisfy all requirements within all areas of application of multicast or group communications, precisely because each area of application has its unique needs. Thus, in devising architectures for group key management, implementers must understand that key management is not an end in itself, but rather an enabler or support for data distribution among group members. To that extent, the implementers must first understand the application that is employing multicast and group communications.

This chapter is roughly divided into two parts, dealing with group key management architectures and protocols, respectively. In the first half of this chapter two basic architectures are discussed, where one is essentially two-layered (or hierarchic) and the other multilayered. The first, called IKAM [1], is a two-tier key management architecture that reflects the two-layered approach found in many aspects of the Internet's design (e.g., intradomain and interdomain routing). The second, called *Iolus* [2] is a multilayered arrangement of domains.

The second half of this chapter looks at specific group key management protocols that have been proposed. In particular, attention is given to the GKMP protocol [3] since it was one of the earliest group key management protocols to be proposed. GKMP was also the first to propose the use of LKHs[1] for the group key determination algorithm. The other group key management protocols discussed are GSAKMP [5] as a counterpart of the unicast ISAKMP, and the GDOI protocol [6] which is based on the SA management framework developed in the IETF, and which takes the IKE protocol as the starting point for group key management.

These protocols represent a sample of the handful of group key management protocols that have been proposed over the years. These are selected for discussion primarily because they have been proposed in the context of the IETF, and have purposely focused on security, key management, and SA management to secure multicast communications. The reader is directed to other efforts, such as [7–9] for related proposals in this area.

1. A more popular name for LKHs (or key trees) is *Wallner trees*, named after the author of the IETF Internet draft describing it [4].

5.1 Architectural issues and motivations

There are a number of motivating reasons why designers of group key management architectures arrange entities and keying material in a given manner. Often, the aim is simply to make key servers or key distributors accessible to as many members as possible. In some designs, limiting the effects of rekeying (e.g., due to membership changes) may be the priority. In others, the prior existence of a semifixed data distribution tree through high-speed pipes or VPNs may be dictating the placement of key servers. In practice, group key management is more likely to be influenced by other elements within the broader IP multicast application that it serves.

Some of the motivations driving certain architectures are as follows:

- *Resilience and reliability of groups*. Often, the continual running of a group is of overriding importance compared to other factors, such as the rate of joins or leaves by individual members. Thus, in such an environment, redundant key servers may be used together with multiple traffic keys.
- *Geographic spread and density of membership.* The location and density (or level of clustering) of members (fixed or mobile) may be the overwhelming design criterion. Here, key servers may be strategically placed at or near the core of a cluster, in order to cater to as many members as possible.
- *Dynamicity of membership*. The rate of joins and leaves, and the average size of membership changes over a given time, may determine the best arrangement of traffic keys, the depth/breadth of the key tree, and the number of such key trees. In general, unless sufficient statistical information has been gathered over a period of time for a given group, the problem of dynamicity is a difficult one to solve.
- *Topology of data flow*. Key servers are often placed in locations that mirror the location of data distribution points (e.g., video streaming servers), in order to make best use of the existing facilities. In addition, the multicast routing topology may indicate that certain points on the routing topology are the most advantageous at which to stand up key servers. This consideration may be the overriding one for content distributors and ISPs that deploy multicast.
- *RM entities*. RM protocols are typically used to provide reliable transport of data delivered through a multicast distribution tree.

A variety of RM protocols have been designed, often with very distinct requirements to satisfy. An RM protocol such as TRACK [10] would deploy key servers and key trees in a different manner to *pragmatic general multicast* (PGM) [11].

- *The existence/nonexistence of trusted entities*. Trust is an important factor in securing multicast, and the trust relationship among entities in the network, and the trust relationship among content providers, distributors, and (access) ISPs may govern the location of group key management entities in the network.
- *Economic and political reasons*. Economic reasons (e.g., cheaper cost) and political reasons (e.g., crypto export) may decide where in the network key distributors are located, and through which parts of the network keying material may be transported.

5.2 IKAM

IKAM [1] proposed an architecture, noting that many aspects of Internet engineering are founded on a two-level hierarchy, which is in turn based on the concept of network domains. Thus, for example, in the area of routing, the notion of intradomain and interdomain is used to subdivide the complex problem of IP routing into manageable units or domains. Each domain thereby has its own protocol(s) to realize efficient and stable routing of IP packets, while a different set of protocols is used across domains to provide connectivity of domains.

The stated objective of IKAM is to foster the development of an Internet-wide solution, while encouraging innovations in solving the many problems that are related to multicast security. Since multicast security has many complex facets related to multicast technologies and security technologies, respectively, the following two-pronged approach was proposed corresponding to a two-level hierarchy:

1. Encourage the growth and evolution of novel, secure solutions for group key management within predefined key management regions (domains) whose scope is determined on a per-case basis. Regions can be defined to be the size of subnets, ASs, or larger. This will allow for the development of independent and innovative solutions that are addressed specifically for such regions, taking into consideration the multicast application being employed.

2. Encourage secure, simple, consistent, and stable interactions among the key management regions that implement the various group key management solutions. This will allow for the development of innovative interregion (interdomain) solutions that can consistently and securely tie together the various regions deploying intraregion (intradomain) group key management protocols.

By defining regions of group key management, various schemes can be used for each region, independent of one another, with the only requirement being that they can interact with a common, simple, interregion group key management protocol.

5.2.1 Domains, areas, and key distributors

The IKAM architecture divides group members into subgroups based on the usage of keys. The basic division is along a domain and one or more areas (see Figure 5.1). This division can be aligned with the network topology, such that members are in fact also grouped according to their location in the network.

In the following discussion, we assume that the latter is the case. This allows physically separated areas to reuse multicast addresses and employ special multicast addresses through administratively scoped multicast. Thus, in order to support multicast groups, the domain is divided into a number of

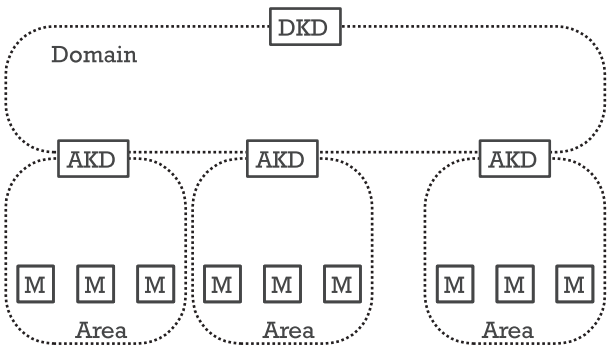


Figure 5.1 The IKAM architecture: Basic model.

administratively scoped areas [12]. A host member of a multicast group is defined to reside within one (and only one) of these areas. The purpose of placing host members in areas is to achieve flexible and efficient key management; particularly in the face of the problem of changes (joins and leaves) in the membership of a multicast group. Two general types of KDs are deployed:

1. At the domain level, a domain key distributor (DKD) entity is defined for the domain, for the purpose of key management.
2. At the area level, an area key distributor (AKD) entity is defined for each area, for the purpose of key management.

Depending on the address allocation approach, each area may be associated with an area multicast address allocation entity (AMAAE), such as the multicast address allocation server (MAAS) of [13], from which the AKD of that area obtains areawide multicast addresses. In addition, at the domain level, a domain multicast address allocation entity (DMAAE) must exist to cater for the domain.

Within the domain, all KDs (both the DKD and AKDs) are members of the domainwide administratively scoped multicast group, called the All-KD-group, which does not extend beyond the domain, and whose membership consists only of KDs. The DKD communicates to the AKDs either through secure one-to-one (unicast) communications, or through the All-KD-group. The All-KD-group is independent of other multicast groups, and exists even when there are no host members of any multicast group in the domain. The AKD communicates to host members residing in its area either through secure one-to-one (unicast) communications, or through a special (i.e., control) multicast group whose scope is limited to that area.

Figure 5.2 shows the IKAM architecture where each of the KDs is located at different domains in the network, and under different network and security administrations.

5.2.2 Multicast groups for data and control

A useful distinction adopted by IKAM is that between IP multicast groups for data transmission, and those for control (keying) messages.

To distinguish these administratively scoped multicast groups for control (i.e., key management) from multicast groups for data, the latter are referred to as data multicast groups, data groups, or simply multicast groups. A control-related group is referred to as control multicast group or simply a control group.

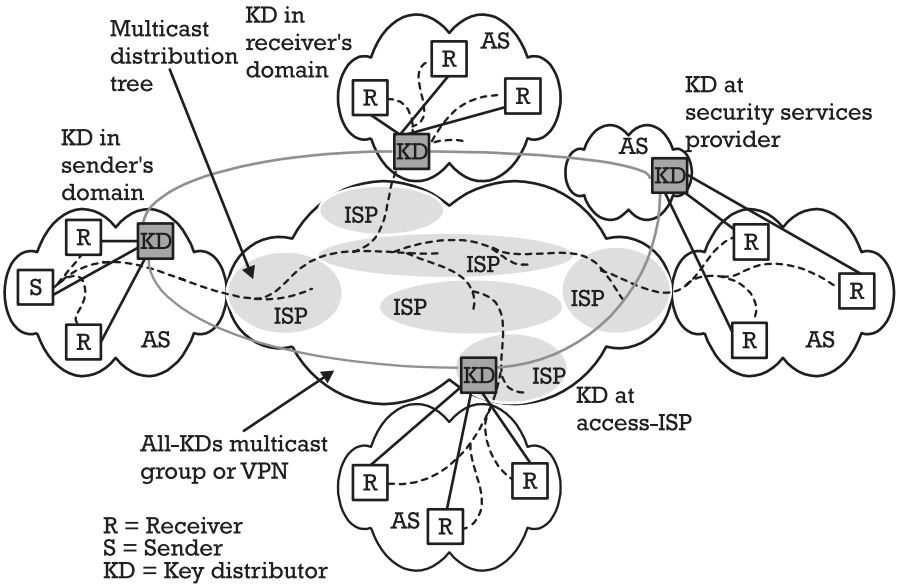


Figure 5.2 The IKAM architecture: Example.

A control group in IKAM is an administratively scoped multicast group that is areawide. It is initiated/owned by the AKD of an area. The purpose of an area control group is for key management relating to an associated data multicast group.

An area control group associated with a data multicast group exists so long as there are members of the corresponding data multicast group in the area. Once a data multicast group ceases to have any members in an area, the AKD of that area may terminate that corresponding area control group.

A special control group in IKAM is the All-KD-group. This is an administratively scoped multicast group that is domainwide, and consists only of the DKD and AKDs. It has a fixed address and is initiated/owned by the DKD. There is only one All-KD-group in a domain, and it is a permanent group, independent of whether any data multicast group members exist in the domain. This is shown in Figure 5.3. Unless specifically mentioned, this section will consider all control groups to be area control groups.

Note that once a host (in an area within the domain) becomes a member of a multicast group, it also must become a member of one of the area control groups of the area within which that host resides. This allows the AKD to communicate with the host member through the area control group.

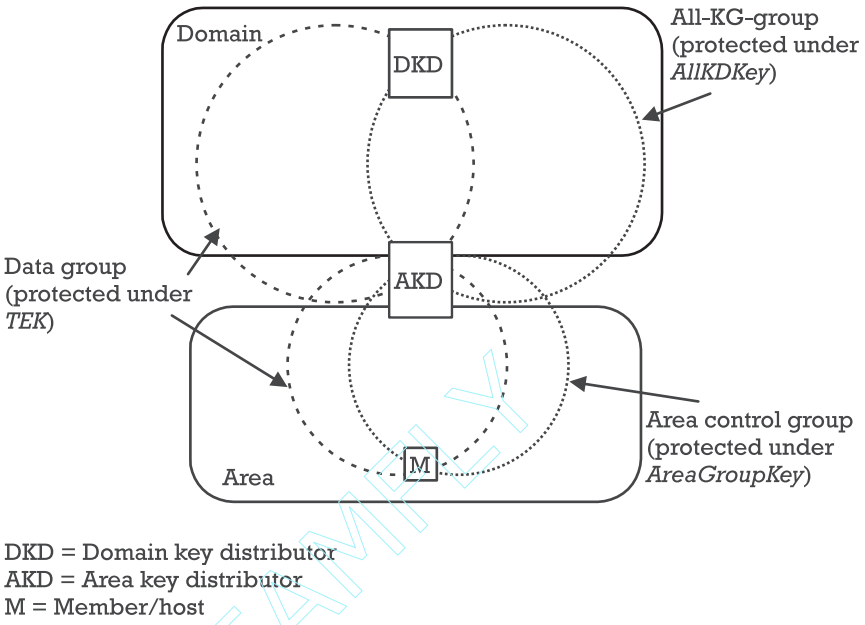


Figure 5.3 The IKAM arrangement of keys for data and control groups.

From the perspective of an AKD, for a multicast group having a host member in its area, the AKD must assign that host member to one of the area control groups.

5.2.3 Keys: Multicast groups and control multicast groups

Similar to other architectures, the domainwide cryptographic key used in IKAM to encipher data traffic is denoted as the TEK. Thus, a multicast group is associated with a domainwide TEK. For each multicast group having a member in the domain, a unique TEK is assigned by the DKD for that multicast group.

A multicast group having a member in an area in the domain is associated with one control group in the area by the AKD of that area. All traffic within an area control group is enciphered in such a way that only the AKD of that area and the intended receivers of the traffic will be able to decipher the traffic. The cryptographic key associated with an area control group is referred to generally as the Area-Group-Key. An Area-Group-Key is selected by the AKD. The Area-Group-Key is unique for each <multicast group, control group> pair.

For the special All-KD-group, an All-KD-Key is assigned by the DKD. The All-KD-Key can in fact be accompanied by an LKH or key array that is shared among the KDs. Since KDs are fixed and do not join/leave groups, such an LKH may be used to provide recovery against the compromise of one or more AKDs. Figure 5.3 illustrates the data group, control groups (i.e., All-KD-group and area control group), and their respective keys.

The ''bootstrapping'' process of each multicast group and control group relies on the establishment of an SA, and a shared private key between a (candidate) member of the group with the KD (DKD or AKDs) that controls the group. It is through this one-to-one secure channel that the parameters for the groups are then given to host members by the AKD, in the case of the multicast group and area control groups, or to the AKDs by the DKD in the case of the All-KD-group.

5.2.4 Control multicast groups: Address allocation

Three possible approaches are identified in IKAM [1] with regard to the use of area control groups (corresponding to a given multicast group) for key management by the AKD:

1. *One control group per area per multicast group*. For each multicast group having members in an area, a separate control group is created within the area. Each separate area control group is associated with a different Area-Group-Key.

   One disadvantage of this approach is the potential lack of multicast addresses within the area. Another disadvantage is the waste of resource related to areawide multicasting, if the area only has very few members of the corresponding multicast group. This approach requires the presence of an AMAAE in each area.

2. *One control group per area for all multicast groups*. In this approach, for each multicast group having members in an area, only one control group is created within the area. Hence, the area control group is shared among members of different multicast groups.

   Although there is only one (shared) area control group, a separate Area-Group-Key is used for each data multicast group. When the AKD wishes to communicate with members of a particular multicast group residing in its area, the AKD will encipher the communications using the appropriate Area-Group-Key. Other unintended recipients will also receive the enciphered packets, but will drop them since they will not be able to decipher them.

The advantage of this approach is that there is only one control group that can be long lived and have a fixed address. Having the fixed address obviates the AMAAE, thereby simplifying the entire design. The main disadvantage of this approach is that bandwidth within the area is wasted when the AKD is performing key management communications with only a small subset of group members residing in the area, since all other unconcerned members are receiving the (enciphered) packets and dropping them. Another related disadvantage is an increased opportunity for cryptanalysis of enciphered packets of control groups, since unconcerned members are receiving these packets and may cryptanalyze them.

3. *One control group per area for several multicast groups*. This approach attempts to bring together the advantages of the previous two. For a fixed number of multicast groups having members in an area, a single control group is created within the area. Thus, within a given area a set of n multicast addresses for n control groups can be selected and fixed. Each multicast group having one or more members in an area is then mapped to one of the n control groups in that area (e.g., by hashing the multicast group address).

   In terms of key management, each <multicast group, control group> pair will be associated with a unique Area-Group-Key. Thus, for example, a host who is a member of two multicast groups, MG1 and MG2, may find that in its area both MG1 and MG2 are mapped into one control group, CG1, with two Area-Group-Keys, AGK1 and AGK2, corresponding to the two multicast groups. Hence, for the control group CG1 the host must maintain the unique triplets <MG1, CG1, AGK1> and <MG2, CG1, AGK2>. The host must be able to distinguish control group packets in CG1 corresponding to the two multicast groups, in order for it to apply the matching key pairs. Tagging methods (e.g., in packet headers) within control packets may be employed to achieve this effect, saving the host the effort of trying the keys.

   In this manner, the design is simplified by obviating the AMAAE for each area, and the problem of unintended members receiving and dropping (enciphered) messages is also somewhat reduced.

5.2.5 Arrangement of keys in the domain

The IKAM architecture aims at aiding the delivery of a TEK to members of a multicast group or a control group. The fact that a host holds a copy of

the TEK is taken to mean that the host has previously been correctly identified by its AKD, and that it has established an SA with its AKD. The private key used in a multicast group or a control group affords confidentiality and group authentication, in the sense that the source of any information enciphered under the key is a valid member of the group. Note that this level of authentication (group authentication) is implicit, and does not provide irrefutable proof of the singular identity of the sender.

IKAM recognizes the benefits of a public key certification infrastructure and is open to such an infrastructure being deployed, with each entity being assigned a public key. IKAM assumes that public key pairs are assigned only to the KDs (DKD and AKDs) and the DMAAE, to allow them to digitally sign information that is to be sent through the multicast group or the control groups. All entities within IKAM must have the certificates corresponding to these public keys.

- *Public keys*. IKAM assumes that all KDs (DKD and AKDs) are assigned public and private keys (asymmetric cryptography) in order for these KDs to digitally sign certain information in such a way that it is verifiable by all hosts in the domain.

  A host member residing in an area is assumed to have a copy of the public key certificates of its AKD and the DKD.

  An AKD is assumed to have a copy of the public key certificates of the DKD. An AKD may obtain the public key certificates of other AKDs from the DKD, with the DKD acting as a domainwide certification authority.

  At the very least, the public key of the DKD must be advertised in a tamper-proof manner (e.g., printed or manually configured), to allow it to be used to vouch for the public keys of the AKDs.

- *Shared private keys (symmetric keys)*. Three types of group-oriented (i.e., shared by a group) private keys (symmetric keys) are used in IKAM:

  1. *TEK*. This is the private key associated with a multicast group that is used by all the group members in the domain to encrypt/decrypt the multicast traffic in the multicast group. This key is assumed to be in possession of the DKD, although it could be generated by the DKD. This key is delivered securely to each AKD, which will in turn deliver the key securely to each group member in its area.

2. *Area-Group-Key*. This is the private key associated with the <multicast group, control group> pair, and is used to encipher the communications in the control group. An Area-Group-Key is generated by the AKD and delivered to each member through a secure channel. An Area-Group-Key is known only to the AKD of an area and the members (of the corresponding multicast group) residing in that area.

3. *All-KD-Key*. This is the private key associated with the special All-KD-group. All the AKDs and the DKD hold a copy of the All-KD-Key. This key is generated by the DKD and delivered to each AKD through a secure channel. This key is used to encipher the communications within the All-KD-group.

Two types of pair-oriented private keys are used in IKAM:

1. *Member-Private-Key*. Each host member in an area pair-wise shares an SA and a Member-Private-Key with the AKD of that area. The SA and the Member-Private-Key are long term, and must be established before the host member joins (or initiates) any multicast groups, and is given a copy of that group's TEK. There is only one SA and one Member-Private-Key shared between the host member and the AKD, independent of the number of multicast groups to which that host member belongs.

2. *AKD-Private-Key*. Each AKD shares a pair-wise SA and an AKD-Private-Key with the DKD. The SA and the AKD-Private-Key are long term, and must be established before any multicast group exists in the domain.

In summary, the private key arrangement from the point of view of IKAM entities is as follows:

- *Hosts:*
  - TEK, per data multicast group;
  - Area-Group-Key corresponding to the <multicast group, control group> pair to which the host belongs;
  - Member-Private-Key shared with its AKD;
- *AKDs:*
  - TEK, per data multicast group (in an area);
  - Area-Group-Key of its area;

- All-KD-Key shared by AKDs and the DKD within the special All-KG-group;
- AKD-Private-Key shared with the DKD;
- Member Private Key of each member residing in its area;

- *DKD:*
  - TEK, per data multicast group (optional);
  - All-KD-Key shared by AKDs and the DKD within the special All-KG-group;
  - AKD-Private-Key of each AKD in the domain.

Note that the DKD does not hold copies of the Member Private Keys. This is in contrast to the approach in [14], in which a central server holds the private keys of all members in the multicast group. If a host is in need of communicating directly through a secure channel to the DKD or any other entity, then the host must establish an SA and a shared private key with the DKD or entity.

Although currently not prescribed, depending on the reliability mechanism to be employed, the DKD may hold copies of the Area-Group-Key within each area in the domain. However, the intent is clear that the DKD is not to replace any AKD.

Each key is assumed to be associated with a key identifier, which uniquely identifies the cryptographic key is question.

5.3 Iolus

Similar to the two-tier IKAM architecture described earlier in this chapter, Iolus [2] uses subgrouping for scalable management of large, secure groups. However, Iolus does not impose a limit on the number of levels in the hierarchy of subgroups. Further, Iolus proposes a decentralized architecture in that the group manager is not aware of any membership changes external to its own immediate subgroup.

Each secure group is managed by a group security controller (GSC). The GSC is ultimately responsible for the security of a group. The group is divided into several subgroups. The GSC manages the top-level subgroup and designates trusted third-party proxies, called group security intermediaries (GSIs), to manage the other subgroups. Each subgroup contains members or

GSIs or both. *Group security agents* is a generic term that refers to both the GSC and the GSIs.

The GSC distributes the group's ACL to the GSIs. With the group's ACL at hand, each GSI independently manages its own subgroup. In particular, a GSI does not need to contact the GSC when a member leaves or joins its subgroup. By keeping subgroups small and their management local, Iolus achieves scalable operation of a secure group.

Independent subgroup management by group security agents results in two types of scalability. First, when membership changes, new key(s) need to be sent to fewer members. Therefore, computation and communication overhead is much less than that in the process of rekeying the whole group. Iolus architecture also provides *1 affects n* scalability. According to this notion, one member should not be able to affect all the members in the group.

Recall that to maintain backward and forward access control, we need to ensure that joining members do not get access to past data, and departing members do not get access to future data. Typically, we achieve this by changing the group key and sending it to the whole group. From their definitions, forward and backward access control and *1 affects n* scalability seem to be at odds with each other. In the rest of this section, we describe how Iolus architecture facilitates these conflicting requirements.

5.3.1 Hierarchical subgrouping

Iolus uses a centralized GSC to manage each secure group. The GSC divides the group into several subgroups and designates trusted third-party entities, called GSIs, to manage them. The subgroups are arranged in a hierarchy, with each GSI being a member of a parent subgroup, apart from being manager of its own subgroup. The GSC itself manages the top-level subgroup. Figure 5.4 illustrates GSI_A being a member of its parent's subgroup, managed by GSI_P.

The GSC can be perceived as the root of the group management tree. The GSIs are essentially the internal nodes, and the members are leaf nodes of such a tree. Note that a subgroup may contain GSIs or members or both. Notice that the root subgroup, that is, the subgroup managed by the GSC, has no parent subgroup. Further, we have subgroups without any GSIs as members. We refer to such subgroups as leaf subgroups. Figure 5.5 illustrates hierarchical subgrouping in Iolus architecture.

Each GSI is responsible for access control and key management within the subgroup. Group security agents use an ACL-supplied by the GSC to

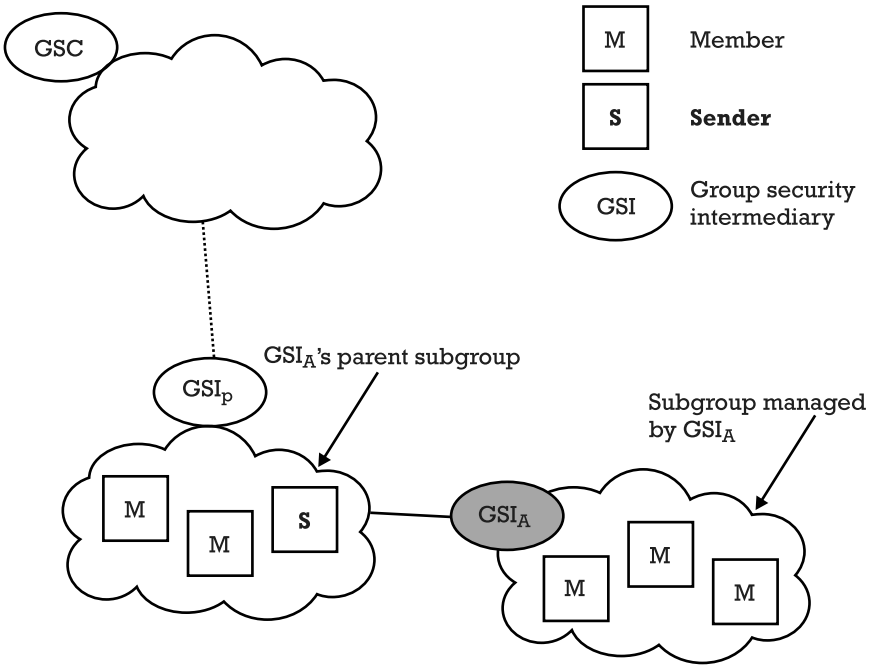


Figure 5.4 Subgroups in Iolus architecture.

determine whether to allow prospective hosts to join the (sub)group. GSIs are responsible for maintaining forward and backward access control in the subgroups. In the Iolus architecture, this means that a group security agent is responsible for join or leave rekeying, as well as data or key forwarding. Note that subgrouping confines join or leave rekeying to a subset of the members. In other words, membership changes do not affect all the members in the group.

Group security agents have an interesting relationship with the subgroups in Iolus. Each subgroup is managed by a single group security agent, but may contain several GSIs as members. Each GSI is thus associated with exactly two subgroups, the subgroup it owns and a parent subgroup. We also refer to the parent subgroup as the upstream (toward the GSC) subgroup.

5.3.2 Subgroup key management

Each group security agent shares a unique key with each of its subgroup members separately. The group security agent establishes these keys when a member joins the subgroup, using a one-to-one secure channel. The group security agent also maintains and distributes a *subgroup key* (SGK). An SGK is

known only to current membership of that subgroup. To maintain forward and backward access control, a group security agent changes its SGK whenever the subgroup membership changes. The group security agent may use any of the schemes described in Chapter 6 to efficiently rekey the SGK. Note that a GSI, as a member of its parent's subgroup, shares a unique secret key with its parent group security agent. Further, it receives its parent SGK as well. Thus, a GSI knows two subgroup keys.

5.3.3 Secure group communication in Iolus

Data forwarding in Iolus is the key process that contributes to *1 affects n* scalability. Iolus introduces the concepts of data and key translation for secure data forwarding. Translation refers to the process of a group security agent decrypting data or keys using one of the SGKs it knows, and encrypting the data with the other SGK it knows. In other words, when a group security agent receives data encrypted with its upstream SGK, it translates the data for its downstream subgroup, and vice versa. Secret data permeates the group supported by this data or key translation by the group security agents.

Data translation is expensive, but it supports forward access control as well as *1 affects n* scalability. Key translation is relatively cheaper, but only supports limited forward access control. Translation in general introduces latency. We discuss these issues in more detail in the following.

Data translation

When a sender has data to send, it encrypts the data with the subgroup key, and sends the encrypted data to the subgroup. Recall that each subgroup may contain several GSIs, and each GSI knows two SGKs. Each GSI decrypts the data, and encrypts and transmits the data for the parent or child subgroups of the sender's subgroup. This process of data translation continues until the data reaches the leaf subgroups, that is, the subgroups without any GSIs as members.

Figure 5.5 illustrates data forwarding in Iolus. Notice the labels corresponding to subgroup data transmissions in the figure. They indicate how secret data propagates through the Iolus framework in steps. Each translation introduces latency, and thus members in the sender's subgroup receive the data first, whereas the members in the furthest subgroup from the sender's subgroup receive the data last. Note that the translation latency is in addition to the network latency.

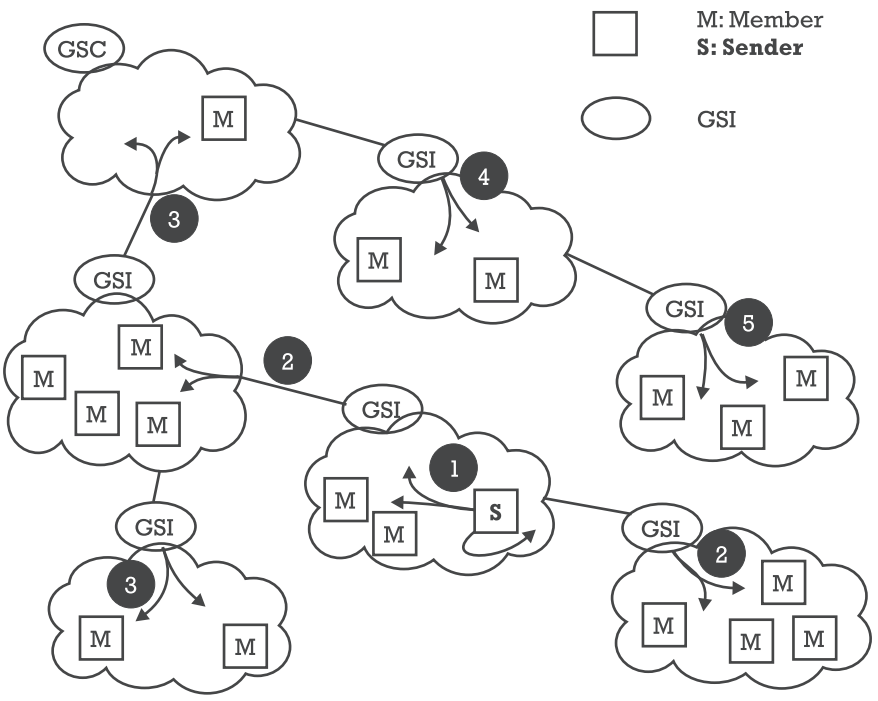


Figure 5.5 Data/key forwarding in Iolus.

Notice that in several subgroups, GSIs that are only members (not managers) translate and forward secret data. In the sender's subgroup, it is the sender, not the local GSI, that transmits data. In all these cases, data may be encrypted using an outdated SGK. Thus hosts that are no longer members in the group may receive data. This can be avoided by having the subgroup managers forward data, instead of the sender or member GSIs.

Key translation

As we discussed, data translation is very expensive. Iolus proposes key translation for efficient data transmission. In this scheme, the sender encrypts the data with a new random key, and multicasts it to the whole group. We refer to the random key as the TEK. The sender forwards the TEK, following the translation process described earlier. Key packets are much smaller than data packets and thus this approach is more efficient.

Key translation, however, introduces some new issues. Since group management is decentralized in Iolus, the sender has no way of knowing

when to change the TEK. Therefore, it changes the TEK periodically. If the period is large, the sender could be using a TEK that departed members may know. However, when the sender rekeys the TEK, departed members will cease to be able to decrypt any further data or key transmissions. In other words, we cannot support strict forward access control. The only way to maintain forward access control and *1 affects n* scalability is to have the sender use a fresh TEK for every data packet it sends. This, while being cheaper than data translation, is expensive.

5.3.4 Limitations of Iolus architecture

We end the discussion on Iolus with a note about the GSIs. Several researchers have pointed out that the use of third-party entities for secure group communication may not be acceptable in many real-world scenarios. Consider a content provider as the sender and its customers as the receivers or members. It may not be acceptable to content providers, that GSIs are able to get access to the secret content.

The dual encryption protocol [15] proposes a scheme to avoid having third-party entities getting access to secret data. The basic idea is to establish a new set of keys, called key group keys, that only members know, and have the sender encrypt TEKs twice, first with the key group keys and next with the SGKs. Since group security agents do not get access to key group keys, they cannot decipher secret data. Another approach, based on reversible parametric sequences [16], provides an alternative solution to this problem.

5.4 Key distribution protocols

In the previous section of this chapter, the issue of architectures supporting scalable group key management was discussed. In the following sections, we describe a number of protocols that actually implement group key management to various degrees. Some of the work is of a research nature, or represents an outcome from research projects, while other work results from industry vendors implementing a protocol dictated by specific needs.

5.4.1 GKMP

GKMP was one of the earliest works addressing key management in the context of a group [3]. GKMP recognizes the creation of groups through

sender-initiated and receiver-initiated operations, and describes key management from the point of view of a group key management application. As part of the group key distribution process, protocol entities pass permission certificates, which contain access control information about a particular site. The access control information is given by a trusted higher authority, who signs the permission certificate. A protocol entity must verify the permissions in the permission certificate and verify the level of service requested, to ensure that it is within the allowable range.

GKMP employs a cooperative key generation process, which initially starts between two protocol entities. The Group Key Controller then distributes the group keys to qualified GKMP entities. This distribution process is a mutually suspicious process, where all actions and identities must be verified.

Compromise recovery is supported in GKMP by way of a signed compromise recovery list (CRL) of compromised entities being distributed at the same time as key distribution. Furthermore, GKMP delegates control of groups to specific group controllers to support the efficient dissemination of CRLs. Each CRL is given a version number, which is delivered during key management. The version number triggers the downloading of the most recent version of the CRL.

GKMP entities

GKMP employs a number of entities and constructs that are innovative and that have been carried over into more recent protocols (such as GSAKMP and GDOI). These entities are as follows:

- *Group controller* (GC). The group controller (GC) is the a group member with authority to perform critical protocol actions. These include creating keys, distributing keys, creating group rekey messages, and reporting on the progress of these actions.

  An important requirement in GKMP is that all group members must have the capability to be a GC, and must assume this duty upon assignment.

- *Group member* (GM). A group member is any group host that is not acting as the GC. A group member is assumed to have the task of assisting the GC in the key creation process, validating the GC's authorization for carry out actions, accepting/requesting keys from/to the GC, maintaining local CRLs, managing local keys, and other tasks.

GKMP introduces a number of important concepts in the engineering of secure groups in the Internet. One of these is the group token (GT) which contains information the members need to ensure that a controller is authorized to create a group and that the group has information pertaining to constraints. These include:

- Group identification;
- GC identification;
- The group action (create, rekey, or delete);
- Group permissions (rules to guide access control);
- The rekey interval (the lifespan of the group key);
- The token version (an identifier to identify the current token);
- The token signature (an asymmetric signature using the GC private key);
- The GC's public key.

GKMP uses various identifiers to address the number of keys used within a GKMP group. These include the Group ID to identify groups, the group TEK (GTEK) ID to identify the group TEK, and the group KEK (GKEK) ID to identify the group KEK. The key identifiers are needed when a key tree or LKH is implemented for the group.

Sender-initiated multicast

The basic operational concept for multicast key management for sender-initiated multicast consists of a number of operations:

- *Identification of group key controller*. The originator of the multicast group creates or obtains a group management certificate from its certification hierarchy. The certificate identifies the holder as responsible for generation and distribution of the group key. The originator relays the membership list to the group key management application.
- *Group key creation*. When the application receives the list of members, it selects a member and initiates the creation of a group key packet (GKP). The packet contains the current group traffic encryption key (GTEK), and a key used to deliver the future GTEK. This key is called the group key encrypting key (GKEK). The packet also contains

additional parameters, selected by the originator's group key management application, defining the actual usage of the keys.

- *Group key distribution*. When the GKP has been created, the GC goes through the list of all members and verifies their respective permissions level, using each member's certificate. The GC then creates a session key package (SKP), which contains a session TEK (STEK) and session KEK (SKEK). To handle future rekeys, the GC then creates a digitally signed group rekey package (GRP), which consists of the earlier created GKP encrypted under the GKEK.
- *Group rekey*. For group rekeying, the originator group key management application selects a member and proceeds to create a new GKP containing a new GTEK. In addition, it creates a new GRP, which is encrypted under the earlier next GKEK.

Receiver-initiated multicast

The receiver-initiated model presents some interesting problems from a security viewpoint, since the end participants are not known a priori. Transferring the notion of GC from above into the receiver-initiated multicast, the following steps describe the process using the controller.

- *Identification of group key controller*. Since there is no a priori list of known members, this approach assumes that one member (e.g., the member that initiated the group) is responsible for initial group establishment, and periodic generation and dissemination of new GRPs. In effect, this member may become the controller, whose identity can be made known to the existing members and newcomers, through various means (e.g., broadcast).
- *Group key creation*. In the receiver-initiated operations, the GKP creation is similar to the sender-initiated case. As before, the controller creates a GKP with the first group member to initiate contact. The group key management application then makes itself know as the GC, which the member validates under the protection of the GTEK.
- *Group key distribution*. After creation of the GKP, as other members contact the controller, an SKP is created, member permissions are validated using their certificates, and a Session Rekey Package is loaded to the member.

- *Group rekey*. The rekeying procedure here is identical to that for the sender-initiated case. The controlling group key management application selects a member, creates a new GKP, creates a new GRP (which is encrypted in the previously distributed next GKEK), and broadcasts it to the group.

5.4.2 GSAKMP

GSAKMP [5] was developed as a framework for creating groups that share cryptographic keys. The framework provides mechanisms to disseminate group security policy, perform access control based upon PKI certificates, generate group keys, and recover from compromise of keys within a group. GSAKMP introduced the notion of a policy token, which is a signed collection of policy information related to a given group.

GSAKMP defines its own header and message format, which is similar to the ISAKMP framework for pair-wise key management. The authors perceived ISAKMP as being insufficient for the needs of dynamic secure group creation and maintenance; hence the definition of GSAKMP. The *dynamic* aspect of GSAKMP is the notion that group members, when provided with enough authority, should be able to take on the function of the GC. This provides scalability and resilience against the GC crashing, or network partitions occurring.

Entities in GSAKMP

GSAKMP defines a number of entities, starting with the GC as the most prominent entity. One important underlying concept within GSAKMP is the notion that the GC can be any member of the group that has (been given) the authority to perform some critical tasks. Among others, the GC is distinguished from ordinary members by its ability to:

- Create and disseminate keys;
- Maintain the rekeying infrastructure;
- Create and maintain a logical key hierarchy or a key array for the purpose of rekeying members.

GSAKMP looks ahead by allowing the group key management infrastructure to evolve into a multiGC arrangement where several members can be a GC within a group, each even with a more specific task of being a

rekey-GC. These are referred to as subordinate controllers (SCs) by GSAKMP. The distinction between a GC and SC reflects GSAKMP's understanding of the need for managing large groups by dividing them into subgroups; each managed by one or more SCs. The policy token, which is known by all members, mentions the members that are delegated to become a SC, and therefore allows new or prospective members to contact their local (or closest) SC to initiate a group join.

Furthermore, GSAKMP reflects an N-tier thinking by introducing the notion of ''subordinate logical key hierarchy'' (or subordinate LKH arrays), which essentially points to each subgroup having its own logical key hierarchy with the aim of protecting and managing the group key.

Group establishment and protocol flows in GSAKMP

GSAKMP recognizes a group life cycle consisting of group definition, group establishment, group maintenance, and group removal. The activities involved in creating a group include:

- Determining access policies for the host/users to join the group;
- Determining authorization policies for entities involved in key dissemination and other actions requiring higher trust levels;
- Determining security mechanisms and algorithms used by members in the group;
- Determining the topology of membership according to the LKHs and compromise recovery algorithms;
- Creation of the policy token containing some or all of the information previously determined.

There are several phases involving differing protocol flows in the creation of a GSAKMP-based group (Figure 5.6).

1. *Group establishment*.
   In GSAKMP the establishment of a group begins when a (potential) member issues a *request to join* (RTJ) message to GC.

   The scheme assumes that at the very least the GC is already in operation, and that group establishment in actuality begins when the first member issues the RTJ. In terms of the group establishment protocol, the GC could in fact issue an invitation (i.e., the push model) and the member could issue a request (i.e., the pull model).

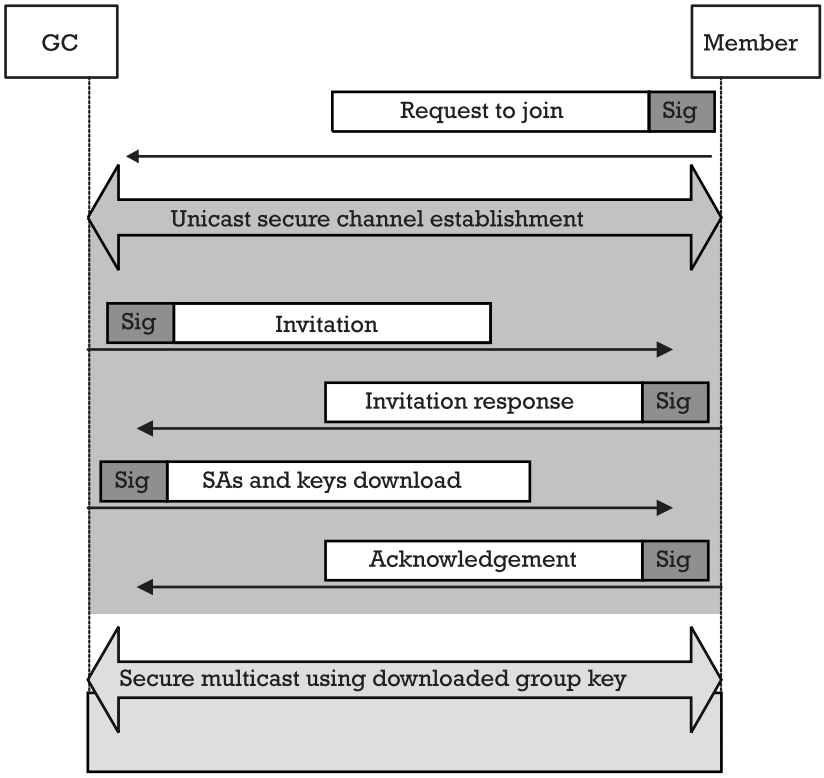


Figure 5.6 GSAKMP flow.

The RTJ must indicate the GID that the member is referring to, and the message must be digitally signed by the member.

The GC then responds to the member with an *invitation* message, which contains among others, the policy token pertaining to the GID that the member requested to join. The message also contains other elements for protection against replay attacks (e.g., nonces), and the identity of the member who requested to join. The GC digitally signs the message and optionally attaches its own certificate.

After the member verifies the signature of the GC on the invitation message (and verifies the GC's certificate), the member verifies the authorization of the GC.[2] Assuming the member has

2. The published GSAKMP documents assume that some authorization mechanism exists for the members to allow them to verify the authority of the GC or subordinate GCs.

verified the authorization of the GC to act on behalf of the group, the member sends back an *invitation response* (IR) message to the GC. This IR message must return the nonces that were contained in the previous *invitation* message, and the member must sign the IR message.

In return, the GC will create and send a signed *key download* message containing the data key (or GTEK), together with the LKH or key tree/array for rekeying purposes later. Note that encryption is performed on all of the necessary fields of the message to protect the keying information from being read in transit. The LKH or key array is often described as ''self protecting,'' because in its delivery the nodes within the hierarchy will be encrypted using keys which are derived from the location of the node within the hierarchy. Furthermore, the key download itself is conducted through a pair-wise secure channel between the GC and the member.

Once the member verifies the authenticity of the *key download* message from the GC and obtains the keying information for the group, it responds to the GC by sending it an *acknowledgment* message. The GC will verify the signature and freshness of the message.

The reader is directed to [5] for detailed information on the payloads of the messages discussed above.

2. *Group maintenance*.
GSAKMP uses the term maintenance to refer to the events and responses related to membership changes, rekeying events, and other emergency situations (such as key compromises).

The first ordinary event that is expected to occur is membership changes, whereby members leave and new members join. When a member joins, the procedure observed is that described above. When a member leaves, then rekeying must occur, based on the position of the member within the LKH. The member-leave event triggers the GC to send out a rekey message containing a (possibly revised) policy token and key array.[3]

Note that all group maintenance messages must be signed by the GC, since it affects all members of the group tied within the LKH.

3. A description of a rearrangement of an LKH can be found in Section 6.3.

Extraordinary events foreseen by GSAKMP includes the compromise of a current data key (or GTEK) and/or the compromise of part or all of the entries within the LKH being used within a group.

3. *Group termination*.
The termination of a group in GSAKMP is effected by the GC sending out a group removal/destruction message which is digitally signed. The understanding is that communications within the group will cease to exist, and that resources used to run the group may also cease to be available.

Group establishment without an underlying SA

GSAKMP allows for the creation of groups through the member-to-GC communication, without reliance on the existence of an underlying security association (Category 1 SA). This approach assumes that the data portion of the key download payload is encrypted, and that the details of the encryption of this data are provided in the key download payload itself.

The key determination for this encryption may be done through a two-party contributory system (à la Diffie-Hellman) using the key creation payloads to carry the contributions of the participants to this key, or may be transferred with the encrypted contents using public key encryption and an enveloping scheme.

GSAKMP policy token

GSAKMP employs a data structure called the policy token to represent security mechanisms (and their parameters) that are used within a group. The elements of a policy token specify the policies that are to be followed by members of a group, and consist of the following:

- *Policy identification.* A group must have some means by which it can identify an instance of group security policy in an unambiguous manner. Failure to correctly identify the group policies, messages, and participants can lead to incorrect and insecure operation. In the simplest form, an instance of a policy token must be associated with a unique GID expressly found inside the policy token.

- *Authorization for group actions.* A group security policy must identify the entities allowed to perform actions that affect group members. Group authorization partially determines the trust embodied by the group as a whole, by defining the parties or entities that are allowed to participate in group activities. Because of the wide range of expected environments, flexible identification of entity authorizations is highly desirable. Authorization given to an entity must be shown as being true and authentic, and coming from another entity that bequeathed that authority.
- *Access control to group information.* Access control policy defines the entities that will have authorization to hold the key protecting the group data.
- *Mechanisms for group security services.* Identification of the security services used to support group communication is required. For example, policy must state the algorithms used to derive session keys and the types of data transforms to be applied to the group content. Each security service can have parameters and policies specific to its implementation.

  In order to establish that the entire GSA is adequate to protect the data, it is necessary to have the full specification of:

  - The group establishment mechanism;
  - The data protection mechanism;
  - The group management mechanism.
- *Verification of group security policy*. Each policy must present evidence of its validity. The means by which the origin, integrity, and freshness of the policy is asserted (for example, via digital signatures) must be known by each group member prior to its acquisition. In the simplest form, this consists of the policy token being digitally signed by the entity authorized to issue the group security policy (e.g., the group owner).

5.4.3 GDOI

The GDOI [6] of ISAKMP represents an effort to specify a ''domain of interpretation'' over the more general ISAKMP key management framework [17]. In essence, the GDOI protocol is a GSA management protocol. The notion of domains of interpretation was introduced in ISAKMP

as a way to allow a more specific usage of ISAKMP for certain areas of application, which may require the addition of extensions to the plain ISAKMP to fulfill those specific needs. More specifically, the GDOI protocol borrows definitions from GSAKMP [5], incorporates the Phase 1 SA of the Internet DOI [18, 19], and proposes new payloads and exchanges according to the ISAKMP and IKE standards.

Although ISAKMP [17] allows for such interpretations, in reality the process is not so straightforward. This fact, together with the reality that the IKE [19] protocol is the de facto Internet key exchange protocol used by over 70 vendors, led the authors of the GDOI protocol to build the protocol using IKE, rather than ISAKMP. The GDOI protocol has its origins in SMuG, which is a research group that was established under the auspices of the IRTF, a sister organization of the IETF. After going through a number of revisions, the GDOI protocol was submitted to the MSEC Working Group in the IETF for formal standardization. At the time of this writing, two major vendors are in the final stages of completing their respective implementations, which have been tested against each other for interoperability.

Mapping GDOI to the GSA Model

Recall that in Section 4.4 the notion of a GSA was described and then defined. A GSA is defined to consist of three categories or types of SAs. The three categories of SAs are called Category 1 SA (or SA1), Category 2 SA (or SA2) and Category 3 SA (or SA3), and they are shown in Figure 4.3. The entities involved are the Group Controller Key Server (GCKS) and the members. Note that in Figure 4.3 the GCKS is also referred to as the KD.

Looking at Figure 4.3, the following provides a mapping between the conceptual SAs in Section 4.4 and how they are implemented in GDOI:

- *Category 1 SA*: This SA is also referred to as SA1 or pull SA (since the member pulls it down from the GCKS). In some literature, this SA is also called registration, since it mimics the registration process that the member has to conduct with the GCKS for the particular multicast group.

  In GDOI, this is implemented as the GDOI Phase 1 and Phase 2 exchanges. As will be explained later, the GDOI Phase 1 is similar to IKE Phase 1, while the GDOI Phase 2 is newly defined to cater for the GSA. GDOI refers to this new Phase 2 as the GROUPKEY-PULL exchange.

- *Category 2 SA*: In Section 4.4 this SA is defined to be used for control messages sent or pushed by the GCKS to the members. Such control

messages may in fact include rekeying messages and SA updates. GDOI refers to this SA as the push SA or rekey SA.

In GDOI, this is implemented as the GROUPKEY-PUSH datagram, where the message is pushed from the GCKS to the members.

- *Category 3 SA*: GDOI refers to this SA as the data security SA since it is used by the application to secure the data traffic (e.g., IPsec).

GDOI and IKE

The ISAKMP protocol [17] is a key management framework for transferring key and authentication data, independent of the key generation process. ISAKMP defines a set of protocol exchanges that set up a secure channel for key management, as well as the exchange of key and authentication data. Generalized payloads for exchanging key generation and authentication data are defined by ISAKMP. These payloads are combined with a DOI, which defines the specifics of key exchange protocol. ISAKMP is intended to support the negotiation of SAs for security protocols at all layers of the network stack, although in practice it is commonly used at the network layer.

The IKE protocol [19] is a widely deployed key exchange protocol. It is primarily used as a key exchange protocol for IPsec, but can be used for other security protocols as well. Hence, the authors of the GDOI protocol saw a number of advantages to making use of existing support for ISAKMP as a key management framework and IKE for the secure channel (namely, the Category 1 SA):

- Reusing much of the existing key management protocol promotes a single key management framework.
- Systems that provide network-layer protection of unicast data will have the same market needs to provide network-layer protection for multicast data.
- Using the same underlying protocol will reduce both the complexity and size of the key management code.
- Implementation can be achieved more expediently.

In attempting to understand GDOI, it is worth reviewing the two phases of IKE, since GDOI uses much of IKE Phase 1. Recall that IKE is logically divided into two exchanges, referred to as Phase 1 and Phase 2. A Phase 1 exchange must be completed before any Phase 2 exchanges are attempted.

Once the Phase 1 exchange has completed, there is no limit to the number of Phase 2 exchanges that can take place, and there may be simultaneous Phase 2 exchanges occurring between IKE peers.

- *Phase 1*: In Phase 1, two peers establish a bidirectional, secure, authenticated channel using payloads and semantics defined in ISAKMP. Several different authentication methods are defined for use in IKE (i.e., manually shared keys, digital signatures, or public key encryption). The two peers negotiate a mutually acceptable set of cryptographic policies, and derive keying material using the Diffie-Hellman or public key encryption algorithms. At the end of Phase 1, the two peers have fully authenticated each other, and have exchanged adequate keying material used to create a secure authenticated channel for Phase 2.
- *Phase 2*: In Phase 2, the two peers negotiate SAs on behalf of IPsec (or other security protocols if another DOI has been defined). IKE Phase 1 provides confidentiality, integrity, replay protection, and the generation of key exchange protocol keying material (i.e., using keying material exchanged during Phase 1).

The secure channel defined by an IKE Phase 1 is used by GDOI to protect GDOI keying material. This is because it can directly provide confidentiality and integrity. Furthermore, the IKE exchanges protect against man-in-the middle, connection hijacking, reflection, and replay attacks. IKE offers some protection against DoS attacks as well.

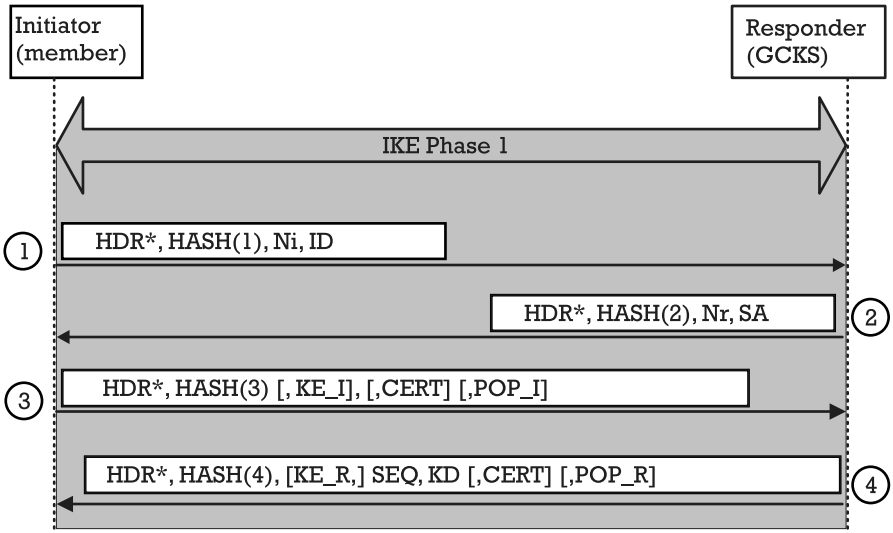
Although GDOI uses IKE Phase 1, it defines a new Phase 2 exchange called GROUPKEY-PULL. In fact, GDOI uses the IKE Phase 1 to protect the new Phase 2 exchange. This new Phase 2 exchange is discussed further below.

