Secure Group Communication for Mobile P2P Groups: A Survey

Zinaida Benenson

Abstract

This report presents an overview over secure group communication paradigms in ad hoc networks, with especial attention to join and leave algorithms. Although classical group communication, after more than 25 years of development, is by now well understood and formalized, group communication for ad hoc networks is still under development. Firstly, the paradigm of groups in distributed computing is considered, distinguishing between classical, multicast and mobile ad hoc groups. Purpose, features, and system architecture of group communication systems are presented in the light of historical evolution of this concept. Further, security issues in group communication are outlined, focusing on the changes to the system architecture which are necessary for making a group communication system secure. Finally, an overview over group key management in group communication is given.

Contents

Chapter 1

Introduction

1.1 Motivation and Structure of this Report

The goal of this survey is to identify issues of secure group communication in mobile peer-to-peer groups with especial attention to join and leave algorithms. However, as the concept of group-oriented system architecture for distributed computing is much older than the concept of ad hoc networks, it is important to understand why and how the groups are used in distributed computing, and what security issues arise with the evolution of this concept. I identified the following topics concerning the concept of secure groups in distributed computing:

1. *Group-oriented system architectures*, see Chapter 2. Group-oriented systems were developed with different goals, such as service replication for high availability (the "classical" groups), multicasting in large groups, scalable applications in pervasive computing environment. Very often the developers do not take security considerations into account, or assume the existence of "some security service". This usually results in the necessity of integrating security mechanisms into the group management architecture *subsequently*.

2. *Secure group management* architectures, see Chapter 3. Work in this area concentrates on securing the group-oriented abstractions, such as the policies for the group organization, access control to the groups (e.g., which processes are allowed to join the group), consistent view on the events in the group (e.g., causal multicasting for consistent communication, consistent information about the group membership). Most of these works assume the existence of the *group key*, which is the

key shared between all group members for confidential communication, but do not go into the details of group key management.

3. *Group key management* is considered in the almost separate area of research, see Chapter 4. These works consider the initial *group key agreement*, which happens at the time of the group initialization, and the subsequent operations needed when the group membership changes. For example, if a member leaves the group, new group key is needed in order to prevent the former member from eavesdropping on the future group communication. On the other hand, if a new member joins the group, the new key should be distributed such that the new member cannot decrypt the past group communication.

In the next section, the results of this study are presented in a concise form. The rest of the report is dedicated to more detailed overview of the above three topics.

1.2 Executive Summary

1.2.1 Purpose and Functionality of Groups

The concept of group communication services in distributed computing exists for about 25 years, probably starting in eighties with ISIS, see [Bir93] for an overview. Groups are used to implement highly available, reliable and fault-tolerant applications, such as brokerage and trading systems, or distributed databases.

As summarized in one of the recent articles [ANRST05], group communication systems provide two main services:

- *Group membership:* The group members are provided with the list of all current group members, called the *view*, and are notified about any membership changes. Group membership can change due to voluntarily joins and leaves, as well as due to communication and process failures.

  An exception constitute *anonymous* groups, e.g., *multicast* groups [Cha05]. There, the members of the group subscribe to the information which is disseminated to the group members by some entities (publishers), but do not communicate with each other directly. In this situation, group members do not need (or even are not allowed) to know about other subscribers. In secure multicast setting, the group

membership is usually tightly coupled with the possession of the group key which enables the participants to decrypt the multicast traffic.

- *Reliable and ordered message delivery:* At least causal order of messages is important in any kind of distributed system. That is, if the content of message m is dependent on the content of message m', then each process should receive m' before it receives m. More stringent message orders are also possible.

The above services are realized through agreement on group membership and on message delivery order. As most kinds of agreement are impossible in asynchronous distributed systems, real systems usually use timeouts to determine communication and process failures. In unstable networks, this may result in correct processes being involuntarily excluded from the groups due to slow communication or network partitioning. This paradigm also applies to peer-to-peer and ad hoc groups consisting of wireless devices, often with relaxed consistency requirements [PH03, LSSI05].

1.2.2 Lifecycle of Groups

The lifecycle of the group consists of the following events.

Group formation (initialization) A group can be started by one or by multiple entities according to some criteria. For example, in ad hoc networks, devices which are in the proximity of each other are often members of the *proximity-based* groups. Other criteria, such as services which the group can offer to outsiders, or a collaborative task which the group should perform (e.g., a video conference, or a collaborative editing session) may apply to the group formation.

Usually, there is a *group initiator* which is responsible for the initial group organization. In dynamic peer groups, this entity is picked dynamically according to some criteria, such as the lowest process identifier, the largest remaining power of the device, or the advanced computational capabilities. The group initiator determines the initial group members. It may also select the *group leader*, and/or coordinate the group key agreement.

Group membership changes Following group membership changes can be identified:

Single join One process sends to the group a request to join.

Mass join Multiple independent process send the join request.

Merge (Fusion) Two or more groups are organized into one group. Sometimes one group becomes the subgroup of another group, sometimes a totally new group emerges.

Single leave (Deletion) One member either leaves the group according to some criterion. This may happen because it moves out of the group's proximity, or becomes unavailable due to some event, e.g., crash, running out of energy, or a communication failure, such as network partition or, in case of wireless communication, some obstacle.

Mass leave Multiple entities independently leave the group at the same time.

Partition (Fission) The group is divided into two or more independent groups.

On each membership change, special protocols are executed. These protocols are often driven by a single entity, the current *group leader*. The role of the group leader may be rotated between the participants on every membership change in order to ensure fairness, as the group leader consumes much more resources than the rest of the group. It sends and receives more messages than the other participants, and sometimes it is also responsible for the bulk of the calculation of the new group key [STW00, KPT04a, ABIS05].

The group ends its existence when the last member leaves the group, or according to some other criteria, depending on the system architecture and application.

1.2.3 Issues of Join and Leave Events

Join and leave events in all three forms (single, mass, and group join/leave) have implications on different levels of the group architecture.

Agreement on new group membership list Especially in case of join events, access to the group should be granted based on some rules. Usually, the access is granted by the current group leader. The group leader may take this decision solely on behalf of the group [LSSI05], or upon an agreement which is usually coordinated by the leader [STM04]. Agreement is also used in case of exclusion of crashed or corrupted members from the group. Decentralized agreement of the type of Byzantine Generals problem [LSP82] is rarely employed due to its very high communication cost and considerable delay.

Access control without agreement on the membership list In case no consistent group membership list is to maintained, it is still important to restrict access to the group, and especially to delete corrupted group members. In case the group membership is determined using a public key certificate [AM02], efficient revocation procedures must be developed.

New group key calculation In case the communication between the group members should remain confidential, new group key should be calculated on each join/leave event. The new key should be independent on the previous group keys, such that joining entities cannot compute the previous keys. On the other hand, the leaving group members should not be able to compute any future group keys.

Chapter 2

The concept of groups in distributed computing

This chapter considers three kinds of groups: classical groups, multicast groups, and ad hoc wireless groups.

