SECURE GROUP COMMUNICATION IN AD HOC NETWORKS

by

Yuying Ding, B.Sc.

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of

Master of Science
School of Computer Science

Carleton University

Ottawa, Ontario

August 7, 2002

©2002, Yuying Ding

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisitions et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-93974-X

Our file Notre référence

ISBN: 0-612-93974-X

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

The Internet has triggered the development of group-oriented applications such as audio and video conference, pay-per-view and distributed simulations. Furthermore, the need for emergent group collaborations, such as disaster recovery or ad hoc conference, has also materialized. The emergence of ad hoc networks provides opportunities for this kind of ad hoc group collaborations. Although research in the area of group communication is one of the hot spots in the present networking field, there is still no ideal solution for accomplishing group collaborations securely and conveniently in highly dynamic situations. The motivation of this thesis is to find a simple and efficient solution for, but not limited to, secure group communication in ad hoc networks.

A new and practical Secure Code Group Communication Mechanism (SCGCM) is presented in this thesis. By applying the orthogonal property of secure codes to group communication, SCGCM not only provides source authentication and data confidentiality but also supports requirements for highly dynamic group communication. Moreover, it effectively addresses both scalability and flexibility.

Although the key objective of SCGCM is to provide ad hoc secure group communication, it works equally well for other kinds of group communications such as pay per view, video conference, etc. Some applications of SCGCM are also presented in this thesis.

Acknowledgements

I would like to thank my supervisor, Professor Evangelos Kranakis, for introducing me this challenging and promising research area. I am also thankful for his strong motivation and support during my research.

I would also like to thank Professor Michel Barbeau for his very helpful and motivational comments and suggestions on my thesis.

In addition, I thank my friends Jeyanthi Hall and Marie-Lise Blain for their kindly assistance in proofreading this thesis.

Table of Contents

Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Tables	viii
List of Figures	ix
1 Introduction	1
1.1 Motivations of the Research	3
1.2 Contributions of the Thesis	4
1.3 Thesis Outline	5
2 Group Communication and Wireless Ad Hoc Technologies	6
2.1 Group Communication Scenarios	7
2.1.1 One-to-many communication	7
2.1.2 Many-to-many communication	7
2.2 Secure Group Communication in Ad Hoc Networks	8
2.2.1 Ad Hoc Networks	8
2.2.2 Security Issues and Requirements of Group Communication in Ad Hoc Networks	12
2.3 Overview of Research on Ad Hoc Networks Security	19
2.4 Overview of Research on Group Key Management	21
2.4.1 Tree Key Graph	22
2.4.2 One-way Function Tree	28
2.4.3 Authenticated Group Diffie-Hellman (A-GDH.2)	32

3	Secure Code Group Communication Mechanism (SCGCM)	37
3.1	Notations and Definitions	38
3.2	SCGCM Overview	40
3.2.1	Assumptions	41
3.2.2	Overview of Data Secure Codes Exchange	42
3.2.3	Overview of SCGCM Secure Group Communication	43
3.3	SCGCM Basic Algorithm	43
3.3.1	Initial Secure Code Exchange	44
3.3.2	SCGCM Secure Group Communication	49
3.3.3	Proof of the correctness of SCGCM Encryption Algorithm	51
3.3.4	Secure Code Update	54
3.3.5	SCGCM(BA)—Join Protocol	56
3.3.6	SCGCM(BA)—Leave Protocol	57
3.4	SCGCM Extension Algorithm I	58
3.4.1	SCGCM(EX_I)—Secure Code Exchange	60
3.4.2	SCGCM(EX_I)—Encryption	62
3.4.3	SCGCM(EX_I)—Decryption	63
3.4.4	SCGCM(EX_I)—Secure Code Update	63
3.4.5	SCGCM(EX_I)—Join Protocol	64
3.4.6	SCGCM(EX_I)—Leave Protocol	65
3.5	SCGCM Extension Algorithm II	66
3.5.1	SCGCM(EX_II)—Secure Code Exchange	67
3.5.2	SCGCM(EX_II)—Encryption	68
3.5.3	SCGCM(EX_II)—Decryption	70
3.5.4	Proof of the Correctness of Extension Algorithm II	70
3.6	Discussion of SCGCM	73
3.6.1	Comparison with other solutions	73
3.6.2	Characteristics of SCGCM algorithms	79
3.6.3	Drawbacks of SCGCM algorithms	81
4	Case Studies of SCGCM Applications	82
4.1	One-to-many communication	83
4.1.1	Pay Per View	83
4.1.2	News Subscription	85
4.2	Many-to-many communication	87
4.2.1	Video Conference	87
4.2.2	Ad Hoc Collaboration	89
5	Conclusions	92

List of Tables

3.1	Notations	39
3.2	Comparison of SCGCM With Other Solutions	74

List of Figures

2.1	An Example of an Ad Hoc Network	9
2.2	Join Protocol for a Tree Key Graph	24
2.3	Leave Protocol for a Tree Key Graph	27
2.4	An Example of a One-Way Function Key Tree	29
3.1	An Example of M_i 's Secure Code Table	44
3.2	Code Exchange Secure Code (CESC) exchange	45
3.3	Code Exchange Secure Code (CESC) transmission	46
3.4	The Impacts of a compromised CESC	56
3.5	An Example of M_i 's Extended Secure Code Table	61
3.6	SCGCM(EX.I)—The Format of a Message sent by a Member	62
3.7	SCGCM(EX.II)—Encryption in Cipher Block Chaining (CBC) mode	69
3.8	SCGCM(EX.II)—Decryption in Cipher Block Chaining (CBC) mode	71

Chapter 1

Introduction

Group-oriented collaborations, such as audio/video conference, Virtual Private Network (VPN), and distributed fault-tolerant computing are among the most important applications of today's networks. These distributed applications require secure transmission of messages among members in a group. A traditional mechanism for achieving this goal is to use a shared group key to encrypt and decrypt the traffic within a session.