New elements in GDOI

The GDOI protocol implements the GSA concept. Thus, although borrowing Phase 1 from IKE (which is a unicast protocol), GDOI by necessity introduced a number of new elements to cater for multicast-related interactions between the GCKS and the members.

The GDOI protocol introduced several new payloads. These are as follows (see Figure 5.7):

- GDOI SA;
- SA KEK (SAK), which follows the SA payload, where the KEK is the key encryption key;



HASH(1) = prf(SKEYID_a, M-ID | Ni | ID)
HASH(2) = prf(SKEYID_a, M-ID | Ni_b | Nr | SA)
HASH(3) = prf(SKEYID_a, M-ID | Ni_b | Nr_b [| KE_I][|POP_I])
HASH(4) = prf(SKEYID_a, M-ID | Ni_b | Nr_b [| KE_R] |SEQ |KD [| POP_R])
POP payload is constructed from prf(Ni | Nr)
* Protected by IKE Phase 1 SA, encryption occurs after HDR

Figure 5.7 GDOI Phase 2 exchange.

- SA TEK (SAT), which follows the SA payload, where the TEK is the traffic encryption key;
- KD Array (denoted as KD, or key download, as borrowed from GSAKMP);
- Sequence number (SEQ);
- Proof of possession (POP).

In addition, the GDOI protocol introduced two new exchanges:

- A Phase 2 exchange to create Category 2 and Category 3 SAs.

  The new Phase 2 exchange called GROUPKEY-PULL, downloads KEK and/or TEK keying material, policy, and attributes for the group member. The GROUPKEY-PULL exchange uses pull behavior, since the member initiates the retrieval of these parameters from the

GCKS. The member is aware of the group through some announcement scheme (such as the SDP), and initiates the pull.

- A datagram to create or modify the Category 2 and Category 3 SAs. The GROUPKEY-PUSH datagram is pushed from the GCKS to the members. The KEK or KEK array protects the GROUPKEY-PUSH message, which creates a new Category 3 or Category 2 SA. When the GROUPKEY-PUSH carries a TEK, it creates a new Category 3 SA. Multiple Category 3 SAs can be specified through the SAT.

Note that the GCKS creates each Category 3 SA with a TEK on behalf of the security protocol that multicasts data. The security protocol is the protocol that will use the TEK for enciphering multicast data before it is transmitted through IP multicast.

A security protocol uses the TEK and ''owns'' the Category 3 SA in the same way that IPsec ESP uses the IKE Phase 2 keys and owns the Phase 2 SA. When the GROUPKEY-PUSH message carries a KEK array (or key hierarchy), it effectively dictates the creation of a new Category 2 SA. The GCKS creates a new Category 2 SA with a KEK array in order to add or remove group members, or to refresh the SAK or SAT [4, 5, 20]. Alternatively, membership may expire when the KEK expires [21], and the GROUPKEY-PUSH message is not used to create Category 2 SAs for the particular group. Use of LKH-style membership management (see Chapter 6) is an option in GDOI.

The new GDOI Phase 2

The new GDOI Phase 2 is shown in Figure 5.7. The goal of the GDOI Phase 2 (or GROUPKEY-PULL exchange) is to establish a Category 2 and/or Category 3 SA at the member for a particular group. The GDOI Phase 2 is protected by the Phase 1 (which is in fact an IKE Phase 1). Furthermore, multiple GROUPKEY-PULL exchanges may occur under a given Phase 1 SA.

One of the items transferred within the GROUPKEY-PULL exchange is the KEK for the group. Since the KEK can in fact contain an array of keys, it is also referred to a KEK array (which reflects the fact that it is a sequence of keys made up from the keys of an LKH or a key tree).

Looking at Figure 5.7, the GROUPKEY-PULL exchange uses many IKE definitions:

- *SKEYID*. IKE Phase 1 computes SKEYID_a from the Diffie-Hellman keying material exchanged in Phase 1. SKEYID_a is the ''key'' in the

keyed hash used in the GROUPKEY-PULL HASH payloads. As with the IKE HASH payload generation (RFC 2409 section 5.5), each GROUPKEY-PULL message hashes a uniquely defined set of values. Nonces permute the HASH and provide some protection against replay attacks. Replay protection is important to protect the GCKS from attacks that a key management server will attract.

- *Nonces*. The GROUPKEY-PULL exchange uses nonces to guarantee ''liveliness,'' or that someone is not replaying a recent GROUPKEY-PULL message. The replay attack is only useful in the context of the current Phase 1. If a GROUPKEY-PULL message is replayed based on a previous IKE Phase 1, the HASH calculation will fail due to a wrong SKEYID_a. The message will fail processing before the nonce is ever evaluated. In order for either peer to get the benefit of the replay protection, it must postpone as much processing as possible until it receives the message in the protocol that proves the peer is live. For example, the responder must not compute the shared Diffie-Hellman number (if key exchange payloads were included) or install the new SAs, until it receives a message with *Nr* included properly in the HASH payload.

  Nonces require an additional message in the protocol exchange to ensure that the GCKS does not add a group member until it proves liveliness. The GROUPKEY-PULL member-initiator expects to find its nonce, *Ni*, in the HASH of a returned message. Furthermore, the GROUPKEY-PULL GCKS-responder expects to see its nonce, *Nr*, in the HASH of a returned message before providing group keying material as in the following exchange.

- *HDR*. In Figure 5.7 HDR is an ISAKMP header payload that uses the Phase 1 cookies and a message identifier (M-ID) as in IKE [19]. Note that nonces are included in the first two exchanges, with the GCKS returning only the SA policy payload before liveliness is proven. The HASH payloads prove that the peer has the Phase 1 secret (SKEYID_a) and the nonce for the exchange identified by M-ID. Once liveliness is established, the last message completes the real processing of downloading the KD payload.

- *ID payload*. In addition to the nonce and HASH payloads, the member initiator identifies the group it wishes to join through the ISAKMP ID payload. The GCKS-responder informs the member-initiator of the current value of the sequence number in the SEQ payload. The sequence number orders the GROUPKEY-PUSH datagrams.

- *SA payload, DOI, TEK, KEK, and SPI*. The GCKS-responder informs the member-initiator of the cryptographic policies of the group in the SA payload, which describes the DOI, KEK, and/or TEK keying material, and authentication transforms. The SPIs are also determined by the GCKS, and downloaded in the SA payload chain.
- *KD payload*. The KEK SA contains the ISAKMP cookie pair for the Category 2 SA, which is not negotiated, but downloaded. The TEK SA also contains an SPI. The second message downloads this SA payload. If a Category 2 SA is defined in the SA payload, then KD will contain the KEK. If one or more Category 3 SAs is defined in the SA payload, then KD will contain the TEKs. This is useful if there is an initial set of TEKs for the particular group, and can obviate the need for future TEK GROUPKEY-PUSH messages.
- *Certificate and POP*. The member may establish an identity in the GROUPKEY-PULL exchange in an optional certificate (CERT) payload that is separate from the Phase 1 identity.

  When the member responder passes a new CERT, a POP payload accompanies it. The POP payload demonstrates that the member or GCKS principal has used the very secret that authenticates that principal (namely, the principal's private key that corresponds to the public key used in the CERT payload). POP_I is an ISAKMP signature (SIG) payload containing a hash of the concatenated nonces *Ni* and *Nr* signed by the member, when the member passes a CERT signed by the group owner, to prove its authorization. POP_R contains the hash of the concatenated nonces *Ni* and *Nr* signed by the GCKS, when the GCKS passes a CERT signed by the group owner, to prove its authority to provide keys for a particular group. The use of the nonce pair for the POP payload, transformed through the IKE pseudo random function (PRF), is designed to withstand compromise of the Category 1 (IKE Phase 1) key.

Updating SAs

One of the fundamental design aspects of the GSA model and GDOI is the use of multicast itself to distribute certain control messages. An example of this use of multicast is the updating of SAs using the GROUPKEY-PUSH message. Note that the message itself is not dependent on IP multicast, and as such can also be pushed using unicast delivery. The GROUPKEY-PUSH message replaces a Category 2 SA KEK (or KEK array), and/or creates a new Category 3 SA. See Figure 5.8.

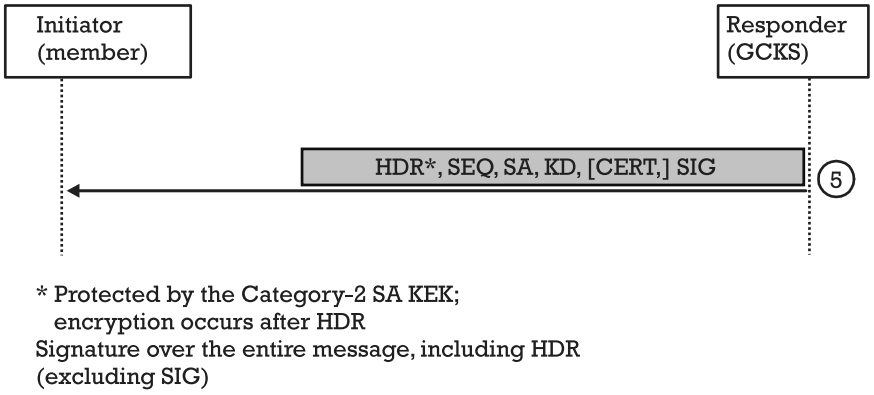


Figure 5.8 GDOI push message.

As before, the SA defines the policy (e.g., crypto suite) and attributes (e.g., SPI) for a Category 2 and/or Category 3 SA. The GCKS optionally provides a CERT payload for verification of the SIG, which is a signature of a hash of the entire message before encryption (including the header and excluding the SIG payload itself). The KD is the key download payload (described above).

If the SA defines an LKH-style KEK array or single KEK, the KD payload contains a KEK or KEK array for a new Category 2 SA, which has a new cookie pair. When the KD payload carries a new KEK SA, a Category 2 SA is replaced with a new SA having the same group identifier and incrementing the same sequence counter, which is initialized in message 4 of Figure 5.7. If the SA defines an SA TEK payload, this informs the member that a new Category 3 SA has been created, with keying material carried in the KD payload.

Functional block diagram of GDOI

As further illustration of the GDOI protocol, Figure 5.9 shows the functional block diagram (FBD) of the GCKS and member.

Members may be senders or receivers of multicast data. There are two functional blocks in the figure which are labeled ''GDOI,'' and there is an arc between them depicting the GDOI message exchange. Some of these functional blocks and the arcs between them are peculiar to an operating system (OS) or vendor product, such as vendor specifications for products that support updates to the IPsec SAD and SPD. The AUTHORIZATION block

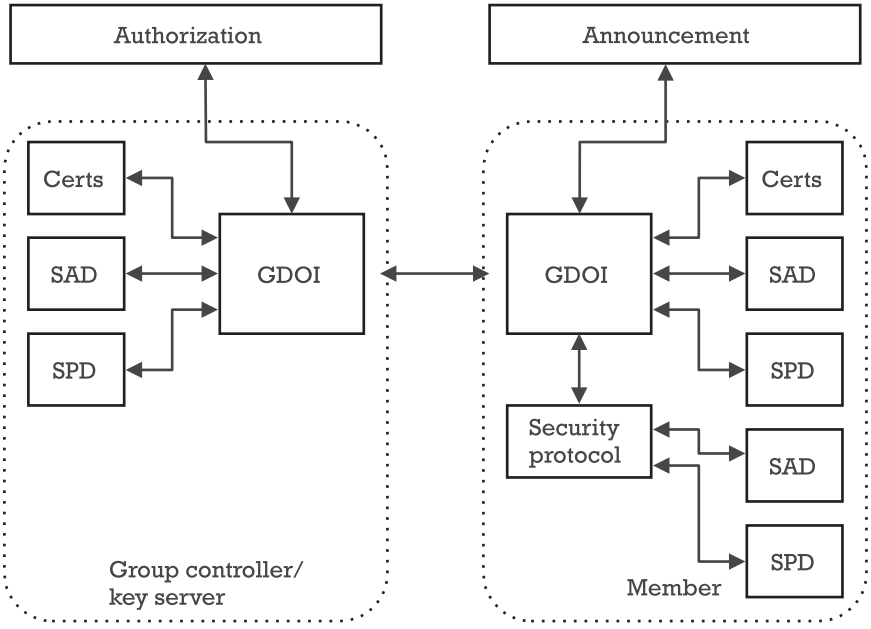


Figure 5.9 GDOI functional block diagram.

is subject to group policy [22], but how this is done is specific to the GCKS implementation. GDOI is open to supporting alternative authorization designs.

Besides the AUTHORIZATION block there is an ANNOUNCEMENT block. The announcement function is how a member receives announcements of secure groups and sessions. The SDP [23] is one form that the announcements might take. The announcement function may be implemented in a session directory tool, an electronic program guide (EPG), or by other means. The announcement function directs GDOI using an API, which is peculiar to the host OS in its specifics.

The reader is directed to the specifications of GDOI in [6] for further details on the GDOI protocol and to [24] for a formal analysis of the GDOI protocol.

5.5 Summary

This chapter has focused on two related and important aspects of multicast and group security; namely, architectures and protocols for group key

management. In the area of architectures, two designs have been described, representing two ends of a spectrum of possible architectures. These are the layered (or hierarchic) architecture (represented by [1]) and the Iolus architecture (represented by [2]).

The second part of the chapter sampled some group key management protocols that have focused on security, key management, and SA management to secure multicast and group communications. These are not meant to be comprehensive, but provide an insight into the complexity of designing protocols for group key management.

The chapter's discussion on the issues and motivations for architectures and protocols points to the fact that group key management cannot be implemented independent of other factors in the wider environment, such as the application of multicast, topological aspects, membership dynamics, and others. These factors must be taken into consideration when architecting security and deploying protocols for group key management.

References

[1] Hardjono, T., B. Cain, and I. Monga, ''Intra-Domain Group Key Management Protocol,'' draft-ietf-ipsec-intragkm-02.txt, IETF, February 2000, work in progress.

[2] Mittra, S., ''Iolus: A Framework for Scalable Secure Multicasting,'' in *Proc. of ACM SIGCOMM*, Cannes, France, September 1997, pp. 277–288.

[3] Harney, H., and C. Muckenhirn, ''Group Key Management Protocol (GKMP) Architecture,'' RFC 2094 (experimental), July 1997.

[4] Wallner, D., E. Harder, and R. Agee. ''Key Management for Multicast: Issues and Architectures,'' RFC 2627 (informational), IETF, June 1999.

[5] Harney, H., et al., ''Group Secure Association Key Management Protocol,'' draft-ietf-msec-gsakmp-sec-00.txt, IETF, March 2001, work in progress.

[6] Baugher, M., et al., ''Group Domain of Interpretation for ISAKMP,'' draft-ietf-msec-gdoi-04.txt, IETF, March 2002, work in progress.

[7] Ballardie, A., ''Scalable Multicast Key Distribution,'' RFC 1949 (experimental), IETF, May 1996.

[8] Harkins, D., and N. Doraswamy, ''A Secure Scalable Multicast Key Management Protocol (MKMP),'' draft-ieft-ipsec-mkmp-00.txt, IETF, November 1997, expired.

[9] Valdvogel, M., et al., ''The VersaKey Framework: Versatile Group Key Management,'' *IEEE JSAC Special Issue on Service Enabling Platforms For Networked Multimedia Systems*, Vol. 17, No. 9, September 1999.

[10] Whetten, B., et al., ''Reliable Multicast Transport Building Block for TRACK,'' draft-ietf-rmt-bb-track-01.txt, IETF, March 2001, work in progress.

[11] Speakman, T., et al., ''PGM Reliable Transport Protocol Specification,'' RFC 3208 (experimental), IETF, December 2001.

[12] Meyer, D., ''Administratively Scoped IP Multicast,'' RFC 2365 (best current practice), IETF, July 1998.

[13] Thaler, D., M. Handley, and D. Estrin, ''The Internet Multicast Address Allocation Architecture,'' RFC 2908 (informational), IETF, September 2000.

[14] Wong, C. K., M. Gouda, and S. S. Lam, ''Secure Group Communications Using Key Graphs,'' *IEEE/ACM Tran. on Networking*, Vol. 8, No. 1, February 2000, pp. 16–30.

[15] Dondeti, L. R., S. Mukherjee, and A. Samal, ''Scalable Secure One-to-Many Group Communication Using Dual Encryption,'' *Computer Communications*, Vol. 23, No. 17, November 2000, pp. 1681–1701.

[16] Molva, R., and A. Pannetrat, ''Scalable Multicast Security in Dynamic Groups,'' in *6th ACM Conference on Computer and Communication Security*, Singapore, November 1999.

[17] Maughan, D., et al., ''Internet Security Association and Key Management Protocol (ISAKMP),'' RFC 2408 (proposed standard), IETF, November 1998.

[18] Piper, D., ''The Internet IP Security Domain of interpretation for ISAKMP,'' RFC 2407 (proposed standard), IETF, November 1998.

[19] Harkins, D., and D. Carrel, ''The Internet Key Exchange (IKE),''RFC 2409 (proposed standard), IETF, November 1998.

[20] Balenson, D., D. McGrew, and A. Sherman, ''Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization,'' draft-irtf-smug-groupkeymgmt-oft-00.txt, IRTF, August 2000, work in progress.

[21] Briscoe, B., ''MARKS: Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences,'' in *Proc. of First International Workshop on Networked Group Communication (NGC)*, Pisa, Italy, November 1999.

[22] Harney, H., A. Colegrove, and P. McDaniel, ''Principles of Policy in Secure Groups,'' in *Proc. of Network and Distributed Systems Security 2001 Internet Society*, San Diego, CA, February 2001.

[23] Handley, M., and V. Jacobson, ''SDP: Session Description Protocol,'' RFC 2327 (proposed standard), IETF, April 1998.

[24] Meadows, C., P. Syverson, and I. Cervesato, ''Formalizing GDOI Group Key Management Requirements in NPATRL,'' in *Proc. 8th ACM Computer and Communications Security Conference (CCS'01)*, Philadelphia, PA, November 2001, pp. 235–244.

CHAPTER

6

Group key management algorithms

Contents

6.1 Batch and periodic rekeying

6.2 MARKS

6.3 LKH

6.4 OFT

6.5 Batch processing of membership changes in key trees

6.6 Reliable transport of rekey messages

6.7 Stateless key revocation algorithms

6.8 Summary

Privacy is a requirement of several group communication applications such as conferencing, whereas access control is a requirement of applications such as stock quote distribution via the Internet. Both group privacy and access control can be enforced by encrypting the group data and distributing the group key to current members. Group members need to hold a common key for group authentication as well (see Chapter 3). In the previous two chapters we discuss GSAs, group key distribution protocols, and architectures for efficient group key management. We focus on group key distribution and scalable rekeying algorithms in this chapter.

Group keys must be changed from time to time for various reasons. The first purpose is to keep the encryption key fresh, that is, to use the key for only a certain amount of time or for encrypting a certain amount of data. The second requirement of rekeying is specific to groups. Consider groups in which the group owner wants to disallow joining members from decrypting past data, and departing members from decrypting future data. To do so, the sender or the GCKS must change the group key each time the group membership changes.

A naïve group rekeying scheme encrypts the group key separately for each of the remaining members of the group. Clearly, such a scheme has computational and communication complexity that is linear in group size, which is inefficient. In the previous chapter, we discussed several secure group management architectures, for example, Iolus, that propose

hierarchical subgrouping for efficient rekeying. At the cost of additional infrastructure, those schemes limit rekeying due to membership changes to the respective subgroup(s), instead of the whole group. But unless the subgroups are very small in size, the naïve rekeying scheme is inefficient for subgroup rekeying as well. Scalable rekeying algorithm design and analysis is the focus of this chapter. We discuss the various proposed group key management algorithms proposed in the literature, and their suitability to practical application scenarios.

First, we consider applications in which the GCKS rekeys the group at fixed instances in the lifetime of the group, and knows each member's departure time when the member joins. MARKS is a mechanism for efficient key distribution to such groups. Members receive random seeds from which they can generate the group encryption keys used during the period(s) of their subscription. Commercial applications such as satellite TV distribution with metered charging (e.g., per hour or per boxing round), may use this rekeying algorithm.

Next, we consider applications in which members may join or leave any time. For military and other highly secretive communications, the GCKS must rekey after each membership change to maintain strict forward and backward access control. However, for commercial applications, periodic or batch processing of membership changes may be sufficient, with provisions for immediate rekeying, to evict a member, for example. Thus, we need a general purpose group key management algorithm that can rekey a group immediately or periodically, as necessary.

A common technique in several proposed group key management schemes is to divide the members into subgroups, each of which has an associated secret key, and distribute the group key encrypted with the subgroup keys. Some of the proposed mechanisms update (some of) the subset keys during rekeying, while others do not. Rekeying is *stateful* in schemes that update subset keys. In other words, stateful rekeying schemes use keys sent in past rekeying messages to protect current rekeying messages. This property requires reliable transport of rekey messages, even if the application does not need reliable data transmission.

One of the earliest and most popular group key management algorithms known as LKH, assigns each member a leaf position in a logical key tree for key distribution. Each internal node of the tree represents a subgroup that contains all the descendant leaf nodes as members. Each member knows the keys of all the subgroups it belongs to, or all the nodes in its path to the root in the key tree. Each internal node's key is protected by its children's keys during rekeying. When a member joins or leaves, the GCKS needs to change and distribute only $O(\log n)$ keys, for a group of size n. Recall that the naïve

rekeying algorithm requires the GCKS to encrypt and transmit the group key $O(n)$ times.

Several improvements to LKH have been proposed in the literature. The first one, known as LKH+, improves join rekeying efficiency, and a second approach introduces a key construction scheme using one-way function chains (OFCs), for efficient rekeying to handle member departures. Note that in LKH and its enhancements mentioned so far, the group manager generates and distributes all the keys.

In another key management scheme based on OFTs, members receive secret information from the GCKS and compute some internal node keys in the tree, including the group key. OFT reduces leave rekeying communication overhead, at the expense of additional computations at the GCKS and the members. Unlike MARKS, LKH, LKH+, OFT, or OFC require no a priori knowledge of group joins or leaves.

Recall that stateful rekeying algorithms, while efficient, require reliable transport of rekey messages irrespective of application requirements. Stateless rekeying algorithms get around this requirement by sending the group key encrypted with predistributed KEKs, in rekeying messages. In these schemes, only the group key is rekeyed and, therefore, rekey messages are independent of past rekey messages. When group membership changes, the remaining members are divided into several disjoin subsets, such that each member belongs to exactly one of the subsets. The GCKS then sends the new group key encrypted with the subset keys, predistributed to each of the subset's members individually. Notice that subset keys themselves are not updated during rekeying.

In the rest of this chapter, we first discuss batch and periodic rekeying. Next, we discuss MARKS and the stateful rekeying algorithms LKH, LKH+, OFC, and OFT. We then discussion stateless rekeying mechanisms based on subset trees and subset difference. Next, we discuss reliable transport of rekey messages. We conclude the chapter with a summary and a discussion on the applicability of group key management algorithms.

6.1 Batch and periodic rekeying

Group rekeying serves two major purposes. First, secret keys need to be refreshed after they have been used to encrypt a certain amount of data, or after they have been in use for a given amount of time, to thwart cryptanalysis. Second and specific to groups, rekeying is used to ensure that only current members can decrypt data. In other words, the GCKS may change keys to disallow new members from getting access to past data

(backward access control), and departed or evicted members from getting access to future data (forward access control). Rekeying to enforce forward and backward access control is more difficult, since it needs to deal with membership changes, and is more frequent in large or dynamic groups. Therefore, we limit the rest of our discussion to rekeying due to membership changes.

Some multicast applications, for example, military communications and conferencing, may require that the group key be changed immediately after each membership change, to enforce *strict* forward and backward access control. It may even be necessary to stop data flow while such groups are being rekeyed [1].

In large and highly dynamic groups, such an immediate rekeying policy may result in frequent rekeying, and might overwhelm the GCKS computationally [2]. Rekey communication overhead may also be seen as too costly for efficient operation of the group. In such cases, it is possible to relax forward and backward access control requirements slightly, to reduce rekeying overhead. A GCKS may choose to rekey a group periodically, or process group membership changes in batches [2–5]. Commercial applications such as Internet or satellite TV distribution may use batch or periodic rekeying, employing immediate rekeying only in exceptional circumstances, for example, when evicting misbehaving members.

Figure 6.1 illustrates immediate, periodic, and batch rekeying. In immediate rekeying, the GCKS spawns a rekeying instance each time the group membership changes. In batch rekeying, the GCKS rekeys after a few group membership changes. The GCKS may choose to rekey, for example, after r group membership changes or, alternatively, after r joins or r leaves. In Figure 6.1, the GCKS rekeys after four membership changes. Finally, the GCKS may rekey every t time units, irrespective of membership dynamics. We refer to this final type of rekeying as periodic rekeying.

6.1.1 Trade-offs in batch rekeying

Batch or periodic rekeying comes at the expense of relaxing forward and backward access control requirements. More precisely, note that when a member departs, the GCKS does not rekey immediately. Instead, it waits until the next rekeying instance (refer to Figure 6.2). But this is against the forward access control rule. Due to batch rekeying, departing members continue to get access to group data for a brief period (until the next rekeying instance) after they leave.

Fortunately, joins in batch rekeying mode can be processed without affecting strict backward access control. The GCKS has two options. The first

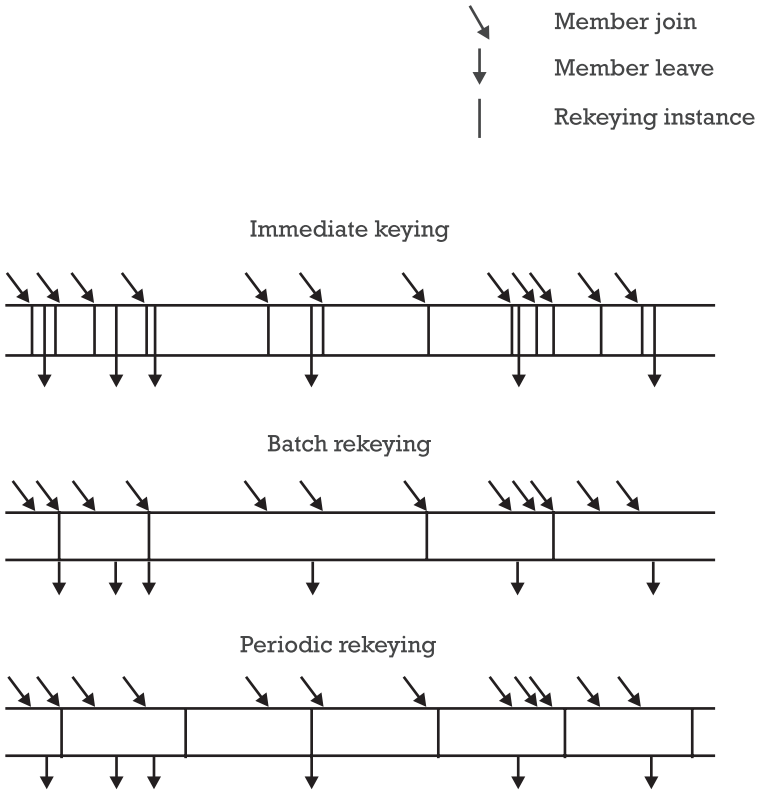


Figure 6.1 Immediate, periodic, and batch rekeying.

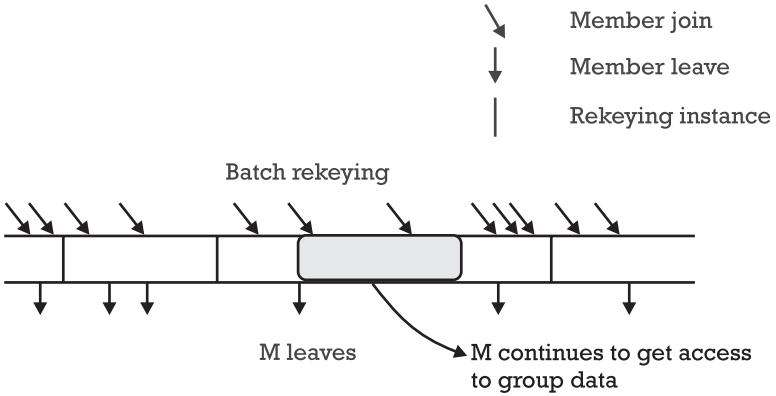


Figure 6.2 Batch rekeying and forward access control.

approach is a variation of LKH+ [6] (see Section 6.3.4). The GCKS applies a one-way function to the group key, and sends the new group key to the new member. The existing members receive a notification about the application of the one-way function, and compute the new group key. New members become part of the key tree only at the next rekeying instance, as determined by the batch or periodic rekeying scheme. In the second approach, joining members are put on hold until the next rekeying instance. They cannot decrypt group data until then.

6.2 MARKS

MARKS is a mechanism for key distribution to a group of members who depart at prespecified time points [7]. Consider a secure group in which members may join the group any time during the group, but the departure time must be known at the time of join. Further assume that the GCKS knows how long the secure group communication lasts.

To manage such secure groups, the GCKS divides the group's life into t time periods. Each time period has a corresponding group key. In all, there are t group keys, $K_1, K_2, \ldots, K_t$. Each member joins the group and requests to be a member between time periods say, $t1$ and $t2$. The GCKS supplies the the group keys, $K_{t1}, K_{t1+1}, \ldots, K_{t2-1}, K_{t2}$, to that member. MARKS [7] presents an efficient mechanism for such key distribution using a hierarchy of seeds.

First we describe the mechanism used to construct a binary hierarchy of seeds. The GCKS starts with a randomly generated seed as the root of the hierarchy. We compute two seeds corresponding to the root's children by applying two different one-way functions g_L and g_R. The GCKS continues this process until the number of leaves in the hierarchy of seeds is t, that is, the number of periods in the group. Figure 6.3 depicts the seed generation mechanism. The GCKS generates the root seed $s_{0,0}$ and uses g_L and g_R to compute $s_{1,0}$ and $s_{1,1}$, corresponding to the left and right children, respectively. In the figure, the seed name $s_{i,j}$ represents the jth seed at depth i.

During registration, the GCKS sends the seeds necessary and sufficient for the member to compute the group keys that the member is authorized to receive. The seed hierarchy facilitates efficient distribution of group keys to members. For example, to allow a member to remain in the group for the entire duration, the GCKS needs to just send the root seed. In general, given a seed corresponding to a node in the hierarchy, one can compute the

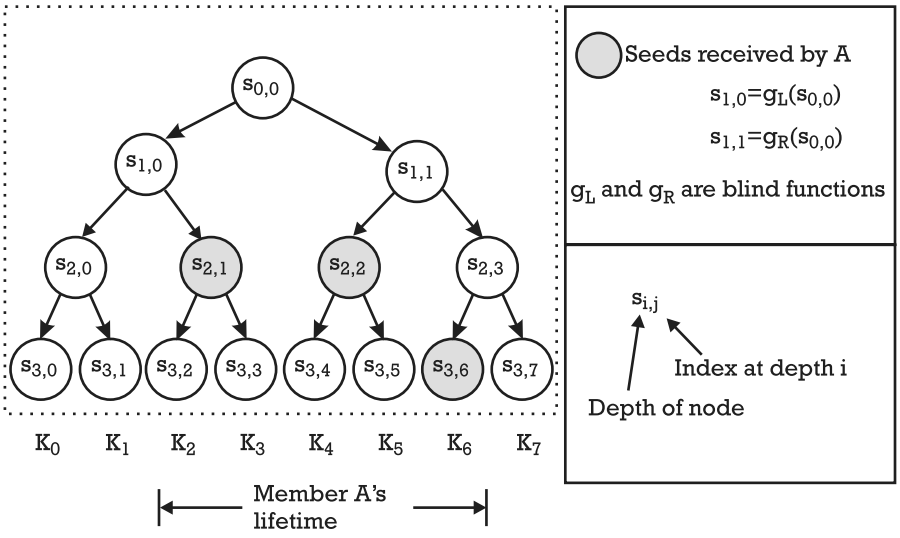


Figure 6.3 Key distribution in MARKS.

seeds of all descendant nodes. In Figure 6.3, say member A is authorized to receive $K_2, K_3, \ldots, K_6$. Instead of sending the five group keys, GCKS sends the three seeds, $s_{2,1}$, $s_{2,2}$ and $s_{3,6}$.

Notice that all key distribution in MARKS is done during the registration protocol. The advantage is that all the keys are transported over unicast; thus it is easy to ensure reliable delivery. The disadvantage, however, is that we cannot eliminate a member without rekeying the entire group, which is very inefficient in large groups.

We conclude this section with an analysis of MARKS key distribution. The minimum number of seeds that a GCKS could send a member is one seed. The single seed could correspond to a leaf node, which reveals one group key, or an internal node, which reveals 2^h group keys, where h is the height of the subtree. The maximum number of seeds are sent to a member who is authorized to receive all group keys except the first and the last one, that is, K_2, K_2, ..., K_{t-1}. The GCKS needs to send the required seeds without revealing K_1 and K_t. Thus it cannot reveal the ancestors of $s_{l,0}$ or of $s_{l,2^{l-1}}$. Instead, the GCKS sends the seeds of siblings of the ancestors of $s_{l,0}$ $s_{l,2^{l-1}}$. In all, the GCKS sends $2\log(t/2)$ seeds. The worst-case one-way function computation overhead applies to a member who stays in the group for the entire duration. It receives the root seed and needs to compute all the seeds in the hierarchy, that is, $2t-2$.

6.3 LKH

Logical key trees [6, 8, 9], where each node represents a KEK, facilitate efficient rekeying of large groups. Each leaf node of the key tree is associated with a member of the group, and the root node represents the group key. Each internal node represents a logical subgroup, and that node's key is known to all members associated with its descendant leaf nodes. The logical subgroup concept is in contrast to hierarchical subgrouping in Iolus [10] (see Section 5.3), where subgroup managers are physical entities. The GCKS holds all the KEKs, and sends only a subset of them to each member.

Each member's unique leaf node position is the reference for key distribution. Members belong to the subgroups represented by each of its leaf node's ancestors in the key tree. Thus, each member receives keys of all nodes in the path from its leaf node to the root. When membership changes, the GCKS changes and distributes only the keys known to the joining/departing member. The number of keys each member holds is equal to the height of the tree, that is, $\log n$. In the rest of this section, we describe LKH key distribution and rekeying mechanisms in detail.

The LKH key distribution rule is that each member receives all the keys in the path from its associated leaf node to the root of the key tree.

Figure 6.4 is an example of an LKH. A, B, ..., H are members, and all other nodes represent keys. Corresponding to each of the members, we have

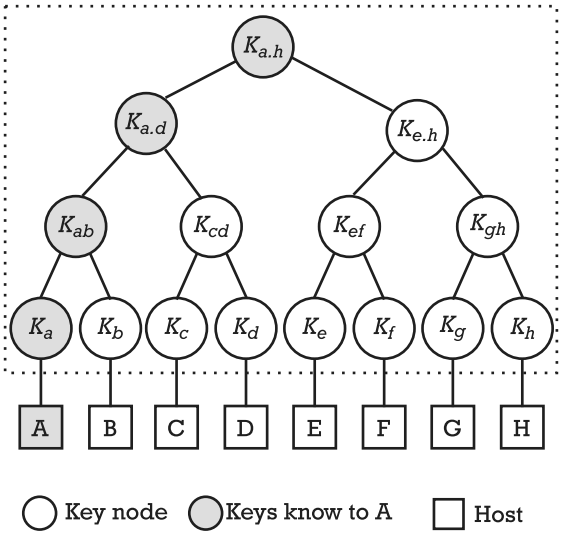


Figure 6.4 An example of a key hierarchy.

leaf key nodes $K_a, K_b, \ldots, K_h$. We now examine some of the key assignments which illustrate the rules explained earlier.

- A receives the keys, $K_a, K_{ab}, K_{a.d}, K_{a.h}$, corresponding to the nodes in its path to the root.
- The GCKS sends $K_{a.d}$ to all the descendants of that node, that is, A, B, C, and D.
- All members receive the group key $K_{a.h}$.

6.3.1 Initializing an LKH

We establish an LKH as follows. First, the GCKS creates a rooted binary tree that has at least as many leaf nodes as there are members. Next, it shares a separate, unique secret key with each one of the group members. These are the leaf node keys. The GCKS then generates keys corresponding to each internal node, and distributes each internal node key encrypted with each of its children's keys. Each internal node key is encrypted with two different keys and sent to the group. In all, the GCKS sends $n-1$ (number of nonleaf nodes in a full tree) keys to a group of size n. Figure 6.5 illustrates initialization of an LKH.

There is an alternative way of initializing a key tree. Recall that the GCKS needs to send each member all the keys in its path to the root. Following this rule, the GCKS separately sends each member the internal node keys it needs, encrypted with the member's leaf node (unique) key. In all, the GCKS needs to send $O(n \log n)$ encrypted keys, and hence this approach is inefficient compared to the one described in the previous paragraph.

6.3.2 Adding a member to a key tree

When a member joins the group, the GCKS needs to add the member to the key tree. There are a couple of different possible scenarios. In the first, we have an empty slot available for the new member. In the second scenario, we have a full and balanced tree with no empty slots. We create an empty slot in a key tree by splitting nodes.

Node splitting creates one or more empty slots in the tree [11]. The GCKS may choose the root node, a leaf node or an internal node to split, depending on how many empty slots it wants to create. For example, splitting a leaf node creates one empty slot, and splitting the root node can

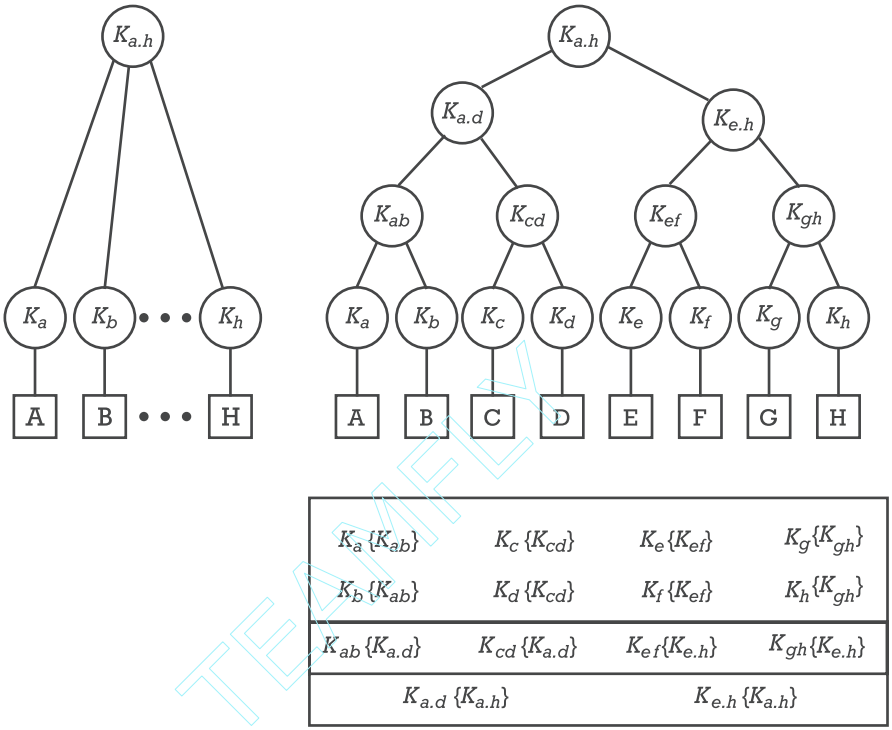


Figure 6.5 Initializing a binary key tree.

accommodate as many new members as the current membership. In general, splitting a node creates 2^l empty slots, where l is the level of the node (assuming leaf nodes are at level 0).

6.3.3 Join rekeying in LKH

After the GCKS finds a leaf node to associate with the new member, it shares the group key and other keys that the new member needs, to be part of the LKH. The GCKS could just send the keys to the new member, encrypted with the unique leaf node key that only the new member and the GCKS know. Notice that if the new member were recording previous transmissions to the group, it could decrypt some of that old data. To keep new members from decrypting old data, that is, to enforce backward access control, the GCKS needs to rekey the internal node keys that the new member receives.

The GCKS generates new keys to replace keys corresponding to the nodes in the path from the new member's position in the tree to the root. It needs

to send all those keys securely to the new member. One way to do this is to send the leaf node key during the registration protocol, and send the internal node keys encrypted with the leaf node key given to the new member, via the rekey protocol. The GCKS also sends each newly generated internal node key encrypted separately with the key it is replacing, to the other members.

In Figure 6.6, E is the new member, and we assume that the GCKS splits D's leaf node to create a position for E in the key tree. It sends K_e to E during registration. It then sends the new keys as follows:

- $K_{a.e}$, $K_{c.e}$, and K_{de} encrypted with K_e for E;
- $K_{a.e}$ encrypted with $K_{a.d}$ for A, B, C, and D;
- $K_{c.e}$ encrypted with K_{cd} for C and D;
- K_{de} encrypted with K_d for D.

Join rekeying complexity in LKH

Consider join rekeying in a full and balanced binary key tree. The GCKS, as explained earlier, needs to change all the internal node keys in the new member's path to the root. In other words, the GCKS needs to send $\log_2 n$ keys, where n is the number of members (leaf nodes in the key tree) in the group. The GCKS encrypts and sends each of those internal node keys twice; once for the new member and once for the existing members. Therefore, the GCKS needs to send $2 \log_2 n$ keys.

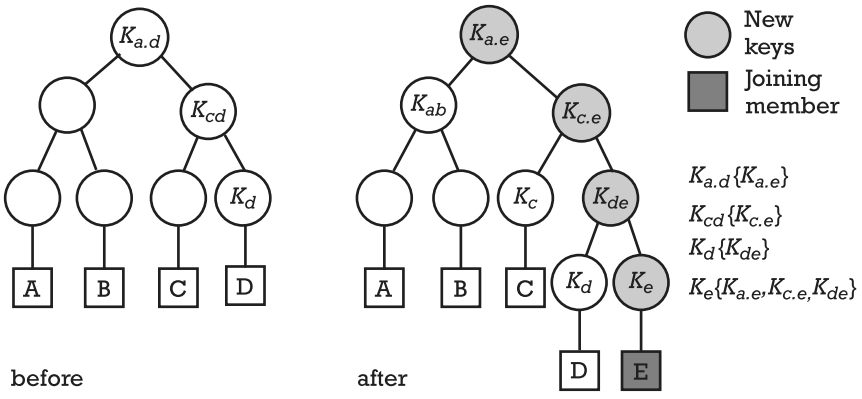


Figure 6.6 Join rekeying in LKH.

6.3.4 Efficient join rekeying using LKH+

The second method of join rekeying works as follows. Recall that the GCKS finds/creates a position for the new member in the key tree. It then generates a new key to serve as the leaf node key of the new member. The new member also receives keys of all internal nodes in its associated leaf node's path to the root. Some of these keys are new, and others replace existing keys that are not going to be used further.

Several researchers suggested [6] generating the replacement internal node keys by applying a one-way function to the keys they are replacing. Note that the advantage is that the GCKS does not need to transmit any keys to existing members. It just needs to notify the respective members of the key updates. The GCKS does need to send the keys to the joining member, however. Thus, join rekeying in LKH+ requires encryption and transmission of only $\log_2 n$ keys, that is, half that in LKH without any enhancements.

6.3.5 Leave rekeying in LKH

When members leave a secure multicast group, the GCKS needs to rekey the group to ensure forward access control. The GCKS needs to change only the keys that the departing member knows, that is, the keys of nodes in the departing member's path to the root.

The GCKS has a choice in key tree maintenance. It can contract (i.e., delete empty nodes and corresponding links) the tree, or just remove the departing member and leave the position empty. Note that the GCKS may assign the empty position to a future member without sacrificing forward or backward access control.

To prevent the departing member from extracting any of the keys from the key tree, the GCKS changes all the keys the member knows, and sends the new keys encrypted with keys that the member does not have access to. In Figure 6.7, E is the departing member. The GCKS eliminates E's position in the tree. Among the keys that E knows, K_e and K_{de} are no longer necessary for maintaining the group key. The GCKS replaces the other keys E knows, that is, $K_{a.e}$ and $K_{c.e}$, with $K_{a.d}$ and K_{cd}, respectively. Now it needs to send the new keys to the remaining members securely, without E being able to decrypt them. Thus, it sends:

- $K_{a.d}$ encrypted with K_{ab} and K_{cd};
- K_{cd} encrypted with K_c and K_d.

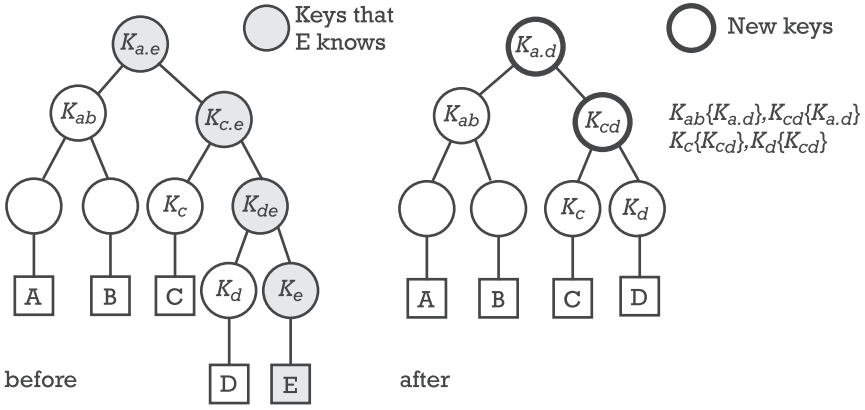


Figure 6.7 Leave rekeying in LKH.

Leave rekeying complexity

When a member leaves, the GCKS changes all the keys that the member knows. Thus the GCKS rekeys as many keys as the length of the departing member's path to the root. In a full and balanced binary tree, that path length is $\log_2 n$. Further, each new key K_i is encrypted twice; once per each child of the internal node i. Thus, leave rekeying requires $2\log_2 n$ key encryptions and transmission.

6.3.6 Efficient leave rekeying using OFCs

It is possible to improve leave rekeying complexity using cryptographic techniques to generate internal nodes keys from a single piece of random information, *ri* [12]. In Figure 6.8, when E leaves, the GCKS generates random data *ri* of length equivalent to the desired key size. It then uses a length-doubling pseudorandom function [13] $PRF(ri)$ to generate $L(ri)R(ri)$, where $|L(ri)| = |R(ri)| = |ri|$. With a cryptographically strong pseudorandom function, given $R(ri)$ or $L(ri)$, it is computationally infeasible to compute ri. The GCKS repeatedly applies the one-way function (PRF) on ri to derive new KEKs and hence the name OFC.

In our example, the GCKS sends ri encrypted with K_c and K_d to members C and D respectively, and $L(ri)$ encrypted with K_{ab} for A and B. The new keys, $K_{cd} = R(ri)$ and $K_{a.d} = R(L(ri))$ are generated from ri. Notice that C and D can compute both K_{cd} and $K_{a.d}$, whereas A and B can compute only $K_{a.d}$, which is pursuant to the LKH key distribution rule.

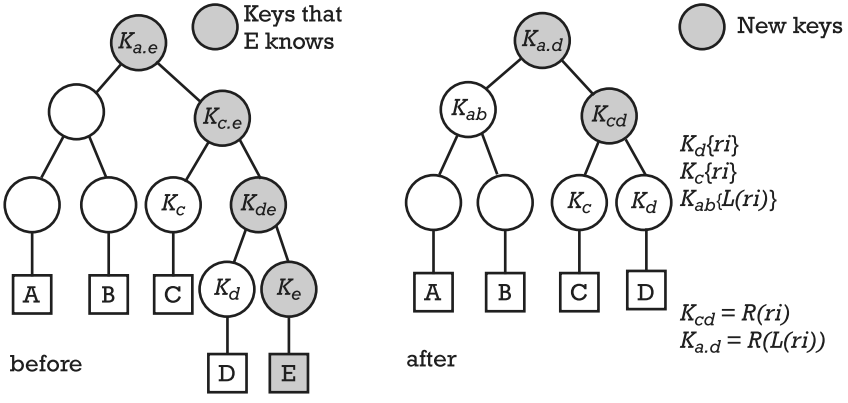


Figure 6.8 Improving leave rekeying efficiency using OFC.

In general, consider a tree of height h in which a member L leaves. L's sibling M gets associated with its parent key node in the tree. The GCKS generates ri and sends it securely to M. The new internal node keys are $R(ri)$, $R(L(ri))$, $R(L(L(ri)))$, ..., $R(L^h(ri))$, starting at M's new parent node and proceeding toward the root. For the members to compute the keys that they are entitled to, the GCKS sends ri, $L(ri)$, ..., $L^h(ri)$ encrypted with sibling node keys of M's ancestors in the key tree, starting with M's own sibling node key. More precisely, the GCKS sends:

- ri encrypted with M's sibling node key;
- $L(ri)$ encrypted with M's parent's sibling node key;
- $L(L(ri))$ encrypted with M's grandparent's sibling node key;
- $L(L(L(ri)))$ encrypted with M's great grand parent's sibling node key.

In all, the GCKS sends $\log_2 n$ encrypted keys. Notice that in this approach, the GCKS sends only half as many encrypted keys as in LKH. Members are responsible for computing the other keys that the GCKS would have sent in LKH without the OFC enhancement.

6.4 OFT

OFTs are also logical key trees used for efficient rekeying, but with better leave rekeying efficiency compared to LKH [11]. OFT is in several ways similar to LKH. The root key is used as the group key, and each member

is associated with a unique leaf node and knows the secret key of that node. Further, each member knows the secret keys of the nodes in its associated leaf key node's path to the root. However, the GCKS does not send those secret keys to the members. Instead, it supplies other keys that the members use to compute their ancestors' keys, including the group key.

The GCKS uses a one-way function g to compute a *blinded key* corresponding to each KEK in the key tree. Given $g(x)$, it is computationally infeasible to compute x. Each internal node key is computed by applying a mixing function f to the blinded keys of its child node keys (see Figure 6.9).

The OFT key distribution rule is that each member receives the blinded keys of the siblings of the nodes in the path from its associated leaf node to the root of the tree.

We illustrate OFT key distribution in Figure 6.9. In the figure, the GCKS shares a unique secret key, K_a, with A via a secure channel established during the registration protocol. It then sends the blinded keys K'_b, K'_{cd}, and $K'_{e.h}$, so A can compute the unblinded keys K_{ab} and $K_{a.d}$, and $K_{a.h}$, respectively.

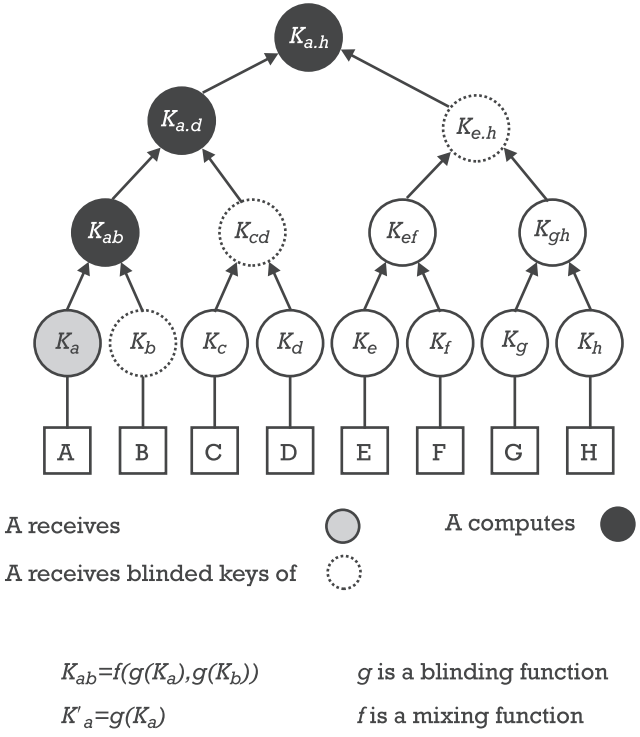


Figure 6.9 OFT key distribution.