2.1 Classical Groups

Classical groups serve as programming paradigms for developing of highly reliable and fault-tolerant systems. In the early stage of the development of group communication systems, small groups of 4-12 processes were used for providing replicated services, fault-tolerance and data replication. Sometimes, these groups had static configurations, join and leave were rare events. Emphasis in theses systems was on consistent system view at all processes. After the group concept proved to be very useful for development of distributed fault-tolerant applications, systems with large groups with participants connected via wide-area networks emerged.

The first rigorously specified groups communication service (GCS) is Isis [Bir93]. It was developed in eighties and found its application in the areas of financial computing (brokerage), database replication, fault-tolerant file storage, reactive control systems, publish-subscribe systems. Many other GCSs followed, such as Totem [AMMS$^+$95], Horus [vRBM96], Transis [DM96], Ensemble [RBD01], Spread [ANRST05].

A classical GCS provides two types of services: group membership service and reliable and ordered message delivery.

Group membership service provides processes with a list of currently alive and connected group members. This list is called *view*. Views may change

due to processes joining and leaving the group, but also due to process crashes, network partitions, and malicious attacks.

One of fundamental characteristics of a group membership service is its reaction to network partitions. A *primary-partition* membership service always runs only on one partition, which is called primary partition. Processes in the parts of the network which are not connected to the primary partition are considered faulty. This kind of service is appropriate in small stable networks, e.g., in wired LANs. On the other hand, *partitionable* group membership service allows multiple network partitions to run the service simultaneously. That is, if a group splits due to network partitions, all its parts continue working. If the network becomes connected again, the parts of the group merge. Partitionable group membership service should be used in WANs and mobile networks. However, there is a fundamental difference in designing a GCS for WANs and for mobile ad hoc networks. In WANs one can safely assume that the network would not partition forever, and any partitioned group will eventually merge. On the other hand, mobile ad hoc networks cannot give this guarantee. Therefore, GCSs for WANs usually cannot be used for mobile ad hoc networks.

Reliable and ordered message delivery makes sure that group members receive messages sent to the group according to some specified semantics. For example, messages may be delivered in causal order, such that if the content of message m depends on content of message m', then all processes in the group are guaranteed to receive m' before m. Another guarantee may be that any message sent to the group is delivered to the processes while all of them have the same group membership view.

A rigorous survey of classical GCSs can be found in [CKV01]. This paper provides a comprehensive overview of services offered by different GCSs, gives an insight (without going into details) into techniques used to implement them, and provides extensive literature on specifications, protocols and impossibility results.

2.2 Multicast Groups

With the development of the Internet, applications based on IP multicast, such as video-conferencing, or multicasting of stock exchange, demanded some changes in the group communication paradigm. This kind of multicast happens over very large wired networks. Multicast groups are very large (thousands of participants) and highly dynamical with with frequent joins and leaves. Group management is often coordinated by a single au-

thority (the multicast source). The group participants are only interested in receiving multicast messages, but do not need a consistent view on group membership. In case of many-to-many multicast, group management may be distributed between several authorities. Multicast groups are not very relevant for this study. A comprehensive overview can be found in [Cha05].

2.3 Wireless Ad Hoc Groups

As wireless personal devices such as laptops, PDAs and mobile phones emerged, the group communication paradigm has to be adapted to the new requirements. Developers of applications for wireless communication face challenges such as unstable links due to interference and obstacles, resource-limited devices, proximity-dependent connectivity. Considered groups are usually small to medium-sized (about 10 to 100 participants), do not have designated leader (all devices are considered as peers, the role of the leader may be rotated among the participants), experience high group dynamics, especially due to unreliable wireless links. In the following, I describe some interesting ideas and important projects from the research area of group communication for mobile ad hoc networks.

A group communication system (GCS) in a mobile ad hoc network should provide the two classical services (1) group membership and (2) reliable message delivery to group members (reliable multicast), despite frequent communication failures due to wireless communication and node mobility.

Classical GCSs consider node crashes and infrequent network partitions as possible failures. In addition to these types of failures, GCSs for mobile ad hoc networks also have to face link failures, sudden short-lived or permanent device disconnections, long-lived or permanent network partitions. These new types of failures pose very serious challenges on the specification of a GCS, as such tasks as agreement on the group view become impossible in presence of link failures, and furthermore, if the network configuration is highly dynamical, the classical GCS protocols may take very long time to stabilize, because after every change in network configuration, such protocol have to go through a communication-intensive recovery phase which, among other things, prevents processes from sending application messages.

Two directions of handling node mobility and wireless communication can be identified:

1. Some authors make additional assumptions on the system model, such as absence of node crashes and obstacles, or perfect communication

links [PB98, RHH01]. It is not clear, however, how these assumptions can be guaranteed in practice.

2. Another way is to adapt the principles of classical GCSs to the new network model. These GCSs change specifications of their services such that the services become possible in the harsh model of mobile wireless networks [BH02, BCM03, LSSI05]. They may relax the specification of group membership and reliable multicast probabilistically [BYFK06, LEH04], or allow that multicast messages be delivered only to the well connected and not too slow group members [Fri03].

2.3.1 Additional assumptions on the network model

In one of the first papers on group communication in mobile networks by Prakash and Baldoni [PB98], a *proximity layer* is suggested which monitors the changes in the neighborhood of a node. The proximity layer enables the execution of the D-proximity test with the goal to find all nodes within the distance D from a given node. Based on the proximity layer, a group membership layer is further proposed. In the first round of the three-round group formation protocol, a node which wishes to form a group sends a request to all nodes in its D-proximity. This request may contain further constraints on the prospective group members, such that not all nodes in D-proximity can join the group. In the second round a node can acknowledge the join request, and in this case the leader sends to the node a confirmation of its group membership.

The proposed group membership protocol is robust to the changes in the D-proximity information during all phases. The underlying assumptions of the protocol are that the nodes and links do not fail, and a collision-free medium access control protocol is used. These assumptions are needed to guarantee accuracy of the D-proximity test in presence of node mobility. The weakest point in the concept of proximity layer is the assumption of fault-free nodes and communication links. Moreover, the presented protocol for D-proximity test requires flooding the network with the discovery messages. Probably, if the D-proximity test would be exchanged for the D-hops-test which finds all hosts within D hops from the given host, the resource-consuming flooding might be avoided.

Roman et al. [RHH01] propose the concept of *safe distance* in order to maintain consistent group membership in ad hoc networks. They also assume that the hosts and the links do not fail, and that the only changes in network configuration are due to node mobility. The nodes move randomly

with bounded velocity. Two nodes are said to be at a safe distance if they are in the range of each other and will remain so for a predefined time period T assuming that they move in opposite directions with maximum speed. Time period T is determined by the time period needed for joining or leaving the group and the network latency. A group is called safe if any two group members are connected by a path along which all consecutive hosts are at a safe distance. Two safe groups are at a safe distance if at least two hosts, one in each of the groups, are at a safe distance.

The system consists only of safe groups with unique identifiers. Considering safe groups gives the advantage of predictable disconnections. An isolated host is considered as a group with only one member. Each group has a group leader which periodically gathers location information from all group members. Every host periodically broadcasts its location information and its group identifier. If it discovers another group at a safe distance, it passes this information to the group leader which initiates the *group merging* protocol which involves agreement on merging between the group leaders. On the other hand, if the group leader identifies that some group of hosts is leaving the safe distance of its group, it initiates the *group splitting* protocol where it determines the members and the leader of the new group.