When a new member joins the group, the group key must be changed to prevent the new member from accessing previous communications; this is called *backward access secrecy*. Similarly, when a member leaves the group, the group key must also be changed to ensure the confidentiality of future messages; this is called *forward access secrecy*. The issues that involve group key generation, distribution, revocation, and storage are collectively called Group Key Management (GKM).

Group key management represents a significant overhead and a potential system bottleneck when the number of group members becomes large or group membership changes frequently. Therefore, it is necessary to have some scalable approaches for group key management, such as those proposed in the Internet Engineering Task

Force (IETF) Group Key Management Architecture [7], [16], and [43].

Since the last decade, group communication has been closely related to multicast communication because multicasting allows a sender to send a single packet, which is automatically forwarded by routers to recipients. This minimizes the number of copies of the original packet that are transmitted across the network. However, with the development of wireless technology (e.g. ad hoc wireless network), current group communication research is not limited to multicast. The Group Security group (GSEC) of the Internet Research Task Force (IRTF), which is formerly called Secure Multicast Group (SMuG), focuses their research on the issues of [33]

“large and small groups that operate on networks, which may be running a bulk distribution method such as broadcast, multicast or anycast.”

Hence, one new requirement for GKM is to be independent of the underlying network. Some GKM proposals which make use of multicasting to distribute group key may not be applicable to this new wireless ad hoc networking environment.

It is a challenge to accomplish secure group collaborations under highly dynamic circumstances such as ad hoc wireless networks. This is mainly due to the special characteristics of ad hoc networks and group communication.

An ad hoc network is a collection of peer-to-peer mobile nodes without any pre-established infrastructure. All participating nodes in ad hoc networks are able to cooperate with each other to transfer a data packet to the destination. They are self-configurable and self-recoverable. However, ad hoc networks are rather vulnerable due to the anonymity and mobility of the nodes, and the use of radio transmission. Moreover, nodes in ad hoc networks generally have a wide range of capabilities in

processing power, storage, and energy usage, etc. Thus, the requirements for an efficient, scalable and flexible GKM in ad hoc networks are more rigorous than in traditional wired networks.

Aside from group key management, authentication is another crucial issue for secure group communication. Usually a secret and non-compromised group key implicitly provides message authentication, which confirms that a message was sent by one of the group members but does not indicate the identity of the member who had sent the message. Source authentication, which confirms the source of the message, sometimes is more desirable.

Data integrity is also required in group communication. However, it will not be discussed in detail because traditional security countermeasures used in two-party communications are still applied here. This thesis mainly focuses on the two basic security issues of group communication—confidentiality and authentication.

1.1 Motivations of the Research

The emergence of mobile ad hoc networks creates opportunities for new forms of collaboration involving interaction between people who are co-located and organized in an unforeseen way [45]. The driving motivation for this research is the need to find an efficient and practical secure group communication solution, which not only supports the rigorous requirements brought up by wireless ad hoc networking paradigm but is also flexible enough to adapt to other group communication environments.

1.2 Contributions of the Thesis

In contrast to conventional secure group communication solutions, Secure Code Group Communication Mechanism (SCGCM) solves the problem from an original point of view. It is based on the concept of where during group collaboration, each party is able to make its own decision so that the relationship between group members is very loose. The changes of group membership have little impact on group communication thereby SCGCM supporting the requirements of dynamic group coalitions in ad hoc networks. In addition, the mechanism is very flexible and scalable to accommodate parties with a wide range of capabilities. SCGCM is able to provide secure group communication without being delayed by any slow members.

SCGCM applies the orthogonal property of codes to group communication area where the length of the message transmitted in this mechanism is independent of the number of members, thus it solves the crux of applying fully distributed mechanism to group communication area. Moreover, SCGCM provides source authentication for group communication in that each member can be identified by a specific secure code. The distinctive features of SCGCM make it a general solution for secure group communication.

The contributions of this work are not limited to ad hoc networking environment. Its contributions cover five research areas of GSEC [33]: Decentralized Group Key Management, Group Key and Membership Management (all group members have different capabilities), Multiple Senders, Non-multicast security, and Security Technologies for closed and open groups.

1.3 Thesis Outline

The layout of this thesis is as follows: Chapter 2 introduces the security issues of group communication and the unique requirements of secure group communication under wireless ad hoc environment. As well, some related research literatures are reviewed. In Chapter 3, the novel Secure Code Group Communication Mechanism (SCGCM) and its two extension algorithms are presented and compared with other solutions. In addition, the characteristics of SCGCM algorithms are discussed. Subsequently, Chapter 4 discusses several applications of SCGCM algorithms. The thesis concludes with a brief summarization and discussion of future works in Chapter 5.

Chapter 2

Group Communication and Wireless Ad Hoc Technologies

Although there have been numerous research efforts on secure group communication during the last decade, few publications exist on secure group communication in ad hoc networks. In this chapter, the issues and requirements of secure group communication in ad hoc networks are studied. In addition, results of related research efforts are reviewed so as to investigate their possible application to secure group communication in ad hoc networks.

Section 2.1 describes some group communication scenarios and their respective security requirements. Then in Section 2.2, wireless ad hoc technologies are introduced followed by a discussion of the security issues of group communication under ad hoc network environment. Section 2.3 presents a survey of the research in the area of ad hoc networks security. Finally, in Section 2.4, some mainstream work of group key management is surveyed.

2.1 Group Communication Scenarios

According to different communication schemes, applications based on group communication can be categorized as one-to-many communication and many-to-many communication.

2.1.1 One-to-many communication

In one-to-many communication, there is one initiator or sender. Only this sender is concerned with traffic security, while others are just passive recipients. Consequently, the sender itself implements the group management scheme among the group. Some examples of one-to-many communication are pay-per-view, news subscription, stock quotation, etc.