Let us examine f and g further with the help of Figures 6.9 and 6.10. In the figures, $K'_b = g(K_b)$ and $f(K_{ab}) = f(K'_a, K'_b)$. Typically, g is a hash (e.g., MD5, SHA-1) function, whereas f is an XOR function.

6.4.1 Initializing an OFT

OFT initialization is efficient if the GCKS sends each blinded key once to the group, instead of sending it separately to each member. We establish an OFT as follows. The GCKS shares a unique secret with each member of the group during registration. That secret serves as the leaf node key. Each member computes the blinded version of its own leaf node key. The GCKS also computes the blinded keys, and sends each leaf node's blinded key, encrypted with its sibling's unblinded key. For example, in Figure 6.10, the GCKS sends K'_b encrypted with K_a. The GCKS then computes the unblinded key of each node, n_i, in the next level of the tree by applying the mixing function to the blinded keys of n_i's children. Similarly, each member

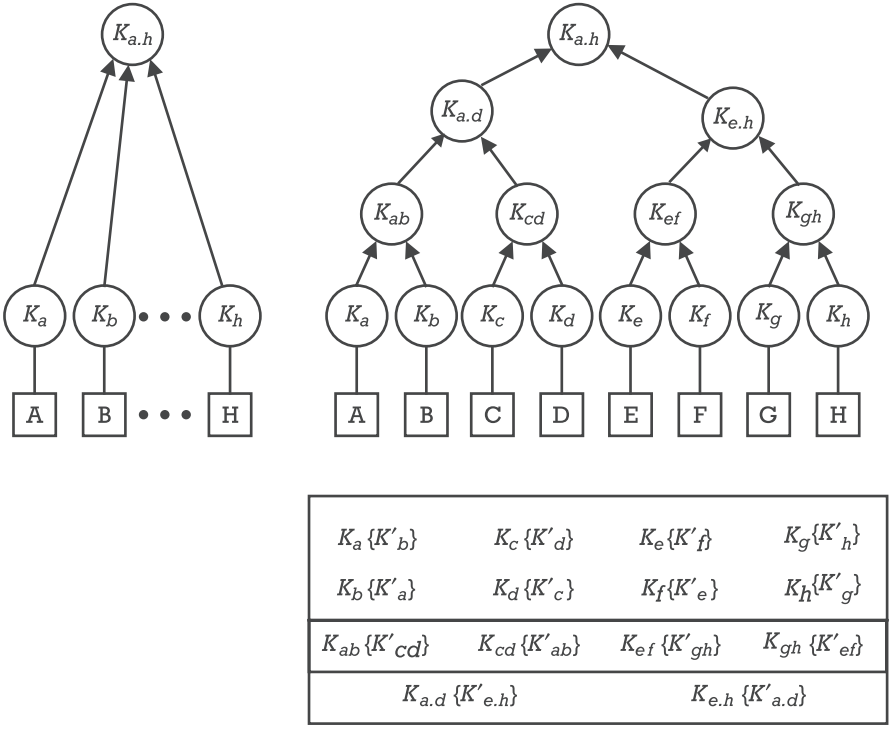


Figure 6.10 Initializing an OFT.

computes its parent node's unblinded key using its own and its sibling's blinded keys. This process continues until all members compute the group key.

In summary, the GCKS sends each blinded key, K'_{n_i}, encrypted with n_i's sibling's unblinded key. Key computation proceeds from the leaf nodes to the root.

6.4.2 Join rekeying in OFT

When a member joins, the GCKS creates a new position in the tree by splitting a node. Note that OFT key computation requires each internal node to have two children. Therefore, it is best to split a leaf node, and thus create a single empty position for the new member. Internal and root node splitting is inefficient at best.

To maintain a balanced and thus efficient key tree, it is desirable if the GCKS splits the nearest leaf node from the root. After accommodating the new member in the key tree, the GCKS shares a unique secret key with it. The GCKS also changes the leaf node key of the new member's sibling, and computes all the new internal node keys. It then sends the blinded keys the new member is entitled to, after encrypting them with the member's unique secret key. Existing members also need access to blinded keys that change, due to the addition of the new member to the key tree. Recall that all the blinded keys in the path from the new member's position to the root change. The GCKS encrypts each of those keys with the corresponding sibling's unblinded key, and sends it to the group.

We illustrate join rekeying in OFT using Figure 6.11. In the figure, E joins and we may split the leaf node corresponding to D to create an empty leaf node for E. The GCKS generates K_e and changes K_d^{old} to K_d. Note that C knows K'_d, and that would violate the OFT key distribution rule. Therefore, the GCKS changes K_d and computes the new keys. It then sends:

- The new leaf node key (unblinded), K_d, and K'_e encrypted with K_d^{old}, for D;
- K'_{ab}, K'_c, and K'_d, encrypted with K_e for E;
- The new blinded keys, $K'_{c.e}$ and K'_{de}, encrypted with K_{ab} and K_c, respectively.

All members then compute the other KEKs including the new group key.

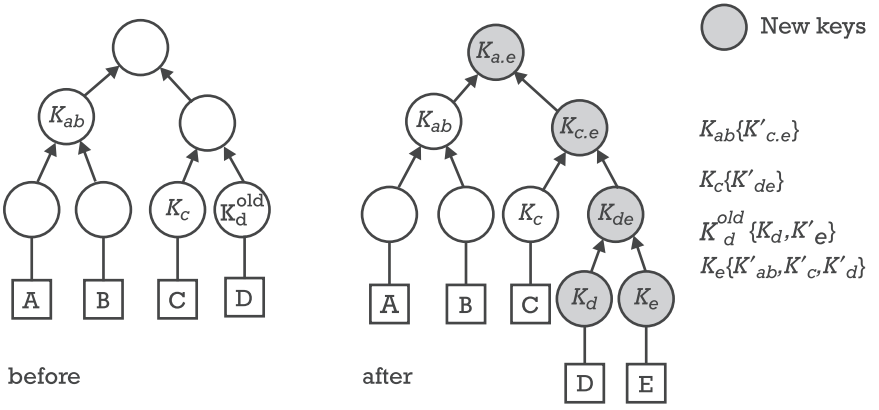


Figure 6.11 Join rekeying in OFT.

Join rekeying complexity in OFT

Consider a member joining a full and balanced binary OFT with n members (after the join). The GCKS needs to send the new member $\log_2 n$ blinded keys. This corresponds to the number of nonleaf nodes in the member's path to the root. Further, the GCKS needs to send $\log_2 n + 1$ blinded keys to the other members. Thus, the GCKS needs to encrypt and send $2\log_2 n + 1$ blinded keys when a member joins the group.

There is also cost associated with one-way function computations at the GCKS during each join. Recall that the new member and its sibling have new leaf node keys. Further, $\log_2 n - 1$ internal node keys in the path from the new member to the root also change. Thus, in all, the GCKS needs to compute $\log_2 n + 1$ new blinded keys, when a member joins. This amounts to $\log_2 n + 1$ one-way function computations.

6.4.3 Leave rekeying in OFT

When a member leaves the group, the GCKS rekeys the group to maintain forward access control. Unlike in LKH, the GCKS does not change all the keys that the departing member knows. Note that it does not need to change all the blinded keys supplied to the departing member. That is because the GCKS does not use blinded keys to send any encrypted keys. Further, the departing member cannot obtain future group keys or other keys, using the keys it has.

First, the GCKS contracts the tree. If the the departing member's sibling is a leaf node, it gets associated with its parent node. Otherwise, the sibling node assumes its parent's position in the OFT. The GCKS then triggers

rekeying as follows. If the sibling is a leaf node, the GCKS changes its unblinded key and sends the new unblinded key encrypted with the current unblinded key. If the sibling is an internal node, the GCKS needs to pick one of the internal node's children to trigger rekeying in the group. By convention, the GCKS chooses the left-most child, and changes its unblinded key. This in effect results in rekeying of all the keys from the rekeyed node's position (departing member's parent node's position) in the tree to the root.

In Figure 6.12, E leaves the group. The GCKS replaces E's parent with E's sibling, D. It then changes K_d, and sends it encrypted with K_d^{old}. Owing to the new K_d, we have two new blinded keys, that is, K_d' and K_{cd}'. The GCKS sends these keys encrypted with their siblings' unblinded keys, that is, K_c and K_{ab}, respectively.

Leave rekeying complexity in OFT

Consider an OFT that is full and balanced after deletion of the departing member from the tree. The GCKS needs to send as many blinded keys as the path length from the rekeyed node to the root. Thus it sends $\log_2 n$ new blinded keys to the group. In addition, the GCKS needs to send the new unblinded key to the rekeyed node. In all, the GCKS sends $\log_2 n + 1$ keys to the group.

Similar to join rekeying, the GCKS needs to perform some one-way function computations as well. Notice that leave rekeying changes one leaf node and $\log_2 n - 1$ internal node keys. The GCKS needs to compute blinded keys of these new keys, and hence perform $\log_2 n$ one-way function computations.

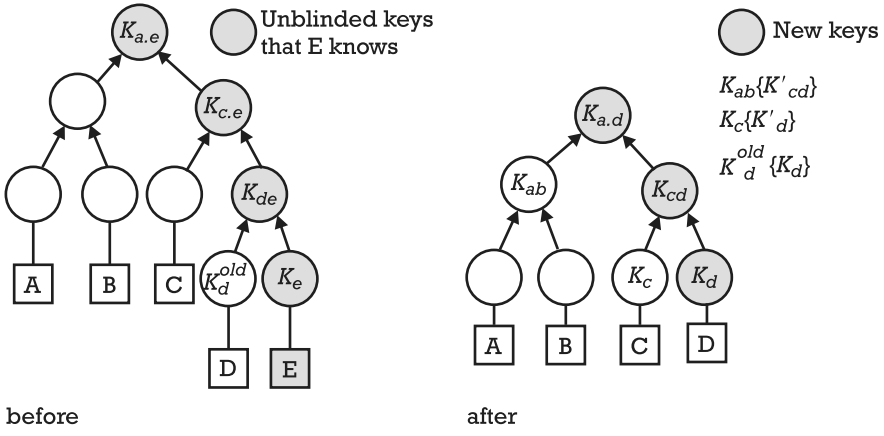


Figure 6.12 Leave rekeying in OFT.

Notice that in terms of number keys sent to the group, OFT sends only half as many keys as LKH to the group. However, the computation cost at the GCKS does not decrease owing to the one-way function computations.

6.5 Batch processing of membership changes in key trees

In this section, we explore batch rekeying in logical key trees. Consider n_j joins and n_l leaves to the group after the last rekeying instance. If $n_j > n_l$, the GCKS needs to create only $n_j - n_l$ slots in the key tree for the additional members [3]. If $n_j \leq n_l$, key tree management is even simpler. The GCKS assigns joining members to departing members' positions in the key tree, and may contract the tree if there are empty positions.

The GCKS may take advantage of the hierarchy in sending group key updates. Depending on new or departed members' positions in the key tree, they may share some internal node keys. All of them certainly share the group key. These shared KEKs need to be rekeyed only once for all the membership changes combined. Analytically, r member departures result in less than $O(rlog_2 n)$ rekeying overhead. If r is sufficiently large, the worst-case rekeying overhead is $O(rlog_2 \frac{n}{r})$ [3]. Yang et al. [3] provide a detailed analysis of the average and worst-case batch rekeying overhead.

6.6 Reliable transport of rekey messages

Stateful group key management schemes such as LKH and OFT require reliable transport of rekey messages, even if the application does not require reliable data transmission. Thus, successful deployment of these GKMAs requires mechanisms for reliable transport. Several solutions have been proposed in the literature, and we briefly explore them in this section.

6.6.1 Repeated retransmission of rekey message

Rekeying messages are typically very small. A GDOI [14] rekey message for immediate rekeying of a group containing tens of thousands of members, using LKH, fits in a single IP packet.[1] Batch rekeying or initialization of a key tree may require a few packets, however. Thus, the simplest solution is to repeatedly resend the rekey message to the group.

1. Assuming a packet of size equal to Ethernet maximum transmission unit (MTU).

When members receive a data packet with an unknown SPI (see Chapter 5), they are expected to contact the GCKS or one of its authorized representatives. However, if a large number of members do not receive the rekey message, there will be an implosion of requests for keys from the members. Thus, we may need slightly more intelligent schemes for reliable transport of rekey messages.

6.6.2 FEC for reliability

LKH has several interesting properties that can be exploited in designing an efficient rekey transport protocol. First, consider that during immediate rekeying using binary key trees, half the members need only one key and only one (or two) member(s) needs, $\log_2 n$ keys. Next, some keys (e.g., internal node keys) are more important than others (e.g., leaf node keys), in that more members need them. For example, all members need the group key. Thus instead of repeatedly sending the entire rekeying message, it may be beneficial to prepare a rekeying message that contains keys based on their importance, that is, their position in the key tree.

We have the following goals in reliable transport of rekey messages. First, we would like to minimize the communication overhead while ensuring that all members receive the new keys that they are entitled to. Second, we would like to reduce the number of rounds in the rekeying process. Finally, we would like to minimize the number of packets processed by each member; especially by members that need fewer new keys.

Yang et al. [3, 5] propose a scheme based on proactive FEC for reliable transport of batch rekeying messages. First, the GCKS divides the rekey message into p packets. It then encodes them using either Reed Solomon [15] or Tornado [16] codes, with a proactivity factor ρ, generating $\lceil(\rho - 1)p\rceil$ additional key packets. Each packet contains a single instance of a key. The proactivity factor, while increasing the number of packets in the current rekeying message, aims to reduce the number of rounds of messages in each rekeying instance. The GCKS then waits for NACKs from members. Each member sends NACKs for keys that it needs, instead of attempting to construct the entire rekey message from the FEC-encoded message. The proactivity factor is computed per round and, after a few rounds, the GCKS may choose to send keys to the remaining members via unicast.

6.6.3 Weighted key assignment for reliable transport

Weighted key assignment (WKA) [17]–based rekeying also sends rekeying messages in several rounds, using NACKs from members for feedback.

Instead of FEC encoding, WKA repeats keys in a rekey message, based on assigned weights for keys. The key weight assignment is based on the number of members that need to receive the key. In each round, the GCKS packs the repeated keys into rekey packets, based on a depth-first or breadth-first search on the key tree. The depth-first assignment works better, since it potentially assigns all the keys required by a member to one rekey packet. Replication in the next round is based on NACKs from the members for packets containing keys it needs, not the entire rekey message.

We end this section with a note that the schemes described here are applicable to stateless rekeying as well. Note that while stateless rekeying schemes are tolerant to rekey packet losses, they do not provide a way to (re)generate lost rekey packets. Thus, if an application requires reliable data transmission, stateless rekey messages also require reliable transport. In that case, the GCKS may use proactive FEC or proactive weighted replication for stateless rekeying as well.

6.7 Stateless key revocation algorithms

LKH, OFT, and the corresponding batch rekeying algorithms encrypt a rekeyed KEK using either an older version of the KEK or another KEK in the key tree. If a member goes off-line and fails to receive KEKs from a rekeying instance, it may not be able to decipher messages from a future rekeying instance. In this section we describe stateless rekeying schemes proposed in the literature [18]. Rekey messages in these schemes are independent of past rekeying messages, since the group key is encrypted with keys sent during the registration phase, that is, over a secure and reliable unicast channel. Thus, unlike LKH-based GKMAs, reliable transport is not required for stateless rekeying algorithms.

Consider a secure group with an overall session membership of size N. Note that the group size we refer to earlier in this section corresponds to the number of members at a given instance in the group's lifetime. Group size changes as members leave or join, whereas session membership size remains constant. More importantly, group size may be much smaller than session size.

The GCKS divides the session membership into several different subsets, and assigns a unique secret key per subset. Subset keys are long lived, and typically do not change during the session. Each member belongs to multiple subsets in the group, and receives or can compute the corresponding secret keys.

When R members leave, the GCKS sends the new group key encrypted so that a member m can decrypt the group key iff $m \in \mathcal{N} \backslash \mathcal{R}$. The GCKS divides the remaining members' set, $\mathcal{N} \backslash \mathcal{R}$, into predefined disjoint subsets, and sends the new group key encrypted with each of those subset keys. Recall that the members receive the subset keys when they join the group.

There are several ways to partition members into subsets, and to predistribute subset keys. We describe two of them, that is, the complete subtree method (STR) and the subset difference method (SDR). Subset keys themselves can be shared keys or, alternatively, the GCKS may generate asymmetric keys, and send the private keys to the corresponding subset members.

6.7.1 STR for membership revocation

This scheme is similar to LKH in structure and initialization, but the definition and the use of the key tree nodes is different. Unlike in LKH, the leaf nodes represent members from the entire session; not just the current membership. Each key tree node represents a subset in the group of all members, containing the descendant leaves of the node. Thus, each member receives the keys of all the nodes in its associated leaf node's path to the root. The similarities with LKH end there, however. The subset keys are never updated or rekeyed. Only the group key changes and is distributed using the subset keys. Figure 6.13 illustrates this key distribution. The subset keys themselves are named after the corresponding subset membership. For example, $K_{a.d}$ is given to all members from A through D. In the figure, member A receives the subset keys K_a, K_{ab}, $K_{a.d}$, and $K_{a.h}$.

Membership revocation in STR works as follows. To process membership changes, the GCKS determines a set $\mathcal{R}$ that contains the hosts whose membership is currently being revoked, and hosts who are not currently members (session membership − group membership). It then computes the directed Steiner tree $ST(\mathcal{R})$, spanning the root and the hosts in $\mathcal{R}$. The GCKS sends the group key encrypted with the subset keys of subtrees that hang off $ST(\mathcal{R})$. Figure 6.14 illustrates membership revocation of C, D, and F following the STR scheme. In the figure, we have three subtrees that hang off the Steiner tree, rooted at K_{ab}, K_e, and K_{gh}. The GCKS sends the new group key encrypted with these subtree keys.

A GCKS using STR sends $O(Rlog_2 \frac{N}{R})$ encrypted keys in removing R members from the group. Each member needs to search for the appropriate subtree key to decrypt the message, and performs a single decryption to

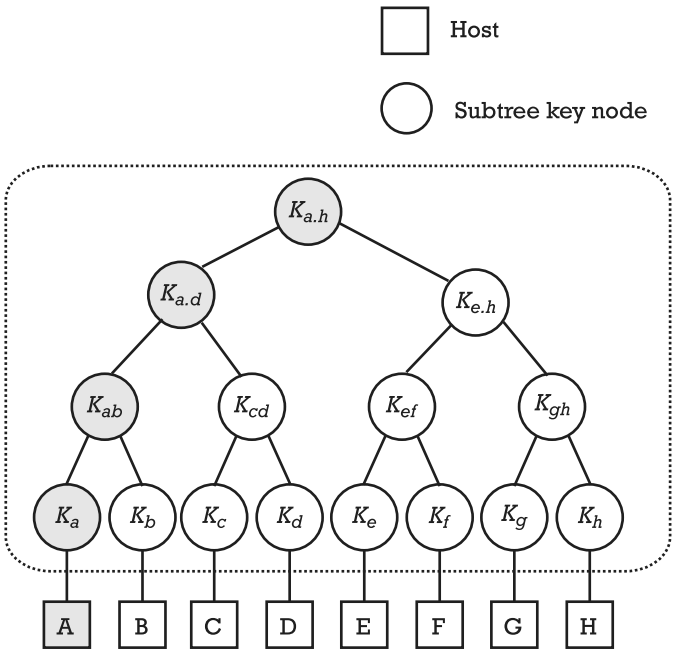


Figure 6.13 Key distribution in STR.

extract the group key. Each member stores $O(log_2 N)$ keys. Note that N and R (used in the context of a session) in this scheme may be much larger than n and r (in the group context) used earlier in this chapter.

6.7.2 SDR for membership revocation

STR may result in too many subsets being formed due to member revocation, and thus result in large communication overhead. SDR aims to lower the overhead by defining more subsets. Furthermore, each member belongs to more subsets, that is, $O(N)$ in this scheme, compared to STR ($O(log_2 N)$) [18].

Each subset in SDR is defined as the difference of two subsets in STR. Thus in SDR, a subset $\mathscr{S}_{i,j} = \mathscr{S}_i - \mathscr{S}_j$, where $\mathscr{S}_i \supset \mathscr{S}_j$. Further, $\mathscr{S}_i$ and $\mathscr{S}_j$ correspond to full binary subtrees of the group's key tree. Figure 6.15 illustrates the concept of subset difference in key trees.

Membership revocation in SDR works as follows [18]. To revoke the membership of hosts in $\mathscr{R}$, the GCKS computes a subset cover of the remaining members. First, it computes the directed Steiner tree $ST(\mathscr{R})$ of

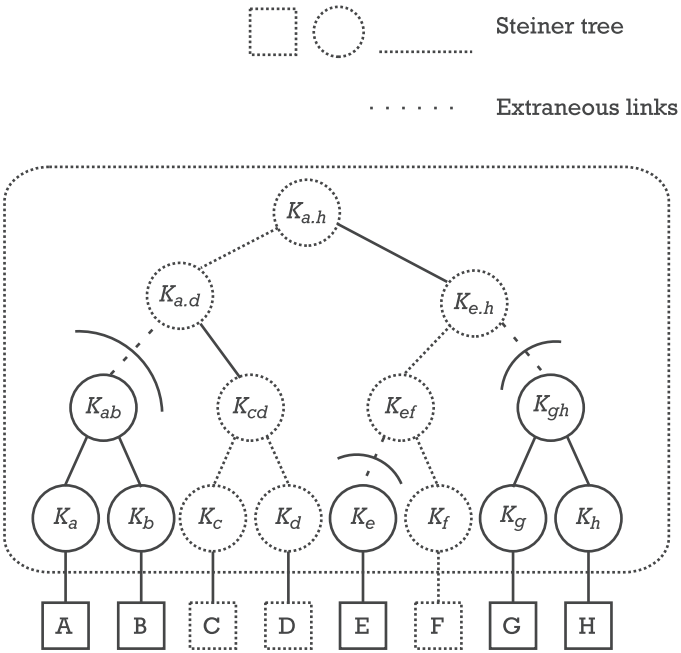


Figure 6.14 STR-based revocation.

the revoked members. It then identifies the maximal chains $[S_{i1}, S_{i2}, \ldots, S_{ik}]$ such that:

- $S_{i1}, S_{i2}, \ldots, S_{ik-1}$ have exactly one child.
- S_{ik} is a leaf node or has two children.
- The parent of S_{i1} is either the root or has two children.

For each such chain with $k \geq 2$, the GCKS sends the group key encrypted with the subset key $K_{i1,ik}$.

Figure 6.16 illustrates membership revocation of C, D, and F in SDR. Following the key distribution algorithm described earlier, we identify two maximal chains in the figure, namely, $[S_2, S_5]$ and $[S_3, S_6, S_{13}]$. Thus the GCKS sends the group key encrypted with $K_{2,5}$ and $K_{3,13}$. By definition of SDR key assignment, only the remaining members that is, A, B, E, G, and H, can decrypt the group key.

Revocation of R members following SDR requires transmission of at most $2R - 1$ encrypted keys. Each member belongs to $O(N)$ subsets, and hence needs to search that space for the subset key required to decrypt the group

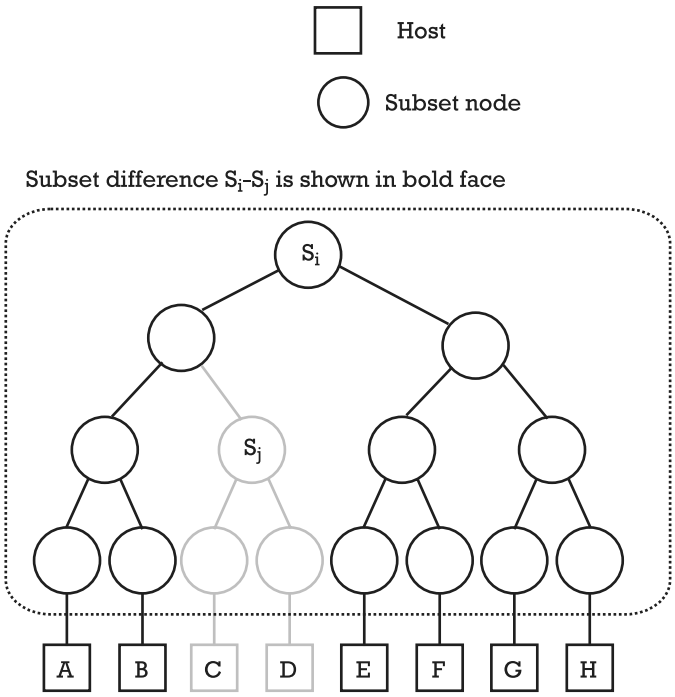


Figure 6.15 Subset difference illustrated.

key. This can be optimized to $O(log^2 N)$. Naor et al. [18] provide a detailed description and analysis of STR and SDR. Note that although SDR and STR appear to use batch processing, they can maintain strict forward and backward access control in immediate rekeying. However, immediate rekeying using SDR or STR is typically more expensive, compared to LKH and other similar stateful rekeying algorithms [19].

6.8 Summary

Group key management is one of the three building blocks of a multicast security solution. This chapter surveys key management algorithms that facilitate efficient distribution of a group key to very large and dynamic groups. We summarize the schemes proposed in the literature for varying application requirements.

The MARKS key distribution scheme is applicable to groups where member departure times are known a priori. The GCKS distributes seeds that

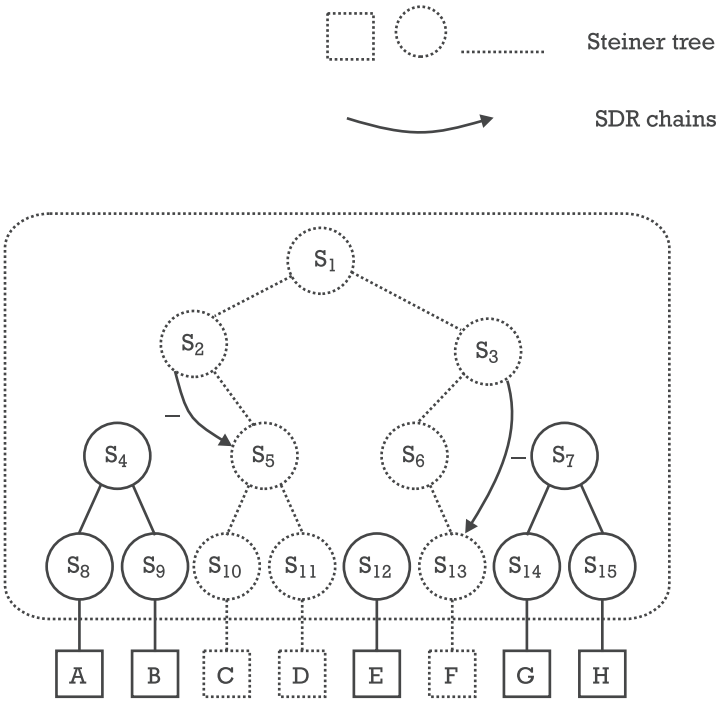


Figure 6.16 Membership revocation in SDR.

members use to compute group keys. Seed distribution is done during member registration using a reliable one-to-one secure channel. No rekeying is necessary or possible in this scheme. Thus there is no need for a reliable multicast channel, but there is also no scope of recovery when seeds or keys are compromised. There is no limitation on members joining, departing, and rejoining the group, as long as the departure information is known at the time of join.

Some key management algorithms use a hierarchy of KEKs to achieve scalability. Updates to KEKs are protected by an older version of the KEK or other KEKs in the key tree. LKH, LKH+, OFT, and OFC are in this category. In LKH, the GCKS is responsible for key generation and distribution; thus it has the largest communication overhead in its class. LKH+ and OFC introduce key computation techniques to reduce communication overhead due to rekeying. Instead of sending all the new keys, the GCKS sends a message or some secret information to the members, and they are responsible for computing the new keys. OFT introduces a member

contributory key determination scheme. Members' keys are used to compute the group key. Thus, OFT reduces communication overhead compared to LKH, but introduces additional computation overhead.

Hierarchical KEK-based schemes are dependent on reliable transmission of rekey messages. If a member fails to receive messages from a rekeying instance, it may not be able to decipher future rekey messages. On the positive side, these schemes enforce strict forward and backward access control. Further, they require no a priori knowledge of membership dynamics.

STR and SDR provide stateless rekeying; thus obviating the need for reliable transport of rekey messages. The GCKS partitions current group membership into several disjoint subsets; each of which is associated with a key. It then sends the group key encrypted with each subset key. Each member belongs to exactly one of those subsets during a rekeying instance, and can decrypt the group key. A major disadvantage of these schemes is that they operate on session membership, which can be quite large compared to group membership. Specifically, STR and SDR may be too expensive if immediate rekeying is the norm in rekeying a group.

In highly dynamic groups, it is possible to sacrifice strict forward and backward access control to reduce rekeying overhead, by batch processing of membership changes. For commercial applications, batch rekeying may be appropriate, with immediate rekeying only to remove compromised or misbehaving members. All the schemes described in this chapter support both batch rekeying and immediate rekeying.

References

[1] DeCleene, B., et al., ''Secure Group Communications for Wireless Networks,'' in *Proc. of the IEEE MILCOM*, Vienna, VA, October 2001, pp. 113–117.

[2] Setia, S., et al., ''Kronos: A Scalable Rekeying Approach for Secure Multicast,'' in *Proc. of IEEE Symposium on Security and Privacy*, Oakland, CA, May 2000.

[3] Yang, Y. R., et al., ''Reliable Group Rekeying: Design and Performance Analysis,'' in *Proc. of ACM SIGCOMM*, San Diego, CA, August 2001.

[4] Chang, I., et al., ''Key Management for Secure Internet Multicast Using Boolean Function Minimization Techniques,'' in *Proc. of IEEE INFOCOM*, New York, March 1999.

[5] Zhang, X. B., et al., ''Protocol Design for Scalable and Reliable Group Rekeying,'' in *Proc. of SPIE Conference on Scalability and Traffic Control in IP Networks*, Denver, CO, August 2001.

[6] Valdvogel, M., et al., ''The VersaKey Framework: Versatile Group Key Management,'' *IEEE JSAC Special Issue on Service Enabling Platforms For Networked Multimedia Systems*, Vol. 17, No. 9, September 1999.

[7] Briscoe, B., ''MARKS: Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences,'' in *Proc. of First International Workshop on Networked Group Communication (NGC)*, Pisa, Italy, November 1999.

[8] Wong, C. K., M. Gouda, and S. S. Lam, ''Secure Group Communications Using Key Graphs,'' *IEEE/ACM Trans. on Networking*, Vol. 8, No. 1, February 2000, pp. 16–30.

[9] Wallner, D., E. Harder, and R. Agee, ''Key Management for Multicast: Issues and Architectures,'' RFC 2627(informational), IETF, June 1999.

[10] Mittra, S., ''Iolus: A Framework for Scalable Secure Multicasting,'' in *Proc. of ACM SIGCOMM*, Cannes, France, September 1997, pp. 217–288.

[11] Balenson, D., D. McGrew, and A. Sherman, ''Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization,'' draft-irtf-smug-groupkeymgmt-oft-00.txt, IRTF, August 2000, work in progress.

[12] Canetti, R., et al., ''Multicast Security: A Taxonomy and Efficient Constructions,'' in *Proc. of IEEE INFOCOM*, New York, March 1999.

[13] Blum, M., and S. Micali, ''How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits,'' *SIAM Journal of Computing*, Vol. 13, No. 4, November 1984.

[14] Baugher, M., et al., ''Group Domain of Interpretation for ISAKMP,'' draft-ietf-msec-gdoi-04.txt, IETF, March 2002, work in progress.

[15] Wicker, Stephen B., and Vijay K. Bhargava (eds.), *Reed-Solomon Codes and Their Applications*, John Wiley & Sons, September 1999.

[16] Luby, M. G., et al., ''Efficient Erasure Correcting Codes,'' *IEEE Trans. on Information Theory*, Vol. 47, No. 2, February 2001, pp. 569–584.

[17] Setia, S., S. Zhu, and S. Jajodia, ''A Comparative Performance Analysis of Reliable Group Rekey Transport Protocols for Secure Multicast,'' in *Proc. of Performance 2002*, Rome, Italy, September 2002.

[18] Naor, D., M. Naor, and J. Lotspiech, ''Revocation and Tracing Schemes for Stateless Receivers,'' in *Advances in Cryptology—CRYPTO*, Santa Barbara, CA: Springer-Verlag Inc., LNCS 2139, August 2001.

[19] Chen, W., and L. R. Dondeti, ''Performance Comparison of Stateful and Stateless Group Rekeying Algorithms,'' in *Proc. of Fourth International Workshop on Networked Group Communication (NGC)*, Boston, MA, October 2002.

CHAPTER

7

Contents

7.1 Group security policy framework

7.2 Classification of group security policy

7.3 Group security policy specification

7.4 Policy negotiation and reconciliation

7.5 Group security policy enforcement

7.6 Summary

Group security policy

The *Merriam-Webster Dictionary* [1] defines policy as ''a definite course or method of actions selected among alternatives and in light of given conditions to guide and determine present and future conditions.'' This definition serves the topic of group security policy very well, for there is a wide range of application requirements and an equally vast number of group security mechanisms.

Applications require multicast security for privacy, data origin authentication, controlled access to group membership, and other similar requirements. The communication could be over the public Internet or intranets, with varying security threats. On the other hand, from the previous chapters, we know that there are several alternative mechanisms for enforcing privacy, key distribution, data origin authentication, etc. In this chapter, we discuss how application security requirements, business models, and *service level agreements* (SLAs) can be translated into group security policy, and distributed and enforced using the group management entities introduced in earlier chapters.

Application requirements are generally specified at a very highlevel to begin with. For example, an application may specify the use of ''strong symmetric key encryption for secure one-way transmission to an average of 10,000 members to enforce periodic billing.'' Such service agreements (between say, content owners and service providers) must be translated into protocols and algorithms, and enforced using a secure multicast communications architecture.

There are several components to the group security policy problem area (see Section 2.5). First, high-level group security requirements, and requirements stipulated by the operating environment, must be translated into policy specifications. Several policy specification languages have been proposed in the literature for describing unicast as well as group security requirements.

In unicast applications, policy may be negotiated, in that the participating entities may come to an agreement on the mechanisms used and on operational behavior. In groups, negotiation or reconciliation is difficult and becomes practically impossible for very large groups. Furthermore, in several unicast and especially multicast applications following the provider-subscriber(s) model, the provider might need the final say in the level of data protection (e.g., use AES instead of 3DES for encryption), the frequency of rekeying, and other similar factors. Thus in groups, the content owner or the provider often dictates the policy, and hosts or end-users for whom the policy is unacceptable may choose to not subscribe to the group. However, providers may change policy (typically for future use), based on feedback from potential customers. For example, if a large number of customer machines have support for 3DES, but not AES, the provider may choose to employ 3DES for encrypting future multicast session data.

The next issue is policy distribution. This is an area more specific to group security policy. The policy distributor (typically the group manager) needs to send information to members about how to decrypt and authenticate group data. This information includes: mechanisms used for data protection; group keys, and instructions on how to use them; expected behavior if a group member does not receive keys; and so forth.

Policy enforcement is the final component of a group security policy system. Enforcement includes allowing only authorized members to receive data, as well as evicting misbehaving members. Policy enforcers must not only have mechanisms to disallow illegal users from receiving group data, but they must also be able to detect unauthorized behavior. In general, the goal of policy enforcement is to ensure that the content owner's requirements are being met, while keeping the group operational.

Some applications might stipulate that the policy data must itself be protected, and divulged only to authorized entities. Furthermore, they may allow the GC some leeway in delegating its policy distribution and enforcement duties. Metapolicy, in the context of group security policy, is a set of rules governing the handling of policy data. Thus metapolicy may specify that a small portion of the group policy may be used for public announcement of the session information, while the rest must be disseminated securely (e.g., along with key distribution). Furthermore,

the GC may be allowed to delegate policy distribution and enforcement duties to either certain members, or to trusted third-party entities.

Before getting into the details of group security policy, we revisit the definition of policy introduced at the beginning of this chapter. Note that there may be changes in the operating environment during the lifetime of a group, and several alternative sets of mechanisms may need to be employed to handle each specific scenario. Thus, group security policy consists of a specific selection of mechanisms to support the application requirements in the given operating environment. The GC must also be able to handle changes in the operating environment or group behavior (e.g., the addition of a new sender). Therefore, the GC must be able to change policy during the operation of the group, and distribute and enforce the new policy without having to reinitialize the group.

The rest of this chapter is organized as follows. First, Section 7.1 describes the various group entities from a policy perspective. Next, Section 7.2 describes a classification of a group security policy. The following section, 7.3, describes group security policy specification languages. Policy negotiation and reconciliation are the topics of Section 7.4. Section 7.5 contains a survey of several group policy systems in the literature. Section 7.6 provides some concluding remarks.

7.1 Group security policy framework

There are typically several different entities with different stakes in the secure operation of a group. For example, cable TV providers may want to distribute a PPV event to customers who paid for the event. Customers having paid for the event expect that the advertised quality of service is maintained. Similarly, in group conferencing, the participants may want to ensure that their communications are not modified en route, and are sent to authorized members whose identities are known to all participants. These and other similar requirements of content owners and group members are translated into group security policy, which is enforced using the various mechanisms and entities (e.g., GCs) described so far in the previous chapters. Policy also needs to take operating environmental conditions (e.g., increased security threats in wireless communications) and infrastructure limitations into consideration.

Group security policy is the third problem area (see Section 2.5) of secure group communication. It ties together the first two problem areas, namely, data origin authentication and group key distribution, with application requirements and the changing conditions of operational

environments. In this section, we introduce several important entities [2, 3] that specify, distribute, and enforce group security policies.

Content owners, owing to the fact that they own the data and are interested in controlling distribution, are at the top of the chain of control to be introduced in this section. They might specify high-level group security policy. Note that content owners could be people (e.g., author Stephen King distributing his new novel over the Internet), corporations (e.g., Microsoft distributing software upgrades to paying customers), or group members themselves (e.g., in a group conference).

The group owner/creator (GOC) is a logical entity that creates policy. If policy is negotiated, the GOC contains the negotiated policy and fills any gaps. It also interprets content owners' or other application requirements to create the list of authorized members (e.g., access control lists). It is responsible for setting up the multicast session, making group announcements, distributing policy to group controller(s), and updating policy during the life of the group, if necessary. The GOC may choose to allow delegation of group control functionality. Finally, the GOC holds the authority to delete the group as well.

The GCKS (see Chapter 5) is the logical entity that distributes and enforces policy. Its two functionalities, that is, membership management and key distribution, may be separated if necessary. Thus, a GCKS may be referred to as a GC as well, in reference to the policy distribution and enforcement functionalities. The GCKS or its delegates are responsible for enforcing access control (verifying membership authorization), policy distribution, and enforcement.

A GCKS authorizes one or more designated hosts to send data to the secure group.[1] Members are expected to disregard data from any illegal senders. Sender(s) also have an important role in enforcing policy. In particular, data protection policy enforcement (other than key distribution itself) is mainly in the sender's control. For example, senders must use the latest keys in protecting data. It is necessary for the GCKS to make sure that the sender has received the latest policy and the keys.

Hosts become members of a secure group when a GCKS verifies their authorization to receive data, and sends the latest policy and group keys. Members have a somewhat limited role in enforcing policy. First, they are expected to accept policy and keys only from an authorized GCKS or its

1. Multicast routing protocols such as PIM allow any host on the Internet to send data to a multicast group. PIM-SSM supporting a one-to-many multicast model only is an exception. The next chapter on multicast routing protocol protection discusses how to enforce authorized sending at the network layer.

authorized delegates (e.g., a subgroup controller or SGC in GSAKMP [4]). Next, members are expected to accept data from authorized senders only. The group policy specifies the list of authorized senders and GCs.

In all types of applications, the members also have an albeit minor interest in the group policy being enforced. Even if end-users are not content owners, they are interested in ensuring that the level of service is maintained, and the protections offered are realized. A particular threat is in a member revealing secret keys or broadcasting decrypted data. Detection of such behavior and policy enforcement is one way to counter such threats. In the simplest case, members not following policy (e.g., to not reveal/share keys to unauthorized entities) are evicted.

In addition to the above group management–related entities, there is another component of importance in a group security policy framework [2]. The group policy repository is a database that stores the group's policy. It is a permanent store of the group's policy, whereas the GCKS and the GOC may store policy in volatile memory. Thus after a reboot they can access the current group policy from the repository. The GOC is authorized to read, create, modify, and delete a group's policy, whereas the GCKS and GPS have a read-only access to the policy repository.

Figure 7.1 shows various entities of a secure group from the policy perspective. Notice the two different types of arrows used to indicate how policy is distributed. The GOC has read and write access to the policy repository, whereas the GCKS can only read from the repository. The GOC is also responsible for supplying the appropriate amount of policy to the announcement mechanism. The announcement may be sent via unicast or multicast to group members. The members can obtain a copy of the announcement, should they fail to receive it when it was sent via multicast. Members have read-only access to the announcements, however.

The GCKS receives policy and any updates to policy from the GOC. It may also download updates from the policy repository. Note that members do not need to receive the entire policy. Several components of the policy may be enforced, without the need to inform the members about them. The GOC may determine what needs to be kept private from members, and the GCKS may only distribute the minimum required for correct enforcement of policy, as well as for better interoperability of secure group communication protocols.

The GCKS distributes policy to authorized members via unicast (during registration) or multicast (during rekeying). Thus, the framework allows for dynamic policy updates. The GOC may change policy during the session and send the new policy to the GCKS. The GCKS may rekey the group to distribute and enforce the new policies. The change in policy, might

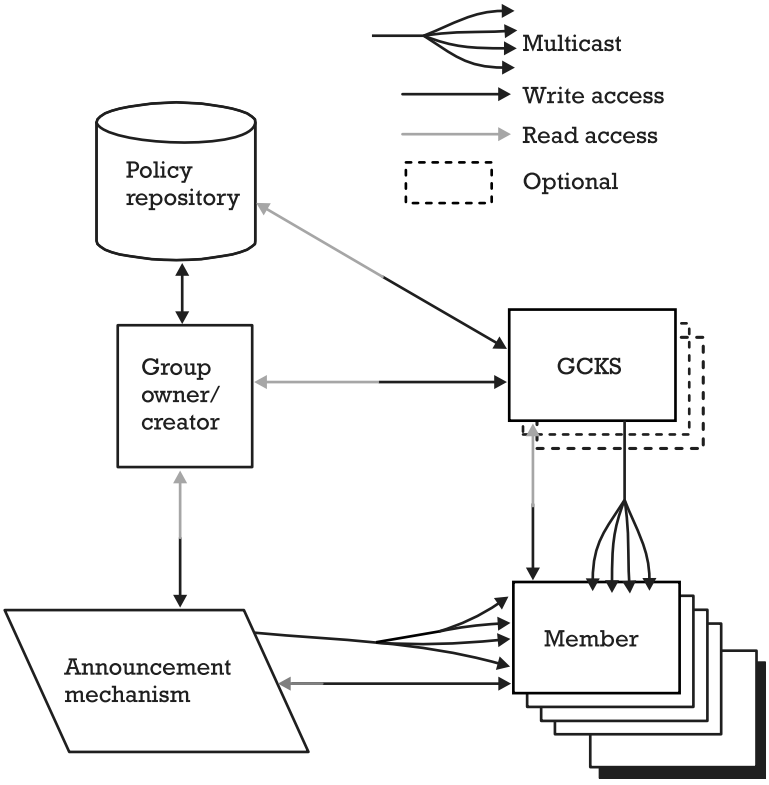


Figure 7.1 Group security policy distribution framework.

result, for example, in eviction of noncompliant members, a change in the cryptographic mechanisms used, a change in rekeying frequency, and so forth.

7.2 Classification of group security policy

Different applications have different requirements for securing group communications. But designing a separate multicast security solution for each class of applications is inefficient. It is better to design a group security system that can support a wide variety of mechanisms, and employ those mechanisms according to the specified policy of a given application. Group security policy can be divided into several categories. In the following we discuss these categories and the options within each category. Note that neither the category list nor the options within it are meant to be exhaustive. They are only intended to provide an insight into the scope of the problem. Other similar classifications are available in the literature [5].

7.2.1 Announcement policy

Secure group announcement policy is a surprisingly tricky issue. The announcement must be informative, to attract potential members to join the group. On the other hand, revealing too much information about a group's policy (e.g., encryption algorithms used or rekeying frequency), might compromise the security of the group.

Prospective members may want to know such information as minimal system and bandwidth requirements, cryptographic algorithm support, and trust model, before subscribing to a group. But such information may also be beneficial to adversaries.

Application requirements help us strike a balance between revealing too much or too little policy during group announcements. For example, for a military application, the announcement might consist of a code word on a predistributed Web page, while most of the policy is revealed only to the authorized members by either a well-known or a prespecified GC. For commercial applications vying to attract customer interest, it might be beneficial to announce the minimum system and bandwidth requirements, data protection policy, cryptographic algorithm support required, and so forth. Thus the members who paid to obtain authorization to join the group will not be denied access to the group at the time of registration.

As mentioned in the previous section, the GOC is responsible for making the announcements. The metapolicy may specify how to appropriately distribute/announce the nature of the group data to be sent, and the group policy (determining which part is public and which is private). The GOC is ultimately responsible for revealing only an appropriate portion of the policy in group announcements.

Closed and open secure groups. Two types of secure groups have been identified in the literature on group security policy [2]. First, closed secure groups enforce privacy not only on data transmission, but also on announcements and the group policy itself. Thus, group announcements are sent to preselected hosts or end-users (e.g., to the executive team within a company or individuals who have a certain security clearance in the military context). In other words, group security policy is also private, and revealed to authorized users only. Open secure groups, on the other hand, make a public announcement about the group, in part indicating how one can get authorization to become a member. The group is secure in that parts of the policy are only revealed after authorization, and, more importantly, data can be decrypted only by authorized members. Commercial applications, such as PPV TV, where members receive information about a program and the price of admission, are examples of open secure groups.

7.2.2 Membership policy

A group may allow members to remain anonymous. Members may buy tokens for the service provided. Any user presenting a token may be authorized to receive group data, without having to authenticate himself or herself. However, more often than not, group owners require that members authenticate themselves.

Membership policy specifies the qualifications to be a member and, sometimes, the duration of membership. For example, membership may be contingent on payments for PPV blocks of time. Alternatively, members over a certain security level, or all vice presidents of a company, may be invited to a secure group meeting.

Finally, membership policy may also specify whether a particular member(s) must be present for the group to operate. For example, in a corporate meeting, if the CEO must leave prematurely, the group may need to be terminated.

7.2.3 Access control or authorization policy

To enforce access control, a GOC may prepare a list of members, known as an ACL, specifying those allowed to participate in the group. The GOC distributes ACLs to the GC, which may subsequently distribute them to the SGCs. Authorization policy may generally be part of a secure group announcement. ACLs may be inclusive or exclusive. The group policy in this regard need only be revealed to the GCs. In other words, members need not be notified about the nature of ACLs.

Distribution of ACLs to the GC and SGCs does not scale well to large and dynamic groups. Authorization certificates and membership tokens work better for group access control, because they are signed by the GOC and verifiable by the GC or the SGCs, without any interaction with the GOC. Thus they scale well to large groups, and are especially suitable for applications that allow members to obtain group membership and join while the group is operational. Members do need to know in advance about certificate usage so they can obtain certificates from GOC-authorized entities. The GOC is also responsible for specifying certificate revocation policy, to allow ejection of misbehaving users from the group.

7.2.4 Data protection policy

A group owner may want to protect the privacy of the group's communications. For IP multicast security, data may need to be encrypted, even if

privacy is not a requirement. More specifically, a GC may use encryption to enforce controlled access to group data. Encryption policy generally consists of encryption algorithms (e.g., 3DES and AES), key length, lifetime and so forth. Replay protection is another typical requirement in secure data communication. Timestamps and sequence numbers are two popular techniques used to guard against replay attacks. Timestamps require clock synchronization, which is difficult to maintain in large groups. Replay protection policy specification may thus be dependent on system or infrastructure support for clock synchronization.

Data authentication policy. Data origin authentication is another component of data protection. A group owner may require group authentication, that is, that the data originated within the group. For other groups, source authentication may be required. Data authentication policy would then include the source authentication algorithm policy. A source authentication algorithm policy (see Chapter 3) may include data streaming parameters, buffering requirements, synchronization requirements, and so forth. Nonrepudiation could be another requirement of data authentication.

During the transition from the old to the new group keys, senders may stop sending data, start using the new key at a specified time, or use both sets of keys for encrypting data [6]. For military applications, it may be necessary to stop data transmission during group rekeying [7], whereas for commercial applications, uninterrupted service may be the driving factor.

7.2.5 Group management delegation policy

In large and distributed groups, group management tasks may be delegated for efficiency. A group's policy could be to delegate group management to trusted third parties or to members themselves [4].

Access control enforcement policy specifies how this functionality may be delegated. For example, a group may require that the GC alone can allow authorized entities to join the group. Small interactive groups may specify that any member may allow a user to join, or that all members must validate each prospective member.

It is also possible to separate group registration from rekeying, and offer both as distributed services. Thus there may be several registration servers that authorize members, download policy and keys, and specify the authorized rekey server(s) that send updates to policy and keys.

7.2.6 Key distribution policy

Key distribution is a major part of secure group communication and thus comprises the majority of the policy as well. We divide the policy into two parts: group registration and rekey policy. The registration policy may specify a single registration server or multiple servers.

Similar to data, key transmissions must be protected. Therefore, key distribution policy contains cryptographic mechanisms (e.g., encryption and message integrity protection algorithms, and replay protection policy) for secure transmission of keys. GDOI SA KEK payload [8] (see Section 5.4.3) , for example, contains the policy enforced in protecting KEKs, including the group key.

Rekeying policy. Rekeying policy may specify (a) the frequency of rekeying, (b) whether immediate or batch rekeying must be employed, (c) the nature of the rekey message transport mechanism (multicast or unicast, reliable or not, reliable transport protocol used, etc.), (d) actions to be taken if a user is out of synchronization and cannot decipher rekey messages, (e) whether forward or backward confidentiality must be enforced, and (f) the group key management algorithm used, and so on. There may also be some group key management algorithm-specific policy. Recall that some of the group key management algorithms are complex, and the members need to know some algorithm parameters to decipher rekey messages correctly. For example, LKH (see Chapter 6) policy may contain algorithm version, key tree degree, and a per-key identity, at a minimum.

Members need not know the entire rekeying policy for correct operation of the secure group. The group owner sends the rekeying policy to the GCKS, and the GCKS (or the group owner itself) determines the minimal subset of the policy that must be sent to the members. In the above list, for example, members need to be aware of forward and backward confidentiality policy or frequency of rekeying. However, they do need to know the cryptographic algorithms used to decrypt and authenticate the messages correctly, and the group key management algorithm policy, to use the keys appropriately.

7.2.7 Compromise recovery policy

The group owner may specify that the GCKS should be able to detect and evict compromised members from the group. Efficient rekeying to remove members may also be a requirement.

Compromise detection could become complicated if many members are compromised and take over the group. Group policy may specify a requirement for protection against such attacks. For example, the policy might stipulate that the group communications must be secure even after *r* members are compromised.

7.3 Group security policy specification

Real-world group security requirements are typically high-level specifications, and often vague. Furthermore, some of the requirements may be conflicting, particularly when there are multiple stakeholders in a secure group. But we need a consistent group security policy specification for the correct and deterministic operation of a group.

Thus on the one hand we have content owners being people, corporations or businesses, specifying policy in natural languages (e.g., English). On the other hand, we need machine-friendly policy specifications for automated negotiation, distribution, and enforcement. Several policy specification languages have been proposed in the literature, for specifying and enforcing quality of service, network access control, and network management policy [9, 10].

Many policy specification languages [9] define roles [4, 10] of various entities, rules of engagement, and actions to be taken when an event occurs. Thus, a policy specification may contain several **if** *event* **then** *action* statements. Different entities in the system may have the role of carrying out the actions; at least they get affected by those actions. The following examples, Ismene [11], CCNT [12], and the *group security policy token* (GSPT), provide an insight into group security policy specification languages.

7.3.1 Ismene policy specification

Ismene considers *provisioning*, and *authorization and access control* as the two central components of group security policy [11]. Next, it recognizes that there is a group policy, specified by the group owner, to be reconciled with local policies of individual members.