2.3.2 Adaptations of classical GCSs to the new network model

We distinguish between deterministic and probabilistic ways to relax the specifications of classical GCS services such that they can be implemented in wireless ad hoc networks.

Deterministic approaches

Briesemeister and Hommel [BH02] extend the idea of partitionable group membership service and propose *localized* group membership service in order to cope with highly mobile ad hoc networks. Each process keeps track of its neighbors via heartbeat messages. Only neighbors of the process can be included into its group view. The authors rigorously specified their group membership service, proved its properties, and applied it to the inter-vehicle communication for traffic jam detection.

Fuzzy group membership is another idea to cope with the node mobility and wireless communication [Fri03]. It is incorporated into the JazzEnsemble group communication system for ad hoc networks [Jaz]. Instead of binary group membership where the members are either considered alive or dead, each group member is associated with a fuzziness level which depends on

ability of the process to send protocol and control messages. Members with various fuzziness levels are treated differently by the system. For example, usually messages are buffered by the group members until every group member acknowledges that it received the message. Only then the messages are delivered. Using fuzzy group membership, messages may be delivered even if some members with a high fuzziness level did not acknowledge them. To the best of my knowledge, JazzEnsemble is the only fully implemented GCS for ad hoc networks.

The next two works on GCSs do not present formal specifications, such that their properties are somewhat unclear.

A location-aware group membership middleware for collaborative applications in pervasive computing AGAPE is presented in [BCM03]. Instead of global views on group membership, location-dependent local views are considered. If for some application, the global group membership view is needed, it can be obtained by merging the local views. The network consists of cells, each cell contains a base station which manages local group membership views in its cell. In particular, it processes join and leave requests, and detects network partitions.

Liu et al. present a group management system for ad hoc networks under assumption that any node is able to detect coming and leaving neighbors (e.g., by means of beacons). Groups are formed according to several attributes. Local attributes apply to individual devices: location, proximity, available resources. Group attributes apply to the whole group, e.g., group size or reputation. Each group has a group leader which manages all group membership operations and group communication. The role of the group leader can be rotated in case of leader failure.

Probabilistic approaches

Luo et al. [LEH04] propose group communication with *probabilistic guarantees* on group membership and reliable multicast. In contrast to the classical group communication systems, no guarantees can be given that all members receive a message in the same view, as the notion of the global group view does not exist in the system. Instead, each group member only has a partial view on group membership. To update their partial views, members exchange membership information. Information is disseminated using *gossiping*: Each host randomly selects a certain number of group members from its view and sends the message to them.

Another way to relax the GCS specifications probabilistically is by utilizing random walks. In [BYFK06], a random walk based group membership

service provides every node with a partial group view such that the nodes in this view are selected uniformly and randomly out of all network nodes. In [DSW06], a mobile agent collects and distributes information about group membership in the network during a random walk. Their group membership algorithm is self-stabilizing such that eventually, with a high probability, all group members become aware of the current group membership.

Chapter 3

Secure group communication systems

Secure group communication systems (GCSs) should offer security guarantees on their two main services: group membership and reliable ordered message delivery to the group members (multicast).

There are two main kinds of secure GCSs: (1) GCSs secure against Byzantine (i.e., arbitrary, including malicious) failures of the group members, and (2) GCSs secure against outsider attacks under the assumptions that all group members behave correctly or fail benignly (e.g., crash, or fail to send or to receive messages). This kind of secure group communication is called the *fortress security model* in [RBD01].

In the following, we first identify requirements on secure group communication, and then give an overview over secure GCSs of both kinds for classical as well as for ad hoc networks.

3.1 Requirements on Secure Group Communication

Following issued should be considered when developing a secure GCS in the fortress as well as in the Byzantine security models:

Access control to the group One of the most important questions is admission to groups. Which entities are allowed to create a group, and on which grounds are new members admitted to the group? Furthermore, how may a member be excluded from the group? What happens in case the

group partitions due to network disconnection? In which cases are some groups allowed to merge? All named issues are actually trust issues and depend on the security policies of the system.

In the fortress security model, access control to the group plays a very important role, as any process admitted to the group is trusted. Of course, also in the Byzantine model, access control is needed in order to limit the number of Byzantine nodes. In classical groups, access control is usually realized using some centralized authentication service. Especially in ad hoc GCSs access control and trust management between the system participants, and between different groups is a very important issue, as these systems cannot rely on always available centralized trust management services, such as certification authorities or centrally managed access control lists.

Group authentication Apart from being able to control which processes may become a member of a group, the group members should also be able to prove to the outside world that they belong to a particular group. That means that the group as a whole should be able to prove its authenticity. For example, if a process wants to join a particular group, it should be impossible for any outsider to impersonate this group to the process.

Secure group membership protocol A group membership protocol should provide the participants with a correct estimation of currently alive and connected group members. It should be impossible (or hard) for an attacker to disrupt the protocol such that an incorrect view is accepted by the participants. For example, if the view change protocol is coordinated by a designated leader, an attacker should not be able to impersonate the leader, or to change in transit the list of the group members. Thus, integrity and authenticity of any group membership view should be assured. Moreover, in the Byzantine model, the protocol should tolerate a Byzantine leader.

Secure group communication After a group is established, the group communication must be protected from eavesdropping and impersonation attacks. That is, confidentiality and integrity of the messages sent in the group must be guaranteed. This is usually achieved through symmetric encryption and authentication using the *group key* which is known only to the group members. Group key management is often studied separately in the literature, as it is a complex issue involving several cryptographic protocols. Group key management includes the initial group key establishment and the subsequent key establishment associated with changes in the group

membership. On every group membership change, the group key should be recalculated in order to prohibit the members which left the group from eavesdropping on further communication, and the members which recently joined the group from decryption of past group messages. Group key establishment protocols are considered in Chapter 4.

On the other hand, if some guarantees exist on message delivery, such as causal or FIFO order, the group communication protocols should be protected from violations of these properties by outside attackers and by compromised group members (in the Byzantine security model).

Interplay between the group key establishment and the group membership protocols During the group membership change, secure group communication using the old group key is impossible, as it is not clear which of the old group members are going to leave the group. On the other hand, it is also impossible to establish a new group key, as the members of the new group membership view are not known yet. Therefore, all messages sent during the group membership protocol should be secured in some other way than by using the group key. For example, this may involve public key cryptography. Moreover, only after the new view is installed at all group members, the group key establishment protocol for this view can start.

3.2 Group Communication Secure against Byzantine Failures

There are not many GCSs which are secure against Byzantine failures. The reason for this lies in a very severe communication load in such systems, as the group members have to reach agreement on every single protocol message. This has a great impact on performance of the whole system. Accordingly, none of the current Byzantine tolerant GCSs was developed with wireless ad hoc networks in mind, as communication overhead is often considered prohibitively high even in wired networks.

Rampart Rampart [Rei95] is the first Byzantine tolerant GCS. It facilitates the realization of trusted services, such as certification authority. These services are replicated to achieve high availability, which makes it easier for an attacker to penetrate one of the replica servers. Therefore, the services should work correctly despite compromise of some of their components by

attackers. If less than one third of sites is compromised, the service continues working correctly.