Different one-to-many communication may have different security requirements. For instance, in Pay-Per-View, there is a broadcaster and a set of registered members. The broadcaster delivers different programs to different privileged sets of members. The security concern is to prevent un-privileged subscribers from viewing the content of the transmitted programs. Hence, *confidentiality* is the most important issue here. However, for stock quotation, which is also publicly available, *source authentication* may be more important than the confidentiality of data.

2.1.2 Many-to-many communication

In many-to-many communication, each group member is able to send and receive messages. Examples of many-to-many communication are audio/video conference, Virtual Private Networks, distributed simulation, Multi-user Virtual Worlds, etc.

The major security issues, which are applicable to each member of the group, are data confidentiality and source authentication.

2.2 Secure Group Communication in Ad Hoc Networks

The need for secure group communication in ad hoc networks is particularly apparent [41] since ad hoc networking refers to on-the-fly groupings of mobile hosts with no prior association. In order to study the security requirements of secure group communication in ad hoc networks, it is necessary to understand the meaning of ad hoc network, its characteristics and security issues which must be addressed. Therefore, in Section 2.2.1, ad hoc networks are briefly reviewed. The security issues and requirements of group communication in ad hoc networks are subsequently discussed in Section 2.2.2.

2.2.1 Ad Hoc Networks

With recent technological advances in laptop computers and wireless devices, such as wireless modems and wireless LANs, wireless communication between mobile users is becoming more popular than ever before. A new paradigm of wireless communication—ad hoc networking has begun to emerge. Figure 2.1 illustrates an example of an ad hoc network. It is comprised of a collection of peer-to-peer mobile nodes with no fixed infrastructure. All participating nodes in ad hoc networks have routing capabilities which permit them to route other nodes' data packets to the destination.

Figure 2.1: An Example of an Ad Hoc Network

Ad hoc networks are self-configurable and self-recoverable thus the establishment and maintenance of ad hoc networks itself represent a form of group collaborations. For example, under disaster recovery situation where infrastructure is unavailable, some nodes collaborate with each other to form an ad hoc network. Another example is a group of individuals at a conference who share files through a temporary established ad hoc network.

Characteristics of Ad hoc networks

In comparison to traditional cellular infrastructure-based wireless systems, ad hoc networks have the following characteristics:

- (1) Rapid establishment

Since ad hoc networks do not rely on any pre-established infrastructure, they can be established rapidly and hence are invaluable in disaster recovery situations or places with non-existing or damaged communication infrastructure. The fast and casual network connectivity also makes it well suited to some circumstances, such as conferences, where the participants can form a temporary network without engaging the services of any pre-existing networks.

(2) Less administrative cost

Ad hoc networks are self-configurable and are able to maintain network connections and routing information without the need for explicit route setup by a system administrator. As a result, the administrative cost is generally less than wired networks.

(3) Easy to fail

There are two implications of the failure of ad hoc networks. On one hand, high mobility, limited resources and battery consumption of mobile nodes make ad hoc networks sensitive to interruptions caused by changes in topology or malfunctioning nodes. Hence, there is a requirement that protocols, designed for ad hoc networks, be not only efficient and robust but also provide power saving.

On the other hand, the radio environment of ad hoc networks is especially vulnerable to interception and impersonation attacks. In order to counter these attacks, some sort of preventative security measures, such as authentication and encryption, are needed.

Ad Hoc Technologies

The main technologies for ad hoc networks are IEEE 802.11 and Bluetooth. Bruno et al. [9] introduce these two technologies and analyze their performance. The 802.11 standard, which is implemented in each node, consists of a single MAC layer and one of multiple physical layers: frequency hopping spread spectrum, direct sequence spread spectrum or infrared light. The MAC protocol of 802.11 is Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). Stations (mobile nodes) of 802.11 can operate in two configurations: independent configuration (ad hoc) and infrastructure configuration [14]. In the first configuration, nodes communicate directly with each other without the support of an infrastructure. Stations in this configuration are considered to be a Basic Service Set (BSS). Without the Extended Service Set (ESS), the stations operate in an Independent BSS (IBSS) (I.e. the ad hoc networks mode). In the second configuration, the stations communicate through an Access Point (AP), which serves the stations in a BSS and forms a bridge between wireless and wired LANs. 802.11 standard uses multiple APs to extend the coverage area to whatever the internals of a distribution system, for instance with bridged wired LANs.

Bluetooth systems operate in the unlicensed 2.4GHZ ISM (Industrial Scientific Medicial) band. All Bluetooth devices share 79 channels (1 Mhz per channel). Using frequency hopping sequences, Bluetooth allows multiple concurrent communications within radio range of each other.

According to the power used, there are three classes of Bluetooth devices. A class 3 device has 1 mW transmission power and a range of 0.1 ~ 10 meters. A class 2 device has 1 ~ 2.5 mW transmission power and a 10 meters range. A class 1 device has a transmission power up to 100mW and a range up to 100 meters [27].

The Bluetooth system provides a point-to-point connection, or a point-to-multipoint connection. In the point-to-multipoint connection, the channel is shared among several Bluetooth units. Two or more units sharing the same channel form a piconet where one unit acts as the master and the other unit(s) acts as slave(s). There can be up to seven active slaves in a piconet. All slaves have to communicate through the master. When two or more piconets are in the proximity and have overlapping coverage areas, a scatternet is formed. Slaves in one piconet can participate in another piconet as either a master or a slave through time division multiplexing. In other words, each piconet operates in its own frequency hopping channel while the devices in multiple piconets participate at the appropriate time via time division multiplexing.

2.2.2 Security Issues and Requirements of Group Communication in Ad Hoc Networks

Confidentiality, authentication, and data integrity are the essential services for secure group communication in wireless ad hoc networks. A brief discussion of these security services and the related issues is provided in this section.

(1) Confidentiality

Confidentiality ensures that the transmitted information is accessible only for reading by authorized parties [35]. It is the core problem of secure group communication. Confidentiality is usually provided through encryption, which requires the management of keys. In group communication, it is referred to as Group Key Management

(GKM) and it includes group key generation, distribution, periodical update, and rekeying when group membership changes.