Ismene policies also follow the **if** ... **then** ... format mentioned earlier. Each Ismene policy statement is a session requirement specification (provisioning) or a policy enforcement action (authorization and access control) statement. A tag identifies each policy statement. Examples of tags include provision, join, confidentiality, integrity, and so forth. The requirements and actions themselves are specified as *conditionals* and *consequences* in the format:

```
tag: conditional(s) : consequence(s);
```

Ismene conditionals stem from the operational environment, the system configuration, and user credentials. A consequence in a provisioning clause may be one of the following. It may define how a requirement may be enforced through *configuration*. Next, a *pick* consequence allows for specification of alternative mechanisms that could be used for implementing a security requirement. Finally, *tag* consequences give structure to policy. Evaluation of policy enforcement action statements results either in an *accept* or a *reconfig* consequence. In summary, Ismene policy language is structured, supports provisioning and group events, and allows for dynamic changes in policy. The following Ismene policy specifies access control using ACLs, message integrity using HMAC-SHA-1, source authentication using TESLA, encryption using AES, and rekeying using LKH.

```
group :=<Ismene example policy>;
provision : :: access_control, group_management;
ACL :=<{Alice}, {Bob}, {Cindy}>;

access_control: is_in_list(ACL) :: allow_member;

group_management : :: key_dist, rekeying, data_protection;
key_dist : :: config(GDOI());
rekeying : :: config(GDOI(GKMA = LKH, encr_algm = AES));
data_protection : :: config(MESP(data_encr, data_integrity));
data_encr : :: config(encr_algm = AES);
data_integrity : :: config(TESLA(MAC_algm = HMAC-SHA-1));
```

7.3.2 CCNT

The dynamic cryptographic context management (DCCM) system [13] specifies a CCNT [12] to facilitate policy/context negotiation. DCCM defines policy as the high-level system and security requirements, and context as the specific set of mechanisms negotiated for a project.

CCNT is described using Backus-Naur Form (BNF) grammar, and consists of specific values for various policy categories or axes. Examples of categories and corresponding values include, level of security (e.g., high, medium, or low), data authentication (e.g., RSA or DSA), entity authentication (e.g., passwords or RADIUS), encryption algorithm (e.g., AES or 3DES), group key management algorithm (e.g., LKH or OFT), eviction restrictions (e.g., cannot evict member X), and so forth. The policy specification itself is the tuple declaring which mechanisms apply to the given project. Notice that CCNT does not allow dynamic changes to policy in response to group events. Using the following CCNT, Policy1(0,0,1,1,~) implies ''access control using

ACLs, key management using GDOI, encryption using 3DES, and integrity protection using MD5, with no source authentication employed.''

Axes:
access_control(ACLs, authorization_certs)
key_management_protocol(GDOI, GSAKMP)
encryption_algorithm(AES, 3DES)
MAC_algorithm(SHA-1, MD5)
source_authentication_scheme(TESLA, Augmented_chaining, tree_block_hashing)

7.3.3 GSPT

The IETF MSEC working group has been working on standardizing a common framework for group security policy. GSAKMP and eventually GDOI may use the resulting structure to specify and distribute policy. The GSPT data structure is a result of that process [14].

A GSPT consists of the security parameters that define the operation of a group. The GSPT specifies the authorization policy, the access control policy, and the mechanisms used for content protection. The token has an identity and is signed by the GO. In the rest of this section, we describe the components within a GSPT in detail.

Components of a GSPT

Figure 7.2 shows the components of a GSPT in detail. The first field is the *token ID*, to uniquely identify a group. Next, we have the *authorization* field, which defines the roles of various entities that manage the group. The *access control* field specifies who can be senders or receivers of the group. Content protection policy is specified by the *mechanisms* field in the GSPT. The final field contains the GO's signature, authenticating the policy token.

Token identification. Members and GCs must be able to uniquely identify a secure group. Note that correct identification of a group is necessary for secure operation of a group. Thus each group is assigned a unique *group identity*. To protect against replay attacks, the GSPT contains a *timestamp*. Recall that the controller or member receiving that token must verify the validity of the token before using it. A receiver must verify that the timestamp in the received GSPT is later than the latest timestamp in the GSPT that it has successfully authenticated.

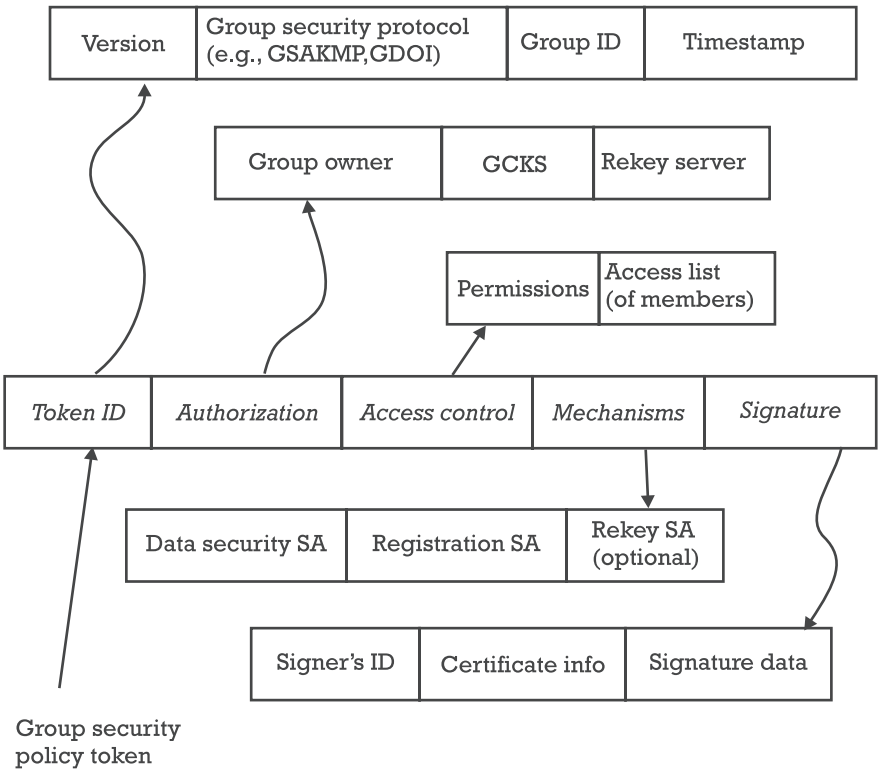


Figure 7.2 GSPT and its components.

Authorization field. Recall that the GOC assigns the responsibility to manage the group to a GC. For efficiency and fault tolerance, the GC may be implemented as a distributed service. Furthermore, a GOC may designate different clusters of servers to provide group registration and rekey functionalities. The authorization field contains the identities of the GCs and their assigned roles. Members are expected to accept group keys and updates to policy from entities specified in the authorization field of the GSPT. The GC has complete control over the operation of the group, and it may further delegate group management tasks to other members. In other words, the entities specified in the authorization field are distributors and enforcers of group policy.

Access control field. The third field in GSPT specifies who can join the group. This field has two subfields, namely, the *permissions* and the *access list*. The permissions could be as simple as member and nonmember, or as

all-encompassing as the hierarchical clearance levels used in military applications. Permissions could specify who can be a sender and who can be an SGC, and so on. An access list might contain the list of user identities that have the specified permissions.

Mechanisms. The IETF group key management architecture (GKMArch) [15] specifies three SAs: the data security SA, the registration SA, and the rekey SA. Each of the SAs is associated with keys and several parameters describing its purpose, lifetime, algorithm used, and so forth. The data security SA specification may contain mechanisms for privacy, integrity, and replay protection of group communications. The registration process consists of access control enforcement and secure download of initial keys, both for protection of future keys, and of data. Thus the registration SA specification may contain mechanisms for secure channel establishment and group membership authorization. The rekey SA policy specifies mechanisms for compromise recovery, group key management, key encryption, key transport, and so forth.

Validity of GSPT. A policy token must be signed to be considered valid by the recipient. Thus the GCKS needs the group owner's signature to consider the GSPT authentic, and a member needs the group owner's or the GCKS' signature to consider the token valid. The policy token's recipient must also know that the token was not modified en route, and is fresh (i.e., distinguishable from a potential replay by an adversary).

7.3.4 Discussion on policy specification languages

We conclude this section with a brief comparison of the three policy specification languages summarized earlier. All three policy languages translate application-level group security requirements into machine-readable form for policy negotiation/distribution and enforcement. CCNT and Ismene specify policy with negotiation or reconciliation in mind, whereas in GSPT, the specification is final. Ismene and GSPT allow dynamic policy changes to adapt to changing security requirements or operational environmental considerations. Finally, Ismene allows policy specification with conditionals and consequences, and is thus more powerful than the other two. However, for most applications, GSPT policy specification may be sufficient, and thus the IETF MSEC working group is in the process of standardizing GSPT for use with GDOI and GSAKMP key distribution protocols.

7.4 Policy negotiation and reconciliation

In a provider-subscriber model, it is intuitively appropriate that the providers mandate who can be a subscriber, the level of protection required, and actions to be taken when a group's security is compromised. In other words, providers may dictate policy. However, considering that the subscribers may be paying for the services (e.g., in content streaming), the members may demand a particular quality of service and maybe even some security guarantees, such as protection against DoS attacks. In interactive applications such as group conferencing, several entities have an approximately equal interest in protecting group communications. Therefore, negotiation may be necessary to determine a mutually satisfactory group security policy. In summary, synthesizing different group entities' requirements at some level is commonplace in policy specification.

The flip side is that, in large groups, satisfying potentially conflicting requirements is a challenge. In the provider-subscriber model, the provider may unilaterally make the final decision in resolving any conflicts. In multisender groups, negotiation may be necessary to arrive at a consensus. Alternatively, there may be a groupwide policy, and several sender-specific policies to be applied to data originating at various senders.

For a familiar example of policy negotiation, recall SA negotiation in the IKE [16] protocol. The initiator sends an SA payload containing severals proposals: each containing a set of transforms. Each transform contains values for attributes such as the encryption algorithm, message authentication algorithm, key length and lifetime, and so forth. The responder may choose a transform that it supports and considers appropriate for protecting the session being negotiated. If the responder does not find any transform that it supports and deems appropriate, it can refuse the request to set up the session.

We now summarize policy reconciliation in Ismene [11, 17] and policy negotiation in the DCCM protocol [18].

7.4.1 Ismene policy reconciliation

Ismene policy reconciliation is via intersection of group and local policies. However, reconciliation is only possible when group policy (specified by the GOC) specifies some choices that can be narrowed down using local policy. Thus if group policy specifies a single mechanism for encryption, there is no need for visiting the local policy specification.

Reconciliation with several local policies may be done in several different ways, including resolving based on the majority's choice, and

resolving with each local policy in the order of the importance of the member in the group. Majority choices may sound attractive at first, but notice that in selecting the majority choice of each component (e.g., encryption algorithm, message integrity, or user authentication policy), we might eliminate all members. For example if A, B and C have the local policies, $<3DES, MD5, Password>$, $<AES, MD5, Cert>$, $<3DES, SHA\text{-}1, Cert>$, reconciliation with majority choices results in the policy $<3DES, MD5, Cert>$, which disallows all three entities from participating in the group.

McDaniel et al., [17] report that, in the worst case, reconciliation of more than two parties is intractable. They suggest the use of heuristics, such as reconciling with each of the local policies, one by one. The other school of thought is to disallow any automated negotiation of group security policy, and have the group owner dictate policy. This is employed in GSPT, described earlier in this chapter, and in GDOI, discussed in Chapter 5.

7.4.2 Policy negotiation in DCCM

DCCM differentiates between high-level policy and *cryptographic context*, which identifies the specific cryptographic mechanisms employed for the level of protection intended in the policy specification. The negotiated cryptographic context applies to all the secure group sessions within a project. Group or project managers negotiate policy: not individual members.

The cryptographic context negotiation protocol (CCNP) for DCCM projects works in three phases: the project initiator's proposal, the negotiators' responses, and the resolution and dissemination by the mediator. The project initiator generates a proposal based on local constraints, project requirements, and organizational requirements. A mediator may be involved in preparing the proposal. The mediator then forwards the proposal to policy negotiators. The negotiators select a subset of the proposed policy, based on local policy mappings (high-level language specification to machine-interpretable rules and algorithms). Negotiators may return multiple choices in the order of preference. The mediator is responsible for collecting the responses and working with the initiator to arrive at an intersection acceptable to all (or most) of the parties. If no single policy (consisting of acceptable parameters in all categories proposed) is acceptable to all negotiators, the mediator may ask the minority negotiators to support the majority's choice. The final step is for the mediator to distribute the project context to the initiator, as well as the members.

We conclude this section with a note on the security threats of policy negotiation. In closed secure groups, policy is private, and thus all policy negotiation messages must be over a secure channel that offers confidentiality and integrity protection. A particular threat to avoid is a man-in-the-middle attack to force negotiating parties to agree on a level of security (policy) lower than what they can support. This attack, known as a *downgrade attack*, is especially affective in open secure groups, where policy negotiation may be in the clear.

7.5 Group security policy enforcement

In this section, we discuss secure group policy distribution and enforcement. The group owner sends policy to the GCKS. The policy specification may include instructions for the GCKS on how much of the policy may be revealed to the members. In the absence of such instructions, the GCKS determines the minimum amount of policy that needs to be sent to members for their participation in the group. Examples of policy that need not be revealed include details of rekeying policy, such as whether immediate or batch rekeying is going to be employed.

The GCKS typically enforces access control, group management delegation, key distribution, and compromise recovery policies. The sender is expected to enforce data protection policy. The members are expected to follow the group security policy, or face eviction from the group.

7.5.1 Policy distribution and enforcement in GDOI

GDOI [8] (see Section 5.4.3) policy has only a relatively limited coverage, compared to some of the other systems described in this chapter. More specifically, GDOI only covers the interactions between the GCKS, a member, and the senders. The communication between the group owner and GCKS is out of scope of the protocol.

The GCKS enforces authorization policy before allowing users to join a secure group. Apart from this functionality, the registration protocol of GDOI securely distributes the group policy and keys. GDOI group policy consists of rekeying and data protection policies. More specifically, the rekey SA specifies the cryptographic mechanisms and other parameters (e.g., key length and lifetime) used to enforce privacy, integrity, authentication, and freshness of key update (rekeying) messages. The key update policy may also contain a group key management algorithm policy. Next, the data protection SA specifies the mechanisms used to protect group communications.

The rekey SA may be used to update both rekeying and data protection policy. It is also possible to keep the policy private from evicted members. The GCKS has the mandate to enforce such metapolicies. The sender is expected to enforce the data protection policy by using the correct (e.g., the latest) keys in encrypting and authenticating group data.

GDOI does not address membership management, however. The GCKS depends on external mechanisms for: determining whether a user's membership has expired, detecting member misbehavior, handling requests for early departures or membership renewal, and so forth. Note, however, that it is possible to initialize and update the GCKS with information such as membership duration, to support some of the above functionality. Absence of member monitoring policy and mechanisms, as well as lack of support for feedback channels from members to the GCKS, are clear handicaps in supporting group membership management tasks.

7.5.2 Antigone policy framework

The Antigone policy framework consists of three layers: the policy layer, the mechanisms layer and, finally, the broadcast transport layer. The policy layer contains the group's policy specification. The mechanisms layer supports the distribution and enforcement of the policy. This layer supports several mechanisms for various group operations, including member join and leave, membership monitoring, and group rekeying. Each of these mechanisms contain microprotocols. Membership monitoring, for instance, uses a microprotocol for processing ''heartbeat'' messages from either a member or a session leader. Other examples of microprotocols include join and leave requests, key distribution, rekey message transport, and data transmission with integrity and confidentiality. The broadcast transport is a group communications abstraction in that applications would send data to the group address, even in the absence of IP multicast support. The broadcast transport layer takes care of group communications, using either unicast or multicast.

Interlayer communication makes Antigone sensitive to group membership behavior and changes in the operating environment. Specifically, the mechanisms layer sends group events to the policy layer. The policy layer, in response, delivers the appropriate course of action to the mechanisms layer. Thus the mechanisms layer enforces group security policy, while being in close interaction with the latest group policy.

The Antigone policy is distributed to members when they are permitted to join the secure group. Unlike GDOI, Antigone does not provide mechanisms to communicate policy changes to members. Note that in

such systems some of the policies can still be changed in reaction to changes in the operational environment, but the new policy cannot be communicated to members. Consequently, some types of Antigone policies, such as data protection policy or rekey message protection policy, cannot change, whereas others, like frequency of rekeying, may change during the life of the group.

7.5.3 GSAKMP policy distribution and enforcement

GSAKMP (see Section 5.4.2) describes policy by assigning roles to various entities of the group. The group owner is either the content owner itself, or a representative thereof, and is thus responsible for issuing policy. The policy might depend on the value of the content, and the threat model under which the group operates [3]. The GC is responsible for policy distribution and enforcement. The GC may delegate these responsibilities (as allowed by the policy) to members, which then act as SGCs. The GC is still responsible for the creation of data encryption keys.

GSAKMP proposes the use of a policy token for specifying group security policy. The policy token consists of group membership, key distribution, and data protection rules of the group. The group owner signs the token for authentication. Members and controllers are expected to verify authenticity and follow or enforce the policy, as applicable.

Only authorized entities can be group members or SGCs in GSAKMP. Once authorized, they get access to group keys and the policy. GSAKMP expects all the group entities to verify the authenticity of the policy and follow it. If the GC observes member or SGC misbehavior, it can evict that entity. However, it is not clear how the GC detects noncompliance.

7.6 Summary

Group security policy is the third problem area of the IETF multicast security framework. It specifies the cryptographic mechanisms employed to protect group data, and actions to be taken in response to group membership behavior or changes in the operational environment.

Content owners specify who can receive group data, and the general level of data protection required. Such high-level requirements must be translated into a machine-friendly specification for automated enforcement. Policy may need to be negotiated at some point. In large groups, policy negotiation is not practical. But, the service providers must have a general understanding of the cryptographic and system capabilities of the prospective members. Besides, the problem has been proven to be intractable.

Small interactive groups contain several members who send data to the group. The senders may have individual security requirements, and may want to negotiate group security policy. Simple heuristic solutions have been proposed in the literature for policy negotiation.

Policy distribution and enforcement is the task of the GC or its authorized subordinates. The metapolicy specifies the parts of group policy that can be publicly announced. The rest of the policy is distributed to authorized membership only.

Key distribution policy is also enforced by the GC. The GC may also monitor compliance and evict misbehaving or malignant members by rekeying the group. Data protection policy is enforced by the senders. Group policy enables members to detect data sent by unauthorized senders.

We conclude this chapter with a note on the state-of-the-art standards-based solution to group security policy specification, enforcement, and distribution. GDOI supports key distribution and data protection policy specification and enforcement. Policy distribution can be restricted to authorized membership only. For more complete specification of policy, the IETF MSEC working group is developing the policy token approach.

References

[1] *Merriam-Webster Dictionary* (on-line version), http://www.m-w.com.

[2] Hardjono, T., and H. Harney, ''Group Security Policy Management for IP Multicast and Group Security,'' in *IFIP Networking*, Pisa, Italy, May 2002, poster.

[3] Harney, H., A. Colegrove, and P. McDaniel, ''Principles of Policy in Secure Groups,'' in *Proc. of Network and Distributed Systems Security Internet Society*, San Diego, CA, February 2001.

[4] Harney, H., et al., ''Group Secure Association Key Management Protocol,'' draft-ietf-msec-gsakmp-sec-00.txt, IETF, March 2001, work in progress.

[5] McDaniel, P., and A. Prakash, *Antigone: Implementing Policy in Secure Group Communication*, Technical Report CSE-TR-426-00, Electrical Engineering and Computer Science, University of Michigan, May 2000.

[6] McDaniel, P., A. Prakash, and P. Honeyman, ''Antigone: A Flexible Framework for Secure Group Communication,'' in *Proc. of the 8th USENIX Security Symposium*, Washington, D.C., August 1999, pp. 99–114.

[7] DeCleene, B., et al., ''Secure Group Communications for Wireless Networks,'' in *Proc. of the IEEE MILCOM*, Vienna, VA, October 2001, pp. 113–117.

[8] Baugher, M., et al., ''Group Domain of Interpretation for ISAKMP,'' draft-ietf-msec-gdoi-04.txt, IETF, March 2002, work in progress.

[9] Sloman, M., and E. Lupu, ''Security and Management Policy Specification,'' *IEEE Network, Special Issue on Policy-Based Networking*, Vol. 16, No. 2, March/April 2002, pp. 10–19.

[10] Meissner, A., L. Wolf, and R. Steinmetz, ''A Novel Group Integrity Concept for Multimedia Multicasting,'' in *Proc. of the 8th International Workshop on Interactive Distributed Multimedia Systems (IDMS)*,'' Lancaster, UK: Springer-Verlag, LNCS 2158, September 2001, pp. 233–244.

[11] McDaniel, P., and A. Prakash, *Ismene: Provisioning and Policy Reconciliation in Secure Group Communication*, Technical Report CSE-TR-438-00, Electrical Engineering and Computer Science, University of Michigan, December 2000.

[12] Balenson, D., et al., *DCCM Cryptographic Context Negotiation Template*, Technical Report TIS Report 0745-2, TIS Labs at Network Associates, Inc., February 1999.

[13] Balenson, D., et al., *DCCM Architecture and System Design*, Technical Report TIS Report 0709, TIS Labs at Network Associates, Inc., June 1998.

[14] Hardjono, T., et al., ''Group Security Policy Token,'' draft-ietf-msec-gspt-00.txt, IETF, September 2001, work in progress.

[15] Baugher, M., et al., ''Group Key Management Architecture,'' draft-ietf-msec-gkmarch-02.txt, IETF, March 2002, work in progress.

[16] Harkins, D., and D. Carrel, ''The Internet Key Exchange (IKE),'' RFC 2409 (proposed standard), IETF, November 1998.

[17] McDaniel, P., and A. Prakash, ''Methods and Limitations of Security Policy Reconciliation,'' in *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, CA, IEEE Computer Society, May 2002, pp. 73–87.

[18] Balenson, D., et al., *DCCM Cryptographic Context Negotiation Protocol*, Technical Report TIS Report 0757, TIS Labs at Network Associates, Inc., February 1999.

CHAPTER

8

Contents

8.1 The three components of multicast security

8.2 Overview of multicast routing

8.3 Security requirements in unicast and multicast routing

8.4 PIM-SM security

8.5 MSDP security

8.6 IGMP security

8.7 Security in other routing protocols

8.8 Summary

Securing multicast routing protocols

At the core of this infrastructure, and underlying its everyday operation, is the routing of IP packets across different ASs that make up the Internet. Similar to other software and hardware systems, routers and routing protocols are designed to follow a specific set of behaviors to achieve the effect of efficiency in IP packet handling, forwarding, and delivery path determination. At the microlevel, each AS executes one or more intradomain routing protocols, while at the macrolevel these ASs are interconnected to each other through one or more interdomain routing protocols.

At both the intradomain and interdomain levels, the correct execution of a protocol based on correct information is crucial to the proper operation of the whole Internet. This is true for both unicast and multicast routing protocols. Since many multicast routing protocols rely on the unicast routing table, it is important to provide security for the information exchanged among routers, and the information stored within routing tables.

The general meaning of the term security in the context of routing protocols is that of protecting the routing information from being illegally modified in transit (between routers) or in storage (within a routing table), and protecting against bogus or false routing information being injected into the network. Thus, a router must be assured that any valid routing information it receives is correct, and it must be provided with the ability to detect and reject false information.

This chapter focuses on the issue of security in multicast routing and multicast routing protocols. It begins by discussing some issues related to security in routing, including security requirements, possible attacks to multicast routing domains, and some possible ways to counter such attacks. The chapter views the multicast distribution tree as consisting of two identifiable parts: the core and the edges (or leaves). Corresponding to this view, the chapter focuses on two protocols as examples of each part of the tree. The PIM-SM protocol provides the core of the distribution tree, while at the edges, the IGMP protocol provides access by hosts onto the tree. The MSDP protocol is briefly discussed as an example of an interdomain routing protocol. This chapter also discusses Core Based Tree (CBT) [1, 2] and HIP [3] multicast routing protocol protection. As the discussions in this chapter focus on security matters, the reader is assumed to have some familiarity with these protocols.

8.1 The three components of multicast security

As mentioned in the previous chapters, there are three components required to achieve IP multicast security: one of which is multicast routing protocol security. The tight relationship among these three components for IP multicast security cannot be overemphasized. All three are crucial to providing security for IP multicast. These components are as follows (see Figure 8.1):

1. *End-to-end data protection*. Data protection consists of the application of cryptographic means end to end, to ensure that data delivered through the multicast group is protected against illegal modifications (data integrity), that the data is verifiable as coming from its original sender (source authentication), and that the data is only readable by valid members of the group (data confidentiality). The use of cryptography entails the delivery and management of keying material to valid members of the group.

   This area was covered in Chapter 4, and does not really pertain to infrastructure protection. However, as we will see, the techniques (such as group key and SA management) developed for content protection are equally applicable to routing control messages.

2. *Multicast distribution tree protection*. The multicast distribution tree protection typically consists of security mechanisms to ensure that tree behavior follows the specified protocol. That is, the multicast

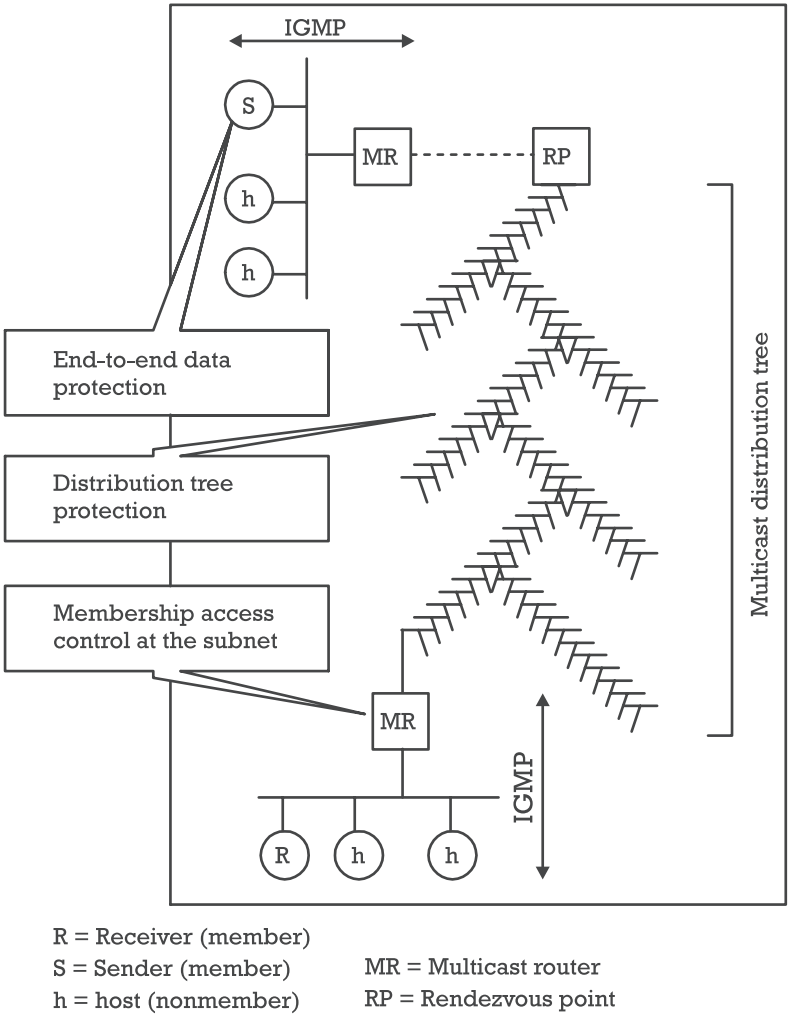


Figure 8.1 Three components of multicast security.

routing protocol that affects the distribution tree must be protected from attacks. In practice, this entails the protection of the control messages exchanged amongst tree entities, resulting in the correct operation of the tree. Examples of this include authentication of control messages in MOSPF and in PIMv2. Although the aim is uniform across all routing protocols, the methods used to achieve tree protection are specific to each multicast routing protocol.

This area is covered in the current chapter.

3. *Membership access control at the subnet level*. Membership access control at the subnet level consists of membership authentication/verification of the Querier (host) in a subnet, in the context of a specific multicast group. In general, this consists of the host proving its membership eligibility to a trusted authority in the subnet (e.g., an authentication server, the subnet router, or others).

   Since this area is closely related to the second area above, it is also covered in the current chapter.

One way to view the last two related areas is by considering the first (distribution tree protection) as dealing with security issues ''inside'' the multicast distribution tree, while the second (access control to the tree) as dealing with security issues at the ''edges'' of the tree. The first area deals almost exclusively with routers and interactions among routers, while the second area deals with hosts and the first hop (last hop) routers that connect the hosts to the distribution tree.

8.1.1 General types of attacks in multicast routing

One of the main factors that makes the basic IP multicast model [4] attractive from the perspective of scalability is the anonymous receiver model underlying it. In this model, any host in a subnet can join a multicast group (through IGMP), by notifying its multicast router of its wish to join a given multicast group.

The basic IP multicast model does not have any security features. Edge multicast (subnet) routers do not maintain identification information about the hosts that join the multicast groups, and they do not pass any identification information about the host to upstream routers in the distribution tree. The lack of maintenance of membership data (containing host identification information) in the multicast distribution tree, including the multicast (subnet) routers, allows IP multicast to scale to a large number of participating hosts.

From the perspective of security, this lack of host identification information represents a problem for access control. One of the possible attacks that exploits the anonymous receiver underpinnings of IP multicast is one in which a host simply joins a multicast group, without any intention of using the data being delivered to it. In such an attack, the user/host essentially extends or pulls the tree toward the subnet, effecting a wastage in resources and state within all the affected routers. In this case, the encryption of the multicast data does not provide any help, since the (encrypted) packets still flow down the distribution tree to the malicious host.

Two general types of attacks that may be carried out to a multicast distribution tree are as follows [5]:

1. *Sender attacks:* Here, the distribution tree is attacked by injection of bogus packets with the correct multicast address, thereby causing the packets to be sent to all receivers. The injection can occur at the leaves of the tree (within subnets with host members) or anywhere within the tree. This attack consumes bandwidth, since the packet would be delivered to all host members. Although such attacks are also possible within unicast, the impact is magnified in multicast precisely due to the replication effect within the distribution tree.
2. *Receiver attacks:* Here nonmembers (from a data/group perspective) simply join the group, causing the tree to expand, and multicast traffic to be forwarded to them. Even if the traffic content is encrypted by the source, the encrypted packets would still be forwarded, thereby consuming bandwidth. The attackers then simply discard the encrypted message, or may use it for cryptanalysis.

Both types of attacks result in the DoS to both the valid sender(s) and receiver(s) in the multicast group.

8.1.2 Multicast routing and security

Routing security has been a contentious issue in the past due a number of reasons. Router vendors and many ISPs do not perceive the security threats to the Internet routing infrastructure as warranting solutions, which the security community deems as minimal.

On one hand, many router vendors and ISPs believe that the security problems that may arise within a routing domain can be solved using the traditional humans- and- telephones approach. On the other hand, security experts argue that the growth of the Internet and its applications will soon outmode the humans- and- telephones approach, and that some security features must be built into the routing protocol itself.

Here are some reasons often quoted by those reluctant to adopt protocol-level security features:

- Among the service provider community, any unusual routing behavior visible from their network operations center (NOC), such as routing black holes, can be easily resolved through one network administrator calling another, since these ISPs typically already have peer relationships and SLAs.

- Since the bulk of the Internet's traffic today is Web related (i.e., HTTP and/or TCP traffic), attacks to the DNS infrastructure would be more attractive to the seasoned hacker. Thus, a hacker would get a greater effect by attacking the global top-level DNS servers (and other DNS servers), than by hacking into a router of an ISP.
- Distributed DoS attacks have historically been done against particular entities (end points) in the Internet, and not the carrier itself. Since many carriers and ISPs are only transport intermediaries, their main focus is on bandwidth management and performance aspects of their network. The end-point application of a connection is outside the scope of their attention.

Notwithstanding these views, there are counter opinions that see the Internet as a public infrastructure, and, in some cases, even a national security infrastructure. As such, any security at the microlevel (e.g., router and routing domain level) contributes to the security and robustness of the Internet infrastructure as a whole. This latter view sees the use of manually keyed MAC protection approaches on control packets in routing protocols as being insufficient and unreliable. Security should be built into all network protocol design, and not be added or retrofitted after the fact. A routing domain should be self protecting, in the sense that it should only accept authentic and authorized control messages, and that it should raise alarms at the NOC when security checks and balances fail.

8.2 Overview of multicast routing

In general terms, a routing protocol establishes the best path of delivery of datagrams (IP packets) from a source to a destination. In the context of IP multicasting, the aim of a multicast routing protocol is to create a *multicast distribution tree*, through which datagrams are delivered from one or more *sources* (or senders) to one or more *receivers*. From a router's perspective, the router executes an instance of the routing protocol, in order to allow the router to determine which interfaces to receive/deliver datagrams, and to inform other routers about its view of the current state of the network. From a domain's perspective, the set of routers as a whole determines the best path of packet delivery to end points (hosts) that reside in that domain, and to end points that reside outside the domain. When a datagram is destined to a single destination, that packet is called a *unicast* datagram and the routing protocol is called a *unicast routing protocol*. When a datagram is

destined to a number of hosts that are members of a *multicast group*, that datagram is typically referred to as a multicast datagram and the routing protocol is referred to as a *multicast routing protocol*. In both cases, delivery is based on a best-effort reliability of the IP datagram.

Multicast datagrams have a *multicast address*, which in IP version 4 (IPv4) is the Class D address ranging from 224.0.0.0 to 239.255.255.255 (i.e., 224.0.0.0/4). This address range is also known as the multicast *group address*. A Class D address has its high-order four bits set to ''1110,'' with the following 28 bits being the group ID. Multicast is different from broadcast in that a broadcast address is used to send a datagram to all hosts within a subnet.

A router typically has a *routing table* (or forwarding table) which contains several fields, including the destination addresses, next hop routers (toward a destination), and metrics. In essence, the router uses its routing table to map datagrams at incoming interfaces to outgoing interfaces, with the aim of processing the datagrams in as little time as possible. In many cases, the multicast routing table is an extension of the unicast routing table, in the sense that the multicast routing protocol makes use of and is dependent upon the unicast routing table. However, in contrast to unicast routing, in multicast routing a datagram may have to be forwarded to more than one next hop routers. Abstractly speaking, a multicast router creates a ''fanout'' effect, where one datagram is replicated and sent out toward multiple downstream routers. As a whole, a multicast routing protocol creates a *distribution tree*, where each multicast router is a fanout point or branch in the tree, and where the root of the distribution tree is the source of the datagram. This is why the multicast routing table entries are usually described in terms of (*source, group*) pairs, corresponding to the source address and the destination multicast (group) address.

When a routing protocol spans multiple domains, they are usually referred to as *interdomain* routing protocols. This is in contrast to *intradomain* routing protocols that execute only within a single domain. Examples of intradomain unicast routing protocols are the Routing Information Protocol (RIP) [6, 7] and the Open Shortest Path First (OSPF) protocol [8]. An example of an interdomain routing protocol is border gateway protocol (BGP) [9].

Another oft-used criterion for classifying multicast routing protocols is whether the protocol is targeted at a dense or sparse population of receivers. Examples of dense-mode multicast routing protocols are DVMRP [10, 11], MOSPF [12, 13], and Protocol Independent Multicast-Dense Mode (PIM-DM), [14], while examples of sparse mode multicast routing protocols are PIM-SM [15], and Core-Based Tree (CBT) [1, 2].

In addition to a multicast routing protocol instance executing on multicast routers, a *group membership protocol* is needed to allow a host to indicate its wish to join/leave a multicast group. A group membership protocol typically executes in hosts, and in the first hop multicast routers that are nearest to the hosts.

In the following we look briefly at two multicast routing protocols: DVMRP and PIM-SM. The reader is directed to [16] for more information on routing protocols in general, and to [17] for multicast routing. In addition, we briefly describe the IGMP and the recent Source Specific Multicast (SSM) paradigm.

8.2.1 Classification of multicast routing protocols

There have been a number of proposed multicast routing protocols in the past few years. Some have been more research oriented with very little industry adoption, while others, though maybe less elegant, have gained much ground in the industry. Although a complete review of all multicast routing protocols is outside the scope of this chapter, in the following we provide a classification and references for readers to follow up:

- Membership management protocol: IGMP;
- Flood and prune and dense mode protocols: DVMRP, PIM-DM, and MOSPF;
- Core-based protocols: PIM-SM and CBT;
- Interdomain protocols: Border Gateway Multicast Protocol (BGMP) and multicast source discovery protocol (MSDP).

Some of these protocols will be discussed below, while for information on the others, the reader is directed to [17].

8.2.2 DVMRP

In networks that have plentiful bandwidth and the receivers are densely distributed, ''flood and prune'' techniques may work quite well. DVMRP [10, 11] is an example of a dense mode protocol. The protocol uses the distance vector distributed routing algorithm to build per source-group multicast delivery trees.

DVMRP implements the *reverse path multicasting* (RPM) algorithm [18] in which a flood and prune approach is adopted. In the flood phase, the first

datagram for any (source, group) pair is flooded across the entire domain. Directly attached routers in subnets (i.e., leaf routers) may then transmit prune messages back toward the source if there are no group members on their directly attached leaf subnets. The prune messages have the effect of removing all branches that do not lead to group members from the tree, thereby establishing a source-based shortest path tree.

Since the prune state for each (source, group) pair has an expiration time in order to remove nonactive groups, for the remaining active (source, group) pairs a subsequent datagram is then broadcast across all downstream routers. The effect is that a new set of prune messages will be transmitted by the leaf routers, thereby prolonging the source-based shortest path tree associated with a given (source, group) pair.

DVMRP also allows a ''graft'' of a previously pruned branch of a group's delivery tree onto that tree again. Such a graft would occur in cases where a router that had just sent a prune message for a (source, group) pair finds that there are new members on the leaf subnet for that group. The router would then send a graft message to the previous hop router for that source, which in turn will cancel the previously received prune message. The graft message is forwarded upstream hop by hop in a reliable manner (via a Graft-Ack message and timeouts) toward the source, until it hits the nearest on-tree router. Being a branch point on the tree, that router would then restore the branch leading to the subnet in question that it previously pruned.

The reader is directed to [10, 11] for the full specification of the DVMRP protocol. Details on RPM is discussed in [18], while an excellent discussion on distance vector routing can be found in the classic work of [19].

8.2.3 PIM-SM

Sparse mode multicast routing protocols are aimed at environments where the receiver population is sparsely distributed. Since sparse mode protocols may be deployed across wide area networks (WANs), they use a different strategy than dense mode protocols, and assume in advance that bandwidth and reliability are at a minimum. The term ''sparse'' here refers not to the lack of receivers in the group, but rather to the wide dispersion of the receivers—possibly across wide geographic regions—which implies that methods such as flooding of the network become impractical.

The PIM-SM protocol is an example of a multicast routing protocol aimed at a sparsely populated region. Features of PIM-SM are [20]:

- PIM-SM maintains the traditional IP multicast service model of receiver-initiated membership.

- PIM-SM uses explicit joins that propagate hop by hop from members' directly connected routers toward the distribution tree.
- PIM-SM builds a shared multicast distribution tree centered at a *Rendezvous Point* (RP), and then builds source-specific trees for those sources whose data traffic warrants it.
- PIM-SM is not dependent on a specific unicast routing protocol: hence the term ''independent.''
- PIM-SM uses soft state mechanisms to adapt to underlying network conditions and group dynamics.

The PIM-SM approach is to require downstream members or receivers to explicitly request to join a given multicast group. This is in contrast to the dense mode version, which adopts a flood and prune strategy. At the heart of the region is the RP. Each multicast group has a shared tree through which receivers hear of sources. The RP is the root of this per-group shared tree, called the *RP tree*. The shared tree (RP tree) is the set of paths connecting all receivers of a group to its RP. A receiver on the RP tree receives packets from all sources of the group, except those sources that were pruned off the RP tree. For reliability, several routers may be configured in preparation to be an RP. These are referred to as *candidate RPs*.

Each subnet that contains a member has a *Designated Router* (DR), which is a PIM-router or a router that is running the PIM-SM protocol. The designated router is the highest IP addressed PIM router on a multiaccess LAN. Typically, the DR sets up multicast route entries, and sends corresponding join/prune and register messages on behalf of directly connected receivers and sources. The designated router may or may not be the same router as the IGMP Querier (see IGMP discussion below). The designated router may or may not be the long-term, last-hop router for the group, or a particular source that is sending to the group. In fact, a router on the LAN that has a lower metric route to the data source, or to the group's RP, may take over that role. Each group has exactly one RP, which is selected from the RP-set using a mapping function from the group address to one RP.

A member must indicate its wish to join a group using the IGMP protocol to the PIM router (Designated Router) in its subnet, which would in turn forward a *PIM-join* message upstream hop by hop to the RP of that group. For hosts wishing to join a group, the DR has the task of forwarding join requests upstream toward the RP. For the source or sender, the designated router has the task of encapsulating the source's datagrams (within

PIM-register messages) and unicasting it to the RP. The RP will in turn decapsulate and transmit the datagrams through the tree to the receivers.

At this point, the RP may either join the source's *Shortest Path Tree* (SPT) or continue letting the designated router encapsulate data packets and unicasting it to the RP, thereby having the designated router rely on the shared tree emanating from the RP. Although the shared tree (RP-tree) provides connectivity to the members of the group, it may not necessarily provide the best delivery path across the network. To improve performance, PIM-SM will allow receivers to ''switch over'' to a shortest path tree rooted at the source, provided the receiver has already begun to receive data from the source. This switchover is determined either by some threshold data rate or by some other metric, although in fact the DR may be configured to never switch over.

Other entities in PIM-SM that are involved in the running of a PIM region are the bootstrap router (BSR) and the set of candidate RPs, from which the RP is selected. The BSR is a dynamically elected router within a PIM-SM domain responsible for constructing the RP set, and originating bootstrap messages. The BSR for a PIM region constructs a set of RP IP addresses—namely the RP set—based on candidate RP advertisements received. The RP set information is distributed to all PIM routers in a domain, in a bootstrap message.

The reader is directed to [15] for the current version of the PIM-SM specifications, and to [20] for the previous specification, for a historical perspective on PIM.

8.2.4 IGMP

Multicast routing protocols typically run on multicast routers, and thus a method is needed to connect a host to its directly neighboring multicast routers, in order for the host to indicate to the multicast routers which groups it wishes to join. The protocol to carry out this function is called the IGMP, which runs between hosts and their multicast routers.

In order to find out if there are still active members of a group in a given leaf subnet, the multicast routers periodically sends out a host membership query message (or *Query* for short). If multiple multicast routers are present on the same leaf subnet or LAN, then one of them is elected to be the *Querier* for that LAN. In this way the querier learns about group membership information, and is therefore able to forward the multicast traffic of those groups to its leaf subnet. Query messages are addressed to the *all hosts group* (224.0.0.1). However, these messages have an IP TTL = 1, which means that the message is transmitted only to the directly attached subnet, and it is not forwarded by other multicast routers.

In response, a host will return a Host Membership Report (or Report for short) for each group to which it belongs. Note that the host will in fact send a Report (for a group) by addressing it to the multicast group, thereby allowing the Report to be heard by other hosts that are also members of that group. Multicast routers will also hear this Report, since all their interfaces are set to promiscuous mode. The Report has an IP TTL = 1 which would prevent the Report from being forwarded outside the subnet where the host resides. In order to avoid host reporting of the same group reports, each host must maintain a (random) delay timer for each group to which it belongs. When hosts are waiting during their respective delay period and a Report for the same group is heard from another host, then all of these waiting hosts must reset their timers (for that group) to a new random value. This behavior reduces traffic due to reports, and spreads the reports over time.

In IGMPv1 and IGMPv2 the multicast routers are not concerned about specific hosts and the groups to which hosts belong. Instead, the routers only want to know whether or not there is a member (at least one) of a group in its subnet. This is true for multicast routers in general (i.e., a member is present at an interface). The Querier in a subnet sends out IGMP Query messages corresponding to a group on a periodic basis. If a matching Report is not heard after a number of queries, the Querier then assumes that there are no longer active members of that group in its subnet and the interface corresponding to this group is removed. To prevent a new member host of a group from waiting too long for a Query message, that host introduces itself by immediately sending a Report to the group. This reduces the join latency for cases where the host is in fact the only member of the group in the subnet and thus where it will take some time for the data packets to arrive at that subnet.

IGMP has evolved through three versions, with the current version being IGMPv3 [21]. The other two versions are IGMPv1 [4] and IGMPv2 [22].

IGMPv2 improves IGMPv1 by introducing a number of features. A Querier election procedure is defined, based on the lowest IP address of the multicast routers. A new *Group-Specific Query* message is introduced, allowing query messages to be sent to a specific multicast group instead of all groups. This Group-Specific Query message is particularly useful when used with another new feature of IGMPv2: a new Leave Group message to be used by hosts. When a host wishes to leave a given group, it sends out a Leave Group message (for the group being departed from) to the all-routers multicast address (224.0.0.2). This allows the Querier to immediately inquire if the host was the last member of the group. The Querier does this by sending Group-Specific Query messages to the same interface where

the earlier Leave Group message was received. If a Report is not heard for that group, the Querier can thus remove the interface corresponding to this group.

IGMPv3 introduces additional new features over IGMPv2. Notable is the support for the *Group-Source Report* message, which allows a host to specify the source from which it wishes to receive data. In particular, an *Inclusion Group-Source Report* message allows a host to set the IP addresses of sources it wishes to hear from, while an *Exclusion Group-Source Report* message allows the host to ask for messages from a given source to be blocked. This is in contrast to IGMPv1 and IGMPv2, where traffic from all the group's sources is received by a host. IGMPv3 also enhances the leave group message of IGMPv2 by supporting a specific *Group-Source Leave* message. The Group-Source Leave message allows a host to set the specific IP addresses of the (source, group) pairs that it wishes to leave (in addition to the usual option of simply leaving the whole group). The reader is directed to [21] for more details on IGMPv3.

8.2.5 SSM

A recent shift in thinking within the multicast routing community has found expression in the notion of *Source Specific Multicast* (SSM). Thus, SSM is not a protocol as such, but rather a model and architecture to achieve more practical and efficient multicast routing for the Internet. The basic model for IP multicast in [4] allows any source (even anonymously) to send to the group by simply transmitting an IP datagram with the multicast address in question. Thus, the basic IP multicast model could be referred to as Any Source Multicast (ASM), since any source can send. In contrast, in SSM, a datagram with source IP address S and single source multicast destination address G is delivered to each upper layer ''socket'' that has specifically requested the reception of datagrams sent to address G by source S, and only to those sockets [23]. SSM uses the notion of a ''channel,'' which is essentially the *network service* identified by (S, G), for SSM address G and source host address S. It is important to note that SSM provides network layer support for *one-to-many* delivery only, although, like the ASM model, SSM is receiver driven, and the set of receivers is unknown to the sender.

The motivations for SSM are explained as follows [23]:

- Elimination of cross delivery of traffic when two sources simultaneously use the same source-specific destination address. The simultaneous use of an SSM destination address by multiple sources is explicitly supported.

- Avoidance of the need for interhost coordination when choosing source-specific addresses.
- Avoidance of many of the router protocols and algorithms that are needed to provide the ASM service model.

The reader is directed to [23] for further information on the motivation and architecture of SSM. A proposal to modify PIM-SM to conform to the SSM model has been proposed in [15].

8.3 Security requirements in unicast and multicast routing

Although each routing protocol—either unicast or multicast—has its unique protocol behavior and therefore security requirements, there are a number of generic or common requirements that would need to be fulfilled in any domain executing these routing protocols. Here, it is useful to distinguish between control packet authentication at the routing protocol level (intradomain) and routing information origin authentication at the AS level (interdomain). These requirements are as follows:

- *Protection of control messages*. The term ''protection'' here means the following functions:
  - *Authentication*, either group authentication or source authentication.
  - *Integrity protection*, against illegal modifications.
  - *Replay protection*, against replay of valid control messages or other bogus messages.

  In essence, all control messages exchanged intradomain (and also interdomain) should be protected against modifications, and routers should be able to detect forged ones and react to them.

  If symmetric key cryptography is used, then *group authentication* is afforded, while if asymmetric key cryptography is deployed, then *source authentication* can be provided. Routers are thus provided with some assurance that control messages were sent by other legitimate routers in the domain that are in possession of the correct keys. Previous efforts relevant to this area have been reported in [24–30]. Note that confidentiality of routing information is rarely required, although encryption may have its uses.
- *Origin authentication of routing information*. When route information is advertised or propagated across several domains or AS, *origin*

authentication of the domain or AS issuing the route information must be provided. Thus, when an AS is forwarding route information from another AS, the recipient must be able to know the original AS that created the route information. This requirement is particularly relevant for interdomain routing protocols, such as BGP in unicast, and MSDP or BGMP in multicast.

In unicast interdomain routing, the BGPv4 [9] allows two BGP routers (border routers) to advertise to each other the available routes in their respective ASs. Once a route is learned, the information may be propagated further to other BGP routers of other ASs. The problem here is that an AS several AS-hops away does not have the ability to verify that the information has not been modified since it was published by the origin AS. The original BGP protocol provides only minimal security in the form of shared secret key–based message digest computation of the TCP connection between two border routers [31].

The Secure-BGP (S-BGP) protocol [32] has attempted to address the security deficiencies of BGP, which have been described in more detail in [33]. S-BGP introduced certificates and digital signatures to allow an AS to prove its ownership of IP address blocks and AS numbers, to prove its identity, and to allow a BGP router of an AS to prove the router's identity and its authorization to ''speak'' on behalf of its AS. S-BGP also introduced a new BGP transitive path attribute, which carries digital signatures (or ''attestations'') over the routing information sent in a BGP update message. A recipient of the update can thus use the signature and certificates to verify the address prefixes and path information mentioned in the message.

S-BGP, however, has not gained much adoption by ISPs for a number of reasons, including the oft-cited increase of message sizes (due to digital signatures and certificates), and the need of AS-level PKIs. Other previous efforts in this space have been reported in [34, 35].

- *Source authentication for domainwide sending*. In many routing protocols, certain entities are given the task of sending out messages to all other routers, either via a domainwide broadcast or via a special multicast group. A group-shared symmetric key approach is insufficient for message integrity protection, as it allows any router (e.g., one that has been compromised or is malfunctioning) to also send out the message carrying a MAC computed using the shared group key. Recipient routers would not be able to verify the

authenticity of the message as coming from the authorized sender. Thus, source authentication is required in this case, which can be achieved using public key–based digital signatures.

- *Efficient key management for routers*. Given that cryptographic methods are the most likely techniques to be used to provide control message authentication, it follows that an efficient method for managing keys is required for routers.

  Key management is necessary for a number of reasons, two of which are as follows. First, a router could be compromised, and its key could be misused. Thus, there must be some way to revoke and replace the keys within routers that are suspected of having been compromised or tampered with. Second, cryptographic keys have a limited lifetime from the perspective of the amount of traffic to which a key is applied, even when control messages may be considerably less in volume compared to data packets. Although the danger from cryptanalysis may be low, keys must be able to be replaced regularly according to some given policy.

  If public key cryptography is to be utilized within routers, efficient methods to manage digital certificates are required. Since routers would typically use device certificates, manual installation would probably be the easiest course of action for initial deployment. However, since routers may be added to or removed from a domain, a more automated certificate management approach may be needed. The issue of PKI management brings with it other related issues, such as the CA selection for intradomain and interdomain messages.

- *Detection of unusual routing behaviors*. Although domainwide fluctuation in routing behaviors can be detected by human operators, additional router-level functions need to be added. These may range from notification (to the operator) caused by failures in control message authentication, to more sophisticated approaches based on sudden changes in traffic volumes within a router's incoming and outgoing interfaces.

  A concrete example in multicast routing would be repeated joins and prunes to the same group from the same subnet (or subtree of the distribution tree) that affects bandwidth in that part of the network. Other examples would be a DoS attack in the form of packets sent to all the Class D multicast addresses. In the context of the PIM-SM protocol, the RP represents a unique target of attacks from bogus register messages and other threats. Another more

subtle attack is one in which the attacker attempts to control or influence the rates of traffic experienced by honest members of the group.

- *Containment of security breaches*. Although they are difficult to define and measure, security breaches within one domain should ideally be confined as much as possible to that domain, with little or no effect propagated to other domains. At the AS level, this also implies that the autonomy of the AS has to be followed through, in that the security mechanisms and policies deployed in one AS should not be dependent on other ASs. A concrete example would be the situation in which a DoS attack occurring in a transit AS affects adjacent ASs.

8.4 PIM-SM security

In this section we look more closely at the security issues and possible solutions for the PIM-SM protocol. A brief introduction to PIM-SM was provided in Section 8.2. However, further information on PIM-SM may be found in [17, 36].

The section is divided into several parts. The first part (Section 8.4.1) provides some historical background on the efforts in the IETF. The second part (Section 8.4.2) discusses the proposal put forward by the PIM working group in the IETF to improve the security of PIM. The third part (Section 8.4.3) discusses a subsequent proposal called *Simple Key Management Protocol* (SKMP), to manage the keys used with a PIM region. This is followed by a discussion on the security issues around the revised PIM specification (Section 8.4.4), for which some security-related suggestions are provided (Section 8.4.5).