Rampart assumes an asynchronous network and comprises a Byzantine tolerant group membership and reliable multicast protocols. In reliable multicast, all messages from the same sender are received by all group members in the FIFO order. These protocols are used to implement atomic multicast, which is core component for realization of replicated services. In atomic multicast, all messages should be received in the same order as they were sent. Access control in Rampart is based on public key cryptography. Each server is given the public keys of all other servers in the system. This happens either manually by an operator, or can rely on a PKI.

SecureRing The SecureRing GCS [KMMS01] also tolerates up to one third Byzantine group members. The processes are organized in a logical ring, and the communication is organized via multicasting a *token*. From this token, the members processor conclude which process will be sending messages in the next communication round. SecureRing utilizes a Byzantine fault detector which, among other things, detects the processes which do not properly handle the token, a Byzantine group membership protocol, and a secure and reliable message delivery protocol.

Secure JazzEnsemble The Byzantine tolerant extension to the JazzEnsemble GCS [Jaz] is described in [DFK06]. Although JazzEnsemble (see Section 2.3.2) was developed for ad hoc networks, its secure version can only be used in wired networks. Its performance, however, is much better than the the performance of the previously presented Byzantine GCSs at the cost of lower number of tolerated Byzantine processes. Thus, its membership protocol tolerates not more than 1/6 fraction of the participants to be Byzantine. According to the authors, the system can be extended to the ad hoc networks. However, the performance of such a system in ad hoc networks remains unclear.

The Byzantine tolerant version of JazzEnsemble was tested on up to 50 processors, whereas the group size of Rampart and SecureRing should be not more than 10 processors.

3.3 Fortress-Inspired Secure Group Communication

In the fortress security model, the members of a group are considered trustworthy, such that the groups have to be protected from outsider attacks.

Secure ISIS The first security architecture of the fortress kind was developed for ISIS [RBG92]. Access to the groups is organized via access control lists (ACLs). Although this method lacks flexibility in large networks with frequent group membership changes, it fitted the properties of ISIS very well, as group joins were supposed to be rare events in this GCS. The process which creates the group (the group initiator) specifies the corresponding ACL as the set of pairs (*owner*, *site*) describing the process owner and the site (machine) where the process runs. Thus, the creating process specifies users and machines which it trusts.

Moreover, the site on which the group initiator resides, also creates a public/private key pair for the group for the purpose of group authentication. Each group member receives the private key, called the *group authentication key*, after it joins the group. The public key is included into the group address. Thus, any process which joins the group firstly has to obtain the authentic address of the group. This is done by asking the name service, which should be implemented as a Byzantine tolerant, highly available service, see e.g. [RB94]. After the address is obtained, the joining process can verify the authenticity of the group it wishes to join.

On the other hand, the authenticity of the joining process can be verified by means of its certificate, which is issues by an *authentication service* which should, as well as the name service, be Byzantine tolerant and highly available. The certificates are actually issued to the sites where the processes run, and all cryptographic keys are also hold by the sites and not by the processes. It is done in order to prevent malicious processes residing on uncorrupted sites from improper using of these keys.

Further, the ISIS security architecture protects group communication using a symmetric *group communication key*. This key is created by the group initiator and is distributed to the group members during the join protocol. If a process sends messages to a group for the first time, its site creates a *connection* to all group members, and generates a symmetric authentication key for this connection. This key is distributed using the group communication key. Thus, different keys are used for each multicast source.

Finally, secure ISIS provides secure causal multicast. In uncorrupted groups, causality of message delivery is assured by the "normal" ISIS causal multicast protocols combined with authentication and integrity protection of messages. However, if a process is a member in several groups, and some of them are corrupted, then the processes in the corrupted groups were shown to be able to manipulate the ISIS protocols in the following way. Assume the process p is a member of groups G and G', and a message m was sent in group G before message m' was sent in group G'. The if the group G' is corrupt, it can manipulate the causal multicast protocol such that m' is delivered to p before m. This is called the *backdating attack*, and secure causal multicast provides protection against this type of attacks.

Enclaves Enclaves [Gon97] is system which enables secure ad hoc collaboration over the Internet. An enclave is created by a designated group leader, and users which want to join the enclave should be able to authenticate each other and the group leader by means of passwords or public key certificates. To join the group, a new user should authenticate to the group leader. Group communication is protected using a shared group key disseminated by the leader. It is not a GCS in the classical sense, as it is not fault-tolerant and does not employ group membership and reliable ordered multicast protocols. If the leader on an enclave fails or becomes disconnected, the the whole enclave ceases to exist.

Secure Spread Spread is a partitionable client/server based GCS for WANs. Each group includes a small number of servers where the resource consuming distributed protocols, such as group membership, run, and a large number of clients. This greatly improves performance and scalability of the system.

In [ANRST05], four versions of a security architecture for Spread are presented and analyzed. They differ in the place where the cryptographic operations and the group key management protocols are executed. In the *layered security architecture*, all cryptographic and key management operations are executed on the clients. The advantage here is that all cryptographic secrets are kept at clients, thus the end user does not need to trust the servers to keep his secrets. However, cryptographic operations and key management slow down the system's performance quite considerably.

On the other hand, the *integrated security architecture* executes cryptographic operations partly on the servers. Three variants of this architecture are presented. In the first variant, group key management and secure group

communication are offloaded to the servers. Clients establish secure channels with the servers. On the second variant, clients encrypt and authenticate messages using a group key which is generated by the servers for each view. This means, among other things, that the messages should be delivered in the same view in which they were sent, as otherwise, the group key will have changed (this is called "sending view delivery"). The third variant is similar, but the it supports a different GCS semantics, called "same view delivery", where the messages need not to be delivered in the view in which they were sent. In this case, the servers need to keep the group keys from the previous views in order to reencrypt the messages which were sent in previous views with the current group key. Most of Secure Spread variants use Tree-Based Group Diffie-Hellman key agreement protocol (TGDH) [KPT04b] which is described in Chapter 4. This protocol is *contributory*: it requires each participants to contribute a secret during the key generation.

Secure Ensemble Ensemble [RBD01] is a partitionable GCS which can support groups of hundreds of members. All processes have access to a trusted authentication and authorization services which can be organized in a centralized fashion (e.g., Kerberos) or in a distributed fashion (e.g., PGP). The emphasis is on use of symmetric key cryptography whenever possible. In particular, no Diffie-Hellman based group key agreement protocols are used. The emphasis in secure Ensemble is made on efficient merge and group rekeying protocols. Users may specify security policies in the form of access control lists (ACLs). Trust relationships are assumed to be symmetric and transitive. Only processes which mutually trust each other can form a group. If the security policy of a group member changes, it requests exclusion of the untrusted member and rekeying. Trusted processes are allowed to join a group without rekeying.

Secure Ensemble comprises a protocol for fast and secure merging of groups which were split due to networks partitions. After the network becomes connected again, the group components from the different partitions are likely to be using different keys, such that a common key should be established in the course of the merge procedure. Actually, in the course of the mutual key establishment, one group component securely switches to the key of another group component. To this purpose, the group leaders of each component engage in authenticated two-party Diffie-Hellman key exchange. Here the trusted authentication service is used. Then one of the group leaders (chosen deterministically on the basis of its name) gives its group key to another group leader, encrypted with the key resulted from

the Diffie-Hellman protocol. Only after this event the actual merge protocol can be started.