RFC2627 [43] clearly explains the general requirement of group key management infrastructure:

“must support bandwidths ranging from kilobits/second to gigabits/second, handle a range of multicast group sizes, and be flexible enough for example to handle such communications environments as wireless and mobile technologies. In addition to these performance and communications requirements, the security requirements of different scenarios are also wide ranging. It is required that users can be added and removed securely and efficiently, both individually and in bulk. The system must be resistant to compromise, insofar as users who have been dropped should not be able to read any subsequent traffic, even if they share their secret information. The costs we seek to minimize are time required for setup, storage space for each end user, and total number of transmissions required for setup, rekey and maintenance. It is also envisioned that any proposed multicast security mechanisms will be implemented no lower than any layer with the characteristics of the network layer of the protocol stack. Bandwidth efficiency for any key management system must also be considered.”

Moreover, the features of an ideal solution to group security in wireless circumstances is pointed out in [41]:

- *Provable security—the ability to demonstrate formally that a given protocol is secure with respect to its goals.*

- *Fairness—ideally, all protocol participants must be treated equally; no one party should avail itself of greater security than any other party, i.e., security must be attained cooperatively.*
- *Efficiency—all protocol costs must be minimized*
- *Adaptability and Fault tolerance—protocols must be able to adapt to dynamic factors, e.g., mobility as well as failure or compromise of protocol participants.*
- *Generality—protocols must be applicable to any communication scenario without prescribing or strongly favoring a specific network topology or communication paradigm.*
- *Scalability—protocols must scale gracefully. Consequently, requirements such as tightly synchronized clocks, simultaneous broadcasts, many-to-many authentication, etc., need to be avoided or minimized.*

To summarize, the requirements of GKM in ad hoc networks can be studied from the *performance and communication requirements* and the *security requirements* aspects.

Performance and Communication Requirements

1. Efficiency

An efficient group key management is very important, especially in ad hoc networks. Group formation should be finished within a reasonable short period of time without being delayed by resource constrained members.

2. Scalability

In wireless ad hoc networks, generally the participating mobile nodes range from Laptop computers to Personal Digital Assistant (PDA) and 3G cell phones. These nodes have different properties of scalability, processing requirements, energy usage, and storage. Therefore, in addition to supporting the expansion of group size, GKM should also be able to accommodate members with a wide range of capabilities. Moreover, GKM should be independent of the underlying networks such that it can be applied across various communication networks.

3. Distributed group key management

It is imperative to have a distributed GKM in ad hoc networks. As we have discussed previously, ad hoc networks are dynamic and are prone to failure; it is hard to have a fixed (and trusted) central key controller. Another disadvantage of having a central controller is that it can become a single point of failure/attack. On the other hand, it is desirable for group participants to be treated equally during secure group communication over ad hoc networks [41].

4. Support for highly dynamic environment

The frequency with which members leave and join ad hoc networks requires that GKM supports a highly dynamic environment. This is a formidable challenge for group key management in ad hoc networks.

Security requirements

1. Key Authentication

According to [25], there are two levels of key authentication—*implicit key authentication* and *explicit key authentication*. *Implicit key authentication* means that a party is assured that no other party aside from a specifically identified second party (and possibly additional identified trusted parties) may gain access to a particular secret key, while *explicit key authentication* is obtained when both *implicit key authentication* and *key confirmation* hold. Here, *key confirmation* assures a given party that a second (possibly unidentified) party actually has possession of a particular secret key.

It is necessary to have key authentication, at least implicit key authentication, for group key generation or distribution. Recall that ad hoc networks are vulnerable to impersonation and inception attacks, therefore key authentication is even more critical. In the case where a group key is generated through collaborations of group members, it is imperative to authenticate the parties before group key negotiation. Similarly, for group key distribution protocols, only authenticated group members should have access to the group key.

In order to have key authentication, direct or indirect *a priori knowledge* (eg. through Trusted Third Party(TTP), or a trusted key controller) between group members is necessary.

2. Perfect Forward Secrecy and Thwarting Known-Key Attacks

Given that ad hoc networks can easily be compromised, reducing the impacts of it happening to systems is very important. *Perfect Forward Secrecy (PFS)* and thwarting *Known-key attack* are two important concepts

used for analyzing key establishment protocols, and therefore should also be considered in GKM. Before proceeding with their definitions, an introduction of the *cryptoperiod* of a key is warranted. “*Cryptoperiod*” refers to the time period over which a key is valid for use by legitimate parties [25]. The use of *cryptoperiod* serves many purposes: to limit the information (related to a specific key) available for cryptanalysis; to limit exposure in case of a compromised key; to limit the use of a particular technology to its estimated effective lifetime; and to limit the time available for computationally intensive cryptanalytic attacks (in applications where long-term key protection is not required) [25].

Hence, a key with short *cryptoperiod* will reduce both the risk and impact of the key being compromised. It is important to have a data transmission key with short *cryptoperiod* in group communication.

The definition of *Perfect forward secrecy* is that compromise of long-term keys does not compromise past session keys [25]. Here, the long-term keys are the keys used to store or to transport other short-term keys. The definition of *known-key attack* is that compromise of past session keys allows either a passive adversary to compromise future session keys, or impersonation by an active adversary in the future [25].

In order to prevent known-key attack, two types of keys can be employed—one is *Traffic-encrypting key* (TEK) used to encrypt the data transmitted over the network and another is *Key-encrypting key* (KEK) used to encrypt TEK or other KEKs [7]. A KEK generally has a longer cryptoperiod than a TEK and it is less likely to be compromised than a TEK because it

is used much less frequently than a TEK. This way, even if past TEKs are compromised, the attackers still cannot compromise future TEKs (i.e. resistant to known-key attack), provided that the KEK is secure and TEKs are random and independent. Moreover, reducing the *cryptoperiod* of a KEK will decrease both the possibilities of the KEK being compromised and the time that the KEK is used for encrypting TEKs. Therefore, it is beneficial to limit the impact of a compromised KEK by using a short period. In other words, this will improve forward secrecy.