8.4.1 Background

The PIM Working Group in the IETF began addressing security issues in PIM-SM in [37]. There, the authors identified the need for control message authentication, and proposed a key arrangement (see Section 8.4.2). The revised PIM-SM specification [15] provided further discussion on security considerations, and a suggestion to use IPsec AH or ESP and a manual method for deploying SAs among a group of routers (see Section 8.4.4).

In studying the definition and use of SAs in IPsec, the authors of the revised PIM-SM [15] rediscovered something that has long been understood within the multicast security community in the SMuG Research Group

(in the IRTF) and the MSEC Working Group (in the IETF), namely, that a pair-wise SA (as defined by IPsec) is unsuitable for multisender and multirecipient messages. Indeed, it is precisely this issue that motivated the development of the GSA in [38, 39], which was then introduced into the SMuG Research Group as [40] and later into the MSEC Working Group in [41]. The immediate applicability of GSAs to PIM routers seeking to share a single (group) SA is straightforward, as suggested again more recently by [42].

In the following, we will discuss PIM security in a chronological fashion, starting with the PIMv2 key arrangement as described in [37] (Section 8.4.2), and the accompanying key management proposal of [43] (Section 8.4.3), and ending with the security issues in the Revised PIM-SM specifications (Sections 8.4.4 and 8.4.5).

8.4.2 PIM authentication

The discussion in the following sections is based on the PIM authentication draft [37] published by the PIM Working Group in the IETF. The PIM authentication draft also suggested a key arrangement for authentication keys, to be applied for control message authentication within a PIM domain. The aim of the PIM authentication draft is that when security is enabled, all PIMv2 messages will use IPsec AH [44]. The authentication mechanism suggested was HMAC-MD5-96 [45, 46] and HMAC-SHA1-96 [47].

The PIM authentication draft identifies the following entities in a PIM domain that require keys: the bootstrap router (BSR), the RP, the designated router, and other PIM routers. All keys are relevant and recognized only within one PIM domain. The keys are as follows:

- *BSR public key*. All BSRs own an RSA [48] key pair[1] and use the BSR private key to sign an entire bootstrap message, while other PIM routers only have the BSR public key to verify the signature. This allows only authorized candidate BSRs to become a BSR. This RSA public key pair is denoted here as (SK_{bsr}, PK_{bsr}).
- *Equal opportunity key*. All PIM routers in the same domain share a single private (symmetric) key used to compute digests or MACs for the protection of PIM control messages. This key is denoted as K_{eq}.

1. Although the first version of the proposal in 1998 required one RSA key pair to be shared by all BSRs, the second version of the proposal (IETF-45, July 1999) recognizes the need for each BSR to have a unique RSA key pair.

This key is used for per hop authentication of control messages by PIM routers in a given PIM domain.

- *RP-key*. All RPs and BSRs share another private (symmetric) key, known as the *RP-key* and denoted as K_{rp}. No other routers have this key. For candidate RP advertisements, the digest is only calculated with the *RP-key* K_{rp} (instead of the equal opportunity key K_{eq}). This achieves the effect that only the authorized candidate RPs can advertise their candidacy to the BSR.

For convenience, the keys of [37] will be referred to as *primary keys* in order to distinguish them from other cryptographic keys.

8.4.3 SKMP for PIMv2

SKMP for PIMv2 [43] attempted to fill in a gap that the PIM authentication draft [37] introduced, namely, the management of the cryptographic keys in a PIM domain. The approach of SKMP is to define additional cryptographic keys to support the main PIM keys of [37], which are referred to in SKMP as primary keys. The supporting keys themselves are referred in to in SKMP as *key management keys* (KM-keys). Figure 8.2 shows both sets of keys. SKMP introduces a DKD into the picture, which is the trusted entity that controls key management for a PIM domain.

The intent is for the DKD to be a separate entity from other PIM entities such as the BSR and RP. This is so as not too overload those PIM entities—which are mainly routers—with key management functions. In addition, DKD is assumed to be a trusted entity with stronger security hardening against attacks. Given that the multiple routers can be candidate BSRs and RPs, a more permanent entity (such as a key server) is more attractive from a performance and security cost perspective. However, there are of course situations where the DKD function could be assigned to a PIM entity.

SKMP restricted public keys

SKMP uses the notion of *restricted* public keys, in the sense that copies of public keys of certain entities are (manually) assigned to (or known by) a restricted type and number of PIM entities. For example, since only certain messages are sent by the BSR to the RP, then only the RP (and candidate RPs) should know the BSR's public key used to verify or decrypt this message.

This approach allows the option of using encryption by the sender (e.g., the BSR) using a private key, whose decryption is possible only by those in

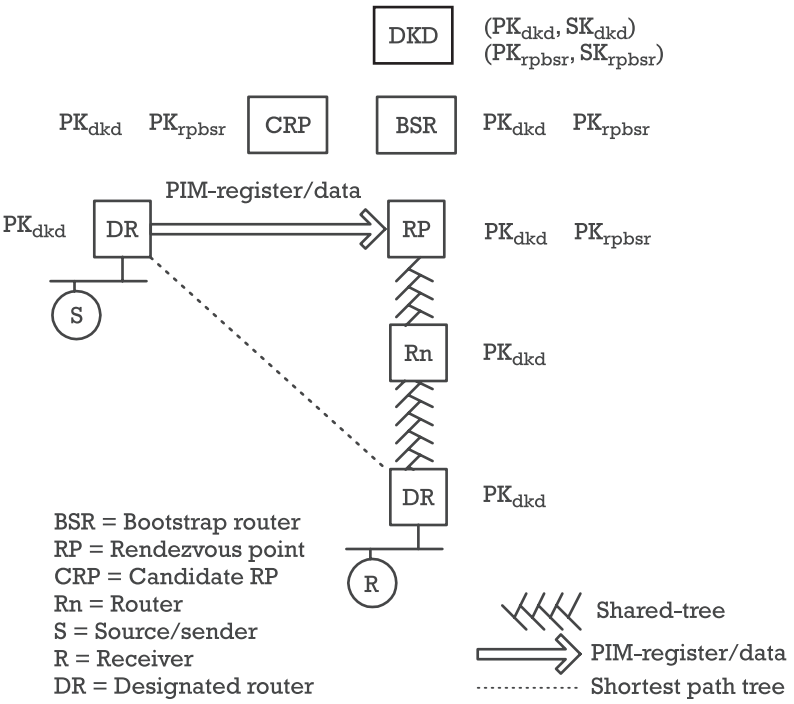


Figure 8.2 Key management keys in SKMP.

possession of the matching public key. These keys will be denoted as restricted public keys in the ensuing discussions. Furthermore, this approach lends itself to a PIM domain deploying a closed private CA for their PKI management, where the public keys and certificates are never known or advertised outside the PIM domain or region.

SKMP key management approach

The KM-keys in SKMP are long-term keys used to protect the delivery of the primary keys. The practical thinking here is that since the KM-keys are used to protect the infrequent rekeying of the primary keys (i.e., far fewer in frequency than the use of the primary keys on control messages), the KM-keys need not themselves be rekeyed frequently.

The KM-keys in SKMP are as follows:

- *DKD public key.* The DKD is assigned the RSA public key pair (PK_{dkd}, SK_{dkd}). Here, the public key PK_{dkd} is only known by PIM routers within the domain. Only the DKD knows the secret key SK_{dkd}.

No other entities know these keys. This allows the DKD to send encrypted (and/or signed) messages which will be readable only by PIM routers.

- *RP-BSR public key.* All candidate RPs (namely, the routers from which the set of RPs will be selected) and the BSR are assigned the restricted RP-BSR public key PK_{rpbsr}, whose matching secret key (namely, SK_{rpbsr}) is only known to the DKD. The DKD also knows the public half PK_{rpbsr}. Note that all RPs know the same public key PK_{rpbsr}.

  In essence, the RP-BSR secret key SK_{rpbsr} is used for the DKD to communicate securely with the BSR and the RPs (including the candidate RPs).

SKMP also proposed the use of two long-term multicast groups which are administratively scoped [49] for use only within the single PIM domain. The first would be used to send keying material to PIM entities involved in the bootstrapping of PIM, and the initial key dissemination. The second group would be used to send keying material to the larger group of routers running the PIM protocol. This is depicted in Figure 8.3. Key management would then be performed using a separate group-key management protocol, such as those described in Chapters 5 and 6.

The reader is directed to [43] for a detailed discussion on the key management approach for PIMv2 proposed by SKMP.

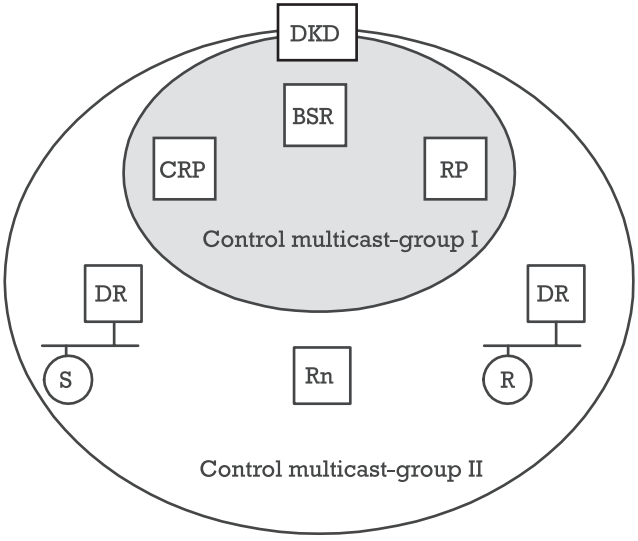


Figure 8.3 Multicast groups for key management.

8.4.4 Revised PIM-SM: Security issues

The recent revised PIM-SM specification [15] identifies a number of security issues in a more pronounced manner than previous versions of the PIM-SM specification. In particular, the revised specification considers potential attacks to the following types of messages (see Section 8.2.3):

- Forged *Join/Prune* messages. These are the messages issued by a DR toward the RP. As mentioned earlier, this has the effect of either pulling the tree into a subnet that has no legitimate members, or of pruning the tree from a subnet that has legitimate members [5, 50].
- Forged *Hello* message. This is the message sent by a router in the process of being a DR. A router that can forge a *Hello* message can effectively set itself up as the DR in the LAN, particularly in the absence of *Assert* messages. Since a DR has an important role in a LAN, including forwarding group traffic to that LAN and register encapsulating (to the RP) any traffic sent from legitimate hosts in that LAN, this issue is a important one for PIM security.
- Forged *Assert* message. An attacker (host or router) that is able to send a forged *Assert* message can trick the legitimate designated forwarder to stop its task of forwarding traffic to that LAN, effectively starving legitimate host members downstream.
- Forged *Register* message. This is the message sent by the DR to the RP which has (encapsulated) data from the source. An attacker that is able to send a forged Register message can trick the RP to subsequently forwarding bogus traffic onto the distribution tree.
- Forged *RegisterStop* message. A forged *RegisterStop* message has the effect of preventing a legitimate DR from registering packets (sent from legitimate hosts downstream) to the RP, effectively preventing those legitimate hosts from being able to send to the group.

Following [37], the revised PIM-SM specification also points to the need of control message authentication using either IPsec AH or IPsec ESP to solve the previous list of potential attacks. However, it goes further than [37] by suggesting specific ways to deploy the IPsec parameters (e.g., SPI and SA):

- *Link-local PIM messages.* The link-local messages here are the *Hello*, *Join/Prune* and *Assert*, which are all sent to the special all-PIM-routers group at address `224.0.0.13`. Here a single IPsec SA is to be used for authentication of all link-local messages on a link, with the IPsec

replay option being enabled. However, there is an inherent problem with using a single (identical) unicast SA for a many-to-many communication, of which the special all-PIM-routers group is an example.

In simple terms, the main issue with this approach is the fact that the (unicast) SA model as it is defined in [51] is simply unsuitable for group communications in which there are multiple senders and multiple receivers. This problem was reported earlier in [52], and is precisely the issue that is addressed by the MSEC Working Group in the GDOI effort.[2]

As an illustration, in the PIM-SM scenario, a PIM router (say, Router A) will receive link-local messages from other PIM routers (say Router B, Router C, and Router D) in the domain. Each of these routers will also receive messages from the others, hence representing a many-to-many communication instance. As defined in the IPsec specification, Router A will use the triple value <SPI, Destination Address, ProtocolType> to index into its SAD and the SPD, in order to finally locate the suitable key (in its possession) to apply to the message. Since there is only a single (identical) SA installed at all the routers for all link-local messages, Router A will not be able to distinguish the sender of any given PIM message, as at the IPsec level only the destination address (namely, a multicast address) is used to index into the SAD/SPD. Granted, IPsec could also scan for the source address, but such behavior is not part of the standard IPsec implementation, and thus cannot be guaranteed across all router vendors.

Furthermore, since a single SA is used in Router A for many senders, the IPsec sequence numbering for replay protection is open to collisions due to the fact that each sender (Router B, Router C, and Router D) will maintain its own sequence numbers. This may, in turn, result in legitimate packets being dropped by Router A.

- *Bootstrap messages.* A BSR sends out bootstrap messages to allow PIM routers to map group addresses to RPs. Since the BSR is elected, only a BSR should be able to send out such messages. Thus, source authentication in this case is paramount.

2. As of this writing, the IPsec ESP specification is being updated to reflect a number of changes, including better support for IP multicast traffic. More specifically, each entry in the SAD must now indicate whether the SA lookup makes use of the source and destination IP addresses (represented by two bits). Although this goes a long way to support SSM, there remain some open issues in ASM traffic processing, in regard to the use of single SA for multisender multicast groups.

The recent work of [42] corrects the digital signature proposal of [43] by suggesting that source authentication for bootstrap messages be done at the PIM level, rather than at the IPsec level. This is because the message is relayed from router to router and a new packet is created at each hop. Therefore, origin authentication would be lost if each router were to create a new packet and apply IPsec. Hence, a digital signature at the PIM level would allow the signature to be carried as part of the PIM payload from the BSR to the intended recipients (PIM routers), which can then be assured of the source authenticity of the information in the payload.

8.4.5 Revised PIM-SM: Possible solutions

As mentioned in Section 8.4.4, the revised PIM-SM specification has seen the need for a group shared SA. However, as explained above and also in Chapter 4, the IPsec definition of a unicast SA does not fit with the many-to-many communications mode.

Bearing in mind that security functions come at a cost, and are best integrated into the design of a protocol, there a number approaches that could be adopted by implementers of PIM-SM:

- *Shared SA with PIM-level sequence number.* In this approach, the IPsec sequence number would be ignored, and replay prevention would be achieved using a PIM-level sequence number as part of the PIM message payload. The shared (unicast) SA would be manually configured into each router.

  This would allow for immediate deployment without having to resort to modifying IPsec. The cost here would be additional PIM-SM design effort, and implementation of the sequence counter as part of PIM-SM.

- *GSA and GDOI.* Since the GSA model [38, 39] and the GDOI protocol [41] were developed for SA and key management in multiparty communications, both lend themselves for use in PIM.

  Here, each router would have to establish a Category 1 SA (unicast) with the GCKS, under which a protected download of the following items would occur:

  - A unique predefined Category 3 SA to send data (i.e., PIM messages) to other routers.

  - *N* predefined Category 3 SAs to receive data (i.e., PIM messages) from other *N* routers.

Other combinations based on the basic GSA are also possible. The work of [53] suggests ways for a router to indicate that it is a sender, and to which group it will be sending.

- *Security at the PIM level.* Another possibility would be to apply all security functions, including digital signatures, at the PIM level, thereby reducing or removing dependence on security features provided by the lower layers of the protocol stack (e.g., IPsec).

Each of the above possible solutions has its advantages and difficulties, either technical, or due to deployment considerations. Of the three, the most promising would be the GSA and GDOI approach, since the GDOI protocol is designed for one-to-many multicast. This would entail minimal changes to a PIM-SM implementation, and would allow PIM-SM to rely on GDOI as a separate layer underneath it. The cost would be that several instances of GDOI would need to run in a PIM-SM domain to match the many-to-many model of control message distribution among PIM entities. This may perhaps require that extra state information and keys be maintained by PIM-SM entities. However, this approach may be tolerable compared to adding security specifics into the PIM-SM protocol, such as sequence numbering and other mechanisms.

To conclude this section, it is clear that there are still a number of broad security issues and deployment problems for multicast routing. Some are unique to PIM-SM, but others are common to several routing protocols (e.g., securing IP packets for multireceiver communications). One point is clear, however, that security must be taken into consideration from the earliest stages of any protocol design effort. Security should not be relegated as an afterthought when the protocol is almost complete or mature, or, even worse, when it is already shipping as a product.

8.5 MSDP security

The aim of MSDP is to allow interdomain multicast routing through the sharing of source information between routing domains. As described in [54, 55], MSDP is a protocol to connect multiple PIM-SM domains together, where each PIM-SM domain uses its own independent RP(s) and does not have to depend on RPs in other domains. The MSDP approach has the advantage that each PIM-SM domain is thus independent of other domains, and that domains that contain only receivers (no senders) may obtain data from groups without having to advertise the group membership.

The source discovery from one domain to another is achieved through peering between MSDP-enabled PIM-SM routers in both domains. When a domain D1 learns of a source in domain D2, the normal source tree building mechanism in PIM-SM will be used to deliver multicast data over an interdomain distribution tree. The learning of sources in other domains is achieved through the use of a *Source-Active* message which is sent among MSDP peers. Thus, when an RP (say, RP1) in a domain (say, D1) learns of a new sender/source within its own domain D1 (through PIM Register messages), RP1 will announce the new source to the MSDP peers (say, RP2) by sending a source-active message to RP2. The source-active message contains the source address (of the new sender), the group address (to which the sender is sending), and the IP address of the RP announcing it (namely, RP1 in domain D1). The inclusion of the address of the announcing RP (here RP1) allows other RPs in other domains to communicate directly to the announcing RP (RP1).

When an MSDP peer (say, RP2 in domain D2) receives a new source-active message (say, from its MSDP peer RP1 in domain D1), it must propagate it to other RPs that are MSDP peers, away from the originating RP via a process referred to in [55] as *peer-Reverse Path Forwarding (RPF) flooding*. In addition, the MSDP peer (namely RP2) must also check to see if there are any group members in domain D2 that are interested in any group announced within the newly received source-active message. This is done by the RP2 checking for an (*,G) entry with a nonempty outgoing interface list, which implies that some system in the domain D2 is interested in the group.

If there is such an entry, then the RP (RP2) must trigger a (S,G) join event toward the data source in the same manner as when a Join/Prune message was received addressed to the RP itself. The effect of this is setting up a branch of the source tree to this domain D2, and subsequent data packets arriving at RP2 via this tree branch are simply forwarded down the existing multicast shared tree inside domain D2. Leaf routers can then join the source tree using the PIM-SM protocol. Figure 8.4 attempts to show this process.

A brief analysis of the security issues in MSDP (as specified in [54]) has been reported in [56], based on the assumption that the same key arrangement of [37] for PIM domains is deployed. The security analysis essentially points to the need of source authentication (or origin authentication) for source-active messages, as these messages are in essence interdomain advertisements.

Since RPs make up the set of MSDP peers, and since source-active messages are propagated to other MSDP peers away from the origin, one possible solution is to deploy public key cryptography and digital signatures at the PIM level, for the payload of the source-active message. Thus, when an

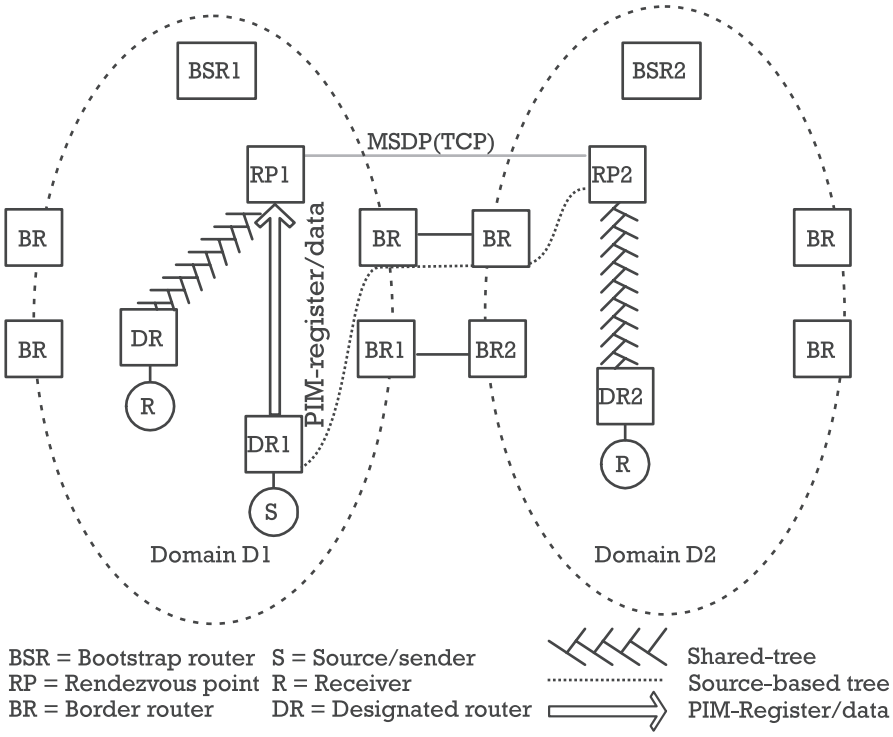


Figure 8.4 MSDP overview.

MSDP peer (say, RP2 in domain D2) receives a new source-active message (say, from its MSDP peer RP1 in domain D1), it should verify the signature of RP1 before it propagates the information to other MSDP peers (say, RP3). If the digital signature was applied to the source-active message at the PIM level (instead of at the TCP level), RP2 could propagate the source-active message unchanged, and the next recipient (RP3) could verify the origin of the source-active message (namely RP1). This approach provides stronger security compared to the simple HMAC preshared key approach commonly advocated for TCP connections. The reader is directed to [56] for further discussion of MSDP security.

8.6 IGMP security

As mentioned in Chapter 1, the second important area in routing infrastructure protection is controlling access to the multicast distribution tree. This concerns not only the issue of authentication of messages used by membership protocols at the edge of the tree (e.g., IGMP [22]) at the

network layer, but also the issue of *authorization*, by which a host proves to the tree that it is entitled to join the tree of a particular multicast group.

The problem of membership management security in IP multicast is a multifaceted one, as it involves a number of factors in its actual application. The basic IP multicast model of [4] focused on the method by which a host would indicate to the multicast router the group(s) that host wanted to send to or receive from. In that original design, a host could even send to a group without receiving from it.

This design is very scalable, as the multicast distribution tree could extend to any part of the network (be it a domain, an AS, or even the whole Internet), with receivers on one part of the tree being unaware of other receivers in a different part of the tree. Furthermore, since membership management was perceived to occur at the edges of the multicast distribution tree, the protocol implementing this function could be independent from the multicast routing protocol that creates the tree. As an example, version 2 of the IGMP protocol [22] essentially provides a signaling mechanism for hosts to indicate their wish to join a given group with a specific source. IGMPv2 does not carry or convey membership access control information in any form, nor does it carry or convey policies regarding group membership.

Multicast membership management security (more loosely referred to as IGMP Security) is difficult to define because it represents the meeting point of several factors, including user/hosts with different connection speeds, network topologies with various restrictions, different (transit) ISPs and routing policies, and various service models used for different multicast service models.

Here, we adopt the following definition of membership management security: It is the authorized access of identified/authentic users/hosts to resources (connectivity, router resources, and data access) on a network (run by carriers or ISPs) for a given multicast application that provides security appropriate to the demands of the application.

There are differing opinions as to which level of security is required for membership management. What is clear, however, is that techniques and solutions must be developed and made available, from which multicast applications can choose as appropriate to their specific needs. For example, a service providing multicasted stock quotes will have different security requirements from a service providing video-on-demand (VoD), which will again be different from a service providing a real-time group chat or neighborhoodwide multiplayer game. Some ISPs may not care how much traffic a user consumes, while others may care about possible waste of bandwidth by users.

8.6.1 Membership authorization and authentication issues

If we accept the notion that a user/host needs authorization to request a multicast router to join a particular group, it follows that some entity must have the power or authority to make the decision for that user/host. This brings into the picture a number of possible authorities that may make that decision. In the case of multicast applications, the main entities that have a vested interest are two, namely, the network service provider (e.g., ISP) and the content provider (e.g., PPV provider). The first provides access to the data packets at the network layer through the use of various network protocols, while the latter provides access to the data content through the use of various cryptographic protection algorithms and key management protocols.

On the abstract level, regardless of which authority made the decision for a given host, there must be agreement or alignment between the authority that provides the network services (including the multicast distribution tree) and the authority that provides access to the content being multicasted. In other words, before any security methods for multicast can be deployed there must be some mutual agreement between the network service provider (e.g., ISP) and the content provider (e.g., PPV provider). This is the basis for the preliminary proposal of [5].

Thus, in the abstract design, a secure group membership management protocol must at least provide a method for the following:

- *Authorization*. A host (or the application on the host) must ''prove'' to the distribution tree that it has permission to send to and/or receive from a given multicast group G.
- *Host identification/authentication*. A host (or the application on the host) must ''prove'' to an entity representing the connection point to the tree that it is who it claims to be. This entity is most likely to be the closest multicast router or the DR, which in turn may rely on another entity [e.g., Authentication Authorization Accounting (AAA) server] to verify both the host identity information and the authorization claims. This requirement is important particularly in shared medium LANs or subnets, where other hosts may be able to intercept or forge IP packets.
- *Message authentication*. Control messages exchanged between a host and the multicast router (representing the connection point to the tree) must, of course, be protected against illegal modifications, and possibly have source authentication.

It is also important to note that the above requirements may be fulfilled at different levels of the communications stack. Thus, for example, there

could be a case where the network (distribution tree) does not wish to deal with authorization since bandwidth is plentiful, and that authorization occurs at the application level. In this case, the distribution tree could be oblivious to hosts connecting to it. Another scenario would be where the multicast group is short lived and where the data traffic is encrypted from the source. In this case, the distribution tree and the membership management protocol at the tree's edges may not care about providing strong security, since the session is short and all states in the routers pertaining to that session would soon be purged in any case.

8.6.2 Membership authorization approaches

The work of [5] provides a preliminary investigation into the notion of multicast membership authorization. The method to convey authorization for a host to join a group is the *multicast access token*. The KD is the assumed point of convergence or agreement between the two authorities (namely, the network service provider and the content provider). This is shown in Figure 8.5.

In Figure 8.5, when a user/host (sender or receiver) wishes to join a group (whose value-carrying data is assumed to be encrypted), it must first obtain the group key from the KD and, at the same time, obtain a proof of membership from the KD (as the content provider authority). This proof is then presented to the network service provider (ISP) at the time when the host requests (at the network level) the multicast router to join the group. This is step 1 of Figure 8.5. This proof of membership is the access token.

The KD must provide the authorization entity (e.g., multicast routers, AAA server) in the network with a matching list of authorized hosts and other security parameters. This is step 2 of Figure 8.5. Alternatively, the KD itself could take this role, eliminating the need of a separate authorization entity. Note that the notion of an authorization entity was also proposed by the work of [57].

At this point, the authorization entity could disseminate copies of the authorization information to the multicast routers in the domain through a special group (e.g., the all-multicast-routers group in PIM-SM), as shown in step 3 of Figure 8.5. This was the approach suggested in [50, 58] in the specific context of the PIM-SM protocol in which PIM routers were already in possession of a cryptographic key (K_{eq} in Section 8.4.2). Alternatively, each router could retrieve the information periodically from the authorization entity or query it as needed.

In step 4, the host presents the access token to the multicast router, as the connection point to the distribution tree. The multicast router verifies

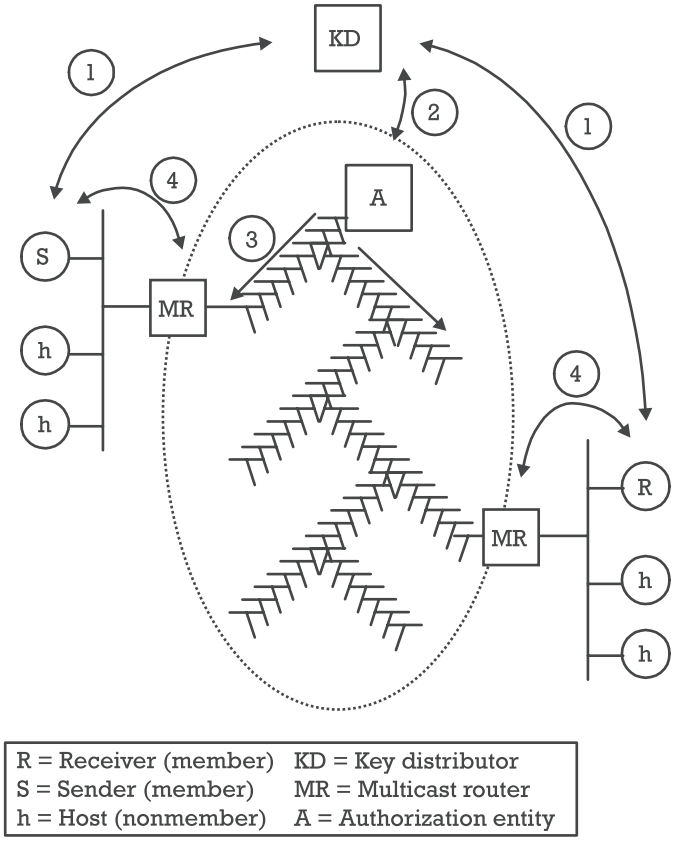


Figure 8.5 Multicast access token concept.

the authenticity of the access token, and checks the authorization information carried in the token. This check could be done against a copy of the authorization information cached at the multicast router, or the multicast router could query the authorization entity. If the host is authorized to join the group, then the network layer membership protocol (e.g., IGMP) can begin to execute.

This concept of a user/host obtaining authorization before requesting a router to join is also adopted in a more elaborate manner in [59]. There, certificates and PKIs play an important role in the design, and the group key manager (GKM) entity (see Chapter 5) is also proposed for reauthorizations. The recent work of [60] proposes a kerberos-like approach, whereby tickets are used for hosts to obtain authorization. The reader is directed to [59, 60] for further details on each design. In the context of IPv6, the work of [61] has begun to address possible solutions for multicast and anycast in IPv6.

8.6.3 Message authentication approaches

Once the host is connected to the tree and is executing the IGMP protocol with the router, IGMP messages must be protected against modification and forgery. This introduces the need of IGMP message authentication. Similar to other protocols in the Internet, the IGMPv3 [21] protocol malfunctions in the face of forged or bogus control messages. The IGMPv3 specification provides some examples of specific scenarios (see Section 8.2.4 for an IGMP overview):

- *Query Message.* A forged Query message from a machine with a lower IP address than the current Querier will cause Querier duties to be assigned to the forger. If the forger then sends no more Query messages, other routers' Other-Querier-Present timer will time out, and one will resume the role of Querier. During this time, if the forger ignores Leave Messages, traffic might flow to groups with no members for up to a period of time specified by the Group Membership Interval.
- *Report messages.* A forged Report message may cause multicast routers to think there are members of a group on a network when there are not. Forged Report messages from the local network are meaningless, since joining a group by a host is generally an unprivileged operation, so a local user may trivially gain the same result without forging any messages. Forged Report messages from external sources are more troublesome.

  A forged Version 1 Report message may put a router into ''version 1 members present'' state for a particular group, meaning that the router will ignore Leave messages. This can cause traffic to flow to groups with no members for up to a period of time specified by the Group Membership Interval. This can be solved by providing routers with a configuration switch to ignore Version 1 messages completely. This breaks automatic compatibility with Version 1 hosts, so should only be used in situations where ''fast leave'' is critical.
- *Leave messages.* A forged Leave message will cause the Querier to send out Group-Specific Queries for the group in question. This causes extra processing on each router and on each member of the group, but it cannot cause loss of desired traffic.

Although mutual authentication between a host and a multicast Router (or between any two entities in general) is perhaps best provided using a public key–based approach, often the message volume and the network

conditions render authentication based on symmetric key MACs to be the most economic option. Thus, the host and the multicast router would share (pair wise or group wise) a common symmetric key which is then used to compute a keyed hash of messages exchanged between the host and the multicast router.

In [5] this shared symmetric key is referred to as the IGMP-key. This shared key could be established at the router either through a key negotiation protocol (such as IKE), or a predefined key being delivered from the KD to individual routers. The work of [62] proposed a message format in IGMPv3 that would carry the MAC values and other parameters.

8.6.4 Open issues

At this point it is worthwhile to consider a number of specific open issues regarding authorization via the access token concept and the need for authentication of IGMP messages. As noted in [53] some IGMP messages are in fact broadcasted in the local subnet, having the characteristics of group messages, and thus have the same group authentication issues as multicasted control messages in routing protocols (e.g., PIM-SM in Section 8.4.4).

Some of the open issues are as follows:

- *Signature method on the access-token.* If public key technology is used, this implies that the multicast router must have the capability to verify public key–based signatures and be installed with the public key (and possibly digital certificate) of the signer, namely the KD.

  Alternatively, all multicast routers could be assigned a symmetric key that all share with the KD, and a MAC or HMAC method used to provide group authentication.

  However, as pointed out by [53], the mode of communicating IGMP messages within a subnet is a many-to-many communication, which in turn prevents the use of IPsec (AH or ESP), due to the IPsec SA model for pair-wise communications (see Section 8.4.4).

- *Layer at which authentication occurs.* If public key–based digital signatures are to be used, then the signatures should be applied at the IGMP level instead at the IP packet level. If a MAC and symmetric key approach is to be adopted, then the MAC authentication can be carried out on either level.

  Note that since the access token in [5] is not actually relayed by the host to the router, and may be presented to the router at a later time, IPsec-level authentication will be unsuitable as an option.

- *Token lifetime and management.* When the KD issues an access token, the token must carry a lifetime, including a used-by-date and an activate-by-date, matching the lifetime of the multicast group. Although the lifetime information may not be useful to the routers, it may help the multicast application software on the host to join/leave groups. This brings up the issue of the need to manage tokens, in that old or expired tokens need to be flushed out of the system and domains.

  In [50], within the specific context of PIM-SM deployment, the access token is managed by introducing a list of valid tokens into the domain through the special all-PIM-routers group. The list allows routers in a PIM domain to verify any access token presented to it by a host. Once used, the token's ID is marked off the list of valid tokens.

  In general, this approach, which is reminiscent of certificates and certificate revocation lists, needs further study to see its impact on the routing domain.

8.7 Security in other routing protocols

In this section, we summarize two multicast routing protocols for which security mechanisms have been proposed in the literature. The CBT [63] multicast routing protocol predates PIM-SM, and it is instructive to see how secure routing works in that context. A secure multicast key distribution (SMKD) protocol has been proposed [64] to protect CBT routing messages as well as data.

HIP [3] is a hierarchical multicast routing protocol based on *ordered CBT* (OCBT) for interdomain multicasting, and any multicast protocol for intradomain multicasting. Keyed HIP (KHIP) [65] is an extension to the HIP protocol to control access to the multicast distribution tree. Similar to SMKD, KHIP protects both routing messages as well as multicast data. Both of them support host authentication and authorization, as well as message integrity. In the following, we briefly describe the routing protocol protection mechanisms proposed in these schemes.

8.7.1 Secure CBT multicasting: SMKD

The SMKD protocol's goal is to enforce group access control. The multicast tree can be extended only by authorized hosts and routers. To enforce this, the group initiator creates an ACL that consists of authorized members of the group, and sends it to the the primary core. The initiator may also generate another list of trusted network nodes. The primary core may share these lists with the secondary cores as well.

A host requests to join the group by sending a self-signed token to the edge CBT router. The token consists of the host's identity, lifetime, and a random number to serve as a nonce. For SMKD, IGMP needs to be modified so that members can send the token along with a join request. The edge router, $R1$, that receives a host's request verifies the host's authenticity and message integrity. $R1$ then sends a CBT join request that includes the host's token as well its own token, toward the core. The whole join request is signed by the edge router. Each CBT router, R_j, in the path to the core verifies the authenticity of R_{j-1}'s request, replaces R_{j-1}'s token with its own token, and signs the modified join request. It then forwards the request toward the core.

The core serves as the group key distribution center (GKDC). It verifies the host's token and the last router R_i's token, and sends the ACL as part of a signed join acknowledgment to R_i. The host must be in the ACL, and R_i must be a trusted network node, for positive verification. The router authorization verification process continues downstream. More specifically, R_i verifies whether R_{i-1} is part of the trusted network nodes' list. After successful authorization, all the routers, $R1, R2, \ldots, R_i$, in the path from the host to the core become trusted authorized agents of the GKDC. They have the ACL and can enforce group access control without having to forward future join requests to the core.

In summary, SMKD uses hop-by-hop authentication and message integrity verification upstream (from host to core or an authorized GKDC agent). The GKDC verifies the host's authorization to participate in the group. There is hop-by-hop router authorization verification downstream. Thus only trusted network nodes can become agents of GKDC.

The complete specification of SMKD includes a group key description as well [64]. But the SMKD protocol only supports join request authorization and key distribution. There is no support for excluding an active (say, misbehaving) member, which is a severe limitation. Once a host or a router is authorized to be part of the multicast distribution tree, it remains authorized until the tree is torn down. Delegation of key distribution functionality to trusted routers makes this scheme efficient.

8.7.2 KHIP

KHIP [65] is a secure version of the HIP multicast protocol. KHIP addresses a more comprehensive threat model than SMKD. The goal is to be able to provide multicast services, even in the presence of untrusted routers on the path from authorized senders to authorized members. This protocol considers replay and flooding attacks, as well as illegal branch creation (pulling of the multicast tree), by malicious or misconfigured routers.

KHIP requires authorized members and routers to obtain authorization certificates. Certificates contain the member's ID, groups it is authorized to join, lifetime, timestamp, its permissions (role as initiator, sender, receiver or terminator of a group), and the member's public key. Certificates are signed by an authorization service. Core routers (or the center point [65]) also need authorization certificates.

Only authorized routers are allowed to create branches, and they must present a valid certificate to a trusted core do be able to do so. The core also needs to present its authorization certificate along with the join acknowledgment messages. KHIP uses nonces for replay protection of join messages. It adds two additional messages to the OCBT join protocol, for exchanging the nonces. All four messages for a router to join the HIP tree include the message initiator's certificate, and are signed. Unlike the normal OCBT join request, KHIP join requests go all the way to a trusted core (or even the center point), not the next hop router [65].

KHIP allows some unauthenticated control (flush and quit) messages, so that branch failure detection can be delegated to untrusted routers as well. This ensures prompt response to link and router failures. However, malicious routers may take advantage of this capability to send bogus quit messages upstream, or flush messages downstream.

KHIP proposes periodic branch teardown and reestablishment. This is to ensure the presence of authorized nodes at both end points (router and core) of a branch. It also mitigates bogus quit and flush messages sent by malicious or misconfigured routers. The periodic branch teardown and reestablishment allows the core to enforce stricter access control compared to that in SMKD, described earlier. When a router's certificate expires, it can no longer pull the tree.

In summary, KHIP offers protection against flooding and replay attacks, but cannot handle other types of DoS attacks, such as a malicious router simply dropping packets or sending illegal flush messages. Also note that the intradomain protocol(s) and the unicast protocols associated with KHIP must be secure as well. Similar to SMKD, KHIP supports integrated group key distribution as well [65]. The trusted routers act as subgroup managers for scalable operation.

8.8 Summary

This chapter has focused on multicast routing security as an integral piece of the whole problem of multicast security. Paramount to the correct operation of any routing protocol is the integrity protection and authentication of

control messages that define the behavior of the protocol. In the context of multicast routing, the problem of control message protection is even more important than in unicast, since any attacks on the multicast distribution tree impact a large number of receivers of data in the multicast group(s). The two areas of focus in this chapter were multicast distribution tree protection and membership access control. The first of these two pertains to the core of the distribution tree, while the second pertains to the edges of the tree, where hosts/receivers join the tree.

One undeniable and overwhelming conclusion of this chapter is the necessity for public key cryptography at the routing level, in order to provide source authentication (e.g., digital signatures). This is true despite the reluctance on the part of routing protocol designers and router vendors to implement this technology in their products. Many types of control messages in multicast routing protocols require the source to be identified and require the message to be integrity protected, something which cannot be achieved with symmetric key cryptography alone.

If routing protocol designers and router vendors want to provide advanced functionalities beyond today's manual configuration of routers (and routing domains), then these advanced functionalities imply that some level of intelligence and decision-making capability must exist on the part of routers and other network elements. Thus, sooner or later, some entity will need to perform some decision making for the other entities, in the domain. The result of this decision must then be made known to the other network entities, and must be observed by these entities. This implies that the decision-making entity must be trusted by the others, and that the dissemination of the decision parameters must be source authentic and integrity protected. Otherwise, there is little point in adding intelligence to such network entities if decisions cannot be relayed with integrity and source authenticity. This notion is not new but echoes, to different extents, earlier efforts, such as [24, 66, 67].

The chapter has focused on the PIM-SM protocol as the main example of a multicast routing protocol, for a number of reasons. Firstly, PIM-SM has become the de facto multicast routing protocol in the industry, and there are production sites that are employing PIM-SM. Secondly, PIM-SM is still undergoing revisions—particularly in the context of the new paradigm of SSM. This allows for security-related input into the PIM-SM evolution process. Thirdly, and not least, the revised PIM-SM is an example of an industry-driven protocol that more recently actually takes security issues into its design considerations.

The second half of this chapter dealt with the issue of membership access control, also referred to as the IGMP security problem, since IGMP is the

de facto protocol to perform group membership management between hosts and (next hop) multicast routers. The main message of this section is that membership access control requires both identification/authentication of the host requesting access, and authorization of the host to join multicast groups. The question then becomes how to satisfy these two requirements in a routing domain, particularly when they involve several layers of the communications stack. Approaches to this problem have been discussed, and open issues are listed.

References

[1] Ballardie, A., ''Core Based Trees (CBT Version 2) Multicast Routing,'' RFC 2189 (experimental), IETF, September 1997.

[2] Ballardie, A., ''Core Based Trees (CBT) Multicast Routing Architecture,'' RFC 2201 (experimental), IETF, September 1997.

[3] Shields, C., and J. J. Garcia-Luna-Aceves, ''HIP—A Protocol for Hierarchical Multicast Routing,'' *Computer Communications*, Vol. 23, November 2000, pp. 628–641.

[4] Deering, S. E., ''Host Extensions for IP multicasting,'' RFC 1112 (standard), IETF, August 1989.

[5] Hardjono, T., and B. Cain, Key Establishment for IGMP Authentication in IP Multicast, *Proc. of the 1st IEEE European Conference on Universal Multiservice Networks (ECUMN 2000)*, Colmar, France, October 2000.

[6] Malkin, G., ''RIP Version 2,'' RFC 2453 (standard), IETF, November 1998.

[7] Baker, F., and R. Atkinson, ''RIP-2 MD5 Authentication,'' RFC 2082 (proposed standard), IETF, Jan. 1997.

[8] Moy, J., ''OSPF Version 2,'' RFC 2328 (standard), IETF, April 1998.

[9] Rekhter, Y., and T. Li, ''A Border Gateway Protocol 4 (BGP-4),'' RFC 1771 (standards track), IETF, 1995.

[10] Pusateri, T., ''Distance Vector Multicast Routing Protocol,'' draft-ietf-idmr-dvmrp-v3-10.txt, IETF, August 2000, work in progress.

[11] Waitzman, D., C. Partridge, and S. E. Deering, ''Distance Vector Multicast Routing Protocol,'' RFC 1075 (experimental), IETF, November 1988.

[12] Moy, J., ''Multicast Extensions to OSPF,'' RFC 1584 (proposed standard), IETF, March 1994.

[13] Moy, J., ''MOSPF: Analysis and Experience,'' RFC 1585 (informational), IETF, March 1994.

[14] Adams, A., J. Nicholas, and W. Siadak, ''Protocol Independent Multicast-Dense Mode (PIM-DM): Protocol Specification (Revised),'' draft-ietf-pim-dm-new-v2-01.txt, IETF, February 2002, work in progress.

[15] Fenner, B., et al., ''Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification (Revised),'' draft-ietf-pim-sm-v2-new-05.txt, IETF, March 2002, work in progress.

[16] Huitema, C., *Routing in the Internet*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, January 2000.

[17] Maufer, T.A., *Deploying IP Multicast in the Enterprise*, Upper Saddle River, NJ: Prentice Hall, 1997.

[18] Deering, S. E., and D. R. Cheriton, ''Multicast Routing in Datagram Internetworks and Extended LANs,'' *ACM Trans. on Computer Systems*, Vol. 8, No. 2, May 1990, pp. 85–110.

[19] Perlman, R., *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*, 2nd ed., Reading, MA: Addison-Wesley, October 1999.

[20] Deering, S., et al., ''Protocol Independent Multicast-Sparse Mode: Motivations and Architecture,'' draft-ietf-pim-arch-05.txt, IETF, August 1998, work in progress.

[21] Cain, B., et al., ''Internet Group Management Protocol, Version 3,'' draft-ietf-idmr-igmp-v3-09.txt, IETF, January 2002, work in progress.

[22] Fenner, W., ''Internet Group Management Protocol,'' RFC 2236 (proposed standard), IETF, November 1997.

[23] Holbrook, H., and B. Cain, ''Source Specific Multicast for IP,'' draft-ietf-ssm-arch-00.txt, IETF, November 2001, work in progress.

[24] Perlman, R., Network Layer Protocols with Byzantine Robustness, Ph.D. dissertation, MIT, Dept. of Electrical Engineering and Computer Science, August 1988.

[25] Murphy, S., and M. Badger, ''Digital Signature Protection of the OSPF Routing Protocol,'' *Proc. of Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 1996.

[26] Hauser, R., T. Przygienda, and G. Tsudik, ''Reducing the Cost of Security in Link-State Routing,'' *Proc. of Network and Distributed Systems Security Symposium (NDSS)*, San Diego, CA, Feb. 1997.

[27] Sirois, K. E., and S. T. Kent, ''Securing the Nimrod Routing Architecture,'' *Proc. of Network and Distributed Systems Security Symposium (NDSS)*, San Diego, CA, Feb. 1997.

[28] Smith, B. R., S. Murthy, and J. J. Garcia-Luna-Aceves, ''Securing Distance Vector Routing Protocols,'' *Proc. of Network and Distributed Systems Security Symposium (NDSS)*, San Diego, CA, Feb. 1997.

[29] Zhang, K., ''Efficient Protocols for Signing Routing Messages,'' *Proc. of Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, 1998, pp. 29–35.

[30] Murphy, S., M. Badger, and B. Wellington, ''OSPF with Digital Signatures,'' RFC 2154 (experimental), IETF, June 1997.

[31] Heffernan, A., ''Protection of BGP Sessions via the TCP MD5 Signature Option,'' RFC 2385 (proposed standard), IETF, Aug. 1998.

[32] Kent, S., C. Lynn, and K. Seo, ''Secure Border Gateway Protocol (Secure-BGP),'' *IEEE Journal on Selected Areas in Communications*, Vol. 18, No. 4, April 2000, pp. 582–592.

[33] Murphy, S., ''BGP Security Analysis,'' draft-murphy-bgp-secr-01.txt, IETF, Aug. 1998, work in progress.

[34] Villamizar, C., et al., ''Routing Policy System Security,'' draft-ietf-rps-auth-01.txt, IETF, May 1998, work in progress.

[35] Bates, T., et al., ''DNS-Based NLRI Origin AS Verification in BGP,'' draft-bates-bgp4-nlri-orig-verif-00.txt, IETF, February 1998, work in progress.

[36] Paul, S., *Multicasting on the Internet and its Applications*, Norwell, MA: Kluwer Academic Press, 1998.

[37] Wei, L., ''Authenticating PIM Version 2 Messages,'' draft-ietf-pim-v2-auth-01.txt, IETF, May 1999, work in progress.

[38] Hardjono, T., M. Baugher, and H. Harney, ''Group Security Association (GSA) Management in IP Multicast,'' *Proc. of the 16th International Conference on Information Security (IFIP/SEC)*, Paris, France, June 2001.

[39] Hardjono, T., M. Baugher, and H. Harney, ''Group Key Management for IP Multicast: Model and Architecture,'' in *IEEE 10th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE 2001)*, MIT, Cambridge, MA, June 2001.

[40] Harney, H., M. Baugher, and T. Hardjono, ''GKM Building Block: Group Security Association (GSA) Definition,'' draft-irtf-smug-gkmbb-gsadef-01.txt, IETF, September 2000, work in progress.

[41] Baugher, M., et al., ''Group Domain of Interpretation for ISAKMP,'' draft-ietf-msec-gdoi-04.txt, IRTF, March 2002, work in progress.

[42] Van Moffaert, A., and O. Paridaens, ''Security Issues in Protocol Independent Multicast-Sparse Mode (PIM-SM),'' draft-irtf-gsec-pim-sm-security-issues-01.txt, IRTF, February 2002, work in progress.

[43] Hardjono, T., and B. Cain, ''Simple Key Management Protocol for PIM,'' draft-ietf-pim-simplekmp-01 txt, IETF, February 2000, work in progress.

[44] Kent, S., and R. Atkinson, ''IP Authentication Header (AH),'' RFC 2402 (proposed standard), IETF, November 1998.

[45] Madson, C., and R. Glenn, ''The Use of HMAC-MD5-96 within ESP and AH,'' RFC 2403 (proposed standard), IETF, November 1998.

[46] Rivest, R., ''The MD5 Message-Digest Algorithm,'' RFC 1321 (informational), IETF, April 1992.

[47] Madson, C., and R. Glenn, ''The Use of HMAC-SHA-1-96 within ESP and AH,'' RFC 2404 (proposed standard), IETF, November 1998.

[48] *PKCS1: RSA Encryption Standard*, RSA Laboratories,1993.

[49] Hardjono, T., B. Cain, and I. Monga, ''Intra-Domain Group Key Management Protocol,'' draft-ietf-ipsec-intragkm-02.txt, IETF, February 2000, work in progress.

[50] Hardjono, T., ''Router-Assistance for Receiver Access Control in PIM-SM,'' in *Proc. of IEEE International Symposium on Computer Communications (ISCC)*, Antibes, France, July 2000.

[51] Kent, S., and R. Atkinson, ''Security Architecture for the Internet Protocol,'' RFC 2401 (proposed standard), IETF, November 1998.

[52] Monga, I., and T. Hardjono, ''Group Security Association (GSA) Definition for IP Multicast,'' draft-irtf-smug-gsadef-00.txt, IETF, February 1999, work in progress.

[53] Van Moffaert, A., and O. Paridaens, ''Security Issues in Internet Group Management Protocol Version 3 (IGMPv3),'' draft-irtf-gsec-igmpv3-security-issues-01.txt, IRTF, February 2002, work in progress.

[54] Farinacci,D., et al., ''Multicast Source Discovery Protocol (MSDP),'' draft-ietf-msdp-spec-03.txt, IETF, January 2000, work in progress.

[55] Meyer, D., and B. Fenner, ''Multicast Source Discovery Protocol (MSDP),'' draft-ietf-msdp-spec-13.txt, IETF, November 2001, work in progress.

[56] Hardjono, T., and B. Cain, ''PIM-SM Security: Interdomain Issues and Solutions,'' in Preneel, Bart (ed.), *Communications and Multimedia Security (CMS'99)*, Leuven, Belgium: Kluwer, September 1999.

[57] Ishikawa, N., N. Yamanouchi, and O. Takahashi, ''IGMP Extensions for Authentication of IP Multicast Senders and Receivers,'' draft-ishikawa-igmp-auth-01.txt, IETF, August 1998, work in progress.

[58] He, H., T. Hardjono, and B. Cain, ''Simple Multicast Receiver Access Control,'' draft-irtf-gsec-smrac-00.txt, IRTF, November 2001, work in progress.

[59] Judge, P., and M. Ammar, ''Gothic: A Group Access Control Architecture for Secure Multicast and Anycast,'' in *Proc. of IEEE INFOCOM*, New York, June 2002.

[60] Coan, B., et al., ''HASM: Hierarchical Application-Level Secure Multicast,'' draft-coan-hasm-00.txt, IRTF, November 2001, work in progress.

[61] Castelluccia, C., and G. Montenegro, ''Securing Group Management in IPv6 with Cryptographically Generated Addresses,'' draft-irtf-gsec-sgmv6-00.txt, IRTF, February 2001, work in progress.

[62] He, H., B. Cain, and T. Hardjono, ''Upload Authentication Information Using IGMPv3,'' draft-he-magma-igmpv3-auth-00.txt, IRTF, November 2001, work in progress.

[63] Ballardie, T., P. Francis, and J. Crowcroft, ''Core Based Trees (CBT): An Architecture for Scalable Inter-domain Multicast Routing,'' in *Proc. of ACM SIGCOMM*, Ithaca, NY, September 1993.

[64] Ballardie, A., ''Scalable Multicast Key Distribution,'' RFC 1949 (experimental), IETF, May 1996.