Secure Ensemble also utilizes a group rekeying protocol which efficiently supports joins, merges, and single leaves. It is called the Diamond protocol and organizes the group participants in a diamondlike graph. The nodes of the graph are participants, and the edges represent secure channels between the participants. Secure channels are created using authenticated Diffie-Hellman key exchange and are, therefore, expensive. For this reason, an algorithm makes sure that, in case of rekeying, the number of new channels to be created is kept as small as possible. For example, when two groups merge, the channels already existent in the groups should be used. The new key is distributed along the edges. The diamond graph is guaranteed to have logarithmic diameter, which favorably influences latency of the rekeying protocol.

3.4 Trust and Relationship Management in Mobile Ad Hoc Groups

As already mentioned above, there exists no rigorously specified secure GCSs which were specifically designed for ad hoc networks. For example, the AGAPE system [BCM03] is said to posses a security layer in its architecture, but no further description of this layer is given. Similarly, the GCS for ad hoc networks described in [LSSI05] uses a group key agreement protocol [ABIS05], and considers reputation as one of the attributes of the system participants, but no further explanations on how these security measures are integrated into the system can be found in the literature. Both systems are briefly described in Section 2.3.2.

Generally, the notion of a group in ad hoc networks is different from the notion of groups in the classical meaning of a group communication system. The groups do not longer consist of the processes, but more often of personal mobile devices closely associated with their users. The requirements of consistent group membership and reliable ordered message delivery to the group members are often relaxed, as shown in Section 2.3.

However, a specific and very difficult question about secure group management in ad hoc networks arises which was not a big issue in the classical groups. This issue is trust and relationship management, especially in connection with the access control to the groups. How should security policies of the system be organized in order to allow devices from different users/domains to cooperate securely on a collaborative task? For example,

if a device wants to join a group, how can the mutual trust be established?

The Resurrecting Duckling [SA00, Sta01] is one of the most famous paradigms for trust management in ad hoc networks. It introduces group formation according to the muster/slave principle, where a dedicated user's device, called *controller*, builds *secure transient associations* with other devices which belong to the same user. An association is built during the *imprinting* process, where the master device translates the shared key to the slave through some secure channel, e.g., physical contact. Topics such as change of the controller (the user gives the device to somebody as a gift, or lends the device), multiple controllers with different access rights (user versus manufacturer), and collaboration of devices which belong to different users are also considered. Further extensions to the Resurrecting Duckling policy model are presented in [Roh04]. They comprise the *ownership model* which helps to establish trust associations between devices which belong to the same user,and the *security policy definition language* which supports authentic key exchange and delegation mechanisms.

In [AM02], a distributed security architecture for access control to groups in ad hoc networks based on public key certificates is presented.

A new group is created by the creating a public/private key pair for the group by its first member, which is called *group leader*. Each new member gets a *member certificate* signed with the group key. The certificates contain the group public key (group identifier), the public key of the new member, validity period, and the signature. Leader can delegate leadership by issuing *leader certificates*. Multiple leaders make the group fault-tolerant. However, a protocol for coordinating the information on group membership is needed (it is not given in the paper). Groups can be nested. A subgroup is created using a *subgroup certificate*.

The main problem in this security architecture is the distribution of membership revocation lists. A best-effort solution is proposed where the revocation lists are propagated from member to member. In case the leader is revoked, the whole group may perish, as the membership certificates would not be valid anymore. The proposed solution utilizes multiple leaders and redundant certificate chains.

Chapter 4

Group key management for secure group communication

The evolution of group key management schemes goes in parallel with the evolution of the group concept in distributed systems. The first group key management schemes were developed for static wired networks with a small number of participants. Key management protocols for large and very large dynamic groups followed, to be used with, e.g., IP multicast. With the advances of wireless ad hoc networks, group key management protocols for this kind of networks emerged.

Group key management protocols are numerous, as well as surveys on this topic. The latter include [DMS99b], [Bha03], [RH03] [CS05], [DB05], [CS05], [Man06]. This chapter gives an overview based on these surveys, and presents some protocols which were developed for ad hoc networks.

4.1 An Overview of Group Key Management Algorithms

Group key is a secret key known to all group members and only to them. It is used to protect messages which are sent in the group from eavesdropping, and sometimes also from malicious changes by outsider attackers. Usually, this is accomplished by means of symmetric encryption and message authentication codes. In case of integrity protection, it is clear that if all participants share the same key, they are able impersonate each other and change message content. Therefore, if impersonation and tampering with message contents is an issue inside the group, other means than the group key should be used for integrity protection. In the following, we only con-

sider the case where all group members want to share a key for protection from outsider attacks.

Key management denotes the set of procedures which establish and maintain keys. To the key maintenance operations belong initial key establishment, key transport, and rekeying.

Key transport Group key management protocols can be classified according to the trust placed on group participants to the system and to each other. In *centralized* approaches, a designated entity (e.g., the group leader, or a key server) is responsible for calculation and distribution of the group key. In *decentralized* approaches, this function is distributed between several hierarchically organized entities. Both centralized and decentralized approaches are also called *key transport* protocols, as in these schemes, some designated entities create and securely transfer the secret key to the group members. In both approaches, group participants trust the key distribution entities to behave correctly according to the security requirements on the group key (see below), and be always available in case rekeying is needed.

Key agreement On the other hand, in distributed, or *contributory* key management protocols, all group participants cooperate in establishing the common secret. This kind of group key establishment is also called *group key agreement*. In this case, even if there is a designated group leader which plays a special part in the protocol (e.g., collects some protocol messages and makes some calculations on them), the security of the resulting key does not depend on it. That is, no group member can influence the key generation process or the resulting key to its benefit.

4.1.1 Security Requirements on Group Key Management

The following security requirements can be identified for group key management protocols.

- *Secrecy* is the most obvious requirement that the current group key be known only to the current group members and cannot be derived by non-participants at any time.
- *Forward secrecy* means that knowing past group keys should not enable calculation of current or future group key. In particular, past group members should not be able to decrypt group messages after they left the group. This means that a rekeying procedure should be started after every leave or partition event.

- *Backward secrecy*: knowing current group key, it is impossible to obtain any group keys used in the past.
- *Perfect forward secrecy* means that compromising any long-term secrets held by the participants should not reveal group keys.
- *Key independence*: knowing a subset of group keys does not enable calculation of any other group keys.

The decision whether satisfying all these requirements is actually necessary for the system depends on the security policy of. For example, if past group members can be trusted not to eavesdrop on the future group communication, or are known to be disconnected from the network, no immediate rekeying after a leave or partition event is required. Moreover, if a group splits due to a network partition, each part of the group may keep the group key in order to make the subsequent group merge more efficient.

Similarly, if a new member joins the group, it may be even necessary for it to know the past group communication. For example, a video conference participant which is late should be able to know what was going on in the part of the meeting which he missed.

4.1.2 Group Key Transport

The description of group key transport protocols in based on the surveys [RH03] and [CS05].

Centralized approaches

In the centralized case, a designated entity, called the key server in the following, distributes the group key to all participants.