(2) Authentication

Besides confidentiality, authentication is another essential security service. It ensures that the originator of a message is correctly identified and that the identity is authentic. Authentication is especially important in wireless ad hoc networks, which is more vulnerable to impersonation attacks than wired networks. There are two kinds of authentication for group communication: message authentication and source authentication. Message authentication only ensures that a message was sent by a group member but does not indicate the actual source of the message. On the other hand, source authentication specifically identifies the originator of a message.

(3) Data integrity service

Data integrity service provides a mechanism for detecting modifications or reordering of messages. Possible modification includes writing, changing, changing status,

deleting, creating, and delaying or replaying of transmitted message [35]. Accordingly, the countermeasures used in two-party communication are still applicable here. For example, while Message Authentication Code (MAC) or one-way hash function can be used to prevent a group message from being changed, a ‘Nonce’ (e.g. ‘random number’, ‘Sequence number’ or ‘Timestamp’) is used to prohibit replay attack. Since the service can be provided merely by applying existing security mechanism, it will not be discussed specifically in this thesis.

2.3 Overview of Research on Ad Hoc Networks Security

In this section, key research initiatives for ad hoc networks security are reviewed. Although they are not necessarily related to group communication, they will increase our understanding of ad hoc network security issues and will provide some security mechanisms to assist group communication in ad hoc networks.

Publications [6], [34], [3], and [46] study the security issues and possible solutions in ad hoc networks. Zhou et al. [46] discuss security issues including secure routing problems in ad hoc networks. They also present a solution which provides an on-line key management service based on threshold cryptography and proactive security. The $(n, t+1)$ key management service of [46] comprises n servers and requires that at least $t+1$ servers be honest to partially sign a certificate. Although it may be a good solution in a moderately dynamic environment, it is restricted by the requirement that a minimal number of honest servers must exist and cooperate in the networks. Providing a Certification Authority (CA) service in fully mobile networks, such as ad hoc networks, is still a challenge today.

Stajano et al. [34] present an interesting resurrecting duckling policy model where a device is imprinted with a secret shared with its mother before it participates in ad hoc networking. Henceforth, the device can authenticate its mother using the shared secret. At the same time, a device can provide services for others according to the access list established by its mother. After the old secret is destroyed, it is possible for the device to be owned by another mother. [34] exploits the policy model to solve the authorization problems in ad hoc networks. Its main focus is on personal area networks.

Balfanz et al. [6] introduce a pre-authentication technique, which is similar to [34]’s imprint technology. They employ a location-limited channel to establish authenticity (not necessary to be secret) among strangers in ad hoc networks. It is assumed that it is not possible to have an active attack without being noticed in a “location-limited channel”. Possible “location-limited channels” are contact, infrared, near-field signalling across the body, and sound. This technique is useful when authentication is required but there is no *a priori* knowledge between group members. Balfanz et al. [6] further demonstrate the application of the pre-authentication technique to secure group communication in ad hoc networks. At first, a member goes to the “location-limited channel” and broadcasts the digest of its Diffie-Hellman public key to other group members. Later, when forming a group key, the digest of DH public key is used as an authenticator of the corresponding member. One benefit of “location-limited channel” is that it renders the Public Key Infrastructure unnecessary.

2.4 Overview of Research on Group Key Management

The previous section reviewed the studies focusing on key management and authentication in ad hoc networks. However, for secure group communication in ad hoc networks, there is still no known satisfied solution. In this section, some mainstream group communication solutions are surveyed. Moreover, these works will be compared with SCGCM, the new security mechanism that we designed, later in this thesis.

The general idea of current solutions for secure group communication is that a group establishes an initial group communication key and then uses the group key to carry out secure group communication. The core problem is Group Key Management (GKM), i.e. how to generate, distribute and re-key a group key. According to the mechanism used to manage a group key, there are non-collaborative key distribution/management techniques ([43], [5], [44], and [13]), which rely on a trusted central key controller to generate and distribute a group key; and collaborative key agreement which requires member collaboration to generate and distribute a group key [10], [4], [37], [36], [21].

Non-collaborative key distribution protocols generally work well in large and relatively static multicast groups, but they may not be very appropriate for wireless ad hoc networks. This is due to the difficulty of maintaining a trusted central controller in ad hoc networks. The center controller, which is responsible for group key generation and distribution, can be both a single point of failure and single point of attack.

In contrast, collaborative key agreement protocols are aimed at relatively small

groups. There are many research efforts (e.g. [10], [4], [37], [36], [21] and [3]) to extend two-party Diffie-Hellman key exchange to multi-party key exchange. It is worth mentioning that [4] and [37] study the Dynamic Peer Groups (DPG) problem, which is very much similar to group communication in ad hoc networks. Collaborative key agreement protocols have some noticeable features, such as non-central controller, distributed key management, etc. However, since most collaborative solutions are based on Diffie-Hellman key exchange, many exponential computations are required to generate or to distribute a group key. Thus, efficiency is a big issue. Another challenge in applying them to ad hoc networks is the need to accommodate a group of participants with a wide range of capabilities.

In the following sections, some representative key distribution protocols and the key agreement protocol for Dynamic Peer Group are reviewed. Please note that most of the non-collaborative key distribution solutions [43], [44], and [13] are tree-based solutions. Although these solutions rely on a trusted central controller for group key management, they are the only solutions that are used in practice. Hence, they will serve as a benchmark against which the performance of the new mechanism will be compared.

2.4.1 Tree Key Graph

Wong et al. [44] have introduced a solution that uses a key graph to manage the group key. The communication and computation cost of a tree key graph in updating a group key is logarithmic to the size of the group. This approach is also recommended in RFC2627 [43].