[65] Shields, C., and J. J. Garcia-Luna-Aceves, ''KHIP—A Scalable Protocol for Secure Multicast Routing,'' in *Proc. of ACM SIGCOMM*, Cambridge, MA, September 1999.

[66] Gong, L., and N. Shacham, ''Elements of Trusted Multicasting,'' in *Proc. IEEE International Conference on Network Protocols*, Boston, MA, October 1994, pp. 23–30.

[67] Ballardie, T., and J. Crowcroft, ''Multicast-Specific Security Threats and Counter-measures,'' in *Proc. of Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 1995, pp. 2–16.

CHAPTER

9

Security in Reliable Multicast protocols

Contents

9.1 Classification of RM protocols

9.2 Generic security requirements for RM protocols

9.3 Security of TRACK protocols

9.4 Security of NORM protocols

9.5 Security of FEC-based protocols

9.6 Summary

Multicast routing protocols typically provide best-effort delivery of data packets from senders to receivers. The ''best-effort'' delivery means just that: No guarantee is given as to the actual delivery of data packets, and no ordering is preserved for packets belonging to a message. This model of IP multicast routing is consistent with unicast routing, and is founded on the same fundamental notion of the connectionless-oriented network underlying the Internet.

Since applications that employ IP multicast require reliability of delivery, the function of reliability must be provided above the multicast routing level. Thus, analogous to the relationship between IP and TCP, a Reliable Multicast (RM) protocol provides the reliability needed for the data packet delivery by the underlying multicast routing protocol.

Since reliable group-oriented communications is by nature more complex than reliable pair-wise communications between two end points, it is hardly surprising that RM protocols are more complex than their unicast counterparts. Furthermore, the Reliable Multicast problem is compounded by the fact that each application has its specific needs, and thus a single general-purpose RM protocol would be impractical, if not impossible, to construct.

The specific needs of each multicast application have resulted in a large number of proposed RM protocols. These, in turn, have their own security requirements and issues. Thus, a

general-purpose security solution for all RM protocols would be impossible to construct due to the sheer size of the design space.

Principle of separation

Similar to the principle of independence between multicast key management (for data protection) and multicast routing [1] which was discussed in Chapter 5, here we would like to put forward an analogous principle. The principle is that of the separation between the security for Reliable Multicast transport and the security for multicast routing.

Obvious as this principle may appear to be, it is worth explaining further. What this principle means is that the instances of security protocols and their application to Reliable Multicast at the transport layer should be independent from the instances of those same security protocols (if they happen to be the same), applied to multicast routing at the network layer. Thus, for example, if an RM protocol instantiation uses a *group key management* (GKM) protocol for key distribution (e.g., for authentication keys) and the multicast routing protocol also happens to rely on the same GKM protocol (and the same GKM entities), then the two instances of the GKM protocol must be independent of one another. Hence, from the perspective of the security-related entities, they are dealing with independent instances of the application of security. An example would be IPsec, which can be used by both the multicast routing protocol and by the RM protocol, even though the same entities may be involved (e.g., the IP layer router that acts as a suppression node for an RM protocol).

Note that this principle of separation is different from (and in fact does not contradict) the basic tenet believed by many in the security community, that the security of the whole is not equivalent to the ''sum'' or composition of the security of the pieces or components (see, for example, [2, 3]). Rather, the principle of separation is driven from the need for security to be manageable, and for the parameters of security to be clearly identifiable and controllable; something which in fact is necessary from the practical perspective of defining and enforcing security.

Cryptographic protection versus heuristics

Although the current authors believe that authentication of control messages is necessary for the correct and robust execution of protocols, there is an alternative view that does not see the need for full-use cryptographic protection for control messages. Instead, this latter opinion suggests that a good RM protocol should be inherently resistant to bogus

control messages. Furthermore, this view holds that, in practice, the frequency of bogus control messages would be so low that it would not impact the RM protocol substantially (so as to warrant cryptographic protection).

By this line of reasoning, a case of suspicious or anomalous behavior of certain receivers and RM entities would be detected and reacted to by the congestion control mechanism and the throughput mechanisms as a whole. Thus, for example, if a bogus NACK was sent by a misbehaving (or compromised) receiver, then it would initially trigger a retransmission of the concerned packet. However, if too many NACKs were sent by that same entity or by a group of entities from the same topological vicinity, then the upstream router would have built-in heuristics and knowledge to throttle the outgoing interface(s) toward that vicinity. It remains unclear whether this heuristic capability will be an inherent part of the congestion control algorithm, or a separate algorithm to counter detected inconsistent information in bogus control messages. Also, the amount of resources consumed by the heuristic algorithm(s) to react to perceived attacks remains to be seen.

We argue that although some RM protocols may provide sufficient innate capabilities to counter bogus control messages, there is some virtue in addressing the use of cryptographic protection on control (and data) messages, and in investigating the mechanics of how this would be achieved in practice. This is the purpose of the current chapter.

In the next section we discuss the classification of RM protocols as defined by [4]. Since there are quite a number of RM protocols that have been proposed—too many to be described here in detail—the reader is directed to [4] for a brief classification of these protocols. In addition, the works of [5–7] provide further details on the various RM protocols. This is followed in Section 9.2 by a discussion on the generic security requirements of all RM protocols. The remainder of the chapter then looks briefly into the specific security requirements and possible solutions for the TRACK, NORM, and FEC families of RM protocols.

9.1 Classification of RM protocols

The problem of providing reliability to IP multicast has been investigated for a number years now, resulting in several RM protocols. An RM protocol is intended to be deployed in conjunction with a multicast routing protocol that creates the multicast distribution tree. Indeed, some RM protocols make use of certain aspects of the distribution tree, such as its topology. Some of

these protocols have remained in the domain of research, while others are in production and are being deployed in a commercial environment.

One broad approach to differentiating RM protocols is to see them as being either *sender-initiated* or *receiver-initiated*. In the first case, the sender or source has the responsibility of detecting lost packets, while in the second case this is the task of the receiver. Usually, sender-initiated protocols use *positive acknowledgments* (ACKs), while receiver-initiated approaches use *negative acknowledgments*. These two basic approaches have their respective drawbacks, either in terms of an explosion in the amount of state information needed to be maintained by the sender and other entities, or the number of messages that may overwhelm the sender and the network as a whole. Thus, *state explosion* and *message implosion* are two common drawbacks in RM approaches, which are typically solved by introducing a hierarchy of *local repair nodes*, and/or by using *suppression techniques*. With local repair nodes, the sender need not deal with lost packets, and they reduce the message implosion at the sender. Local repair nodes also allow the possibility of various entities performing the repair function, including specialized servers or other group members. Suppression techniques (e.g., for NACKs) reduce both implosion and state, at the sender.

In the context of the IETF, the problem area of RM has been addressed under the IRTF within the Reliable Multicast Research Group (RMRG) for several years. In early 2000 a corresponding IETF working group was created, called the RMT Working Group. One of the tasks of the RMT Working Group is to standardize certain components or elements (building blocks) that are common across multiple RM protocols. Examples would be congestion control and good throughput mechanisms. To this end, the RMT Working Group provided a definition of the design space for the RM protocols [4] and attempted to categorize the protocols (or their components) according to some aspects.

Two of these are discussed below, and are used to provide a framework of discussion for the various security issues and solutions in subsequent sections.

9.1.1 Good throughput strategies

The work of [4] identifies two important aspects of RM protocols, namely, congestion control and good throughput. These two aspects are like two sides of the same coin, in the sense that packet loss is symptomatic of congestion in the network, which can be alleviated through good throughput mechanisms. Hence, one way to categorize RM protocols is to group them according to the good throughput strategy that they use:

- *ACK-based strategy*. Here the basic idea is that a positive acknowledgment (known simply as ACK) is sent by the receiver for every received data packet. Packets that are lost could be resent by the sender or by any receiver that has obtained it. In essence, congestion control occurs at the sender. This pure model is impractical, since it suffers from the problem of implosion of ACKs at the sender.

  An improvement over the pure model can be achieved by imposing a structure on the paths taken by the ACK messages, and to provide intermediate entities that can process them. An example is the hierarchic or tree structure used to arrange the receivers into groups or subgroups. Here, a receiver would send ACKs to a designated parent node, instead of to the sender. A parent node acts as an aggregator of ACKs to its own parent node, and so on. The nodes in the ACK tree are typically used only for ACK aggregation, and not for data packet retransmission. The cost of this improvement is the tree-building process, which can range from being dynamic (e.g., Tree-Based Reliable Multicast or TRAM [8]) to statically configured (e.g., Reliable Multicast Transport Protocol, RMTP-II [9]). Other proposed structures include a ring formation in which an ACK token is passed around [10], while other suggestions on the basic ACK model include placing multiple ACKs inside a data packet [11].

- *NACK-based strategy*. In a NACK-based strategy, rather than a receiver or node sending an ACK for the packets it has received, the receiver or node instead sends a NACK for those packets it did *not* receive. The general aim is two fold:

  - *One NACK*. Only a single NACK (for a data packet) must reach the sender (or retransmission entity) as soon as the packet loss is noticed by the receivers.

  - *One retransmit copy*. Only one copy of the retransmitted data packet must be received by the nodes or receivers that experience the packet loss.

  Since only a single NACK is needed, a method called *NACK suppression* is employed to discard NACKs for the same data packet. An example of this approach is the Scalable Reliable Multicast (SRM) protocol [12] and the pragmatic general multicast (PGM) protocol [13].

- *Packet-level redundancy strategy*. Another strategy is to add redundancy within each packet (or within separate accompanying packets) in such a way that a receiver needs only to obtain a certain subset of the packets in order to reconstruct the complete message. Thus, the sender must create *encoding packets* for each *round* (or collection) of data packets. The analogy is that with FEC approaches that are used to recover from corruption of data, but in this case it is performed at the unit of packets.

  Solutions based on FEC can be *proactive* or *reactive* (or a combination of both). In the first case, the sender decides (before sending the packets) how many encoding packets per round need to be sent. In the reactive case, the sender starts by sending only the original data packets, and reacts when it learns of packet losses (e.g., through ACKs or NACKs). After the sender learns that some packets are missing, it then computes the encoding packets that are calculated for the worst receiver: meaning the receiver is experiencing the worst loss. An example of a protocol that employs packet-level redundancy is [14].

9.1.2 Network entity participation and support

Another possible way of classifying RM protocols is to view them from the perspective of the network entities that participate in the protocol. The work of [4] recognizes four types of possible support that these network entities may provide to create reliability:

1. *Router-assisted approaches*. In this category of RM protocols, routers that are part of the multicast distribution tree help in providing reliability to the data. The thinking in these designs is that since the routers are already participating in the normal data distribution tree, they are in a good position to assist in the delivery of missing packets, in the aggregation of feedbacks, and in the suppression of certain feedback types (e.g., multiple NACKs). Since routers typically do not have large amounts of spare memory or central processing unit (CPU) power, the amount of functionality that can be added to routers is limited. The PGM protocol [13] is an example of this category.

2. *Server assisted approaches*. In these approaches, the RM protocol relies on the use of (nonrouter) network entities to help in the data delivery to recipients, or in the aggregation of feedback coming from

recipients. These entities are typically not senders/receivers in the multicast distribution tree, and they are not defined as part of the multicast routing protocol. Thus, they are present on the multicast distribution tree solely to provide support for the RM protocol. As such, the term ''server'' is often used, since these entities must have capabilities beyond the usual router. An example is the RMTP-II [9] protocol.

- *Layered approaches*. In layered approaches, several multicast groups are used concurrently to deliver different sets of data packets, possibly at differing speeds. Redundancy is added to the complete set of packets to allow a receiver to establish the complete message just from a minimal of N out of M pieces (where $N \leq M$). By sending different sets over different multicast groups, the receiver has an increased chance of obtaining the sufficient amount of packets to reconstruct the entire message.
- *Approaches without assistance*. In this category of RM protocols, only the senders and receivers are involved in the RM protocol. Thus, only they maintain any state information pertaining to reliability. An example of this type of RM protocol is SRM [12].

9.2 Generic security requirements for RM protocols

Similar to multicast routing protocols (and other communications protocols), RM protocols also define their respective set of control messages to inform different entities in the system about the state of the system as a whole. Indeed, for this purpose some RM protocols even establish a logical tree-like structure that parallels the multicast distribution tree at the routing level. To ensure the correct functioning of an RM protocol according to its specifications, there are a number of basic security requirements that need to be fulfilled, some of which mirror the requirements listed in Chapter 8.

Basic protection requirements

Assuming data/content protection is a separate issue, the broad requirement in Reliable Multicast protocols is for the protection of all control messages. More specifically, the term ''protection'' means the following functions:

- *Authentication*, either group authentication or source authentication;
- *Integrity protection*, against illegal modifications;

- *Replay protection*, against the replay of valid control messages or other bogus messages.

The mechanisms to achieve these would depend on the specific RM protocol.

Layer of protection application

The issue of the layer at which cryptographic protection is applied comes to the foreground in any discussion regarding security of Reliable Multicast. Although each RM protocol instantiation will have its specific needs, it is useful in discussions to provide a distinction among packet-level protection, transport-level protection, and application-level protection. This is because certain techniques may only be applicable to certain layers of the IP stack, and those techniques may be unsuitable for certain RM protocols. Thus, for example, source authentication could be best applied at the application layer for cases where the RM protocol uses the sender as the sole retransmission entity. In FEC-based protocols, integrity protection of data could be applied at the application layer, to allow the receiver to detect inconsistencies due to a bogus packet.

Note that, in practice, multiple protection methods may need to be deployed at different layers. This is in order to solve the specific needs of the multicast application that is relying on a specific RM protocol, which is in turn using a given multicast routing protocol, all the while taking into consideration the principle of separation in security.

Specific requirements of Reliable Multicast

More RM-specific security requirements based on the above protection requirements are as follows:

- Protection of the *structure-building* phase and the operational phase of RM protocols (some protocols distinguish between the two, since they employ a procedure to create state at the entities, which is then used to arrange the flow and delivery of ACKs, NACKs, and other messages);
- Protection of ACKs, NACKs or other protocol-specific status-reporting messages sent by receivers in the multicast group;
- Protection of aggregated ACKs, NACKs or other protocol-specific messages sent by aggregators or suppression nodes;

- Protection of retransmitted data packets, whether they come from the sender or other retransmission (repair) entities, via unicast, group multicast, or subgroup multicast;
- Distinguishability by a receiver of an original (not previously sent) data packet from a retransmitted data packet, independent of the actual entity that sent it;
- Detection of anomalous behavior in both the data distribution tree and the RM structure (e.g., tree) by both receivers and RM entities;
- An emergency procedure in the face of suspected DoS attacks, which could possibly include a graceful degradation in the quality of service delivered to the application employing multicast;
- Management of cryptographic keys and other keying material (e.g., SAs) used to provide protection to ACKs, NACKs or other protocol-specific status-reporting messages.

9.3 Security of TRACK protocols

The TRACK family of RM protocols employs a tree structure to organize the delivery of ACKs and NACKs [15]. All receivers are arranged into local regions, where each region is assigned a repair entity (or repair node) to aid in providing reliability [15]. The tree structure essentially acts as the *control channel* over the data channel (namely, the multicast distribution tree).

These groups are arranged hierarchically in the form of a tree rooted at a sender, with the repair entity representing the interior nodes of the tree, and the receivers as the leaf nodes. Each receiver is assigned (through a protocol-specific mechanism) a repair entity in the region in which the receiver resides. Thus, the repair entity can be said to be the parent node of the receiver. Periodically, receivers send control messages (ACKs or NACKs) to their parent repair node, selectively acknowledging the packets they have received, and requesting the ones they have missed. Retransmission is then performed by that parent node.

For scalability, there might be multiple layers of repair entities, at the top of which is the sender who needs to obtain the ACKs or NACKs pertaining to some data packet. Hence, each repair node sends its control messages to the repair node at the next level up the hierarchy. This process is repeated until the messages reach the sender, informing it of the status of the group, and notifying it when it is allowed to advance its transmission window. Here, in the case of ACKs, the repair nodes effectively aggregate the selective positive

acknowledgments from the receivers. In addition, in the case of NACKs, the repair nodes perform the suppression of NACKs, thereby removing the NACK implosion problem at the sender [15].

A repair node maintains a local multicast group consisting of only its children and, in turn, it subscribes to the local multicast group of its parent. A repair node uses this local multicast group for retransmissions to its children, which also provides suppression of other negative retransmission requests for that packet at other children. In order to provide reliability of data packets, the repair node of a region must know the groups that its children has joined, and must also obtain data packets from those groups. Thus, the repair node of a region must in fact join every multicast group that its descendants have joined [15].

9.3.1 Model of TRACK

Figure 9.1 attempts to illustrate a general model of Reliable Multicast based on ACKs and NACKs, viewing it from the network entities that participate in the protocol. In Figure 9.1, a multicast distribution tree is shown with a number of receivers (R) at the leaves of the tree. A number of network entities are given the task of assisting in the management of the ACK messages, denoted as the *designated node* (D) and the *top node* (T).

Each designated node is charged to cater for a number of receivers, who send their ACKs to the designated node. The designated nodes, in turn, are arranged hierarchically in a tree structure where the upper-level nodes receive (aggregated) ACKs from the lower-level nodes. The top node is the entity at the root of the tree that provides feedback to the sender.

Figure 9.1 attempts to provide an abstraction of the various TRACK protocols, notably RMTP-II and TRAM. Thus, in terms of specific protocols, it is quite possible that the designated nodes are implemented as hosts, routers or servers depending on the protocol. Similarly, the top node could be implemented using the same entity as the sender, or it could be a special retransmission server. Hopefully, the model provides a framework by which one can identify and discuss security issues and requirements to be fulfilled by (secure) RM protocols that exhibit properties like the model, leaving details to the specifics of each RM protocol.

9.3.2 RMTP-II

The RMTP-II [9] protocol uses a static approach to tree building. The tree structure is established through the configuration of a number of *designated receiver nodes* (DR), without any dynamic election process. RMTP-II places

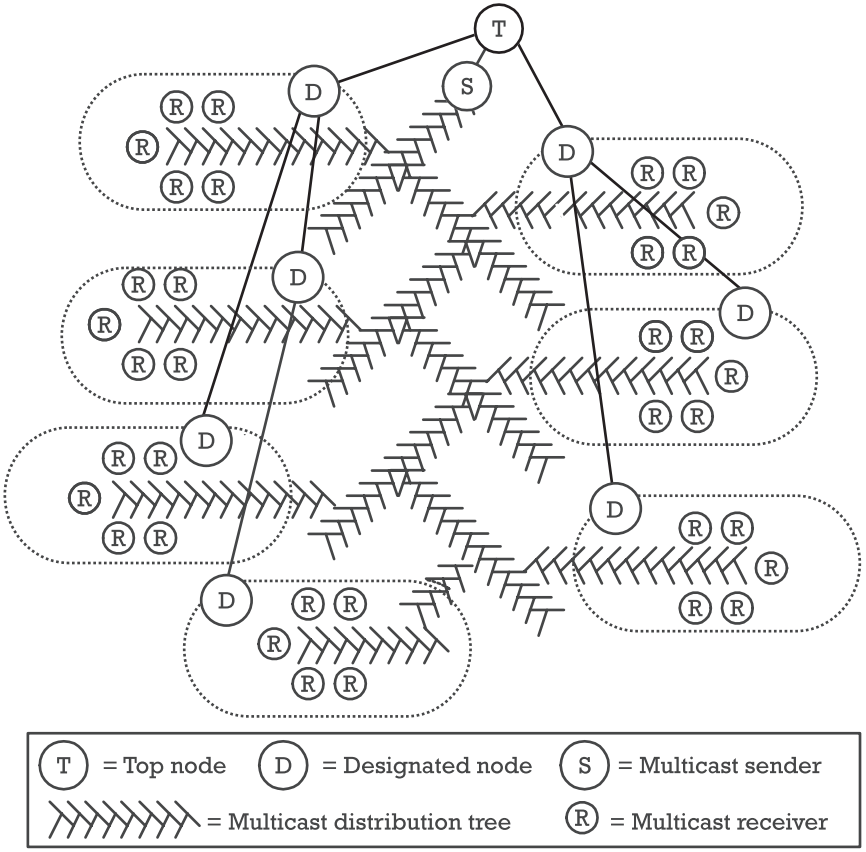


Figure 9.1 General model of TRACK protocols.

receivers into local regions, where each region is assigned to a DR. These groups are arranged hierarchically as a tree rooted at the *top node* (TN), with the DRs representing the nodes of the tree, and the receivers as the leaves. Each *receiver node* (RN) is assigned to a local designated receiver nodes. The sender node connects directly to the TN, at the top layer of the hierarchy.

The receivers send their ACKs to the DR of their region. Retransmission of lost packets may be done locally by the DR or globally from the sender. Each DR sends its control messages to the DR at the next level up the hierarchy. This process is repeated until the TN, which then forwards the control messages to the appropriate sender. The DRs and TN aggregate the positive acknowledgments from the receivers, and suppress the redundant NACKs. A DR maintains a local multicast group to just its children, and subscribes to the local multicast group of its parent. For

retransmissions to its children, a DR can use either unicast or this local multicast group. Figure 9.2 illustrates an RMTP-II tree.

Additional RMTP-II security requirements

RMTP-II has the same broad security requirements as the generic requirements mentioned in Section 9.2. However, in addition, RMTP-II also has a specific requirement, namely, the *optional data access* by RMTP-II nodes [15]. Another way to express this is that the DRs and TN must have nondecipherability of data packets as an option.

Recall that RMTP-II requires that a DR join all of the multicast groups that its descendants have joined. Recall also that the sender (content provider/distributor) may use encryption to provide content access control. Requiring a DR to join data multicast groups for the purpose of caching and retransmissions has implications when RMTP-II is deployed in some

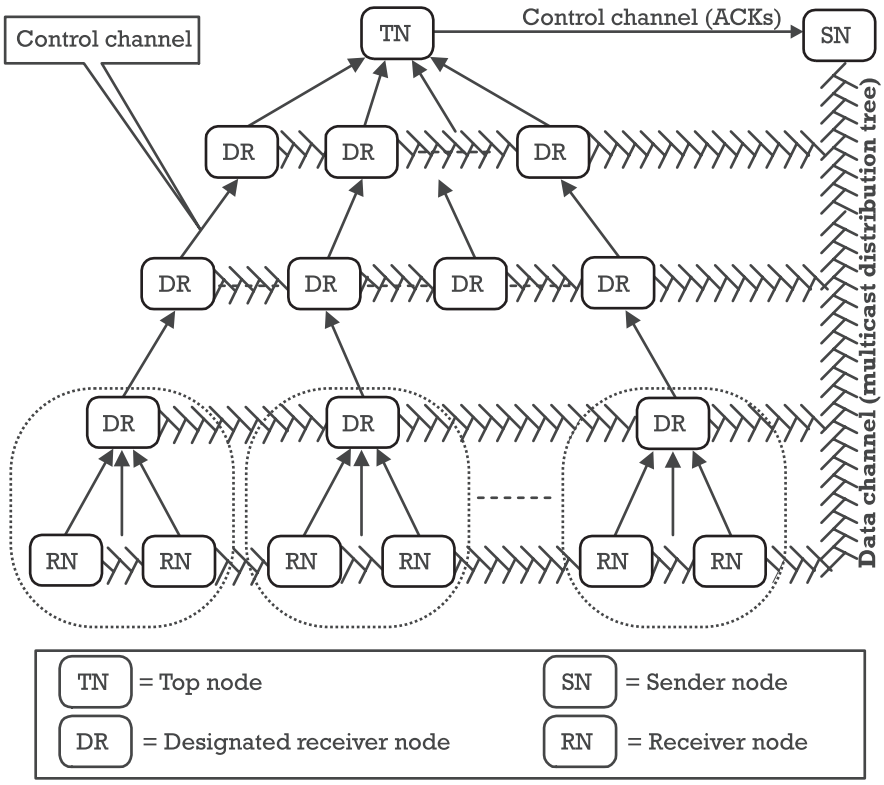


Figure 9.2 RMTP-II tree structure.

commercial environments. More specifically, DRs may be administered as part of a reliability service offered by third parties such as ISPs. These third parties may refuse the ability to decipher data packets, in order to avoid the liabilities of having access to the data contents. Thus, from their perspective, RMTP-II must allow them to prove to the content provider/distributor that they do not possess the means to read or alter the contents transmitted through the multicast groups.

In concrete terms, this requirement translates to the specific exclusion of all DRs from the process of obtaining the data encryption/decryption key through the GKM protocol. If a single GKM protocol instance is used to manage keys for data packet encryption and keys for RMTP-II control message protection, then the GKM protocol must be able to distinguish between senders/receivers and RMTP-II entities. In the ISP scenario, it is quite likely that the senders/receivers would need to obtain the data encryption/decryption key from the content distributor, and obtain the RMTP-II–related keys from the ISP.

Possible RMTP-II key arrangement

One way to satisfy the requirement of optional data access by RMTP-II nodes is to separate data encryption from control message authentication [15]. The idea is that separate cryptographic keys are used for achieving confidentiality and authentication at different layers of the communications stack (Figure 9.3). To achieve this, the work of [15] introduces three symmetric keys, namely, the *Group-Key*, the *RM-Key*, and the *Region-Key*. Although these keys could instead be public keys, the work of [15] focuses on the case where they are symmetric keys, therefore affording only group authentication using symmetric key–based approaches (e.g., MAC). Their usage is described in the following.

A receiver at the leaves of the RMTP-II tree must be given a copy of the RM-Key and the Group-Key in order for the receiver to be able to authenticate and decrypt the data packets.

A sender must be aware of a deployment of the protocol where the RMTP-II nodes are forbidden (or refuse) to have the ability to decrypt data. This means that the nodes will not have a copy of the Group-Key.

When a message is to be sent to the multicast group, the source must first encrypt the message at the application layer using the Group-Key. If needed, the source may also apply source authentication at the application layer.

The encrypted message is then passed down the communications stack (at the source) to the RM layer, where the necessary RM-related headers/

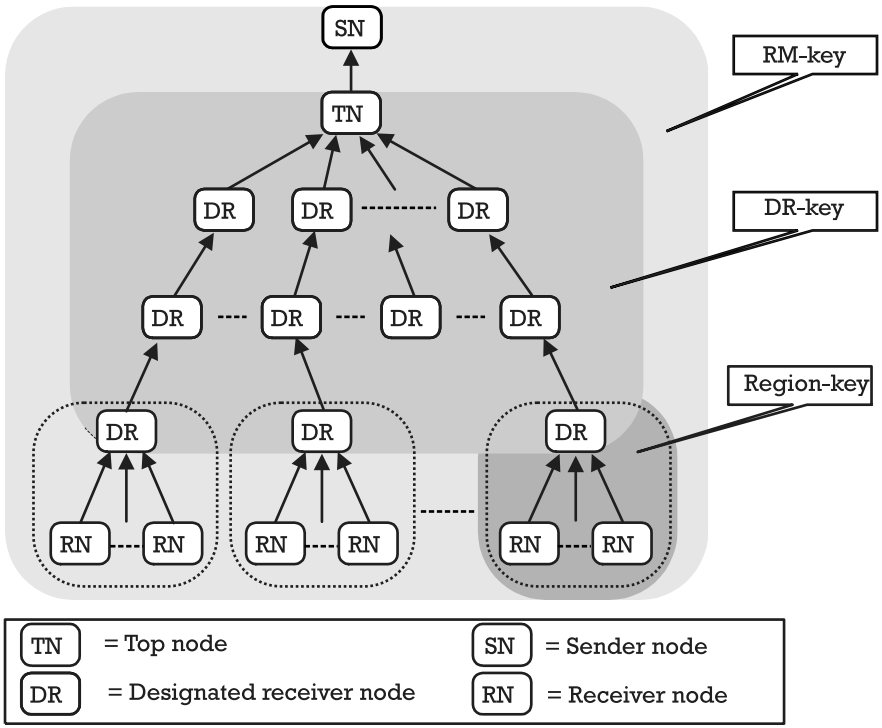


Figure 9.3 RMTP-II key arrangement.

trailers and other parameters are added to the message. At this point, group authentication using the RM-Key is applied at the RM layer.

In order to satisfy the requirement of optional data access by RMTP-II nodes, a DR node will possess a copy of the RM-Key, but not the Group-Key. This means that the DR will only be able to group-authenticate a packet, but not decrypt it.

When a DR has to retransmit a (encrypted) packet, it will strip away the authentication information and parameters set by the sender, and apply its own authentication using its Region-Key. The Region-Key is a unique symmetric key, shared only by receivers within a given RMTP region. This retransmission is either via unicast or via subgroup multicast. Thus, the Region-Key is in effect a ''subgroup'' key. Depending on the exact structure of the RMTP hierarchy, other parent (or ancestor) DRs may also be given a copy of a Region-Key, to provide further reliability in the case when a DR crashes or becomes unavailable.

Finally, in order to protect the RMTP-II tree itself, another symmetric key, called the DR-Key, is used by the RMTP-II entities. The DR-Key is

a symmetric key shared by all DRs and the TN within a given RMTP hierarchy. The key is independent of any data stream, and is used to authenticate control packets exchanged among the DRs/TN. Should a child-DR request a retransmission of a lost data packet from its parent-DR, then the parent-DR could unicast the (encrypted) packet to the child-DR, authenticated using the DR-Key. Before the child-DR retransmits this data packet to its own region, it must first authenticate the packet from the parent-DR using the DR-Key. It would then apply authentication using its Region-Key, before subgroup-multicasting it to its own children receivers.

9.3.3 TRAM

The TRAM protocol is another example of the TRACK class of RM protocols. In TRAM, each receiver belongs to one of many *repair groups*, which are dynamically created from a local subset of receivers, through an election process. Within a repair group, one receiver becomes the *group head*, while the remainder function as members of the group. With the exception of the sender, each group head must also become a member of some other repair group, thereby interconnecting the repair groups. Once having a group head, the receivers can then report lost and successfully received packets to the group head using a selective acknowledgment mechanism [8]. Since repair groups are local, a group head would be located close to its members, reducing or obviating latency.

One important aspect of TRAM is the construction of the *repair tree*, which is done top down. The sender initiates the tree construction process by multicasting a beacon message or data to the multicast group. Before being able to attach to the tree, a receiver must discover or find a repair head that can take on the receiver as a member of a repair group. This is done by the member advertising the fact that it is seeking a repair head. Similarly, a repair head may advertise that it is seeking members. The advertisement is sent via multicast, using an expanding-ring-search, with some scope control imposed over the message. A receiver uses the shortest TTL distance as the basis for deciding the closest repair head with which it will affiliate itself. On its part, a repair head must cache all the received data messages, perform repairs, and process acknowledgments from members in its repair group.

Additional TRAM security requirements

From the security perspective, TRAM has the same broad security requirements as those listed in Section 9.2. In addition, some of the other TRAM-specific requirements are as follows:

- Beacon messages sent out by the sender must be provided with source authentication to provide assurance to receivers that they came from the true sender.
- Multicasted advertisements, either from a head-seeking member or from a receiver seeking a head, must be authentic. This brings up the issue of whether source authentication or group-authentication should be provided. Furthermore, since the advertisements use expanding-ring-search, the authentication method must take into consideration this TTL-based ring search.

TRAM distinguishes among receivers by designating them as Eager-Head receivers, Reluctant-Head receivers or Member-Only receivers. This opens the possibility of limited deployment of public keys, where only Eager-Head members are assigned a public key pair since they are the most likely to become the head of a repair group. Copies of the public keys of Eager-Head members could then be distributed based on the topological distribution of the Eager-Heads, since other receivers will likely be affiliated with the same subset of Eager-Heads (i.e., those that are nearby). The possibility of using public key cryptography in TRAM, and the protection of the tree-building phase of TRAM, need to be investigated further. The reader is directed to [8, 16] for further details on the tree-building process in TRAM, and the other control messages used by TRAM, for tree management.

9.4 Security of NORM protocols

Instead of using positive ACKs to indicate to a sender or a retransmission entity the successful delivery of a given data packet, NORM protocols employ NACKs to indicate packets that were not successfully received [17]. There is an inherent difference in philosophy between ACK-based and NACK-based approaches: positive ACKs essentially determine the transmit buffer management strategy, while NACKs—which should be fewer in number—essentially determine repair and reliability.

The NACK-based strategy has a number of advantages. One advantage is that a single NACK can benefit multiple receivers. That is, only one NACK has to be transmitted by a receiver in order to effect a retransmission of a (missing) packet that benefits multiple receivers. To prevent multiple NACKs (for the same packet) from overwhelming the source (or retransmission node), *NACK-suppression* entities are usually used by NORM protocols. These entities simply discard superfluous NACKs.

NORM protocols have two nice scalability features [4, 18]:

- The source does not need to know which receivers are missing any given data packet: hence it does not need to keep track of the group membership.
- The first NACK sent by a receiver for a given missing packet can suppress further NACKs for the same packet sent by other receivers.

Thanks to the first feature, in the ideal case, memory and processing requirements at the source are independent of the number of receivers. Thanks to the second feature, the volume of feedback traffic is (almost) constant, independent of the receiver population size.

9.4.1 Model of NORM

Figure 9.4 attempts to show a basic model of NORM protocols. A set of suppression nodes (SNs) are employed to discard superfluous NACKs, and are placed at strategic locations in the multicast distribution tree.

When a receiver sends a NACK toward the sender (or a known retransmission entity), the appropriate suppression node will intercept the NACK, and verify that it is not another NACK for the same data packet.

Although not shown, depending on the particular NORM protocol, the suppression node may in fact also be the retransmission entity. Alternatively, the retransmission entity could be associated with one (or more) suppression nodes, and will only retransmit packets upon obtaining instructions to do so from the suppression node.

Security requirements of NORM protocols

In general, NORM protocols would need to satisfy the security requirements mentioned earlier in Section 9.2. One immediate concern is for the authenticity of the NACK messages being sent by receivers toward the sender (or other retransmission entities) upstream. The NACK messages must be protected against modification of parts of their payload, and overall protection must be provided against bogus NACKs being injected into the routing path of the control messages.

Authentication of NACK messages can be provided through public key digital signatures (source authentication) or through symmetric key cryptography (group authentication). With public key–based authentication, each receiver would be assigned a public key pair, and a suppression node would need to know the public keys (certificates) of all receivers

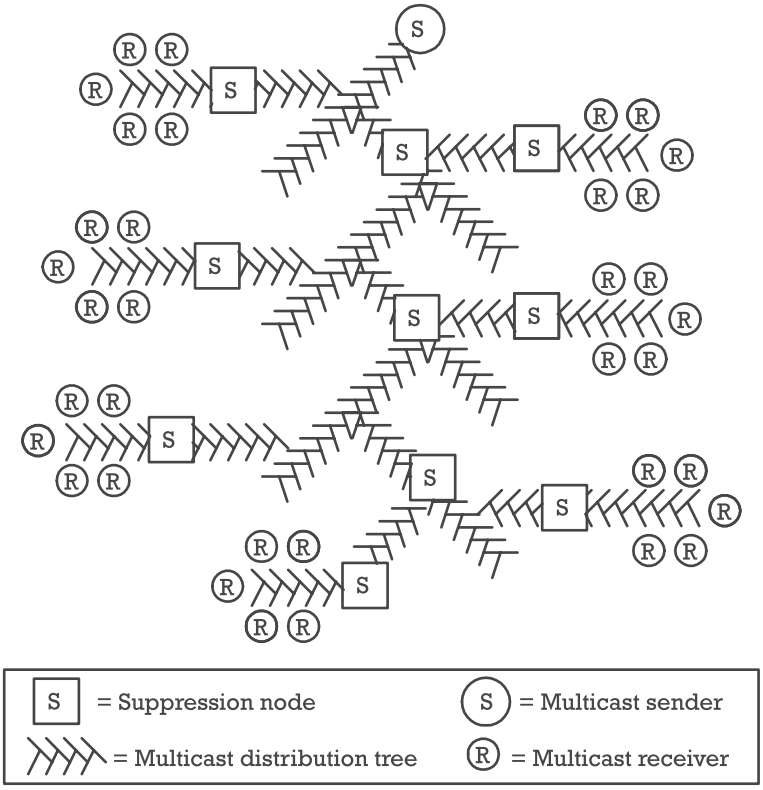


Figure 9.4 General model of NORM protocols.

downstream to it. This, in turn, requires the distribution and management of certificates in the domain.

The second option, using symmetric key–based authentication, is a double-edged sword. Here, the advantage is that retransmissions can be done by any receiver or suppression node that holds a copy of the symmetric key. On the down side, any dishonest member could also transmit bogus messages and thereby confuse the other receivers and even deny them service. Since symmetric key–based authentication provides the most promising approach in the near future to router vendors and ISPs, we further discuss possible key arrangements for NORM protocols.

Possible key arrangement for NORM protocols

There are many ways in which symmetric keys can be introduced and assigned to different NORM entities, through the use of appropriate key

management approaches. In the following, two broad key arrangement strategies are discussed.

1. *Flat arrangement*. In this approach, all entities involved in the NORM protocol share a common (group) symmetric key, called the NACK-key. This is shown in Figure 9.5. The entities of main focus in this case are the receivers, the sender(s), and the SNs. This approach achieves group authentication (i.e., via MACs), in which the recipient of any NACK message can conclude that the NACK was sent by a holder of the NACK-key. The NACK-key should be used to authenticate NACK messages only, and should never be applied to data messages, even if the data is authenticated using the same technology or even the same GKM-protocol instantiation. When a

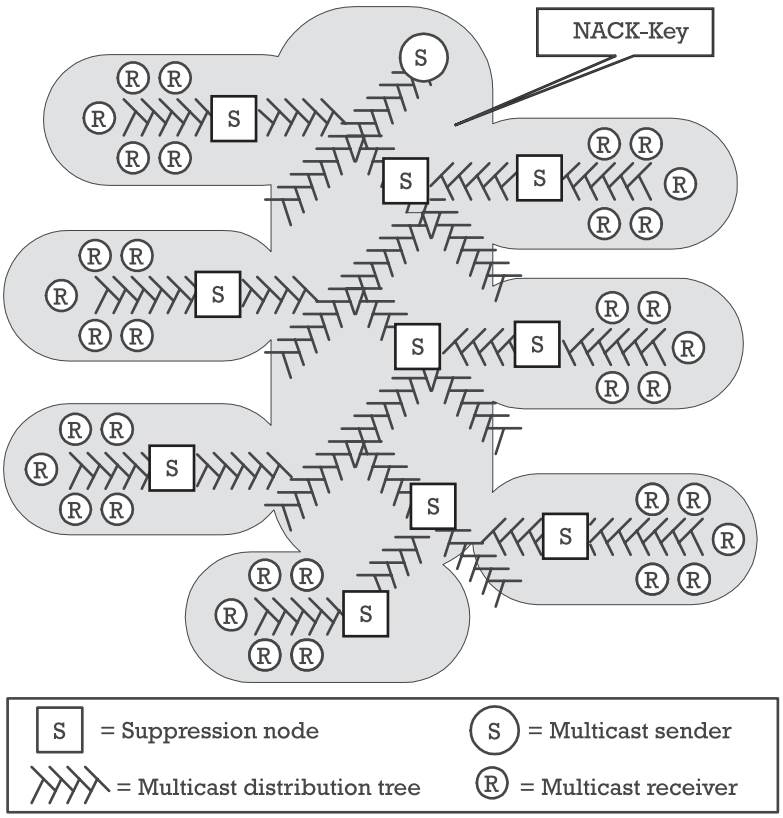


Figure 9.5 Flat arrangement of NACK-keys.

receiver loses a packet, it will apply the NACK-key to the NACK message that it transmits upstream.

The main advantage of this flat arrangement is that no topology information is needed by the GKM protocol, since it treats all involved entities equally and provides each with the same NACK-key. Furthermore, depending on the specific NORM protocol, SNs may be able to forward a NACK message without needing to modify the message (e.g., to indicate that it originated from that SN), since all relevant entities are in possession of the same NACK-key. Hence, the SN is only required to verify the group authenticity of the received NACK message, and then forward the message toward the sender or another repair entity. This latter feature depends on whether the NACK message is modified by the SN (e.g., TTL, hop-count, or other fields).

The disadvantage in having the NACK-key arrangement is that when a member leaves, the (old or current) NACK-key must be replaced, in order to prevent the ex-member from issuing unauthorized NACK messages using the (old) NACK-key. Rekeying the NACK-key when group membership changes ensures that only authorized members/entities can send group-authentic NACK messages. However, rekeying both keys when a member leaves may create an unbearable overhead on the GKM protocol entities and the SNs. One possible way around this problem is for the SNs to be given several NACK-keys (with key identifiers or key IDs) in advance by the KD for future use, while the receivers are given the NACK-keys on a per-key basis or other release frequency. This reduces the need for the SNs to be rekeyed each time there is a change in group membership, since the key IDs allow the SNs to swap keys and erase terminated (old) NACK-keys.

2. *Logical hierarchical arrangement*. An improvement over the flat (groupwide) arrangement of a common NACK-key is the configuration of receivers into logical subgroups from the keying perspective. Here, each subgroup of receivers is associated with one (or few) SN. This association can be based on the topology of the network (e.g., all receivers closest to an SN), statistics on some network characteristics, or other aspects of the NORM protocol. Alternatively, the logical arrangement of receivers and SNs can simply be superimposed.

   Each subgroup then shares a symmetric key (referred to here as SG-key or Subgroup-key) with the SN associated with that logical

subgroup. This permits an SN to authenticate NACK messages coming only from its subgroup members (who are in possession of the corresponding SG-key), with the alternative of dropping a message that fails authentication or forwarding it upstream (with the hope that another SN upstream may be able to authenticate it). This is shown in Figure 9.6.

In considering the logical, hierarchical NACK-key arrangement, several arrangements can be constructed for the sharing of keys between SNs and members, although the principle of hierarchical arrangement of keys largely remains the same. An important consideration regarding these key arrangements is the impact on the entity that performs retransmissions, since that entity may have to hold several keys in order to verify NACK messages.

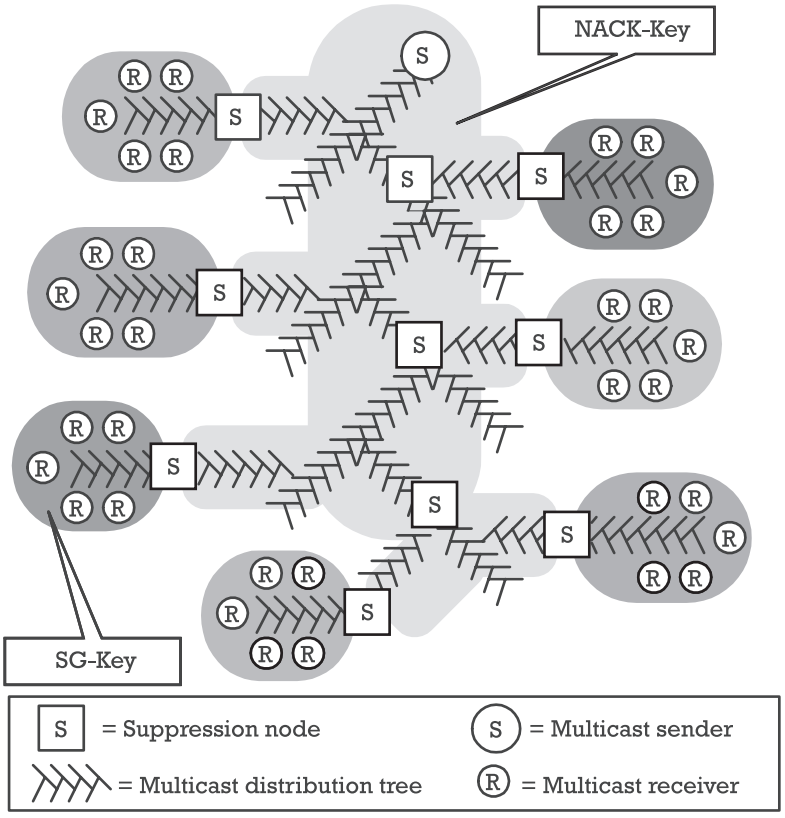


Figure 9.6 Hierarchic arrangement of NACK-keys and SG-keys.

9.4.2 PGM

The PGM [13] reliable transport protocol represents an interesting instance of a NORM protocol. PGM makes use of the same path (in reverse) as the multicast distribution tree to deliver the selective NACKs sent by receivers toward the source.

NACKs for missing packets travel in the reverse direction along the path of the distribution tree. A receiver will repeatedly send (via unicast) a NACK to the last-hop PGM network element on the multicast distribution tree. The network elements at the multicast routing layer (i.e., the network layer) forward the NACKs upstream, PGM-hop-by-PGM-hop, to the source. The term ''PGM-hop'' implies that network elements in the distribution path must be PGM entities, and must participate in the PGM protocol instance. A NACK, in fact, travels through the same sequence of PGM network elements (on the distribution tree) as the data packet (to which that NACK corresponds). Retransmissions are done either by the source or by a repair entity called the *designated local repairer* (DLR), which could be a router on the multicast distribution tree or another entity designed to perform retransmissions.

Since the NACK messages themselves could be lost, PGM defines a network-layer hop-by-hop procedure for reliable NACK forwarding. When a PGM network element receives a NACK at an interface, it will in return transmit a *NACK confirmation* (NCF) message out that same interface. In order to indicate to other PGM network elements that it has already received a NACK for a given data packet, the NCF message is in fact sent via multicast to the group. PGM network entities do not propagate NCF messages, since they are essentially a hop-by-hop reliability mechanism for the NACKs. The repair data packets are constrained in their delivery to only those network segments from which NACKs originated, namely, where there are members that did not receive the original transmission of the data packet from the source.

In order to create PGM state information within network elements, a *source path message* (SPM) is sent interleaved among the data packets by the source. The transport session identifier (TSI)—which identifies sessions, groups, and sources—allows the SPM to effect a source path state within the network elements for individual sessions/sources (i.e., per TSIs).

Additional PGM security requirements

Besides the security requirements in Section 9.2, PGM has a number of additional requirements that need to be addressed.

- *Protection of PGM trees*. The use of SPMs, NACKs, and NCFs essentially creates an overlay PGM tree over the multicast distribution tree. This implies that all the downstream control messages (i.e., SPMs and NCFs) and upstream control messages (i.e., NACKs) must be protected against illegal modification and other attacks (e.g., replay) mentioned in Section 9.2. For NCFs, since PGM network elements use multicast to send out NCFs, this implies that a separate instance of a secure multicast group is needed to protect the multicasted NCFs. For SPMs, since the source is the entity that sends out SPMs interleaved with the data for a given TSI, this implies that SPM protection must be separate from data protection, since the SPMs are addressed to the PGM network elements, instead of receivers.
- *Protection of PGM leaves*. When a receiver unicasts a NACK repeatedly to the last-hop PGM network element, the NACK must be authentic, integrity-protected, and replay-protected.

Possible PGM key arrangement

Ideally, PGM network elements should have their unique public key pairs. This may be attractive, depending on the amount of NCFs, SPMs, and NACKs exchanged within a domain. An alternative to public key digital signatures is to employ shared symmetric keys, which is discussed in the following (see Figure 9.7).

Here, one symmetric key—the PGM-key—could be used per TSI/session/source to provide a long-term cryptographic key that is shared among the PGM network elements within the domain, excluding the sources and receivers. The PGM-key is used to provide protection of all NCFs, SPMs, and NACKs in the domain independent of the TSI/session/source. In effect, the interior of the PGM tree is protected separately from the PGM leaves, where the sources are considered to be at the leaves.

The cost of using a PGM-key that is shared only by PGM network elements in the interior of the PGM tree, is the need for the last-hop (next-hop) PGM network element to provide separate protection when dealing with sources and receivers:

- *Last-Hop-key*. For NACKs issued by a receiver, the receiver and its last-hop PGM network element would share a symmetric Last-Hop-key (LH-key). The receiver's last-hop PGM network element then substitutes that NACK with its own NACK, and provides protection using the PGM-key. Other PGM network elements upstream can then authenticate that (substituted) NACK without having to share a

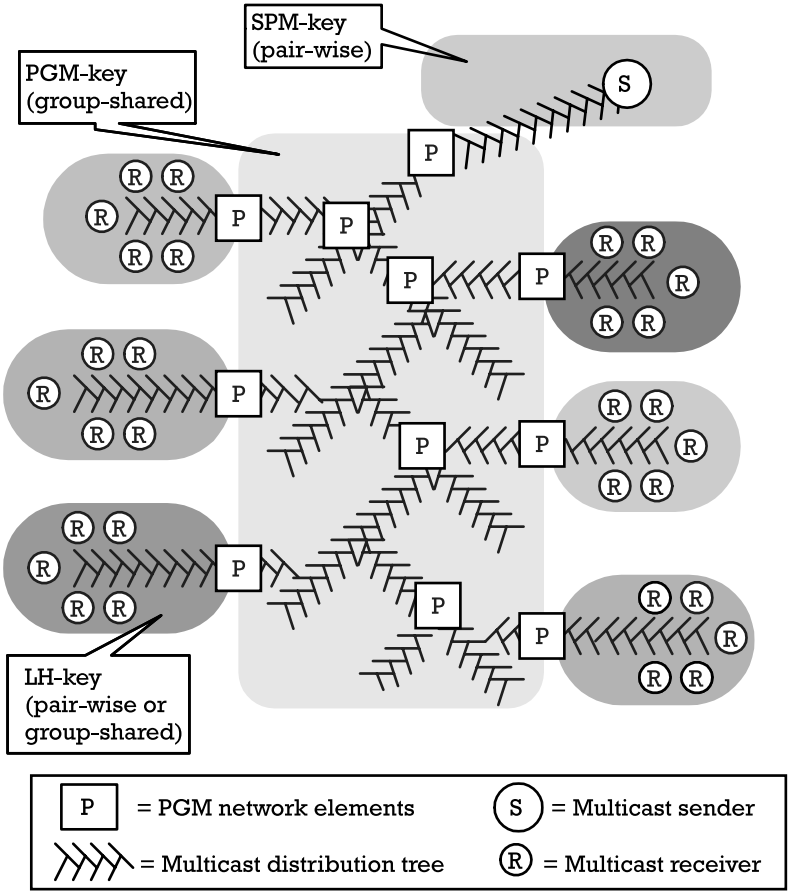


Figure 9.7 Possible key arrangement for PGM.

key with the receiver, since they share the PGM-key with that last-hop PGM network element. Note that an LH-key can be pair-wise unique between a receiver and its last-hop PGM network element, or it can be shared by a set of receivers downstream to a last-hop PGM network element.

- *SPM-key*. For SPMs interleaved (by a source) among the data packets, a separate SPM-key could be established as a pair-wise key between the source and its next-hop PGM network element. If other PGM network elements are required to also authenticate SPMs (in addition to authentication by the source's next-hop PGM network element), then this SPM-key would need to be distributed to all PGM network elements in the interior of the PGM tree.

One way to accomplish this distribution is for the source's next-hop PGM network element to encrypt the SPM-key under the PGM-key, followed by the transmission of the encrypted result to the other PGM network element. The transmission could be done through the same path as the SPMs (e.g., via the multicast distribution tree). Other non-PGM entities would not be able to obtain the SPM-key, since they are not in possession of the PGM-key. In essence, the source's next-hop PGM network element vouches for the SPM-key. This solution may require changes to the current PGM specifications.

9.5 Security of FEC-based protocols

As mentioned previously, the implosion of messages at the sender represents one problem faced by many RM procotols, which can be addressed using techniques such as local repair and suppression. Another approach—which is a departure from the sender-initiated and receiver-initiated frame of thought—is to simply dispense with ACKs or NACKs aimed at the sender. This is the basis of the FEC-based RM protocols, for which the reader is directed to [19, 20] for detailed information about packet-based FECs.

In the FEC-based strategy, no back channel, in the sense of ACKs and NACKs, is assumed. Thus receivers need not send messages to the sender, and the sender will therefore not suffer the implosion problem. Instead, the sender incorporates *redundancy* into the data, to help the receiver in reconstructing the original message. In the most primitive case, the sender could send the same messages repeatedly—which is perhaps acceptable for some limited environments. However, a more intelligent way is to redundantly encode packets using (n, k)-encoding. In the packet-level (n, k)-encoding the k source packets are encoded into n packets, where n is greater than k. The receiver then needs only to obtain any unique k packets out of the sent n packets to reconstruct the original message.

As mentioned in [21], most—if not all—FEC-based approaches are especially vulnerable to DoS attacks by attackers who try to send forged packets to the session that would prevent successful reconstruction by receivers, or that would cause inaccurate reconstruction of large portions of the data object by receivers.

To overcome potential DoS attacks, protection in FEC-based schemes should be provided both at the application layer and at the packet level (see Section 9.2). At the application layer, integrity protection—and possibly group authentication or source authentication—should be provided to the

message (e.g., file) before it is delivered to the FEC layer where the (n, k)-encoding will be applied. Then, each of the n packets would need individual integrity protection and authentication, so that a receiver could immediately discard bad packets, instead of passing them up from the network layer to the transport layer or application layer.