In the *pairwise key approach*, the key server shared pairwise keys with each participant. For example, in [HM97], apart from pairwise keys and the group key, all current group participants know a group key encryption key (gKEK). If a new participant joins the group, the server generates a new group key and a new gKEK. These keys are sent to the new member using the key it shares with key server, and to the old group member using the old gKEK.

The *secure lock* [CC89] uses the Chinese Reminder Theorem (CRT) for traffic encryption. The key serves shares pairwise keys with each participant. The traffic is actually encrypted with a random key, and this random key is then "locked" such that it becomes the solution to a particular system of

linear congruences. The server computes the lock using the CRT and the pairwise keys, and sends the lock to the members together with the group message. It is not clear, however, how join and leave events can be handled by the system. The idea of secure lock is extended to the decentralized key management in [SLBE02].

Most efficient approach to rekeying in the centralized case is the *hierarchy of keys* approach. Here, the key server shares keys with subgroups of the participants, in addition to the pairwise keys. If in the previous approaches, each participant needed to be sent a separate message encrypted with its personal key, in the hierarchical approach most group members get the new group key encrypted with the key of its subgroup. On the other hand, group members have to store not only the pairwise keys, but also a number of subgroup keys, as the subgroups are organized hierarchically (e.g., in a tree). Thus, the hierarchical approach trades off storage for number of transmitted messages. Below, some approaches are outlined.

Logical key hierarchy was proposed independently in [WHA99] and [WGL00]. The key server maintains a tree with subgroup keys in the intermediate nodes and the individual keys in the leaves. Each node knows, apart from the individual keys shared with the key server, all keys on the path to the root. In root, the group key is stored. As the depths of the balanced binary three is logarithmical in the number of the leaves, each member stores a logarithmical number of keys, and the number of rekey messages is also logarithmic in the number of group members instead of linear, as in previously described approaches.

In [WGL00], an an extension of LKH to k-ary trees [WGL00] is proposed. One-way function trees (OFT) [SM03] enables the group members to calculate the new keys based on the previous keys using a one-way function, which further reduces the number of rekey messages.

In the Diamond protocol [RBD01], no hierarchical key tree is maintained. The participants are required to be able to establish pairwise keys on demand. The group is organized into a diamondlike graph with a designated leader. Graph edges correspond to the secure channels established between the nodes. Each time a member or a subgroup joins or leaves, the leader calculates the most efficient rekeying schedule and, if necessary, rebalances the diamond structure. Nodes pass the key to each other over secure channels according to the diamond graph.

The performance of the protocols is compared in [RH03] and [CS05] in detail, including message and storage overhead for join and leave events. The pairwise key approach exhibits linear complexity. Secure lock, although most efficient in number of messages, poses serious load on the server and

can be used only for small groups. All tree-based protocols have logarithmic communication and storage complexity at the members, and linear storage complexity at the key server. The Diamond protocol has linear message complexity and requires a small number of Diffie-Hellman key exchanges per membership change event.

Decentralized approaches

In the decentralized approaches, instead of one key server, there is a hierarchy of key servers which distribute the new keys to the group members, see [RH03] and [CS05] for examples and analysis.

4.1.3 Group Key Agreement

The contributory protocols are very enthusiastically studies in the cryptographic community. Indeed, they pose a very interesting problem. In addition to the mentioned above security requirements, a key agreement protocol should satisfy the following:

- The key management protocols should be *contributory* meaning that all members should contribute to the key, and it should be infeasible to derive the key in the absence of one contribution.
- There is no central authority, although a protocol leader may be selected for a particular execution. However, the role of the leader should be interchangeable.
- The derived key cannot be predicted or influenced by any protocol participant, including the leader. That is, the resulting key should be random.
- *Authenticated* GKA makes sure that all group member, and only them, compute the group key.

There exist more than 20 group key agreement (GKA) protocols, and there are also numerous comparison papers on their performance and security. Thus, [RH03] compare 8 protocols with respect to their number of rounds, communication and computational complexity. In [CS05], 11 protocols are analyzed. Finally, [Man06] presents over 20 GKA protocols which are analyzed with respect to their security. This survey does not include some of the protocols described in the previously mentioned surveys. Efficiency is not considered in the latter survey.

GKA protocols can be classified according to their communication pattern, to the use of broadcast communication, and to the use of Diffie-Hellman (DH) key agreement.

Ring based GKA In some protocols, members are organized in a ring. The first member starts the protocol by sending its contribution which is usually a DH public exponent g_r, where g is generator of a cyclic group and r a fresh random value, to its neighbor. Each participant then computes its contribution using a fresh random value and the message received from its neighbor. At some points of the protocol, a designated member (or members) broadcast the resulting partial group key values. The round and message complexity of these protocols is linear in the number of participants. The CLIQUES protocol suite [STW00] is an example of ring-based cooperation.

Hierarchical GKA In the hierarchical GKA protocols, the members are organized according to some structure. For example, in the Octopus protocol [BW98], the participants are divided into four groups. Each group agrees on a partial group key value using DH, and then the subgroup leaders agree on the final group key. In Hypercube [BW98], the users are organized into a d-dimensional hypercube, i.e., a graph where each user is connected to d other users.

Most popular form of hierarchical GKA protocols is tree-based cooperation. If a balanced tree is used, the costs of group operations, especially, the number of partial group key value calculations, is logarithmic. For example, in the Tree-based Group Diffie-Hellman (TGDH) protocol the participants are organized into leaves of a balanced binary tree, and compute the partial DH key values starting from the tree leaves, such that in each round, the partial values in one tree level are computed.

Protocols dLKH (distributed Logical Key Hierarchy) [RBD00] and dOFT (distributed One-way Function Tree) [DMS99a] are the contributory variants of the centralized LKH and OFT protocols (Section 4.1.2) which also use balanced binary trees for cooperation.

In the STR protocol [KPT04a] uses the linear binary tree (the binary tree with linear depth) for cooperation and provides communication-efficient protocols with especially efficient join and merge operations.

Trenary trees are used in [LKKR03] and [DBS04]. These protocols are based on three-party DH key exchange [Jou00].

Broadcast based GKA Broadcast based protocols have constant number of rounds. For example, in three-round Burmester-Desmedt (BD) protocol [BD05] each participant broadcasts intermediate values to all other participants in each round. The communication and computational load is shared equally between all parties. In the one-round protocol [BN03], a designated participant experiences a much more severe computational load than all other participants. Among other things, it has to compute $n-1$ public key encryptions, whereas all other participants are not required to to any cryptographic operations apart from computing the group key using a cryptographic hash function.

Efficiency comparisons As the number of GKA protocols is very large, a comparison of all these protocols would be too complex. Usually, one has to choose some GKA protocols which are appropriate in a certain situation, and then compare their complexities. Such comparisons are numerous in the literature. Usually, protocols are compared with respect to the join, leave, merge and partition operations. Appropriate metrics are round and message complexity, number of broadcast and unicast messages, and computational complexity (e.g., number of modular exponentiations). For example, [Bha03] compares CLIQUES, TGDH and STR. The performance of CLIQUES, TGDH, STR and the BD protocols protocols is also compared in [AKNRT04] and [ZAFL05]. The latter paper also proposes two new GKA protocols. In [Man05], the above protocols are compared not only with respect to the above metrics, but also with respect to message size and memory requirements.