In [44], a triple (U, K, R) is used to represent a secure group, where U is a set of

users, K is a set of keys and R is a binary relation set between U and K . A directed acyclic key graph, comprised of u -nodes (users) and k -nodes (keys), is established according to the relation defined in R .

Let u denote an arbitrary user, k an arbitrary key, then the key set of a user u is $\{k | (u, k) \in R\}$ and the user set of a key k is $\{u | (u, k) \in R\}$. According to the relation, when a user u leaves a group, the associated keys should be changed also. The central key controller securely distributes the new keys using keys that are unknown to u .

Wong et al. [44] consider the re-keying problem in star, tree, and complete key graphs. In addition, they present three rekeying strategies, *user-oriented*, *key-oriented* and *group-oriented*. Here, only the protocols based on *group-oriented* tree key graph are discussed since the star key graph is just a simplified version of tree graph. The use of the tree graph will make it easier to understand the process of the protocols; and furthermore, a group-oriented strategy will reduce the key controller's per rekey message overheads. As we are concerned with the possible application of TKG to peer-to-peer environment, it will be beneficial to equalize the overhead of all group participants.

In the tree key graph, each leaf node corresponds to a secret key of a user (member) and the internal nodes represent auxiliary subgroup keys. In the following, two group-oriented rekeying protocols—a join protocol and a leave protocol are presented.

TKG—Join protocol

Figure 2.2 shows an example of the tree based key management. When a new member u_9 joins the group, the key controller broadcasts a concatenated message $\{k'_{Grp}\}_{k_{Grp}}, \{k_{789}\}_{k_{78}}$ to all old members and $\{u_1, u_2, \dots, u_8\}$ and one unicast message

k_{Grp} : old group key

k'_{Grp} : new group key

User i , denoted as u_i , stores all the keys on the path from the leaf to the root

The shadowed key nodes are updated, and key server broadcasts the new group keys k'_{Grp} , k_{789} to old users and unicast k'_{Grp} , k_{789} to the new user u_9

Figure 2.2: Join Protocol for a Tree Key Graph

Tree Key Graph—Join Protocol

Suppose a user u wants to join the group. The central key controller s decides a joining point for it, which is x_j . Let x_0 denote the root of the tree, x_{i-1} the parent of x_i for $i = 1, \dots, j$; K_{j+1} denote the secret key of u (i.e. k_u), while $K_0, \dots, K_j$ represent the old keys of $x_0, \dots, x_j$, and $K'_0, \dots, K'_j$ represent the new randomly generated keys. Here K_0 is the old group key and K'_0 is the new group key.

1. $u \rightarrow s$: a new user u sends a join request to the central key controller s .
2. s authenticates u and distributes a secret key k_u to u
3. $s \rightarrow$ all old members: $\{K'_0\}_{K_0}, \dots, \{K'_j\}_{K_j}$ The key controller s uses the old group key and subgroup keys to encrypt the corresponding new keys and broadcasts the concatenated message.
4. $s \rightarrow u$: $\{K'_0, \dots, K'_j\}_{k_u}$ The key controller unicasts all the keys including the group key and subgroup keys on the path (from the joining point to the root) to the new user.

$\{k'_{Grp}, k_{789}\}_{k_9}$ to the new user u_9 .

TKG—Leave protocol

When a user u leaves, all keys along the path from the leaving point to the root node need to be changed correspondingly.

Figure 2.3 shows that when user u_6 is granted to leave, the central key server broadcasts one concatenated message $\{k'_{Grp}\}_{k_{123}}, \{k'_{Grp}\}_{k_{45}}, \{k'_{Grp}\}_{k_{789}}, \{k_{45}\}_{k_4}, \{k_{45}\}_{k_5}$ to all remaining group members.

Tree Key Graph—Leave Protocol

Let x_{j+1} denote the deleted k-node for k_u , x_j the leaving point, x_0 the root, x_{i-1} the parent of x_i for $i = 1, \dots, j$, $K'_0, \dots, K'_j$ the new keys of $x_0, \dots, x_j$, $J_1, \dots, J_r$ the keys at the children of x_i in the new key tree, and L_i represent $\{K'_i\}_{J_1}, \dots, \{K'_i\}_{J_r}$.

1. $u \rightarrow s : \{\text{leave request}\}_{k_u}$
 u sends leave request to key controller s and the key controller grants the leave request.
2. Central key controller s finds the leaving point, i.e the parent of k_u , and then removes k_u from the tree.
3. $s \rightarrow$ all remained members: $L_0, \dots, L_j$
 The keys on the path from u to the root are updated. The key controller s creates the rekey message by encrypting the parent's new (sub)group key individually using each of the children's (new) subgroup keys.

TKG—Discussion

TKG achieves *backward access secrecy* by updating the group key and the related subgroup keys in the *join protocol*. Since the new group key and subgroup keys are irrelevant to the old ones, the new joining member is not able to deduce the old keys from its new keys and access pervious secure group communication. Similarly, the *leave protocol* ensures *forward access secrecy* by updating the old group and subgroup keys with the corresponding subgroup keys that the departing member does not know.

The storage cost and communication complexity of the Tree Key Graph are analyzed as follows.

Suppose the tree key graph is a k -nary balanced tree and there are N group members. While the number of keys that each group member stores is equal to the

k_{Grp} : old group key

k'_{Grp} : new group key

u_i denotes user i , it stores all the keys on the path from the leaf to the root

u_6 leaves so that all the keys on the path from k_6 to k_{Grp} (shadowed nodes) are updated.