At the packet level, group authentication or source authentication could be provided. The shortcoming of group authentication via a group-shared symmetric key is that any dishonest receiver can forge packets. On the other hand, public key–based digital signatures on a per-packet basis is too expensive in term of resource consumption.

One possible source-authentication method is to use the TESLA protocol [22], which was described earlier in Chapter 3. TESLA provides comprehensive support for source authentication of a multicast stream using MACs. TESLA uses the approach of committing to a MAC key first, and revealing it after a preadvertised delay. The applicability of TESLA would depend on the specific application of the FEC-based RM protocol, and, more specifically, on whether receivers can tolerate the delayed authentication imposed by TESLA.

The reader is directed to [22] for further details on TESLA, and to [23] for a proposal on a framework to use source authentication within the context of IPsec.

9.6 Summary

This chapter has focused on the broad problem of the security of RM protocols, which provide the reliability needed over the data packet delivery by the underlying multicast routing protocol. RM protocols are in general more complex than unicast reliable transport protocols (e.g., TCP). Furthermore, each application that uses an RM protocol has its specific needs. The specific needs of each multicast application has prompted several proposals for RM, each having its own specific security requirements.

The three basic requirements of RM protocols are the same for other protocols in general: namely, the need for control messages to be authentic, the need to be integrity-protected, and the need to be replay-protected. Section 9.2 provides further RM-specific requirements that are common across several RM protocols.

The three families of RM protocols discussed are the TRACK protocols, the NORM protocols, and the FEC-based protocols. The TRACK family of RM protocols employs a tree structure to organize the delivery of ACKs and NACKs. All receivers are arranged into local regions, where each region is assigned a repair entity to aid in providing reliability. The tree structure

essentially acts as the control channel over the data channel (namely, the multicast distribution tree). In contrast, the NORM protocols employ NACKs to indicate packets that were not successfully received. Suppression entities in NORM protocols help in discarding superfluous NACKs, thereby avoiding the implosion problem at the sender. In the FEC-based strategy, no back channel is assumed, and the sender instead incorporates redundancy into the data to help the receiver in reconstructing the original message. Thus, the strategy avoids the implosion problem entirely.

The security issues of an RM protocol from each of the three families were discussed. In particular, attention was given to the RMTP-II protocol and the TRAM protocol from the TRACK family, while the PGM protocol was selected from the NORM family. These specific protocol instantiations were chosen due to the fact that they have seen deployment in production environments, as opposed to laboratories only. Since the security of FEC-based protocols is tightly related to the TESLA proposal discussed in Chapter 3, only a limited discussion was provided, in order to prevent repetition of the earlier discussion.

Since a general-purpose security solution for all RM protocols would be impossible to construct due to the sheer size of the design space, this chapter has approached the subject of RM protocol security more from a practical survey perspective. It has provided security requirements for families of RM protocols, and for three specific RM protocols (namely, RMTP-II, TRAM, and PGM) it has suggested some cryptographic key arrangements to help achieve the authentication and integrity of control messages used in those protocols. However, further security analysis and design will need to be conducted on these three RM protocols (and others like them), in order to gain some assurance of their security.

References

[1] Hardjono, T., B. Cain, and N. Doraswamy, ''A Framework for Group Key Management for Multicast Security,'' draft-ietf-ipsec-gkmframework-03.txt, IETF, August 2000, work in progress.

[2] McCullough, D., ''Noninterference and the Composability of Security Properties,'' in *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, CA: IEEE Computer Society, April 1988, pp. 177–186.

[3] McLean, J., and C. Meadows, ''Composable Security Properties,'' in *Proc. of the IEEE Computer Security Foundations Workshop II*, IEEE Computer Society, 1989.

[4] Handley, M., et al., ''The Reliable Multicast Design Space for Bulk Data Transfer,'' RFC 2887 (informational), IETF, August 2000.

[5] Paul, S., *Multicasting on the Internet and Its Applications*, Norwell, MA: Kluwer Academic Press, 1998.

[6] Towsley, D., J. Kurose, and S. Pingali, ''A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols,'' *IEEE JSAC*, Vol. 15, No. 3, April 1997.

[7] Levine, B., and J. J. Garcia-Luna-Aceves, ''A Comparison of Reliable Multicast Protocols,'' *ACM Multimedia Systems Journal*, Vol. 6, No. 5, August 1998, pp. 334–348.

[8] Chiu, D., et al., *TRAM: A Tree-Based Reliable Multicast Protocol*, Technical Report SML TR-98-66, Sun Microsystems, July 1998.

[9] Montgomery, T., et al., ''The RMTP-II Protocol,'' draft-whetten-rmtp-ii-00.txt, IETF, April 1998, work in progress.

[10] Whetten, B., T. Montgomery, and S. Kaplan, ''A High Perfomance Totally Ordered Multicast Protocol,'' in Birman, K. P., F. Mattern, and A. Schipper (eds.), *Theory and Practice in Distributed Systems: International Workshop*, New York: Springer-Verlag, 1995, pp. 33–57.

[11] Miller, K., et al., ''Starburst Multicast File Transfer Protocol (MFTP) Specification,'' draft-miller-mftp-spec-03.txt, IRTF, April 1998, work in progress.

[12] Floyd, S., et al., ''A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing,'' *IEEE/ACM Trans. on Networking*, Vol. 5, No. 6, December 1997, pp. 784–803.

[13] Speakman, T., et al., ''PGM Reliable Transport Protocol Specification,'' RFC 3208 (experimental), IETF, December 2001.

[14] Byers, J. W., et al., ''A Digital Fountain Approach to Reliable Distribution of Bulk Data,'' in *Proc. of ACM SIGCOMM*, Vancouver, Canada, September 1998.

[15] Hardjono, T., and B. Whetten, ''Security Requirements for TRACK,'' draft-ietf-rmt-pi-track-security-00.txt, IETF, June 2000, work in progress.

[16] Kadansky, M., J. Wesley, and J. Provino, ''Tree-Based Reliable Multicast (TRAM),'' draft-kadansky-rmt-tram-02.txt, IETF, January 2000, work in progress.

[17] Adamson, B., et al., ''Nack-Oriented Reliable Multicast Protocol (NORM),'' draft-ietf-rmt-pi-norm-04.txt, IETF, March 2002, work in progress.

[18] Hardjono, T., L. Vicisano, and L. Dondeti, ''Security Considerations for NORM Protocols,'' March 2001, Internet draft in preparation.

[19] Huitema, C., ''The Case for Packet Level FEC,'' in *Proc. of the IFIP 5th International Workshop on Protocols for High Speed Networks*, Sophia Antipolis, France: IFIP, October 1996.

[20] Luby, M., et al., ''The Use of Forward Error Correction in Reliable Multicast,'' draft-ietf-rmt-info-fec-02.txt, IETF, February 2002, work in progress.

[21] Luby, M., et al., ''Asynchronous Layered Coding Protocol Instantiation,'' draft-ietf-rmt-pi-alc-08.txt, IETF, April 2002, work in progress.

[22] Perrig, A., et al., ''Efficient and Secure Source Authentication for Multicast,'' in *Proc. of Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2001, pp. 35–46.

[23] Canetti, R., P. Rohatgi, and P. Cheng, ''Multicast Data Security Transformations: Requirements, Considerations, and Proposed Design,'' draft-irtf-smug-data-transforms-00.txt, IRTF, June 2000, work in progress.

CHAPTER

10

Applications of multicast and their security

Contents

10.1 Stock market data distribution

10.2 Local area IP Television

10.3 Nonreal-time multicast distribution

10.4 SecureGroups project

10.5 Summary

Multicast and group communications have been studied for over 2 decades now, and various aspects of this area have been put to practical use. Much of the interest in this broad area has been driven by the wide scope of applications, and by the enormous potential of multicast and group communications.

In this chapter we briefly look into a few examples or case studies where multicast has been deployed to satisfy one or more requirements of the specific area of application. Each example reflects the decision of its respective architects to deploy multicast in their network, and—due to the specific architectures—each has deployed specific protocols for routing, reliability, and other functions.

For each of these examples, the current chapter discusses the general security requirements and possible approaches to satisfy them. The discussion on the various security issues surrounding these cases is not intended to be comprehensive, due to the fact that considerable familiarity with the field of application, together with a comprehensive security and threat analysis, is needed to arrive at a strong security solution. Instead, these case studies are intended to show that multicast and group communications are indeed being deployed in real-life scenarios, that these scenarios have serious security requirements to be fulfilled, and that the technologies that have been discussed in the previous chapters have immediate applicability in answering some or all of these requirements.

10.1 Stock market data distribution

The distribution of stock market data has been one interesting application of multicast, both for private and public networks.

In the public network case, there has been interest in providing stock-tick information on the user's desktop, albeit with a 10- or 20-minute delay. Here, multicast at the IP layer provides a very compelling method of delivery of the information on the Internet, to a wide audience of users.

The private networks case refers to those networks that provide very specific high-quality information in real-time, to a closed membership of receivers. An example of this later case is the Securities Industry Automation Corporation (SIAC) network, described in the following.

10.1.1 Background

SIAC operates and maintains two separate computer environments to process trade and quote information on behalf of the Consolidated Tape/Quote Association (CTA/CQ). The trade information is processed by the Consolidated Tape System (CTS) while the quote information is handled by the Consolidated Quotation System (CQS). Both systems run on fault-tolerant computer platforms at different physical sites, to provide redundancy in the face of possible disasters. The thinking is that if a site disaster should occur at either location, all of the computer processing would be transferred to the surviving site at reduced capacity (65% capacity). The aim is to have the surviving site recover the services within 2 hours, while a full-site disaster can be recovered on a next-day basis. SIAC's two operating sites are located on two separate power grids, and have multiple redundant communications paths connecting the two facilities. The sites have uninterrupted power supplies, as well as emergency generator backups. All of SIAC's fault tolerant systems utilize a SIAC-developed software environment that allows for a common operations interface and internal processing infrastructure [1].

10.1.2 Network topology

The CTS and CQS receive their data from the nine market centers over network-based TCP/IP connections. Each market center has redundant communication paths into the two operating environments, and each uses diverse common telephone carriers to send its trade and quote data to SIAC. SIAC uses a high-bandwidth router backbone network to simultaneously distribute, via IP multicast, trade and quote information to the 66 CTS and 62 CQS subscribers (data recipients).

These data recipients receive their data over T-1 and T-3 communications facilities from both sites, through diverse common carriers. Independent of where the system is actually located (of the two sites) both streams of data are simultaneously distributed out of both sites using a SIAC-developed multicast packet replicator (MPR). This provides the data recipients with live redundant streams.

SIAC's design and implementation of IP multicast technology allows trade and quote data to be distributed to recipients in a fair manner over a network, and eliminates any dependency where one data recipient having a problem might impact another data recipient. If a recipient experiences loss in data due to problems with its own system, an automated retransmission facility allows the recipient to request and receive the retransmissions.

The CTS and CQS receive trade and quote information from the nine market centers using a standard message format. Each system validates its respective message formats, verifies the information against its databases (e.g., valid symbol, etc.), consolidates the information with the other market centers' information, and disseminates the information to the data recipients over its respective common standard message formats via the IP multicast network. A timestamp is included in every trade and quote message, and they are in turn stored in the system for both on-line and after hours processing. Figure 10.1 shows a general architecture of the SIAC CTS/CQS system, with its multicast network to its subscribers. The diagram also shows a possible use of a key server—one at the CTS/CQS and one at each subscriber, in order to carry out key distribution and key management for the cryptographic keys used to protect the data being transmitted through the multicast network.

10.1.3 Security requirements and possible approaches

Aside from the requirements of transmission reliability and timeliness of delivery, there are a number of security-related requirements that need to be fulfilled by the data distribution service. Two foremost requirements are data integrity and source authentication. (In some cases data confidentiality through encryption may be deployed, depending on the specific requirements of the subscriber of the service.)

Data integrity is crucial, because a subscriber needs the assurance that the stock market information being transmitted has not been modified in transit. The accuracy of stock information at any given time is paramount for the subscriber, and has drastic implications on the financial future of the subscriber. Sophisticated attacks on the stock price information of a given listed company can result in the financial demise of that company.

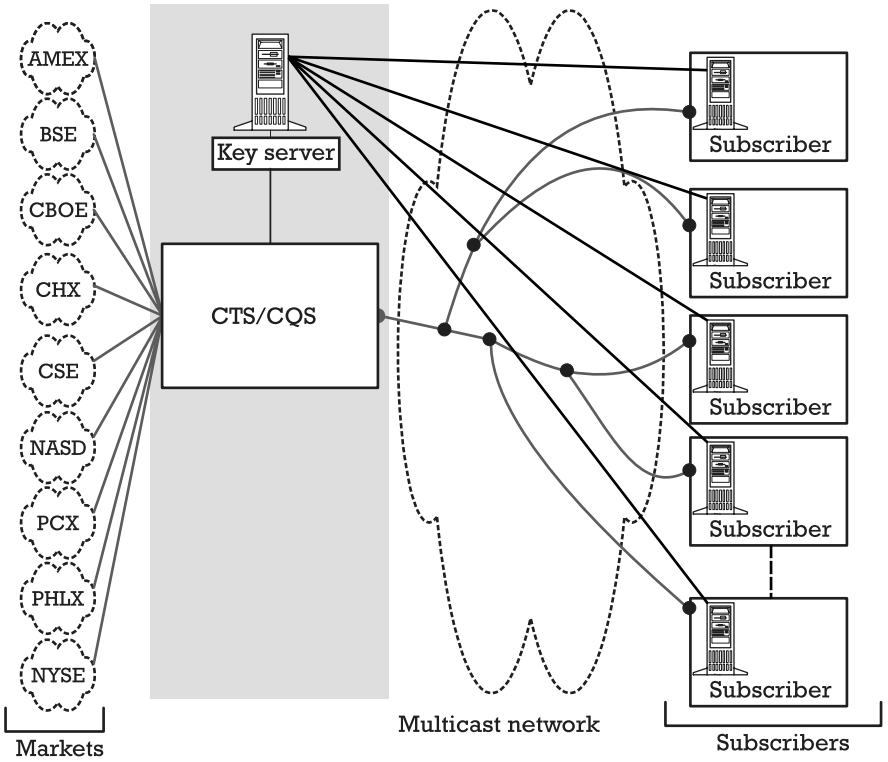


Figure 10.1 Stock market example.

Source authentication of timely stock market data is important, because a subscriber needs assurance of the origin of the information. Although in theory a subscriber may rely on another subscriber to relay accurate information with a few seconds delay, in practice every subscriber can only afford to trust a single entity—in this case SIAC and its system—to provide it with authentic information. The enormous value of the data being transmitted makes integrity, source authentication, and timeliness paramount requirements.

Although the current authors are not privy to the detailed workings of SIAC's stock market data distribution network, there are some general approaches that can be adopted to minimize potential attacks to the network. These include the following:

- *Closed network.* Since the membership of subscribers is not expected to be dynamic over shorts periods of time, and since the value of the data is high, providing a closed network in itself already provides

some level of security. Each subscriber could be assigned a permanent IP address, and administratively scoped multicast could be used to prevent IP multicast packets from leaving the network.

- *Unidirectional multicast.* Since the source of the transmission in this example is the CTS/CQS system at SIAC, and since the subscribers do not transmit data to other subscribers, a unidirectional multicast approach is perhaps the most appropriate for this case. At the multicast routing level, this can be achieved through the appropriate setup of IGMP (version 3), and of the multicast routing protocol, PIM-SSM.
- *Authentication and integrity.* Data integrity can be provided using a keyed hash function (MAC), with the MAC-key being shared only by legitimate members (subscribers) of the group. Source authentication can be achieved using asymmetric cryptography (e.g., public key–based digital signatures), but it is typically more expensive in computational cost. Chapter 3 discusses several approaches to providing authentication and integrity.

Although not widely tested or deployed, another possible idea proposed in [2] is to establish a multicast VPN, by using a multicast address on a VPN, based on IPsec ESP in tunnel mode. This is shown in Figure 10.2. This idea is particularly attractive here since the subscribers/receivers are fixed, and the multicast VPN can be used instead of a one-to-one IPsec VPN from the sender to each subscriber. In effect, in the multicast VPN idea, a one-to-many tunnel is established from the sender (SIAC) to the many receivers (subscribers). From a routing perspective, the multicast routing protocol executing in the network should simply forward the IPsec packets (with a multicast destination address) to the receivers in the group. In order to provide the IPsec SA management for the group of subscribers, the GSA definition and its management using the GDOI approach can be used (see Chapters 4 and 5).

10.2 Local area IP Television

The Internet remains an attractive distribution network for content to IP-enabled television and other network devices. A number of companies continue to invest in developing content-related technologies, including content delivery networks (CDNs) and IP-based consumer electronic devices, such as IP-enabled television, entertainment centers, and networked game consoles. At the consumer's end, IP multicast provides a most

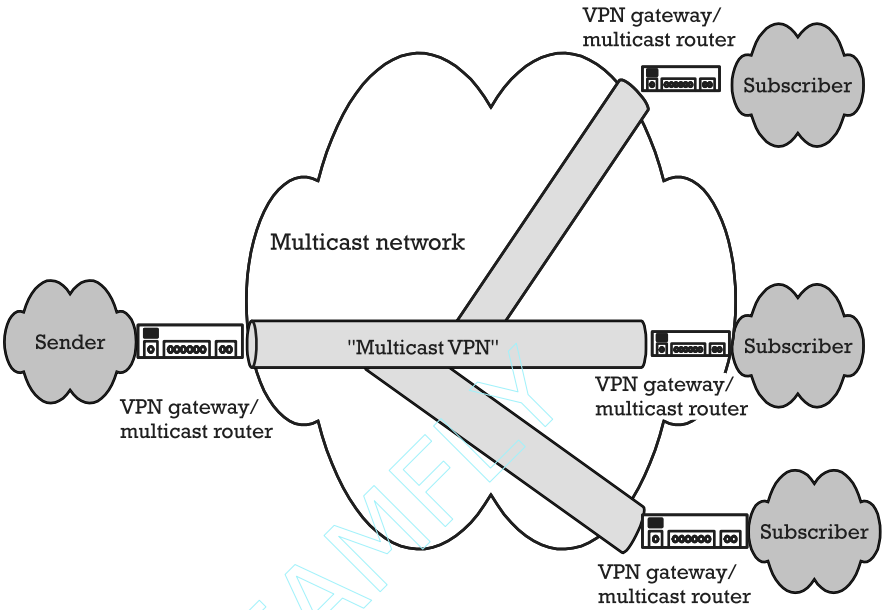


Figure 10.2 Notion of multicast VPNs.

attractive delivery mechanism for local area delivery of content/programming to a well-defined network with a reasonable amount of bandwidth.

10.2.1 Background

A number of companies are currently developing systems for content delivery to densely inhabited environments with rich bandwidth. An example of such an environment would be apartment complexes with LAN capabilities, and with high-speed access to the open Internet at several nodes.

The central concept is that for such apartment buildings IP multicast can be used to deliver content either in near real-time, or in a stored video approach. The user's IP-TV can then join certain multicast groups, either short-lived groups or long-lived sessions. The multicast groups can be relaying content in near real-time or in a prerecorded (stored) mode, that is replayed at frequent intervals. Since the environment is local, and is based on LANs or wireless LANs, a dense-mode multicast routing protocol can be used.

In addition, users in the apartment complex could create their own multicast groups to share various activities, such as live chats, multiplayer games, and even their own closed network ''TV broadcasting,'' with programs of their own creation.

10.2.2 Network topology

A general illustration of this application of multicast is shown in Figure 10.3. In this Figure, an apartment building consists of several floors, each of which in turn contains a number of receivers in the form of IP Television devices. Although the diagram shows a separate router for each floor, other configurations are possible, and would be largely determined by the distribution of users and the type/speed of the underlying network. Thus, for example, it is possible that several floors may share a single router.

At the other extreme, it is also quite possible that the entire building shares one single subnet, thereby possibly located only a single hop from the actual source, which could be the *Content Management Server* (CMS).

Figure 10.3 shows several entities, notably the content management server (CMS), the *key server*, the *video server*, and the *video store*. The intent is to convey the notion that several functions need to be supported in such an environment to make IP multicast attractive. Thus for example, the CMS could be the heart of the system, which controls the incoming content from the content distributor. The CMS would record and store content that was delivered on a regular basis (e.g., news or stock quotes) and it would request (to the content distributor) content that was preordered by one or more users ahead of time.

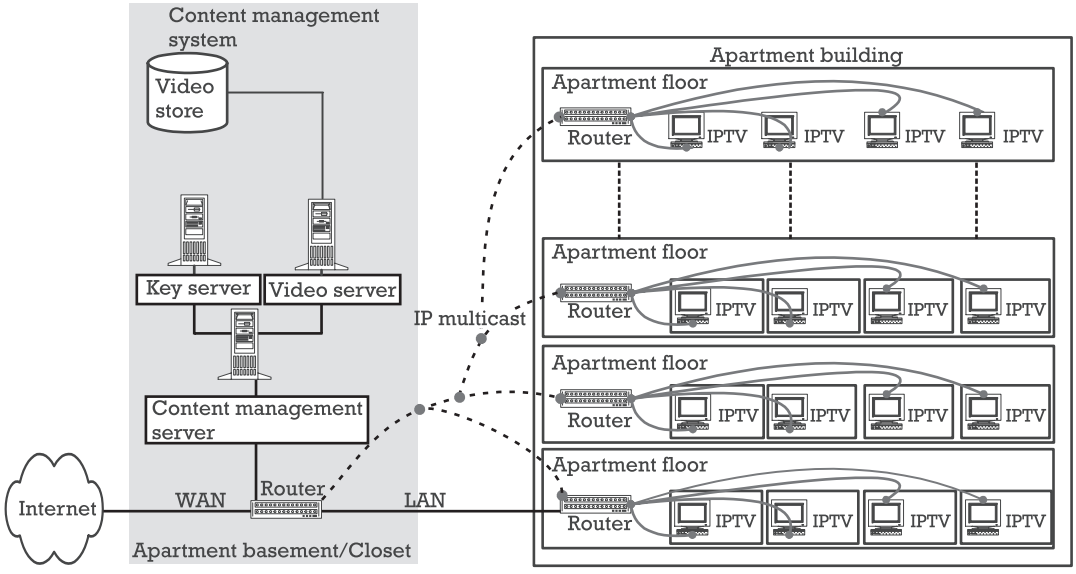


Figure 10.3 Local area IP television example.

An important function of the system is to provide controlled access to the various multicast groups in the IP-TV network. This is the role of the key server, which runs a group key management protcol, with the IP-TV receivers as its host/members. For each multicast group (or session), an IP-TV receiver (i.e., a group member) would need to obtain an decryption key (or group key) from the the key server.

From a multicast routing perspective, possible protocols would include dense-mode protocols such as DVMRP, PIM-DM, and MOSPF (See Chapter 8 or [3]). Using the topology in Figure 10.3, each router in the apartment floor would be a multicast router, and would also run the IGMP protocol with its respective members (namely the IP-TV receivers). The source for externally based content would be the CMS. In addition, each IP-TV receiver could be a source for its own multicast group (provided it was not transmitting rights-carrying or copyrighted contents).

As such, an important assumption in this scenario is that the IP-TV client contains built-in copyright protection functions or incorporates a digital rights management (DRM) client. Such a DRM capability would minimize the threat from illegal copying and/or redistribution of the content by users. Without copyright protection, a user could simply forward contents to other users, either in the apartment complex or at external locations (e.g., through a VPN). Currently, there are a number of efforts in the various content-related industries. An example is the OpenCable Architecture Platform (OCAP) proposal from CableLabs, which would incorporate DRM capabilities into next generation IP-enabled set-top-boxes.

The above example of local area IP-TV provides an attractive source of additional revenue for apartment complex owners, and may even become the core business for service providers that carry out the installation, management, billing, and maintenance of this service based on off-the-shelf hardware and software.

10.2.3 Security requirements and possible approaches

In considering the security requirements of the IP-TV scenario, it is useful to distinguish between externally sourced content being retransmitted by the CMS, and internally sourced content being transmitted by a user for the user's private multicast group. In the first case, the CMS would provide this service, and charge users for accessing the content (e.g., movies and TV programs). Billing could be based on a PPV model, and each program could employ a separate multicast group with a limited lifetime. In the second case, billing could be based on either volume of traffic or a fixed monthly price per owner/creator of each multicast group.

From a security perspective the value of the content being delivered through the multicast groups determines the security requirements of the scheme. The requirements include, among others, the following:

- *User/client identification and authentication.* For billing purposes, when an IP-TV client requests to join a given multicast group or session being transmitted by the CMS, and asks the key server for a key to decipher the content, that client must be identified and authenticated by the key server.
- *Confidentiality.* For valuable content (e.g., movies) encryption provides a means for controlling access to the content. As such, confidentiality (in the sense of encryption) is an important requirement. For private (local) multicast groups, the group owner/creator should be provided with the option of providing confidentiality for the group's traffic.
- *Source authentication.* For certain types of content, the transmission source of the content matters. For example, for private (local) multicast groups, members should be able to identify senders to the group. For groups owned by the CMS, the IP-TV client may be configured to accept content that originated from the CMS only.
- *Copy protection.* Although outside the scope of multicast security, copy protection or DRM functions should be provided as part and parcel of the IP-TV client or application. This is especially true for rights-carrying content (which includes most commercial content, such as movies and TV programs), in order to prevent the content from being illegally copied and retransmitted by a user.

10.3 Nonreal-time multicast distribution

Consider distribution of the next Windows operating system software update to Microsoft's customers, distribution of a training video to all Ford dealers, or distribution of software or database updates to all Wal-Mart stores. Considering the large number of receivers, these applications are better served by a multicast-based push distribution model, than a unicast based pull model.

Typically, data transmission in these applications does not need to be in real time; however, reliable transmission is a requirement. In most cases, the sender has the data available in advance, and has the buffer space to hold the data. In other words, there are no limits on sender-side buffering.

On the other hand, since receivers are typically end-user devices, receiver-side buffering must be kept to a minimum. Consider also that receivers may have heterogeneous connectivity to the Internet. Some of them may have broadband access, whereas others may be using a dial-up connection. Thus, there may be multiple multicast sessions serving the receivers; for example, one for each service level. Receivers may also need to be grouped based on the number of hops to the sender.

In the following, we describe a protocol called MFTP [4, 5] that supports applications such as those listed earlier.[1]

10.3.1 MFTP

MFTP supports simultaneous file transfer to a group of members. An MFTP server streams data, packaged into *data transmission unit* (DTU) messages, to receivers, without waiting to process feedback. Members send feedback, aggregated into ranges of DTUs, after the first *pass* of the transmission has been completed. Intermediate agents may be employed to further aggregate receivers' feedback for scalable operation. The MFTP server processes feedback after each pass, consisting of either file or *repair data* transmission.

MFTP data streams are announced on a *public* group, and users are invited to join a *private* group to receive the data. MFTP operates over UDP and consists of two protocols: the multicast control protocol (MCP), and the multicast data protocol (MDP). MCP provides a functionality similar to that of IGMP, at the application layer. MCP messages include *join* and *leave* messages from the server to receivers, and a *query* message from a client to a server, to find a public address. MCP also has an *echo* message for an MFTP entity (server or client) to ''ping'' another MFTP entity. MDP is used for reliable data transfers as well as group announcements. *Announce* is an MDP message from the server to the public group address, and a *registration* message is from a prospective member to the server. The server sends DTUs to members via the private group address, and may ask for member status. The members may respond with an ACK or a NACK. Other MDP messages are *completion* or *abort* from the server, and *done* or *quit* from a member.

MFTP supports several different types of groups: closed groups, open limited groups, and open unlimited groups. The classification is based on how tightly the sender can control data download to receivers. In closed groups, the sender invites members to join the group, and ensures that all of them register and receive data. In open limited groups, members are invited,

1. There are several similar protocols such as FCAST [6] reported in the literature.

but are not obligated to join. Once a member registers, however, the sender must ensure that the member receives data. Finally, in open unlimited groups, the sender does not specify a member list in the group announcement, and thus any host can register to receive data. Members may also leave the group when they are no longer interested in the data being sent.

An example MFTP session works as follows. A server sends the group announcement on the public group that all potential participants are expected to monitor. The announcement contains information such as the type of group (e.g., closed or open limited), the list of members that must respond, the nature of the group data to be sent and so forth. Closed groups are typically employed when the sender has a list of members that must receive the data transmission. Thus after receiving an announcement of a closed group, members whose identities are in the members' list, must register. For reliability, the sender repeats the announcement several times. The sender may also include acknowledgments for member registrations in announcement messages. For example, it may delete the members that have successfully registered from the members' list.

An MFTP registration message conveys a member's intent to join the group. In closed groups, members must register and express at least their lack of interest in joining the group. Other parameters in the registration message include the sender's address, port, member address, and so forth. The sender's and members' parameters are repeated for the benefit of intermediate feedback aggregators. These entities may consolidate the registration messages, and just send client addresses to the sender, along with their intent, if necessary.

For data transmission, the sender (MFTP server) packages data into DTU messages. The server then sends the whole file to the private multicast group, without waiting to process feedback. Members are expected to store the received data and note any missing or corrupted DTUs. To avoid feedback implosion, MFTP groups DTUs into blocks. Thus, MFTP divides the file to be transmitted into equal-sized blocks and the blocks into data that can fit into a single DTU message.

Data transmission proceeds in several passes. The first pass contains the whole file, whereas the subsequent passes contain only the DTUs that were reported to be missing by at least one member. The DTUs are stamped with the same block and the DTU number in the header, so that a retransmitted DTU is indistinguishable from the copy that was originally transmitted.

The MFTP server requests the clients to report the file receive status after each pass. The *status response* MDP message from the clients contains a bitmap of the missing or corrupted DTUs per block. Retransmission passes may continue either until all clients receive data, or when a *delivery time*

limit timer expires. When the timer expires, the server just sends an abort message to the remaining clients.

Several real-world applications use MFTP for reliable and efficient data/file delivery to large groups. For example, the U.S. National Weather Service plans to collect Doppler Weather Surveillance Radar (WSR-88D) data from all radars in the country, and distribute them to all WSR-88D government and nongovernment users, using MFTP [7]. This centralized radar data collection and dissemination is called Radar Product Central Collection/Distribution Service (RPCCDS). RPCCDS plans to provide data as individual files via unicast ftp or as a tape archive (tar) file to closed or open groups as defined by MFTP.

Several large corporations including Ford Motor Company, Boeing, Toys ''R'' Us, and Sherwin Williams also use MFTP to send software or inventory database updates to dealerships, branch offices, or franchise stores. Traditionally, such updates are sent via unicast or on CD-ROMs shipped via mail, which may take a few days to a week to reach all the receivers.

10.3.2 Security requirements of MFTP applications

As noted earlier, MFTP is being used widely in corporate intranet settings as well as for file distribution over the Internet. There are several security requirements to consider, however. First, servers may want to ensure that only authorized receivers get access to the MFTP data. Next, most applications need to know that the data is being sent by the claimer source. Also, consider that some of the data that needs to be sent via MFTP could be proprietary or sensitive, and needs to be confidential. Examples of such data include a company's inventory, automobile dealer training material, and so forth.

Aside from the content protection and access control requirements, MFTP protocol security is also of concern. MCP messages contain NACKs, data reception status, registration requests, and so forth. Some MDP messages such as *abort* and *done*, also convey control information between MFTP entities. MFTP architecture also supports the use of aggregators to reduce the implosion of NACK messages. Modification of any of these messages may cause denial of service to one or more (when aggregation is involved) receivers.

10.3.3 Security solutions for MFTP

In this section, we explore possible solutions based on typical MFTP applications for the above-listed security requirements.

Access control and confidentiality

A rudimentary form of access control in MFTP can be achieved by the server allowing registration messages only from authenticated, predesignated hosts. This applies to closed as well as open groups. The disadvantage to this simplistic solution is that eavesdroppers or hosts on a LAN shared with an authorized host can get access to the confidential data sent using MFTP.

The solution to both access control and confidentiality of MFTP data is encryption and distribution of the keys to only the authorized members. We may use IPsec ESP [8] or MESP [9] for the multicast data transmission. For key distribution, we need a closer look at the applications' requirements. First, notice that members are generally interested in downloading a complete file via MFTP. Thus, the typical membership span is for the lifetime of a group. In other words, there is no need for rekeying due to membership changes.

During member registration, the server verifies whether the prospective member is authorized to receive the MFTP data. This might involve authentication of the host, and a check to see if the host is in the list of members that an MFTP server distributes as part of the announcement. For access control and key distribution, the MFTP server could use the GDOI [10] (see Chapter 5) registration protocol. Using GDOI, the MFTP server downloads a common key via unicast to all the authorized members.

Data origin authentication

Data origin authentication can be achieved in MFTP rather easily. First, let us examine the properties of MFTP applications that influence our choice of a source authentication protocol. In most cases, the data (a tar file, for example) is available to the sender in advance. The receivers wait until they receive the complete file. Thus the sender could compute a digest of the entire file and send the signed digest along with the file. While this solution is simple, it may lead to DoS attacks on the receiver's buffer space. Notice that the receiver needs to receive the entire file before being able to determine whether the contents are authentic. Even if the adversary can introduce only one fake packet, the entire file may need to be discarded. For similar reasons, hash chaining will not work. Thus the best solution is block hashing, so that each packet can be authenticated as it is received. Notice, however, that the computation and communication overhead in this approach is quite high. If overhead is a concern, we may use TESLA. The disadvantages there, however, are the need for clock synchronization and probabilistic authentication. Thus, receivers may need to request retransmission of some DTUs, since the TESLA security condition could not be verified.

Chapter 3 has a detailed description of the aforementioned source authentication schemes.

Protecting MFTP control messages

Most of the MDP and MCP control messages are sent via unicast, while some of them are sent by the server to the MFTP clients via multicast. Few others are sent upstream by the feedback aggregators. First, let us consider the control messages from the clients to the server or vice versa, without involving the aggregators. Notice that if GDOI is used for registration, there is a shared secret key between each receiver and the sender. The client can use this key (or a key derived from the secret) to authenticate the control messages using HMAC [11]. Servers can use techniques similar to those used for MDP data communication (see Section 10.3.3). Aggregators are typically involved in unicast communication, and those messages can be authenticated by sharing MAC keys between clients and aggregators, aggregators and aggregators, and aggregators and the server.

10.4 SecureGroups project

In military and other mission-critical applications, several group communication scenarios are common. For example, command and control offices disseminate information and orders to soldiers in the battlefield or training grounds, and spy satellites and planes send maps and photographs of target areas to multiple command and control centers. Apart from these one-to-many communication scenarios, there may be many-to-many communication between soldiers in the battlefield. Securing such group communications is essential. Another factor to consider is the possibility of group communication between coalitions of countries, with the potential for dynamic partition and merging of coalition partners.

While some of the command and control centers are stationary, most entities are mobile, and the communication is typically over wireless links to those mobile units. We must also consider that devices of heterogeneous computation and communication capacity will be in use. For example, soldiers may carry handheld computers, whereas tanks may be equipped with server-class machines.

The SecureGroups [12] project is part of a Defense Advanced Research Project Agency (DARPA)-funded program to investigate secure group communication in dynamic coalition environment. For scalable key distribution, the intradomain key management part of the IKAM protocol, described in Chapter 5, is being used for key distribution. When members are mobile, hierarchical node-based group key management (decentralized)

works better than LKH-based (centralized GCKS) approaches. If the AKDs are geographically distributed, mobile members can get access to new group keys as long as they are near one of the AKDs.

An airborne warning and control system (AWACS) plane could serve as the DKD, and tanks could serve as AKDs. This model works well for coalitions as well. Each coalition partner could form a secure group at the area level, and the primary partner could control the domain-level key distribution.

The DKD is responsible for generating the TEKs, and distributing them to authorized AKDs. The AKDs are responsible for enforcing access control to the area, as well as forwarding the TEKs to authorized area members. The DKD may be colocated with the sender, and that has a special application for military applications. For strict confidentiality, data transmission must be stopped during rekeying. Otherwise, evicted members may get access to some secret data sent during the rekeying process. If the DKD is colocated with the sender, we can stop data transmission without any latency.

10.4.1 Impact of mobility on group key management

Mobility complicates key management by allowing members to not only leave or join a session, but also to transfer between networks while remaining in the session. Since a mobile user may accumulate information about the local security services for each area he or she visits, the key management system must consider the level of trust to impart to these mobile members, and the performance implications should the member leave the session. Furthermore, as a member moves, the network latency between the member and the key management services may change, and result in additional performance degradation. Some rekeying strategies include: immediate rekeying of the area key, delaying rekeying until the member leaves the domain, or something in between (such as rekeying when a member holds more than a given number of area keys, or when more than a given number of nonmembers hold an area key).

Member registration is the most computationally expensive phase in group key distribution. Thus it is important that the registration process is not repeated when members move between areas. Thus AKDs should be able to securely hand off members to other AKDs. The All-AKD-key may be used for such hand-offs.

AKDs themselves may be mobile as well. That may result in some members being too far away from their AKD. In that case, members may join another AKD or elect an AKD from among themselves. Thus secure election algorithms are needed for AKD election.

10.5 Summary

Despite the slow deployment of multicast, there are several corporations and organizations using multicast for efficient data transmission in intranets or over the Internet. Multicast security will only increase multicast use. There are several applications that need, for example, the concept of a GSA and the key download feature of GDOI, for efficient secure group communication. Otherwise, the sender has to separately encrypt data for each member, thus undoing the benefits of multicast.

This chapter describes security requirements and solutions for real-world applications involving real-time and nonreal-time multimedia and data transmissions. Stock quote distribution and IP-television are the real-time applications, and MFTP is an example in the nonreal-time category. We also discuss security requirements and some solutions for key distribution to mobile members in wireless environments.

The lesson is that most of the applications' requirements can be satisfied with group security architectures, algorithms, and protocols proposed in the literature and being standardized at the IETF. Notice also that in most commercial applications, immediate rekeying of highly dynamic and large groups is not a requirement. Efficient group key initialization, batch rekeying, and source authentication schemes are of immediate concern.

There are also some open research problems such as the reliable transmission of rekey messages, and efficient group key management for mobile groups communicating over wireless links.

References

[1] Demchack, Tom., ''Memo on (Stock Market) Data Dissemination,'' http://admin.spa.org/liz/newfisd/news/sec_051001memo.html, May 2001, Securities and Exchange Commission (SEC) memo.

[2] Hardjono, T., ''Multicast Tunnels Using IPsec ESP,'' in *Proc. of the 10th IEEE Workshop on Local and Metropolitan Area Networks*, Sydney, Autralia, November 1999.

[3] Paul, S., *Multicasting on the Internet and Its Applications*, Norwell, MA: Kluwer Academic Press, 1998.

[4] Miller, C. K., *Multicast Networking and Applications*, Reading, MA: Addison-Wesley, 1998.

[5] Miller, K., et al., ''Starburst Multicast File Transfer Protocol (MFTP) Specification,'' draft-miller-mftp-spec-03.txt, IRTF, April 1998, work in progress.

[6] Gemmell, J., E. Scholler, and J. Gray, ''Fcast Multicast File Distribution,'' *IEEE Network*, Vol. 14, No. 1, January 2000, pp. 58–68.

[7] *Operations Demonstation Plan for the Radar Product Central Collection/Distribution Service (RPCCDS),* TechReport, U.S. Department of Commerce, June 2000.

[8] Kent, S., and R. Atkinson, ''IP Encapsulating Security Payload (ESP),'' RFC 2406 (proposed standard), IETF, November 1998.

[9] Canetti, R., P. Rohatgi, and P. Cheng, ''Multicast Data Security Transformations: Requirements, Considerations, and Proposed Design,'' draft-irtf-smug-data-transforms-00.txt, IRTF, June 2000, work in progress.

[10] Baugher, M., et al., ''Group Domain of Interpretation for ISAKMP,'' draft-ietf-msec-gdoi-04.txt, IETF, March 2002, work in progress.

[11] Krawczyk, H., M. Bellare, and R. Canetti, ''HMAC: Keyed-Hashing for Message Authentication,'' RFC 2104 (informational), IETF, February 1997.

[12] DeCleene, B., et al., ''Secure Group Communications for Wireless Networks,'' in *Proc. of the IEEE MILCOM*, Vienna, VA, October 2001, pp. 113–117.

CHAPTER

11

Conclusion and future work

Contents

11.1 IETF multicast security framework

11.2 Secure multicast data transmission

11.3 Group key distribution

11.4 Policy

11.5 Infrastructure protection

11.6 Future direction and final words

Multicast security standardization and research efforts have come a long way, but there is a great deal of work to be done as well. For example, there are several applications that currently need a way to send the same key securely to a large number of registered customers. This key may need to be changed daily or just monthly in some cases. Thus, for many applications, implementation and deployment of a GSA establishment protocol along with IPsec provides a working solution. More precisely, deployment of IPsec ESP for secure multicast data transmission, and GDOI or GSAKMP for key distribution meets the security requirements of many contemporary applications of multicast.

Commercial applications in general are expected to employ batch rekeying for efficiency. In such applications, immediate rekeying may be employed only to revoke a misbehaving member. Immediate rekeying to maintain strict forward and backward confidentiality may be a requirement in military applications and in interactive group conferencing. GDOI and GSAKMP do support transport of rekeying algorithms for immediate or batch rekeying.

However, there are some open areas for research as well as standardization in group key distribution. Reliable transport of rekey messages, secure group membership management, and GSA synchronization are areas where research and standardization efforts are ongoing. Several elegant solutions for the reliable transport of rekey messages have been proposed in the literature, and the standardization efforts are well under way.

Membership management and member deregistration are areas that need more work.

Group authentication and simple forms of source authentication can also be supported using IPsec ESP. Source authentication techniques that amortize the cost of digital signatures need MESP or AMESP, which have been proposed, and are on their way to standardization. Replay protection in the presence of multiple senders, and IPsec support for SSM are open areas for standardization.

Several group policy systems have been proposed and prototyped, and the GSPT standardization is making good progress. GSPT integrated with GDOI or GSAKMP, along with MESP (or AMESP), will provide protocol and policy support to most algorithms for multicast security.

In summary, sufficient work has been done in the IETF MSEC working group, for development and deployment of simple multicast security solutions. The previous chapter illustrates this point in detail, in the case of example applications such as MFTP and stock quote distribution. In the rest of this chapter, we summarize the standards and research work done in multicast and group security, and discuss ongoing and future work.

11.1 IETF multicast security framework

The IETF multicast security framework serves as a starting reference for application designers, in that they need to find solutions for secure multicast data transmission, group key distribution, and secure group policy management. The framework also points out how a centralized or a distributed design might incorporate the three problem areas (see Chapter 2). Note that the security framework only addresses issues related to multicast content protection. Multicast routing protocol protection is discussed in the latter sections of this chapter.

11.2 Secure multicast data transmission

There are three major requirements for secure data transmission: data confidentiality, message integrity (also known as data origin authentication), and protection against replay attacks. IPsec has been designed with a single source and one or more destinations in mind. Thus, IPsec as defined in RFC2401[1] provides confidentiality, rudimentary message integrity, and

1. RFC2401 is currently being revised, and the new IPsec specification may have additional support for multicast data protection.

protection against replay attacks for one-to-many multicast data transmissions. Multisender multicast introduces new challenges in that replay protection requires a separate per-sender sequence number to be maintained by all the receivers. General-purpose message integrity or data origin authentication is the topic of Section 11.2.2 (also see Chapter 3).

The use of the IPsec SA identifier for SSM is currently a point of contention. SSM is the prevalent model of multicast from one sender to many receivers. Recall that an IPsec SA is identified by the triple <*SPI, destination address, protocol ID(ESP/AH)*>. The SPI is chosen by the receiver and the destination address, and could be either a unicast or a multicast address. In multicast, there are multiple receivers and thus the receiver can no longer choose the SPI. Instead, in secure multicast protocols (e.g., GDOI or GSAKMP), the GCKS chooses the SPI. The destination address is a *class D* IP address. However, in SSM, a multicast group is identified by the tuple <*source address, destination address*>. Thus, (S_1, G) and (S_2, G) are two different groups, but the IPsec SA identifier does not allow expression of such distinct groups. The simple solution is to add the source address to the SA identifier, but that comes at the cost of modifying existing IPsec implementations.

Another suggestion is that a unique SPI, along with the destination address, is sufficient to avoid any SA identifier collisions. However, if different GCKSs manage the two groups (S_1, G) and (S_2, G), and if they select the same random number for an SPI, a member subscribing to the two secure groups cannot determine whether a particular packet was sent by S_1 or S_2, since the SA identifier would be same.

However, the current SPI definition works for single-sender multicast using classic IP multicast (also known as any-source multicast), where a group address alone uniquely identifies a multicast group, and thus the SA identifier triple is unique per secure group.

11.2.1 Group authentication

Applications where group members are content with knowing that one of the other members has sent the data can use IPsec ESP for maintaining message integrity. An example would be a group of routers that accept routing updates only from each other. We can form a secure group of such routers by distributing a common secret seed to all of them. That seed can be used to derive a group encryption key and a group authentication key. Note, however, that most applications have receivers that must be able to determine the source of the multicast data.

11.2.2 Source authentication

Several source authentication mechanisms, described in Chapter 3, amortize the cost of digital signatures over multiple packets, and in some cases over the entire content sent in a multicast session. While there is a choice of source authentication schemes, each of them is designed with specific application requirements and resource constraints at the sender or the receivers, in mind.

For example, hash chaining is best for reliable or semireliable (limited bursty losses) data transfers. Block hashing on the other hand facilitates immediate authentication of data over lossy channels, but at the expense of increased communication overhead. MAC chaining is best suited for applications where the sender and receivers have limited buffer space, are in loose synchronization with each other, and have bandwidth and computational constraints.

IPsec offers support only for group authentication, but work is under way at the IETF on the design of MESP and AMESP to support the various data source authentication schemes proposed in the literature.

Replay protection. Replay protection for multicast traffic is a topic of active discussion in the IETF IPsec Working Group. In IPsec, the sender uses a 32-bit sequence number[2] for replay protection. For single-sender multicast, this works without any modification. It is very difficult, if not impossible, to extend the concept of a single sequence number for multisender multicast. Receivers must maintain a separate sequence number per sender.

11.3 Group key distribution

For confidentiality or group authentication, the GCKS needs to share a common key with members of the group. GDOI and GSAKMP are protocols that serve the purpose.

For group key establishment, the GCKS needs to share a separate key with each member of the group. When a member joins, the GCKS can send the new group key encrypted with the old key to the group members. Membership revocation, on the other hand, requires the GCKS to send the new group key encrypted separately for each (or some) of the remaining

2. A 64-bit extended sequence number is currently being proposed in the revised version of ESP, for replay protection of long or high data rate sessions.

members. This naive form of rekeying does not scale to large groups. However, there are several ways to improve the efficiency of rekeying: by taking advantage of application requirements, or with smart key arrangements.

In MFTP applications, the sender divides data into blocks for efficient NACK aggregation. We might take advantage of that property and encrypt each block of data with a different group key. If access to the group is in terms of blocks of data or time, per-block encryption is appropriate. Examples of such applications include PPV event distribution, where blocks of time are sold. MARKS is a scheme designed for efficient key distribution to members whose departure time is known at the time of their join.

For groups where members may join and leave at any time, and where strict forward and backward confidentiality must be maintained, the LKH or its variants can be used for efficient rekeying. Military communications and interactive groups are examples of applications that require immediate rekeying.

For most commercial applications, batch or periodic rekeying is appropriate. The GCKS processes member joins or departures in a batch, for efficiency. However, members' departure times do not need to be known a priori. Thus, the GCKS may evict a member as necessary; conversely, members may choose to leave the group when they wish.

11.3.1 Reliable transport of rekey messages

Group rekeying messages are sent using multicast for efficient transmission. However, Reliable Multicast protocol standardization is ongoing at the IETF. Fortunately, rekey messages are often small, and take up only one or two IP packets for even large groups. Thus, it makes sense to design special-purpose reliable transport mechanisms for sending rekey messages.

The simplest form of reliable transport is to send a rekey message more than once. However, that does not guarantee that all members will receive the rekey messages. For such guarantees, application designers need to use the FEC-based transport of LKH keys. This scheme relies on member feedback to retransmit lost packets. After a few rounds of FEC, if any members have still not received the keys they need, the GCKS sends keys to members using unicast.

A more efficient scheme is based on the idea of assigning weights to LKH keys. The group key, since it needs to be sent to all members, gets the highest weight. LKH keys corresponding to parent nodes of leaf keys get the least weight. The weights are used by the GCKS to determine the redundancy of each key in a rekey message. This scheme also uses NACKs for feedback,

but the retransmission messages consist of repeated keys, with the number of repetitions based on the weights of keys. Standardization of reliable transport of rekey messages is underway.

Stateful and stateless rekeying

In stateful rekeying algorithms such as LKH, the GCKS uses keys sent via the rekeying protocol to protect KEKs as well as the group key. Thus, when stateful rekeying is being used, if a member was off-line during a rekeying instance, it could not decipher any future rekey messages. On the contrary, in stateless rekeying algorithms, the GCKS uses the keys it shares with members during the registration phase, to send the new group key. Thus stateless rekeying needs to be employed in applications where members may go off-line frequently. However, stateless rekeying is expensive for immediate rekeying.

Note also that stateless rekeying is not really reliable. It only handles members going off-line. If a rekey message is lost in transmission, members cannot decipher data encrypted with the group key sent in the rekey message.

11.3.2 Secure multicast group management

Secure multicast group management is a topic that has not received much attention. The GCKS may need to control data transmission, monitor the sender(s) and members closely, debug multicast data flow, and so on. For example, military applications may require that data streaming be stopped during rekeying. This is to ensure that members to be evicted do not receive any more secure data, once the decision to evict them has been made.

Another requirement may be for a GCKS to keep a close watch on the members. For example, it may be necessary to stop multicast data from flowing to LANs where all authorized members have gone off-line. This practice makes it difficult for nonmembers to get access to encrypted data for cryptanalysis. Membership management of large groups is a difficult problem. Consider that processing heart-beat messages from all authorized members is computation intensive, and does not scale. More importantly, this may be a requirement only in highly secure groups.

GSA synchronization

During the lifetime of a group, a member may find its GSA to be out of sync with the sender's GSA. There are several possible solutions, with subtle

variations, for the member to synchronize with the sender. We may require that the member in question to go through the registration process again. Alternatively, the member and the GCKS may keep the unicast SA established during initial registration active until the member leaves the group. Both of these schemes have advantages and disadvantages. The first is efficient because the GCKS does not need to maintain additional state, but is inefficient considering that the entire registration process must be repeated. The second approach avoids having to repeat the registration process, but at the cost of keeping additional state.

Deregistration

In the event of a graceful shutdown, or when a member does not want to be charged anymore for data, it may want to notify the GCKS. Thus, we may need a back channel for communication between members and the GCKS. This is similar to member registration, but serves the opposite purpose, and hence the name, deregistration.

Similar to the out-of-sync case, deregistration could be achieved by keeping the one-to-one secure communication channel between every member and the GCKS open, or by establishing a secure channel anew, for the purpose of deregistration.

11.3.3 Distributed group key management

A centralized GCKS is a single point of failure and attack, and is a performance bottleneck. Several solutions have been proposed to mitigate this problem. First, the services provided by the GCKS may be distributed to several entities. Second, we may separate the registration and rekey functionalities. Alternatively, the GCKS may delegate its registration as well as rekey functionalities to members or trusted third-party entities (e.g., IKAM, Iolus, or DEP; see Chapter 5).

It is also possible to distribute the computation and communication overhead during rekeying, to the members. Distributed versions of LKH and OFT have been proposed in the literature to achieve this purpose.

11.3.4 Secure group communication between mobile members in wireless environments

Consider mobile members served by an infrastructure of group managers, as in IKAM or Iolus. In addition to joining or leaving the group itself, members may move between subgroups (or areas, in the case of IKAM). Mobility may

necessitate rekeying, since members may join one subgroup and leave from another. Thus, evicted or departed members may hold keys of subgroups they visited even after they leave the group, which may lead to the compromise of group keys. In summary, subgroup managers must keep track of members that move and rekey their subgroup, when members who visited the subgroup eventually leave the group. A simpler, but more expensive solution would be for the subgroup to be rekeyed whenever a member moves from the subgroup.

Member mobility may be processed as a leave from one subgroup combined with a join into a second subgroup. However, if the first subgroup manager can ''introduce'' the member to the second subgroup manager, the transfer could avoid repeating the expensive registration protocol in its entirety. A member or the subgroup manager determining the right time for member hand-off is another interesting problem.

Subgroup managers themselves may be mobile, and that may result in members of a subgroup out of reach of a manager. In that case, the group manager may select a new subgroup manager, or the members may elect a subgroup manager.

11.4 Policy

Policy is an important piece of the multicast security puzzle in that it provides a way for the GCKS or the group owner to specify the mechanisms used to manage the group, and it provides information to members on how to use the keys sent, decrypt data, and so forth.