4.2 Group Key Management in Ad Hoc Networks

In the setting of ad hoc networks, it is often argued that the contributory key agreement should be preferred, as participants of an ad hoc network do not trust each other. However, trust issues really depend on the considered system. For example, if all properly authenticated and authorized group members are trusted, and the authorization takes part before the group key establishment, there should be no objection to the centralized approach where the group key is calculated and distributed by the group leader. Centralized group key management algorithms are often much more computation and communication effective than the contributory ones, as the latter are often based of Diffie-Hellman key exchange, or on otherwise usage of modular exponentiations.

Asokan et al. [AG00] describe a password-based GKA protocol for small ad hoc groups. If people in a room want to share a group key, they share a short secret (e.g., write it on the blackboard), and then perform a series of modular exponentiations and communication in order to derive a strong shared secret from this weak shared secret.

Yasinsac et al. [YTCC02] propose a verifiably contributory GKA where each participant can be sure that its contribution was indeed used in the final key calculation.

Augot et al. [ABIS05] present This protocol is integrated into the GCS for ad hoc networks [LSSI05], see Section 2.3.2.

Manulis [Man05] distinguishes between protocols for homogeneous groups where all members experience approximately the same load, and the protocols for heterogeneous groups where some members have to execute more operations or to send more (or larger) messages than the other. Furthermore, he presents optimized versions of CLIQUES, BD, STR and TGDH which use elliptic curve cryptography and compares their performance.

Chapter 5

Conclusion

The concept of group communication in ad hoc networks is different from the classical group communication systems (GCSs), which were designed for LANs or WANs. Groups do not longer consist of the processes, but more often of personal mobile devices closely associated with their users. Classical GCSs provided to the application consistent group membership and reliable ordered group communication (multicast). In ad hoc networks, these consistency requirements have to be relaxed considerably. For example, group members are only provided with partial membership information, and with probabilistic guarantees on message delivery.

Also the purpose of groups has changed. Classical GCSs were designed as middleware for developing fault-tolerant applications. Members of ad hoc groups often engage in peer-to-peer collaboration without synchronization with the other group members, and the main questions are trust and relationship management, especially access control to the group.

In light of this paradigm shift, interesting questions arise which are worth investigation. For example, how can partial membership information be combined with group key agreement protocols. These protocols usually assume that all group members are known. Also the concept of fuzzy group membership, which yields efficient group membership protocols for ad hoc networks, has not been combined with a key establishment protocol yet. Another question to investigate is a possibility for two groups which use different key management schemes to merge and choose the appropriate key management scheme optimally.

Bibliography

[ABIS05] Daniel Augot, Raghav Bhaskar, Valerie Issarny, and Daniele Sacchetti. An efficient group key agreement protocol for ad hoc networks. In *First International IEEE WoWMoM Workshop on Trust, Security and Privacy for Ubiquitous Computing*, 2005.

[AG00] N. Asokan and Philip Ginzboorg. Key-agreement in ad-hoc networks. *Computer Communications*, 23(17):1627–1637, 2000.

[AKNRT04] Yair Amir, Yongdae Kim, Cristina Nita-Rotaru, and Gene Tsudik. On the performance of group key agreement protocols. *ACM Trans. Inf. Syst. Secur.*, 7(3):457–488, 2004.

[AM02] T. Aura and S. Maki. Towards a survivable security architecture for ad-hoc networks. In *Security protocols 9th international workshop*, LNCS 2467, 2002.

[AMMS+95] Y. Amir, L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, and P. Ciarfella. The totem single-ring ordering and membership protocol. *ACM Trans. Comput. Syst.*, 13(4):311–342, 1995.

[ANRST05] Yair Amir, Cristina Nita-Rotaru, Jonathan Stanton, and Gene Tsudik. Secure spread: An integrated architecture for secure group communication. *IEEE Trans. Dependable Secur. Comput.*, 2(3):248–261, 2005.

[BCM03] Dario Bottazzi, Antonio Corradi, and Rebecca Montanari. Agape: a location-aware group membership middleware for pervasive computing environments. *8th IEEE International Symposium on Computers and Communications*, 2003.

[BD05] Mike Burmester and Yvo Desmedt. A secure and scalable group key exchange system. *Information Processing Letters*, 94(3):137–143, May 2005.

[BH02] Linda Briesemeister and Günter Hommel. Localized group membership service for ad hoc networks. In *International Workshop on Ad Hoc Networking (IWAHN)*, pages 94–100, August 2002.

[Bha03] Raghav Bhaskar. Group key agreement in ad hoc networks. Technical report, INRIA, 2003.

[Bir93] Kenneth P. Birman. The process group approach to reliable distributed computing. *Commun. ACM*, 36(12):37–53, 1993.

[BN03] Colin Boyd and Juan Manuel González Nieto. Round-optimal contributory conference key agreement. In *PKC '03: Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography*, pages 161–174, London, UK, 2003. Springer-Verlag.

[BW98] Klaus Becker and Uta Wille. Communication complexity of group key distribution. In *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*, pages 1–6, New York, NY, USA, 1998. ACM Press.

[BYFK06] Ziv Bar-Yossef, Roy Friedman, and Gabriel Kliot. RaWMS: random walk based lightweight membership service for wireless ad hoc network. In *MobiHoc '06: Proceedings of the seventh ACM international symposium on Mobile ad hoc networking and computing*, pages 238–249, New York, NY, USA, 2006. ACM Press.

[CC89] Guang-huei Chiou and Wen-Tsuen Chen. Secure broadcasting using the secure lock. *IEEE Trans. Softw. Eng.*, 15(8):929–934, 1989.

[Cha05] Yacine Challal. *Group communication security*. PhD thesis, Universite de Technologie Compiegne, France, 2005.

[CKV01] Gregory V. Chockler, Idid Keidar, and Roman Vitenberg. Group communication specifications: a comprehensive study. *ACM Comput. Surv.*, 33(4):427–469, 2001.

[CS05] Yacine Challal and Hamida Seba. Group key management protocols: A novel taxonomy. *International Journal of Information Technology*, 2(1), 2005.

[DB05] Ratna Dutta and Rana Barua. Overview of key agreement protocols. Cryptology ePrint Archive, Report 2005/289, 2005.

[DBS04] R. Dutta, R. Barua, and P. Sarkar. Provably secure authenticated tree based group key agreement. In *Proceedings of ICICS'04*, LNCS. Springer-Verlag, 2004.

[DFK06] Vadim Drabkin, Roy Friedman, and Alon Kama. Practical byzantine group communication. In *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, page 36, Washington, DC, USA, 2006. IEEE Computer Society.

[DM96] Danny Dolev and Dalia Malki. The transis approach to high availability cluster communication. *Commun. ACM*, 39(4):64–70, 1996.

[DMS99a] L. Dondeti, S. Mukherjee, and A. Samal. A distributed group key management scheme for secure many-to-many communication. Technical report, Technical Report PINTL-TR-207-99, 1999.

[DMS99b] L. Dondeti, S. Mukherjee, and A. Samal. Survey and comparison of secure group communication protocols, 1999.