Key controller broadcasts new group key and related updated subgroup key to remained users

Figure 2.3: Leave Protocol for a Tree Key Graph

height of the tree, the number of keys stored by the central key controller is equal to the number of nodes (including k -nodes and internal nodes) in the tree. First of all, we need to compute the number of nodes in the tree. Let h denote the height of the tree. The number of nodes in the tree is:

$$1 + k + k^2 + \dots + k^h = \frac{k^{h+1} - 1}{k - 1} \quad (2.4.1)$$

Also, for k -nary balanced tree, the height of the tree is $\log_k N$. Therefore, the number of keys stored by the server is $\frac{kN-1}{k-1}$ whereas each user stores $\log_k N$ keys.

When a new member joins, the key controller authenticates and distributes a secret key to the new member first, which is called Key Encryption Key(KEK) in [43] (or Key-encrypting key in [7]), and then updates the related group keys. Let l denote the length of a key. Therefore, there is a rekey broadcast message of $(\log_k N \times l)$ bits, and a unicast to the new member of $(\log_k N \times l)$ bits.

Remark 2.4.1. Since we are examining a Tree Key Graph that takes group oriented strategy into account, we analyze it by considering the number of broadcast (instead of multicast) and unicast messages which are transmitted and the number of bits per broadcast and per unicast message(s). Except when explicitly stated, these performance parameters are used throughout the thesis.

The number of encryptions performed by the key controller when a new member joins the group is $2(h - 1)$, including $(h - 1)$ group and subgroup key upgrade encryptions and $(h - 1)$ encryptions of the keys on the path from the new member to the root. The number of decryptions done by the new member is $\log_k N - 1$ and the average number of decryption by other members is given to be $\frac{k}{k-1}$ in [44].

When a member leaves, a broadcast message is sent $(k \log_k N \times l)$ bits to update subgroup keys and group keys while the number of encryptions performed by the center is $k(\log_k N - 1)$, the average number of encryption executed by the members is given as $\frac{k}{k-1}$ in [44].

2.4.2 One-way Function Tree

Balenson et al. [5] introduce a binary One-way Function Tree (OFT) solution. In contrast to the tree-based key management solution proposed in [44], each user of OFT owns two keys: one is a normal secret key k_x , called unblinded node key; and

Figure 2.4: An Example of a One-Way Function Key Tree

the other is a blinded node key k'_x generated from the unblinded node key using a public one-way function $g(.)$, i.e. $k'_x = g(k_x)$.

Figure 2.4 shows an example of One-way Function Tree. Each leaf node stores not only its unblinded node key but also the blinded node keys of sibling nodes on the path from its leaf to the root. The central server computes the unblinded node key k_x for an internal node x from the blinded node keys of its two children, i.e. $k_x = f(k'_{left(x)}, k'_{right(x)})$, where f is a mix function such as *XOR*.

The join protocol and leave protocol are introduced in the following subsections.

OFT—Join Protocol

When a new member joins the group, a leaf node x , which is closest to the root, is split into two nodes, denoted as $left(x)$ and $right(x)$. The old member becomes $left(x)$, and the new member becomes $right(x)$.

OFT—Join Protocol

1. $s \rightarrow left(x), right(x) : k_{left(x)}, k_{right(x)}$.

Central key controller s assigns a new key to $left(x)$ and $right(x)$ respectively. The unblinded node key of the old member is changed in order to prevent the former sibling from colluding with another group member to find an unblinded key that is not on its path to the root.

2. $s \rightarrow$ the appropriate subgroups (subtrees) with the siblings of $right(x)$ as the roots : the central key controller s computes new values of the blinded node keys for the nodes on the path from the leaf to the root and encrypts them using the appropriate sibling keys. The encrypted blinded node keys are then broadcast to the corresponding subgroups.
3. $s \rightarrow right(x)$: The new member $right(x)$ is given a set of blinded node keys in a unicast transmission.

OFT—Leave Protocol

When a member y leaves the group, its sibling x combines with its parent p . If the sibling of y is the root of a subtree, then the subtree is moved one level closer to the root.

OFT—Leave Protocol

1. $s \rightarrow x : k_x$; s assigns a new unblinded leaf key to x .
2. $s \rightarrow$ the appropriate subgroups (subtrees) that siblings of x are roots :
 $\{\text{blinded node key}\}_{k_{\text{sibling}}}$
 Key controller s broadcasts new blinded node keys to the appropriate subgroups that siblings of x are (sub)roots of the corresponding subtrees.

OFT—Discussion

The security of OFT systems depends on an invariant called OFT Security Invariant—each member knows the unblinded node keys on the path from its node to the root, and the blinded node keys that are siblings to its path to the root, and no other blinded or unblinded keys [5].

The combination of the above security invariant, the strength of the one-way function and the random selection of leaf keys renders OFT perfect *backward access secrecy* and perfect *forward access secrecy*. After a member leaves, the keys along the path from the member's node to the root change accordingly. Thus, the departing member knows only the blinded keys of the siblings of the nodes along the path from it to the root, which are insufficient to compute directly any unblinded key. Moreover, arbitrary coalitions of the departing members cannot directly compute the group key nor any unblinded node key. Similarly, when a new member joins the group, the new member is not able to compute directly any unblinded keys that are not on the path from it to the root.

Suppose the OFT is a balanced tree and there are N group members. Each member stores $\log_2 N + 1$ keys while the center stores $2N$ keys. When a new member

joins the group, the center carries out a blinded key broadcast with $O(\log_2 N)$ bits and an unicast with $O(\log_2 N)$ bits. On the other hand, when a member leaves, it performs only a blinded key broadcast with $O(\log_2 N)$ bits.

As the auxiliary subgroup keys used in tree-based algorithms such as [44] and [5] increases the storage requirement of each member to a logarithmic value. Thus, the communication overheads required for key update are reduced to a value which is equally logarithmic. Moreover, these subgroup keys can also be used to transmit subgroup messages. Hence, these algorithms are useful for some hierarchical group collaborative applications in which cooperations are not only carried out among all group members but can also be restricted to a small subgroup only.

2.4.3 Authenticated Group Diffie-Hellman (A-GDH.2)

Ateniese et al. [4] present an Authenticated Group Diffie-Hellman (A-GDH.2) protocol for Dynamic Peer Groups (DPG) contexts. Examples of DPG's include replicated servers (e.g database, web, time), audio and video conference, etc. [4].