The group security policy token standardization goes a long way in achieving this purpose. That is a work in progress at the MSEC Working Group. Secure group announcements, group policy distribution, and updates are also problems that are currently under investigation.

11.5 Infrastructure protection

As mentioned at the outset and also in Chapter 8, infrastructure protection represents an important aspect of providing overall security to multicast. Two aspects of multicast infrastructure protection are the protection of the multicast distribution tree and membership access control at the subnet level.

The two generic classification of attacks are sender attacks and receiver attacks. In the first case, a malicious sender sends bogus packets to a

multicast group (addressed to the group's multicast address), with the effect that all members of that group receive the bogus packets. In the latter case, a user (nonmember of a group) issues a join request to the group, which in effect extends or pulls the multicast distribution tree to that user's subnet. The aim of this latter type of attack is to waste resources, such as bandwidth and state in routers, therefore content encryption does not aid in preventing such attacks. In order to protect against both types of attacks, security needs to be provided at the routing level (at the core of the distribution tree) and at the host membership level (at the edges of the distribution tree).

Unicast and multicast routing have been studied and developed for over two decades now, and thus several routing protocols exist today. Multicast routing protocols can be classified into those that pertain to intradomain routing (e.g., flood and prune protocols, dense-mode protocols, and core-based protocols) and those that focus on connecting domains together (interdomain). Chapter 8 discusses a number of these protocols.

The nature of routing in the Internet creates a number of security requirements relating to the routing protocols being deployed. These include the protection of control messages being exchanged by routing-related entities within a routing domain, origin authentication of route advertisements, source authentication in domainwide sending of both data and control packets, detection of routing misbehavior, and others. Chapter 8 discusses PIM-SM as a case study of a multicast routing protocol for a sparse distribution of receivers, and explores its security requirements and possible solutions that can be deployed. Also discussed in the context of PIM-SM security are the SKMP key management protocol for PIM-SM, and the MSDP interdomain protocol that connects together multiple PIM-SM domains.

Complementing routing protection is the need for membership management security at the edges of the distribution tree. Membership management provides control over members accessing the tree (sending or receiving packets), and more specifically control over nonmembers accessing the distribution tree. The basic IP multicast model defined originally in RFC1112 focused on the method by which a host would indicate to the multicast router the group(s) that host wanted to send to or receive from. The model did not incorporate any membership authentication features, and did not propagate host/user identity information within the distribution tree. In that original design, a host could even send to a group without receiving from it. The advantage of that basic model is its scalability.

Thus, in the broader picture, membership management security must include the authorized access of identified/authentic users/hosts to resources (connectivity, router resources, and data access) on a network, for a given multicast application. The specific requirements for membership

management security include host identification/authentication, authorization for a host to send/receive, and the authentication of control messages exchanged between a host and the multicast router at the edge of the distribution tree.

11.6 Future direction and final words

As mentioned in the beginning of this book, the area of multicast security is still in its infancy, even though the broader fields of IP multicast and of group communications have been studied for well over a decade now.

The work in this book represents a snapshot of the current state of affairs in the area of multicast and group security in the Internet, as reflected by the IETF as the primary standards-setting body for IP-related protocols. Much work still needs to be done in the area of multicast security. Currently, efforts are continuing in the IETF and IRTF in providing solutions to the multicast security problem.

Within the MSEC Working Group in the IETF, progress has been made in completing a number of work items, covering architectures, protocols, and algorithms:

Problem area 1: Secure multicast data handling. Currently both the MESP and AMESP draft proposals are being further refined and finalized, with the aim of progressing to proposed standard. Similarly, work continues on the TESLA proposal for source authentication. Close collaboration is maintained with the RMT Working Group: notably with the FEC-based efforts.

Source authentication remains a difficult problem to solve. As mentioned previously, public key cryptography tend to be computationally expensive for a per-packet usage. Hence, further research into source authentication—particularly in the context of lossy channels and unreliable transport—needs to be carried out.

Problem area 2: Management of keying material. The area of group key management has progressed considerably, with work on the GKM architecture nearing completion.

Corresponding to the GKM architecture are two GKM protocols: namely, the GDOI and GSAKMP (light version), both of which are also nearing completion. These implement the GSA model defined earlier.

Another related effort in this area is the Multimedia Internet Keying (MIKEY) protocol, which is particularly relevant to the real-time multimedia

environment. The protocol is designed to work with multimedia-related protocols, notably the Session Initiation Protocol (SIP) and the Secure Real-Time Transport Protocol (SRTP). This work item is also nearing completion.

Problem area 3: Multicast security policies. As mentioned previously, although seemingly straightforward, multicast security policy represents a difficult area of work, mainly due to the dependencies that policy has with other protocols, and with the specific area of application. Thus, for example, there is the issue of which part of a group policy should be made public through the announcement mechanism employed by the application.

Current work in the IETF has been derived from previous efforts related to the GKMP and GSAKMP protocols, both of which define a policy token to convey policy-related information and parameters. Some work from the Antigone system has also been introduced into the IETF.

Looking further ahead, there are some open issues that will need to be brought into the MSEC Working Group in the IETF once these issues are scoped and defined, and some solutions are proposed (i.e., in the GSEC Research Group). These include:

- *Many-to-many multicast security*. Here there are multiple senders and receivers. Although the current GSA model and definition is extensible to multiple senders, one open issue is with IPsec, notably, the need to use the source address when identifying SAs and related parameters.
- *Distributed group key management*. This is where multiple key servers and policy servers may cater to different subsets of receivers and different parts of the Internet with different membership densities.

  The IKAM architecture and the GSAKMP protocol have addressed this problem to different degrees, and thus will be the basis for conducting further work on this issue. Solutions will build on existing protocols from the various IETF working groups.
- *Multicast security in mobile wireless and ad hoc networks*. The area of wireless ad hoc networks represents a new, emerging paradigm in computing and communications. Deploying multicast in those environments will introduce new issues, including those pertaining to security.

  Although currently out of scope for the MSEC Working Group, this area of work will be addressed in the immediate future by the GSEC Research Group in the IRTF.

Finally, as these developments in the MSEC Working Group and GSEC Research Group continue—together with developments in other protocols and technologies—it is inevitable that some parts of this book will become out of date. Our hope, however, is that this book can be a reference point to newcomers and oldtimers alike, and that the book can provide fundamental concepts and notions that are solid and applicable, regardless of the future shape of the solutions to the multicast security problem.

Glossary

Administratively scoped multicast The concept of administratively scoped (admin-scoped) multicast, described in RFC2365, pertains to the scoping of multicast-related messages within a network. The key properties of administratively scoped IP multicast are, firstly, that packets addressed to administratively scoped multicast addresses do not cross configured administrative boundaries, and, secondly, that administratively scoped multicast addresses are locally assigned (and thus are not required to be unique across administrative boundaries). RFC2365 defines the administratively scoped IPv4 multicast space to be in the range 239.0.0.0 to 239.255.255.255.

Amortization of digital signatures Digital signing each data packet supports individual packet authenticity verification. However, this introduces large computational and communication overhead. Practical source authentication schemes amortize the cost of a digital signature over multiple packets.

Back channel In the context of group key management, the back channel is used by a member of the group to report its status and other messages to the KD or the GCKS.

Backward rekey The rekeying of a TEK in a group, in order to prevent new members from decrypting previous traffic in the group, which they may have recorded. A new TEK replaces an existing one whenever one or more new members join the group. It is also known as *backward secrecy* or *backward access control*.

Batch rekeying Group rekeying to process several membership changes, that is, member departures and/or joins. Typical thresholds for batch

rekeying include number of membership changes, member joins or departures. Also refer to *periodic rekeying*.

Blinding function One-way functions transform a given value x into y, such that given y, it is computationally infeasible to compute x. One-way functions are sometimes referred to as blinding functions.

Border gateway protocol version 4 (BGP-4) BGP-4 is the primary exterior routing protocol deployed today for the global Internet. Described in RFC 1771, BGP-4 is essentially based on the distance-vector algorithm, with some additional features. Overall, the function of BGP-4 in an AS is to advertise routes (or BGP paths) available in that AS to the other BGP peers in other ASs. This is done by BGP peers pair-wise exchanging their respective routing tables over a TCP connection. Since these routing tables are large, in practice only the changes (deltas) are exchanged. The set of BGP-related RFCs are RFC 1772 (BGP application), RFC 1773 (BGP experience) and RFC 1774 (BGP protocol analysis).

Building blocks approach The approach to solving a complex problem by subdividing the problem into manageable ''blocks.'' Here, each block must serve a well-defined function, and its relationship with other blocks must be clearly defined. Besides being more manageable, the approach inherently has a number of advantages, including the use and reuse of the functional block independently of the whole, and the ability to combine different blocks to satisfy multiple functions.

Categories of SAs The GSA model defines three categories of SAs that make up a GSA. The term ''category'' is used to indicate the three SAs that are used in group key management to achieve different purposes.

Class D address See *Multicast address*.

Closed and open secure groups *Closed secure groups* enforce privacy, not only on data transmission, but also on group announcements as well as on the group policy itself. Thus, group announcements are sent to preselected hosts or end-users (e.g., to the executive team within a company or individuals who have a certain security clearance in the military context). *Open secure groups*, on the other hand, make a public announcement about the group, in part indicating how one can get authorization to become a member.

Content owners Content owners, owing to the fact that they own the data and are interested in controlling distribution, are at the top of the chain

of control in issuing policy. They might specify high-level group security policy.

Control group A multicast group whose traffic consists only of control messages, including key management messages, typically from one multicast support entity (e.g., key servers, routers) to other support entities or group members. As an example, in the PIM-SM protocol, there is a special multicast group address reserved only for PIM routers.

Core based tree (CBT) protocol CBT is a multicast routing architecture that builds a single delivery tree per group, which is shared by all of the group's senders and receivers. Most multicast algorithms build one multicast tree per sender (a subnetwork); the tree being rooted at the sender's subnetwork. The primary advantage of the shared tree approach is that it typically offers more favorable scaling characteristics than all other multicast algorithms. The CBT architecture is described in RFC 2201.

Data group A multicast group whose traffic consists only of data messages from a member sender (i.e., end-user) to the other member receivers.

Data/key translation The process by an intermediary entity of decryption of a message (under a key), followed immediately by encryption of the message (under a different key). Typically, the intermediary entity or ''translation entity'' is not the end-customer or intended recipient of the message.

Denial of quality of service (DQoS) attack In the context of IP multicast, this attack is not a complete DoS in the sense of traditionally known DoS attacks, but rather an intolerable degradation in the quality of service provided by IP multicast to its application (e.g., PPV or video streaming).

Distance vector multicast routing protocol (DVMRP) DVMRP is an multicast routing protocol based on the distance-vector algorithm or the Bellman-Ford algorithm. Described in RFC 1075, DVMRP is an interior routing protocol, meaning that it functions within the boundaries of a single AS. Like in other distance-vector routing protocols, a router simply informs its neighbors of its routing table. A receiving router then recalculates the lowest cost of delivery of packets for each network path. This is simply done by choosing the neighbor who advertised the lowest cost. The result is then added into that router's routing table. Distance-vector algorithms are described in RFC 1058.

Distance-vector routing protocols Distance-vector routing protocols are those that are based on the distance-vector algorithm to compute available routes or paths. In distance-vector routing protocols, each router simply inform its neighbors of its routing table. For each network path, the receiving routers pick the neighbor advertising the lowest cost, and then add this entry into its routing table (for future advertisement).

Forward rekey The rekeying of a TEK in a group, in order to prevent departing members from decrypting future traffic in the group. A new TEK replaces an existing one whenever one or more members leave the group. It is also known as *forward secrecy* or *forward access control*.

Group address See *Multicast address*.

Group authentication Provides a weak form of message integrity in that receivers can verify whether data has been modified by nonmembers in transit. Also see *source authentication*.

Group key management The set of functions, protocols, and procedures for the management of cryptographic keys, security associations, policies, keying material, and other parameters pertaining to entities communicating within a group.

Group owner/creator (GOC) The GOC is a logical entity that *creates* policy. If policy is negotiated, the GOC contains the negotiated policy and fills any gaps. Otherwise, it interprets content owners' or application requirements to create the list of authorized members (e.g., ACLs), creates the group, and distributes policy to the group manager or controller.

Group policy distribution Policy negotiation may not converge in groups. Thus, a policy distributor (typically the group manager) needs to send information to members about how to decrypt and authenticate group data. This information includes: mechanisms used for data protection; group keys, and instructions on how to use them; expected behavior if a group member does not receive keys, and so forth.

Group security association (GSA) The multicast counterpart of the unicast SA. A GSA is defined to consist of an aggregate three SAs: namely, Category 1 SA (or SA1), Category 2 SA (or SA2), and Category 3 SA (or SA3). SA1 is used for the (bidirectional) unicast communications between a member and the KD. SA2 is used for the multicast of control messages (unidirectional) from the KD. SA3 is used for the multicast of data messages (unidirectional) from the sender.

Group Security Association Database (GSAD) The SAD, containing parameters pertaining to a GSA of a multicast group.

Group Security Policy Database (GSPD) The SPD, containing parameters pertaining to the SA policies of a multicast group.

Hash chaining A popular digital signature cost amortization technique for efficient source authentication. The sender divides the data stream into several blocks and signs the first/last block, which includes the hashes of other blocks. Those blocks contain hashes of one or more blocks in the stream, thus forming a chain of hashes.

Heuristics-based protection In the context of Reliable Multicast (RM), this refers to the use of heuristics-based algorithms to detect and react to DoS attacks to RM entities. This is in contrast to a cryptographic approach in which the integrity and authentication of all control messages are achieved through cryptographic means.

Immediate rekeying Rekeying after each membership change to enforce strict forward or backward access control. The sender may even stop data transmission to during immediate rekeying. Also see *batch* and *periodic rekeying*.

Independence of group key management The notion that, regardless of the scope of a group key management protocol, such a protocol must be independent of (or decoupled from) the underlying multicast routing protocol, thereby allowing it to be used in conjunction with various multicast routing protocols. For a group key management protocol to be independent from multicast routing protocols, the group key management protocol must not rely on the structures (e.g., multicast distribution tree) and mechanisms inherent to any particular routing protocol.

Internet group management protocol (IGMP) IGMP is the protocol used for a host to indicate its wish to join (leave) a multicast group. Initially described in 1989 in RFC 1112, IGMPv2 is described in RFC 2236, while the most recent version, namely IGMPv3, is described in RFC 3376. Multicast routers use IGMP queries and reports to find out if there are still active members of a group in a given leaf subnet. If multiple multicast routers are present on the same leaf subnet or LAN, then one of them is elected to be the Querier for that LAN. In this way, the Querier learns about group membership information, and is therefore able to forward multicast traffic of those groups to its leaf subnet.

Key encryption key (KEK) The cryptographic key used to encrypt the TEK and other keying material in a multicast group.

Key management algorithms In the context of group key management, an *algorithm* is used to maintain the logical arrangement of keys held by the members and other entities. Thus, for example, some algorithms maintain a logical tree structure where members are placed at the leaves of the tree, with each leaf having a key derived from other keys located in the interior nodes of the tree.

Key management architectures In the context of group key management, the term *architecture* is used to express the concept or notion that entities that are involved in group key management are purposely arranged or configured in relation to one another to achieve an intended effect. Two common architectures in group key management are *hierarchic* and *flat* architectures.

Key management protocols In the context of group key management, the term *protocol* is used to refer to the procedures, message exchanges, and message payloads that govern the behavior of the entities involved in supporting a group (e.g., servers), and those participating in a group (e.g., hosts).

Layered reliable multicast The category of Reliable Multicast protocols in which several multicast groups are used concurrently to deliver different sets of data packets; possibly at differing speeds. Redundancy is added to the complete set of packets to allow a receiver to establish the complete message from just a minimal of N out of M pieces (where $N \leqslant M$). By sending different sets over different multicast groups, the receiver has an increased chance of obtaining a sufficient amount of packets to reconstruct the entire message.

Link state routing protocols Routers participating in a link state routing protocol have a complete knowledge about the entire topology. Periodically, a router verifies the status of all its neighboring routers (i.e., link state) and then this link status information is advertised by the router to other routers in the AS through a link state advertisement, which is flooded throughout the network. This message will be received by all the routers within the routing domain. Each router then updates its view of the topology. To reach any destination network, the router first computes the routes by using Dijkstra's shortest path algorithm.

Logical key hierarchy (LKH) LKH is a logical hierarchy of keys for efficient group rekeying. Each node of the tree represents a key, with the root node representing the group key, and each leaf node representing a key known to exactly one member and the group controller/key server. Each member knows all the keys in its path to the root.

Metapolicy A metapolicy in the context of group security policy is a set of rules governing the handling of policy data. For example, a meta policy in groups may specify that a small portion of the group policy may be used for public announcement of the session information, while the rest must be disseminated securely (e.g., along with key distribution).

Multicast address Multicast datagrams have a *multicast address*, which, in IPv4 is the Class D address ranging from 224.0.0.0 to 239.255.255.255 (i.e., 224.0.0.0/4). This address range is also known as the multicast *group address*. A Class D address has its high-order four bits set to ''1110,'' with the following 28-bits being the group ID. Multicast is different from *broadcast* in that a *broadcast address* is used to send a datagram to all hosts within a subnet.

Multicast distribution tree The state information in routers and network elements within an AS, with regard to a multicast group. Since each of these routers typically has one incoming interface and multiple outgoing interfaces, the routers logically represent a branch point or fan out point in a logical tree-like structure that is rooted at the sender(s) within the group. Hence the term ''distribution tree.''

Multicast open shortest path first (MOSPF) protocol RFC 1584 describes the multicast extensions to OSPF. Building on the notion of areas within an OSPF routing domain, in MOSPF each router in an area maintains a local group database that holds information about the groups present on all the attached networks for which this router acts as designated router. The local group database is indexed based on <*multicast group, attached network*> pairs. For each multicast group in its group database, a router will flood group-membership link state advertisements throughout the area. These group-membership link state advertisements list the router that acts as the designated router for the network with group members. MOSPF is *data driven*, meaning that the multicast distribution tree is created only when the first datagram for a <*source, group*> pair is received, at which point the router that receives the datagram builds the shortest-path tree for this particular source using the Dijkstra's algorithm.

Multicast routers IP routers and network elements that run or execute an instance of a multicast routing protocol. Multicast routers typically are

also unicast routers, since they also run an instance of a unicast routing protocol.

Multicast routing protocols Routing protocols that are designed to deliver IP packets from one or more source IP addresses to a destination IP *multicast address*. The destination address is a special address in the Class D range of addresses in IPv4, and is also referred to as the group address. Examples are the DVMRP, MOSPF, and PIM protocols.

Multicast source discovery protocol (MSDP) MSDP connects multiple PIM-SM domains together, thereby allowing interdomain multicast routing through the sharing of source information between routing domains. Since each PIM-SM domain uses its own independent RP(s), it does not have to depend on RPs in other domains. The MSDP approach has the advantage that each PIM-SM domain is independent of the other, and domains that contain only receivers (no senders) may obtain data from groups without having to advertise the group membership.

Negative acknowledgment (NACK) Control message used in RM protocols to indicate loss of a packet at a receiver.

Negative-acknowledgment oriented Reliable Multicast (NORM) protocols The family of RM protocols in which NACKs are employed to indicate packets that were not successfully received. NORM protocols have two attractive scalability features; namely, that the source does not need to know which receivers are missing any given data packet (hence it doesn't need to keep track of the group membership), and that the first NACK sent by a receiver for a given missing packet can suppress further NACKs for the same packet sent by other receivers.

Open shortest path first (OSPF) protocol OSPF is a link state routing protocol described in RFC 1583. It is a protocol used for interior routing within an AS. Like other protocols that use the link state algorithm, each router in the AS holds only a partial map of the network. When links go down (and up), this link status information is advertised by the router to other routers in the AS through a link state advertisement, which is flooded throughout the network. Upon receiving a link state advertisement, these routers then recalculate the available routes.

Origin authentication Origin authentication in the context of interdomain routing—notably in the BGP-4 and S-BGP protocols—refers to the ability to authenticate the publisher of a given piece of routing update information. This ability is needed if the information has been passed along from one BGP peer to another (i.e., from one AS to another). In this way,

the recipient AS (i.e., BGP router) several AS hops away, can be assured of the origin and freshness of the information before that AS (i.e., BGP router) adds the information to its routing table. Another more generalized term is ''source authentication.''

Packet-level redundancy A good throughput strategy deployed in Reliable Multicast protocols, in which redundancy is added within each packet (or within separate accompanying packets), in such a way that a receiver needs only to obtain a certain subset of the packets in order to reconstruct the complete message. The sender must create encoding packets for each round (or collection) of data packets.

Periodic rekeying Group rekeying after a prespecified amount of time passes, typically resulting in batch processing of membership changes. Note that periodic rekeying is applicable even if no membership changes occur. Also refer to *batch rekeying*.

Positive acknowledgment (ACK) Control message used in Reliable Multicast protocols to indicate successful receipt of a packet by a receiver.

Principle of separation In the context of Reliable Multicast and multicast routing, the principle refers to the separation of the security for reliable multicast transport from the security for multicast routing. What this principle means is that the instances of security protocols and their application to RM protocols at the transport layer should be independent from the instances of those same security protocols applied to multicast routing protocols at the network layer.

Probabilistic packet authentication Some source authentication mechanisms may not be able to verify the authenticity of some packets due to the loss of the message integrity information contained in other packets. Thus packet losses during transmission may result in some packets being dropped. Such schemes only support probabilistic authentication of multicast packets.

Protocol independent multicast-sparse mode (PIM-SM) PIM-SM is a multicast routing protocol for a sparsely distributed population of receivers. Described in RFC 2362, PIM-SM has a number of features. PIM-SM maintains the traditional IP multicast service model of receiver-initiated membership. PIM-SM uses explicit joins that propagate hop by hop from members' directly connected routers toward the distribution tree. PIM-SM builds a shared multicast distribution tree centered at a Rendezvous Point (RP), and then builds source-specific trees for those sources whose data traffic warrants it. The term ''independent'' means that PIM-SM is not dependent on a specific unicast routing protocol.

Protocol instantiations (PI) A specific protocol realization that implements one or more building blocks.

Receiver access control Controlling access by receivers/members of a multicast group to the distribution tree. The aim is to allow only members of the group to attach to the distribution tree, to obtain IP packets destined to the group.

Receiver attacks Attacks to the multicast distribution tree by way of nonmembers (from a data/group perspective) simply joining the group. This causes the tree to expand, and multicast traffic to be forwarded to them. Even if the traffic content is encrypted by the source, the encrypted packets would still be forwarded regardless, thereby consuming bandwidth.

Receiver-initiated Reliable Multicast The underlying approach or strategy in RM, in which the receiver has the responsibility of detecting lost packets and reacting to the loss. Usually sender-initiated protocols use NACKs. For example, a receiver would send a NACK (toward the sender or source) for every packet it misses.

Repair nodes Repair nodes are used in a number of RM protocols to alleviate the problems of state explosion and message implosion. Local repair nodes prevent the sender from having to deal with lost packets and reduce the message implosion at the sender. Local repair nodes also allow the possibility of various entities performing the repair, including specialized servers or other group members.

Router assisted Reliable Multicast The category of RM protocols in which routers that are part of the multicast distribution tree help in providing reliability to the data. These routers may assist in the delivery of missing packets, in the aggregation of feedbacks, and in the suppression of certain feedback types (e.g., multiple NACKs).

Routing information protocol (RIP) RIP is one of the earliest developed unicast routing protocols for the Internet. Initially described in RFC 1058, it was replaced by RFC 1388 (in January 1993), and then by RIPv2, described in RFC 1723 (November 1994). RIPv2 allows RIP messages to carry more information, including payload for a simple authentication method. Another important addition to RIPv2 was the support for subnet masks. RIP is an example of a distance-vector routing protocol.

Routing protocols Communications protocols to create and control state information within routers and network elements, in order for them to route IP packets in the most efficient manner from a source IP address to a

destination IP address in the IP network. A routing protocol establishes the best path of delivery of datagrams (IP packets) from a source to a destination.

Secure-BGP (S-BGP) The S-BGP protocol addresses the security deficiencies of the BGP-4 protocol, such as the lack of origin authentication. S-BGP introduced certificates and digital signatures to allow an AS to prove its ownership of IP address blocks and AS numbers, to prove its identity, and to allow a BGP router of an AS to prove the router's identity and its authorization to ''speak'' on behalf of its AS. S-BGP also introduced a new BGP transitive path attribute, which carries digital signatures (or attestations) over the routing information sent in a BGP update message. A recipient of the update can thus use the signature and certificates to verify the address prefixes and path information mentioned in the message.

Sender access control Controlling access by senders/members of a multicast group to the distribution tree. The aim is to allow only members of the group to send IP packets destined to the group.

Sender attacks Attacks to a multicast distribution tree in which bogus packets with the correct multicast address are sent to the group by an attacker. This attack consumes bandwidth, since the packet would be delivered to all members. Although such attacks are also possible within unicast, the impact is magnified in multicast precisely due to the replication effect within the distribution tree.

Sender-initiated Reliable Multicast The underlying approach or strategy in RM, in which the sender or source has the responsibility of detecting lost packets and reacting to the loss. Usually, sender-initiated protocols use ACKs. For example, a receiver would send an ACK (toward the sender or source) for every packet it receives.

Server assisted Reliable Multicast The category of RM protocols in which the protocol relies on the use of (nonrouter) network entities to help in data delivery to recipients, or in the aggregation of feedback coming from recipients. These entities are typically not senders/receivers in the multicast distribution tree, and they are not defined as part of the multicast routing protocol.

Source authentication With source authentication of group data, receivers can verify to themselves that data has not been modified in transit. Source authentication may not imply non-repudiation. Source authentication techniques typically rely on digital signatures, but may use

amortization techniques to reduce per-packet computation and communication overhead.

Source specific multicast (SSM) The limitations and deployment problems with the current Internet standard multicast—the term used to refer to such current protocols as PIM-SM, MSDP, and M-BGP—have resulted in the SSM concept. In PIM it is technically a subset of PIM-SM for one-to-many multicast. SSM removes the need for an RP, for Class D address allocation (since all joins contain (S, G) pair information), and a simpler implementation (compared to the full PIM-SM). SSM is designed to work with IGMPv3. The SSM Working Group in the IETF was established in the year 2000.

Suppression nodes Suppression nodes are used in a number of RM protocols to alleviate the problems of state explosion and message implosion. Suppression nodes are used in particular by receiver-initiated reliable multicast protocols that employ NACKs for lost packets. When multiple receivers lose the same data packet, rather than multiple NACKs (for the same lost data packet) overwhelming the sender or source, a suppression node can discard NACKs referring to the same lost data packet, and forward only a single NACK to the sender or source.

Traffic encryption key (TEK) The cryptographic key used to encrypt data traffic in a multicast group.

Tree-based ACK (TRACK) protocols The family of RM protocols in which a tree structure is used to organize the delivery of ACKs or NACKs. Typically, the receivers are arranged into local regions, where each region is assigned a repair entity (or repair node) to aid in providing reliability. The tree structure essentially acts as the control channel over the data channel (namely, the multicast distribution tree).

Tree building phase The tree building phase in RM protocols is the phase in which the protocol constructs a tree structure used to organize the delivery of ACKs or NACKs.

Unicast routing protocols Routing protocols that are designed to deliver IP packets from a single source IP address to a single destination IP address. Examples are the RIP and OSPF protocols.

About the authors

Thomas Hardjono is the principal scientist at VeriSign. His current area of work covers new technologies in security, including certificates and PKI management, WLAN security, digital rights management, Web services security, cryptographic protocols and algorithms, and multicast and network security. Over the years, he has been responsible for a number of roles, ranging from software engineer to principal architect at a number of organizations, including NTT, Bay Networks, and Nortel Networks. Dr. Hardjono has been the cochair of the Secure Multicast Group (SMuG) in the Internet Research Task Force (IRTF) since its establishment in early 1998. He oversaw the transition of the work in SMuG into the Multicast Security (MSEC) Working Group in the Internet Engineering Task Force (IETF), and has been the cochair of the MSEC Working Group since its establishment in 2000. He has a Ph.D. in computer science from the University of New South Wales and is a member of the IEEE and ACM.

Lakshminath R. Dondeti is a senior research engineer in the Strategic Protocols and Standards group in the Advanced Technology Investments division of Nortel Networks. His doctoral dissertation was in the area of secure group communication, and he has been conducting active research in the areas of group key distribution protocol design and performance analysis for several years. He is an active contributor to the IRTF SMuG, GSEC, and IETF MSEC Working Groups, and was one of the first to implement the group domain of interpretation (GDOI) protocol. His current interests are in the areas of wireless and ad hoc secure multicast content and routing protocol security. He is currently a cochair of the IRTF GSEC Research Group. He has a Ph.D. in computer science from the University of Nebraska–Lincoln, and is a member of the IEEE ComSoc and ACM SIGCOMM.

Index

A

Access control, 4
  backward, 8
  forward, 8
  to group information, 117
  membership, 184
  MFTP, 265
  policy, 166
Access control lists (ACLs), 12
All or nothing flaws, 49
Announcement policy, 165
Anonymity, 25
Antigone policy framework, 177–78
Any Source Multicast (ASM), 193
Application layer MESP (AMESP), 6
Applications
  local area IP TV, 257–61
  many-to-many, 24
  multicast, 253–68
  nonreal-time multicast distribution, 261–66
  PPV, 11
  SecureGroups project, 266–67
  of secure multicasting, 13
  stock market data distribution, 254–57
Area multicast address allocation entity (AMAAE), 96, 100
Attacks
  DoS, 2
  downgrade, 176
  DQoS, 21
  receiver, 185, 278
  sender, 185, 278
Augmented chaining, 59
  defined, 59
  delay, 60
  illustrated, 60
Authenticated key exchange (AKE), 77
Authentication, 45–70
  of control messages, 183
  data integrity and, 47
  data origin, 265–66
  group, 6, 32, 39–40, 48–49
  hash chaining for, 55–61
  hop-by-hop, 215
  host, 209
  immediate, 66–67
  issues, 46–50
  MAC-based, 5
  MAC support, 47–48
  membership, 209–10
  message, 209
  origin, 194–95
  PIM, 198–99
  policy, 167
  problem components, 46–47
  source, 6, 32, 39, 47, 49–50, 195–96
Authorization, 208, 209–11

B

Backus-Naur Form (BNF), 170
Backward rekey, 82

Batch rekeying, 131–34
forward access control and, 133
illustrated, 133
joins in, 132
trade-offs, 132–34
Block hashing, 51–52, 54, 69
Bootstrap routers (BSRs), 191
Border Gateway Multicast Protocol (BGMP), 188
Border Gateway Protocol (BGP), 187
Building blocks, 34–42
advantages, 35–36
criteria, 37
functional, 38–42
motivation for, 34–38
multicast data confidentiality, 38–39
multicast group authentication, 39–40
multicast group membership management, 40
multicast key management, 40–41
multicast policy management, 41–42
multicast source authentication/data integrity, 39
sharing of standardized technologies, 37

C

CCNT, 11, 170–71
defined, 170
dynamic changes and, 170
See also Policy specification
Compromise recovery policy, 168–69
Content distribution network (CDN), 26
Control groups, 99–100
Control messages
authentication of, 183
MFTP, 266
protection of, 194
Core-Based Tree (CBT), 187
Cryptographic context negotiation protocol (CCNP), 175
Cryptographic context negotiation template. *See* CCNT

D

Data integrity, 39, 47
Data protection policy, 166–67
DCCM
CCNP for, 175
policy negotiation in, 175–76
Denial of quality of service (DQoS) attacks, 21
Denial of service (DoS) attacks, 2
distributed, 186
FEC-based protocols and, 247
Deregistration, 40, 277
Designated local repairer (DLR), 244
Digital signatures
individual packet authentication, 51–55
for source authentication, 50–55
Disclosure delay, 63
Distance vector multicast routing protocol (DVMRP), 20, 187, 188–89
defined, 188
"graft," 189
RPM algorithm, 188
Distributed group key management, 277
Domain multicast address allocation entity (DMAAE), 96
Downgrade attack, 176

E

Efficient multichained stream signature (EMSS), 58
Encapsulating security payload (ESP), 5–6, 68–69
definition, 68
multicast, 6

F

FEC-based protocols, 247–48
DoS attacks and, 247
redundancy, 247
security, 247–48
Forward error correction (FEC), 58
proactive/reactive solutions, 228
for reliability, 149
See also FEC-based protocols
Forward migration path, 78
Forward rekey, 82
Future directions, 280–82

G

GDOI protocol, 92, 117–26, 271
defined, 117–18
exchanges, 121–22

functional block diagram, 125–26
IKE and, 119–20
mapping, to GSA model, 118–19
membership management and, 177
new elements in, 120–22
new Phase 2, 122–24
payloads, 120–21
policy coverage, 176
policy distribution, 176–77
policy enforcement, 176–77
push message, 125
updating SAs, 124–25
See also Key distribution protocols
GKMP protocol, 92, 108–12
compromise recovery support, 109
cooperative key generation process, 109
defined, 108–9
entities, 109–10
group controller (GC), 109
group key controller identification, 110, 111
group key creation, 110–11
group key distribution, 111
group member (GM), 109
group rekey, 111, 112
group token (GT), 110
LKHs, 92
receiver-initiated multicast, 111–12
sender-initiated multicast, 110–11
See also Key distribution protocols
Group authentication, 39–40, 68–69
advantage, 39–40
applications, 48
defined, 6, 32, 48
providing, 48–49
in secure multicast data transmission, 273
See also Authentication
Group controller and key server (GCKS), 7
blinded key, 143, 144
defined, 7, 29
entities, 29, 30
for group key establishment, 274
for policy distribution/enforcement, 11
registration, 7, 8
secret key, 143
with STR, 151
Group controller (GC), 109
Group domain of interpretation, 9
Group key, 73
controller, 110, 111
creation, 110, 111
Group key distribution, 2, 111, 274–78
architectures, 9
distributed group key management, 277
reliable transport, 275–76
secure group communication, 277–78
secure multicast group management, 276–77
Group key management, 22, 73–88
abstractions, 80
algorithms, 10, 129–56
architectural issues/motivations, 93–94
architectures, 91–108
defined, 73, 87
distributed, 277
model for, 74–76
problem classification, 86–87
protocols, 108–26, 224
requirements, 76–79
scalability, 23
security requirements, 79–82
summary, 88
Group key manager (GKM) entity, 211
GROUPKEY-PULL exchange, 123, 124
Group management
delegation policy, 167
secure multicast, 276–77
Group member (GM), 109
Group membership
access control, 184
density, 93
dynamics, 8–9, 93
geographic spread of, 93
management, 40
protocol, 188
Group owner/creator (GOC), 162
Group rekeying, 8–9
GKMP, 111, 112
purposes, 131
Groups
control, 99–100
for data and control, 96–98
establishment, 113–15, 116
maintenance, 115–16
policy distribution in, 12
reliability of, 93
repair, 237

Groups (continued)
resilience of, 93
scalable registration/initialization of, 8
termination, 116
Group secrecy (GS), 68–69
Group security agents, 104, 105
defined, 104
keys, 105
subgroups and, 105
See also Iolus
Group security association (GSA), 9
database (GSAD), 41
defining, 85–86
definition illustration, 84
management, 82–86
mapping GDOI to, 118–19
model, 83–85
multiple SAs, 81
synchronization, 276–77
Group security association key management protocol. *See* GSAKMP protocol
Group security controller (GSC), 103
Group security intermediaries (GSIs), 103–4
Group security policy, 159–79
access control policy, 166
announcement policy, 165
authorization policy, 166
classification of, 164–69
components, 12
compromise recovery policy, 168–69
data, 160
data authentication policy, 167
database (GSPD), 41
data protection policy, 166–67
defined, 2, 159
distribution, 160, 163, 176–78
distribution framework, 164
enforcement, 160, 176–78
framework, 161–64
group management delegation policy, 167
group owner/creator (GOC), 162
identification, 116
key distribution policy, 168
membership policy, 166
negotiation, 160, 175–76
problem components, 160
reconciliation, 175–76
rekeying policy, 168
specification, 11, 169–73
summary, 178–79
updating, 163
verification of, 117
Group security policy token (GSPT), 169, 171–73
access control field, 172–73
authorization field, 172
components, 171–73
defined, 171
illustrated, 172
mechanisms, 173
security parameters, 171
standardization, 278
token identification, 171
validity of, 173
Group TEK (GTEK), 110
Group token (GT), 110
GSAKMP protocol, 9, 92, 112–17, 271
defined, 112
entities, 112–13
flow illustration, 114
group establishment, 113–15
group establishment without underlying SA, 116
group maintenance, 115–16
group termination, 115
header and message format, 112
policy distribution, 178
policy enforcement, 178
policy token, 116–17
protocol flows, 113–16
subordinate controllers (SCs), 113
See also Key distribution protocols

H

Hash chaining, 55–61
augmented chaining and, 59, 60
defined, 55
efficiency, 56
EMSS and, 58
graph representation, 56–57
graph representation illustration, 57
illustrated, 55
limitations, 56
piggybacking and, 59–60
variations, 57
See also Authentication

I

Identities, 25
IKAM, 94–103
  architecture example, 97
  basic model, 95
  defined, 92, 94
  key arrangement, 98, 100–103
  keys, 98–99
  multicast groups, 96–98
  objective, 94
  private keys, 101–3
  public keys, 101
  two-level hierarchy, 94–95
  See also Group key management
IKE
  defined, 7
  GDOI and, 119–20
  Phase 1, 120
  Phase 2, 120
  SA negotiation in, 174
  use of, 80
  uses, 119
Infrastructure protection, 12–13, 278–80
Integrity protection, 2
Interdomain routing protocols, 187
Internet Engineering Task Force (IETF), *xvii*, *xviii*, 42, 280, 281
  group key management architecture (GKMArch), 173
  MSEC Working Group, 280, 281, 282
  multicast security efforts, 25–26
  multicast security Reference Framework, 27–28, 272
  problem scope, 25–30
  Secure Multicast Reference Framework, 81
Internet group management protocol (IGMP), 2, 3, 21, 191–93, 207–14, 217
  defined, 191
  IGMPv1, 192
  IGMPv2, 192, 208
  IGMPv3, 193
  Leave message, 212
  membership authorization approaches, 210–11
  message authentication approaches, 212–13
  multicast access token, 210, 211, 213
  open issues, 213–14
  Querier, 190
  Query message, 212
  Report message, 212
  security, 207–14
  SMKD and, 215
Internet Research Task Force (IRTF), *xviii*, 280
Internet security association and key management protocol (ISAKMP), 9
  defined, 119
  framework, 76
  HDR, 123
  unicast SA as defined in, 79
  versatility, 78
Internet service providers (ISPs), 4
Internet standard multicast (ISM), 4–5
Intradomain routing protocols, 187
Iolus, 103–8
  data/key forwarding in, 107
  data translation, 106–7
  defined, 92, 103
  group security agents, 104, 105
  GSC, 103
  GSIs, 103–4
  hierarchical subgrouping, 104–5
  key translation, 107–8
  limitations, 108
  secure group communication, 106–8
  subgroup illustration, 105
  subgroup key management, 105–6
  See also Group key management
IP multicast
  communications, 3
  design, 3
  scalability, 1–2
IPsec, 36
Ismene, 169–70
  characteristics, 170
  conditionals, 170
  defined, 169
  policies, 169
  policy reconciliation, 174–75
  See also Policy specification

J

Join rekeying, 138–40
  complexity, 139, 146
  illustrated, 139
  with LKH, 138–39
  with LKH+, 140

Join rekeying (continued)
in OFT, 145–46
See also Rekeying

K

Key distribution center (KDC), 73
Key distribution policy, 168
Key distribution protocols, 108–26
GDOI, 117–26
GKMP, 108–12
GSAKMP, 112–17
Key distributors (KDs)
area (AKDs), 96, 97, 98, 99
control channels, 75
domain (DKDs), 96, 97
preferred, 75
trusted, 73
Keyed HIP (KHIP), 214, 215–16
defined, 215
goal, 215
nonces for replay protection, 216
periodic branch teardown/reestablishment, 216
unauthenticated control, 216
Keyed material management, 7–11, 12–13, 280–81
defined, 30
multicast data handling, 30, 31–32
multicast security policies, 31, 33–34
problem areas, 30–34
solutions, 32
Key Encryption Keys (KEKs), 10, 91, 150
hierarchical schemes, 156
shared, 148

L

Leave rekeying, 140–41
complexity, 141, 147
illustrated, 141
improving, 142
with OFCs, 141–42
in OFT, 146–48
See also Rekeying
LKH, 10, 136–42
defined, 130
improvements, 131
initializing, 137
join rekeying in, 138–39
key distribution rule, 136
key hierarchy example, 136
key tree member, adding, 137–38
LKH+, 131
join rekeying with, 140
requirements, 140
Local area IP TV, 257–61
approaches, 261
background, 258
Content Management Server (CMS), 259
defined, 257–58
illustrated, 259
network topology, 259–60
security requirements, 260–61
See also Applications
Logical key hierarchies. *See* LKH
Loose synchronization, 63

M

MAC-based source authentication, 61–68
packet processing, 65
of packets by sender, 64–65
TESLA and, 61–64, 65–68
See also Source authentication
Management
group key, 10, 22, 23, 73–88, 91–108
GSA, 82–86
keyed material, 7–11, 12–13, 30–34
multicast key, 22, 40–41
multicast policy, 41–42
multicast security policy, 33
unicast key, 22
Many-to-many multicast applications, 24
MARKS, 131, 134–35, 154
analysis, 135
defined, 130, 134
key distribution in, 135
Membership policy, 166
Message authentication code (MAC)
authentication, 5
authentication support, 47–48
computation, 67
keys, 62
MFTP, 13, 262–66
access control, 265
application use of, 264
confidentiality, 265

control messages, 266
data origin authentication, 265–66
data streams, 262
defined, 262
group support, 262–63
registration message, 263
security requirements, 264
security solutions, 264–66
server requests, 263
session example, 263
MSEC Working Group, 280, 281, 282
Multicast
addresses, 187
applications of, 253–68
content protection, 5–12
defined, 2–3
VPNs, 257, 258
Multicast access token, 210
concept illustration, 211
defined, 210
lifetime/management, 214
signature method on, 213
See also Internet group management protocol (IGMP)
Multicast address allocation server (MAAS), 96
Multicast data
authentication, 45–70
confidentiality, 38–39
encrypting, 3
handling, 30, 31–32
Multicast data protocol (MDP), 262
Multicast distribution tree
controlled access to, 20–21
creating, 186
protection, 182–83
Multicast ESP (MESP), 68–69
application layer, 6
defined, 6
goals, 68
IETF working group, 68
Multicast extensions to open shortest path first (MOSPF), 20, 187
Multicast ftp. *See* MFTP
Multicast key management, 22, 40–41
components, 40
defined, 40
Multicast policy management, 41–42
Multicast routing, 279
attack types, 184–85
overview, 186–94
security and, 185–86
security requirements, 194–97
Multicast routing protocols, 181–218
aim of, 186
classification of, 188
defined, 187
execution of, 181
interdomain, 187
intradomain, 187
security in, 214–16
Multicast security
components, 182–86
components illustration, 183
end-to-end protection, 182
IETF efforts, 25–26
IETF reference framework, 27–28
motivation for, 2–5
multicast distribution tree protection, 182–83
problem areas, 2
problem scope, 17–18
as problem to solve, 1
Multicast security policies, 11–12, 281
creation, 33
defined, 31
information, 34
management, 33
translation, 33, 34
Multicast source discovery protocol (MSDP), 188, 205–7
advantage, 205
defined, 205
overview, 207
peers, 206, 207
security, 205–7
Source-Active message, 206
Multimedia Internet Keying (MIKEY) protocol, 280

N

Negative acknowledgments (NACKs), 226
aggregated, 230
NORM protocols, 238
protection of, 230

Negative acknowledgments (NACKs) (continued)
protocols, 24
suppression, 227
Network operations center (NOC), 185
Nonreal-time multicast distribution, 261–66
defined, 261–62
MFTP, 262–64
security requirements, 264
security solutions, 264–66
See also Applications
Nonrepudiation, 6
NORM protocols, 238–47, 248–49
defined, 238
flat arrangement, 241–42
key arrangement, 240–43
logical hierarchical arrangement, 242–43
model illustration, 240
model of, 239–43
NACKs, 238
NACK-suppression entities, 238
PGM, 244–47
scalability features, 239
security requirements, 239–40
See also Reliable Multicast (RM) protocols

O

OFC, 131, 141–42
OFT, 142–48
defined, 131, 142–43
initializing, 144–45
join rekeying in, 145–46
key distribution, 143
leave rekeying in, 146–48
Open Shortest Path First (OSPF) protocol, 187
Ordered CBT (OCBT), 214
Organization, this book, *xviii–xix*
<*ind*>Origin authentication, 2, 194–95

P

Pay-per-view (PPV) application, 11
Perfect forward secrecy (PFS), 77
PGM, 244–47
defined, 244
designated local repairer (DLR), 244
key arrangement, 245–47
Last-Hop key, 245–46
leaves protection, 245
NACK confirmation (NCF) message, 244
security requirements, 244–45
source path message (SPM), 244
SPM-key, 246–47
tree protection, 245
See also NORM protocols
Piggybacking, 59–60
PIM-DM, 187
PIM-SM, 188, 189–91, 217
Assert message, 202
background, 197–98
bootstrap messages, 203
defined, 182
entities, 190–91
features, 189–90
Hello message, 202
Join/Prune messages, 202
link-local messages, 202–3
message attacks, 202
Register message, 202
Register/Stop message, 202
revised, security issues, 202–4
revised, solutions, 204–5
security, 197–205
Policy distribution, 160, 163
in GDOI, 176–77
GSAKMP, 178
Policy enforcement, 160, 176–78
in GDOI, 176–77
GSAKMP, 178
See also Group security policy
Policy negotiation, 160, 174–76
in DCCM, 175–76
example, 174
See also Group security policy
Policy server, 30
Policy specification, 169–73
CCNT, 170–71
GSPT, 171–73
Ismene, 169–70
languages, 173
See also Group security policy
Positive acknowledgments (ACKs), 226
aggregated, 230
protection of, 230

Pragmatic general multicast (PGM), 227
Private keys
  IKAM, 101–3
  pair-oriented, 102
  shared, 101–2
Protocol Independent Multicast (PIM), 3
  authentication, 198–99
  BSR public key, 198
  Dense Mode. *See* PIM-DM
  equal opportunity key, 198–99
  RP-key, 199
  Sparse Mode. *See* PIM-SM
  working group, 187
Public key infrastructure (PKI), 78, 101
Public keys
  BSR, 198
  IKAM, 101
  SKMP, 199–200

R

Real-time transport protocol (RTP), 36
Receiver attacks, 185, 278
Receivers
  heterogeneous, 67
  packet processing at, 65–66
  TESLA, 64
  TESLA packet processing at, 65–66
Reference framework, 27–28, 42
  centralized/distributed designs, 30
  defined, 28
  development of, 27
  GCKS, 29
  horizontal view, 28
  illustrated, 27
  policy server, 30
  sender/receiver, 29
  vertical view, 28
Rekeying
  batch, 131–34
  group, 8–9, 111, 112
  immediate, 133
  join, 138–39, 140, 145–46
  leave, 140–42, 146–48
  periodic, 131–32
  policy, 168
  stateful, 276
  stateless, 276
Rekey messages, 148–50
  reliable transport of, 10–11, 275–76
  repeated retransmission of, 148–49
  small, 10
Reliable Multicast Research Group (RMRG), 26, 226
Reliable Multicast (RM) protocols, 13, 18, 223–49
  ACK-based strategy, 227
  classification of, 225–29
  cryptographic protection vs. heuristics, 224–25
  defined, 223
  deployment, 225
  differentiating, 226
  entities, 93–94
  FEC-based, 247–48
  function of, 23
  generic security requirements, 229–31
  layer of protection application, 230
  NACK-based strategy, 227
  network entity participation/support, 228–29
  NORM, 238–47
  operation, 24
  PGM, 244–47
  principle of separation, 224
  protection requirements, 229–30
  RMTP-II, 232–37
  security of, 23–24
  security requirements, 229–31
  specific requirements, 230–31
  summary, 248–49
  throughput strategies, 226–28
  TRACK, 231–38
  TRAM, 237–38
Reliable Multicast Transport (RMT), 26, 34, 37
Reliable transport
  FEC for, 149
  of rekey messages, 148–50
  WKA for, 149–50
Reverse path multicasting (RPM) algorithm, 188
RMTP-II, 232–37
  defined, 232
  designated receiver nodes (DR), 232
  DR-Key, 236–37
  Group-Key, 235
  key arrangement, 235–37
  optional data access, 234

RMTP-II (continued)
receiver nodes (RN), 233
Region-Key, 235
RM-Key, 235
security requirements, 234–35
symmetric keys, 235
top node (TN), 233
tree structure, 234
See also Reliable Multicast (RM) protocols; TRACK protocols
Routing Information Protocol (RIP), 187
Routing tables, 187

S

SDR, 151, 156
description/analysis, 154
illustrated, 154
member revocation illustration, 155
for membership revocation, 152–54
See also STR
Secure-BGP (S-BGP) protocol, 195
Secure group communication, 277–78
SecureGroups project, 266–67
airborne warning and control system (AWACS), 267
defined, 266
key group management, 267
mobility impact, 267
See also Applications
Secure hash algorithm (SHA), 52
Secure multicast data transmission, 272–74, 280
group authentication, 273
requirements, 272
source authentication, 274
Secure multicast key distribution (SMKD) protocol, 214–15
goal, 214
hop-by-hop authentication, 215
IGMP and, 215
specification, 215
Secure Real-Time Transport Protocol (SRTP), 281
Security association databases (SADs), 36
Security associations (SAs)
management functions, 75
minimum number of, 82
SA1, 83, 85
SA2, 83, 85–86
SA3, 83, 86
updating, 124–25
Sender attacks, 185, 278
Service level agreements (SLAs), 159
Session Initiation Protocol (SIP), 281
Shared private keys, 101–2
Simple Key Management Protocol (SKMP), 197
key management approach, 200–201
key management keys in, 200
multicast groups, 201
for PIMv2, 199–201
restricted public keys, 199–200
SMuG
defined, 26
near-standard documents, 26
Reference Framework, 27
Source authentication, 6, 32, 39
defined, 48
digital signatures for, 50–55
for domainwide sending, 195–96
lossy streaming, 50
MAC-based, 61–68
mechanisms, 47
providing, 49
real-time streaming, 50
reliable bulk data transmission, 49
reliable streaming, 49–50
in secure multicast data transmission, 274
See also Authentication
Source path message (SPM), 244
Source-Specific Multicast (SSM), 193–94
‘‘channels,’’ 193
defined, 4, 193
model, 12, 194
motivations, 193–94
Star hashing, 51–52
defined, 51
illustrated, 52
Stateful rekeying, 276
Stateless key revocation algorithms, 150–51
Stateless rekeying, 276
Station-to-station (STS) protocol, 77
Stock market data distribution, 254–57
approaches, 256–57
background, 254
Consolidated Quotation System (CQS), 254, 255, 257

Consolidated Tape System (CTS), 254, 255, 257
defined, 254
illustrated, 256
network topology, 254–55
security requirements, 255–57
See also Applications
STR, 151, 156
description/analysis, 154
GCKS using, 151
key distribution, 152
for membership revocation, 151–52
revocation, 153
subsets, 152
See also SDR
Streaming
lossy, 50
real-time, 50
reliable, 49–50
Subgroup keys (SGKs), 105–6
Subgroup managers (SGMs), 9

T
Timed efficient stream loss-tolerant authentication (TESLA), 248
applicability analysis, 67–68
defined, 61
disclosure delay, 63
enhancements, 66–67
immediate authentication, 66–67
initialization, 63–64
intervals, 66
key chain commitment, 63
loose synchronization, 63
loss tolerance, 66
MAC key chaining, 62
MAC keys, 62
packet contents, 65
packet processing at receivers in, 65–66
receivers, 64
security condition, verifying, 65
time intervals, 62
TRACK protocols, 231–38, 248–49
defined, 231
model, 232
model illustration, 233
repair node, 232
RMTP-II, 232–37
scalability, 231
TRAM, 237–38
tree structure, 231
See also Reliable multicast (RM) protocols
Traffic encryption key (TEK), 110
TRAM, 237–38
defined, 37
public key cryptography in, 238
repair groups, 237
repair tree, 237
security requirements, 237–38
See also Reliable multicast (RM) protocols; TRACK protocols
Tree hashing, 52–55
block hash computation in, 54
computational overhead, 54
defined, 52
hash verification in, 54
illustrated, 53
Trust relationships, 24–25

U
Unicast routing, 279
protocols, 186
security requirements, 194–97
User datagram protocol (UDP), 38

V
Virtual private networks (VPNs), 13, 257, 258

W
Weighted key assignment (WKA), 149–50
defined, 149
for reliable transport, 149–50
repeat keys, 150