[DSW06] Shlomi Dolev, Elad Schiller, and Jennifer Welch. Random walk for self-stabilizing group communication in ad-hoc networks. *IEEE Transactions on Mobile Computing*, 5(7), July 2006.

[Fri03] Roy Friedman. Fuzzy group membership. In A. Schiper, A.A. Shvartsman, H. Weatherspoon, and B.Y. Zhao, editors, *Future Directions in Distributed Computing: Research and Position Papers*, volume 2584 of *LNCS*, pages 114 – 118. Springer, 2003.

[Gon97] Li Gong. Enclaves: Enabling secure collaboration over the Internet. *IEEE Journal on Selected Areas in Communications*, pages 567–575, 1997.

[HM97] H. Harney and C. Muckenhirn. Group key management protocol (gkmp) specification. RFC2093, 1997.

[Jaz] JazzEnsemble Homepage.
http://dsl.cs.technion.ac.il/projects/JazzEnsemble/.

[Jou00] Antoine Joux. A one round protocol for tripartite diffie-hellman. In *ANTS-IV: Proceedings of the 4th International Symposium on Algorithmic Number Theory*, pages 385–394, London, UK, 2000. Springer-Verlag.

[KMMS01] Kim Potter Kihlstrom, L. E. Moser, and P. M. Melliar-Smith. The SecureRing group communication system. *ACM Trans. Inf. Syst. Secur.*, 4(4):371–406, 2001.

[KPT04a] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Group key agreement efficient in communication. *IEEE Transactions on Computers*, 53(7):905–921, 2004.

[KPT04b] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Tree-based group key agreement. *ACM Trans. Inf. Syst. Secur.*, 7(1):60–96, 2004.

[LEH04] Jun Luo, Patrick Th. Eugster, and Jean-Pierre Hubaux. Pilot: Probabilistic lightweight group communication system for ad hoc networks. *IEEE Transactions on Mobile Computing*, 3(2):164–179, 2004.

[LKKR03] Sangwon Lee, Yongdae Kim, Kwangjo Kim, and Dae-Hyun Ryu. An efficient tree-based group key agreement using bilinear map. In *First International Conference on Applied Cryptography and Network Security (ACNS'03)*, pages 357–371, 2003.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[LSSI05] Jinshan Liu, Daniele Sacchetti, Francoise Sailhan, and Valerie Issarny. Group management for mobile ad hoc networks: design, implementation and experiment. In *MDM '05: Proceedings of the 6th international conference on Mobile data management*, pages 192–199, New York, NY, USA, 2005. ACM Press.

[Man05] Mark Manulis. Contributory group key agreement protocols, revisited for mobile ad-hoc groups. In *International Workshop on Wireless and Sensor Networks Security (WSNS 2005)*, 2005.

[Man06] Mark Manulis. Security-Focused Survey on Group Key Exchange Protocols. Technical Report 2006/03, Horst-Görtz Institute, Network and Data Security Group, November 2006.

[PB98] Ravi Prakash and Roberto Baldoni. Architecture for group communication in mobile systems. In *SRDS '98: Proceedings of the The 17th IEEE Symposium on Reliable Distributed Systems*, page 235, Washington, DC, USA, 1998. IEEE Computer Society.

[PH03] Pushkar Pradhan and Abdelsalam (Sumi) Helal. An efficient algorithm for maintaining consistent group membership in ad hoc networks. In *ICDCSW '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 428, Washington, DC, USA, 2003. IEEE Computer Society.

[RB94] Michael K. Reiter and Kenneth P. Birman. How to securely replicate services. *ACM Trans. Program. Lang. Syst.*, 16(3):986–1009, 1994.

[RBD00] Ohad Rodeh, Ken Birman, and Danny Dolev. Optimized group rekey for group communications systems. In *Network and Distributed Systems Security*, 2000.

[RBD01] Ohad Rodeh, Kenneth P. Birman, and Danny Dolev. The architecture and performance of security protocols in the Ensemble group communication system: Using diamonds to guard the castle. *ACM Trans. Inf. Syst. Secur.*, 4(3):289–319, 2001.

[RBG92] Michael Reiter, Kenneth Birman, and Li Gong. Integrating security in a group oriented distributed system. In *SP '92: Proceedings of the 1992 IEEE Symposium on Security and Privacy*, page 18, Washington, DC, USA, 1992. IEEE Computer Society.

[Rei95] Michael K. Reiter. The rampart toolkit for building high-integrity services. In *Selected Papers from the International*

Workshop on Theory and Practice in Distributed Systems, pages 99–110, London, UK, 1995. Springer-Verlag.

[RH03] Sandro Rafaeli and David Hutchison. A survey of key management for secure group communication. *ACM Comput. Surv.*, 35(3):309–329, 2003.

[RHH01] Gruia-Catalin Roman, Qingfeng Huang, and Ali Hazemi. Consistent group membership in ad hoc networks. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*, pages 381–388, Washington, DC, USA, 2001. IEEE Computer Society.

[Roh04] Christian Rohner. Building secure communities in spontaneously networked environments. In *4th Scandinavian Workshop on Wireless Ad-hoc Networks (AdHoc'04)*, 2004.

[SA00] Frank Stajano and Ross J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols*, pages 172–194, London, UK, 2000. Springer-Verlag.

[SLBE02] Oskar Scheikl, Jonathan Lane, Robert Boyer, and Mohamed Eltoweissy. Multi-level secure multicast: The rethinking of secure locks. In *ICPPW '02: Proceedings of the 2002 International Conference on Parallel Processing Workshops*, page 17, Washington, DC, USA, 2002. IEEE Computer Society.

[SM03] Alan T. Sherman and David A. McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE Trans. Softw. Eng.*, 29(5):444–458, 2003.

[Sta01] Frank Stajano. The resurrecting duckling - what next? In *Revised Papers from the 8th International Workshop on Security Protocols*, pages 204–214, London, UK, 2001. Springer-Verlag.

[STM04] Narasimha Prasad Subraveti, Soontaree Tanaraksiritavorn, and Shivakant Mishra. Flexible intrusion tolerant group membership protocol. In *ICPADS '04: Proceedings of the Parallel and Distributed Systems, Tenth International Conference on (ICPADS'04)*, page 437, Washington, DC, USA, 2004. IEEE Computer Society.

[STW00] Michael Steiner, Gene Tsudik, and Michael Waidner. Key agreement in dynamic peer groups. *IEEE Trans. Parallel Distrib. Syst.*, 11(8):769–780, 2000.

[vRBM96] Robbert van Renesse, Kenneth P. Birman, and Silvano Maffeis. Horus: a flexible group communication system. *Commun. ACM*, 39(4):76–83, 1996.

[WGL00] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Trans. Netw.*, 8(1):16–30, 2000.

[WHA99] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. RFC 2627, 1999.

[YTCC02] Alec Yasinsac, Vikram Thakur, Stephen Carter, and Ilkay Cubukcu. A family of protocols for group key generation in ad hoc networks. In *IASTED International Conference on Communications and Computer Networks (CCN)*, 2002.

[ZAFL05] Shanyu Zheng, Jim Alves-Foss, and Stephen S. Lee. Performance of group key agreement protocols over multiple operations. In *International Conference on Parallel and Distributed Computing Systems (IASTED PDCS)*, pages 600–606, November 2005.