Suppose there are N group members arranged in a (logical) ring and that the last group member M_N is trusted and has distinct secrets shared with each of the other members. These secrets can be established using Diffie-Hellman public key scheme. In particular, it is assumed that either M_N stores the M_i 's public DH key or is able to access to it securely. Likewise, M_i can either store the M_N 's public DH key or is able to access to it in a secure manner. Therefore, each member is able to compute the secret key shared between M_N and itself. To generate a group key, each party contributes a random number and finally the last member uses the shared key to encrypt the group key.

This protocol achieves implicit key authentication because only the group member with the corresponding private DH key is able to decrypt the broadcast message sent by M_N . The key agreement protocol is explained below.

Authenticated Group DH – Key Agreement

Let p be a prime, q a prime divisor of $p - 1$, i.e. $q|(p - 1)$, G the unique cyclic subgroup of Z_p^* of order q , and α a generator of G .

1. M_i selects a number $r_i \in_R Z_q^*$

Each member selects a random number.

2. $M_i \rightarrow M_{i+1}$, where $0 < i < N$: $\{\alpha^{\frac{r_1 \cdots r_i}{r_j}} | j \in [1, i]\}, \alpha^{r_1 \cdots r_i}$

The first member M_1 sends α^{r_1} to M_2 ; Each of the other members (M_i) computes the key materials received from the previous member with its contribution r_i , and then sends them to the next member.

3. $M_N \rightarrow \text{All } M_i$: $\{\alpha^{\frac{r_1 \cdots r_N}{r_i} \cdot K_{iN}} | i \in [1, N]\}$.

When last member receives the key materials, it computes them with its own contribution, and then encrypts them with the corresponding shared secrets. Finally, it broadcasts the concatenated and encrypted partial group key message so that only valid group members can recover the group key.

4. When a member M_i receives the above message, it retrieves the group key by computing $\alpha^{(\frac{r_1 \cdots r_N}{r_i} \cdot K_{iN}) \cdot K_{iN}^{-1} \cdot r_i} = \alpha^{r_1 \cdots r_N}$.

When a new member joins, it will become the last member M_{N+1} . The former last member M_N selects a new random number r'_N , computes and sends the resulting message $\{\alpha^{\frac{r'_N r_1 \cdots r_N}{r_i} \cdot K_{iN}} | i \in [1, N]\}, \alpha^{r'_N r_1 \cdots r_N}$ to the new member. After the new member adds its contribution, it constructs a new group key and distributes it using a broadcast message $\{\alpha^{\frac{r'_N r_1 \cdots r_{N+1}}{r_i} \cdot K_{iN} \cdot K_{iN+1}} | i \in [1, N + 1]\}$ to all other members.

Ateniese et al. [4] do not give the Leave Protocol. However, another paper [37] does define the leave protocol of GDH, called Group Key Agreement Member Exclusion. Since GDH is the basis of A-GDH.2, we can use it for reference purpose. The general idea is that when a member leaves the group, the last member M_N selects a new random number r'_N and takes $r_N \cdot r'_N$ as its new contribution to recompute a new group key.

A-GDH.2 provides implicit key authentication. Each member is assured that non-members will not have access to the group key as long as it is protected by all group members. Ateniese et al. [4] prove it using the following example.

Suppose an active attacker S wants to share a key with other group members. S successfully impersonates a member M_i such that it substitutes M_i 's contribution r_i with s_i , i.e. $\{\alpha^{\frac{r_1 \cdots r_i}{r_j}} \mid j \in [1, i]\}$, $\alpha^{r_1 \cdots r_i}$ is replaced with $\{\alpha^{\frac{r_1 \cdots s_i}{r_j}} \mid j \in [1, i]\}$, $\alpha^{r_1 \cdots s_i}$. Consequently, the final group key becomes $\alpha^{r_1, \dots, s_i, \dots, r_N}$. However, attacker S still cannot retrieve the key because it does not know K_{iN} .

Ateniese et al. further discuss a possible attack to A-GDH.2 that enables the attacker to share a key with the last member. However, the authors point out that this kind of attack can be prevented by *Key Confirmation*.

A-GDH.2—Discussion

The join protocol of A-GDH.2 achieves *backward access secrecy* due to the difficulty of the Diffie-Hellman problem, i.e. it is computationally impossible for the new joining member recovering the previous group key(s) from the new group key. Likewise, the leave protocol accomplishes *forward access secrecy* because of the difficulty of the Diffie-Hellman problem.

A-GDH.2 has some desirable features such as key authentication, contributory key agreement and accomplishing secure group communication under unsecured networks. Moreover, the key controller of A-GDH.2 is not fixed; it can be dynamically changed/selected.

However, the computation cost for initial key establishment and re-key is very expensive. Using A-GDH.2 to establish a group key results in N rounds of exponential computations where the number of exponential computations in each round depends on the position of the member in the ring. Hence, the computation costs of different members are not balanced. Another disadvantage of A-GDH.2 is that since group members are arranged in a ring and group key establishment information are exchanged in sequence, group key formation may be delayed by a member with inferior processing capabilities.

Moreover, the leave protocol works well if the departing member is not the last member. However, if it is the last member, then a new group controller has to be selected. Thus, any member capable of becoming a new controller has to store the last broadcast message, i.e. $O(N)$ storage. In addition, each of the other members should be able to establish a secret key with the new controller. This is also true for the join protocol—each of the other members should (be able to) have a DH key shared with the new member (controller). Because it is not easy to predict the new group controller, especially when the former controller leaves, the underlying assumption of A-GDH.2 is that each pair of group members should (or be able to) have a shared DH key. The shared secret can be obtained by each member either by storing the new controller's DH public key or by accessing a trusted public directory. If a storage method is used, then there is $O(N)$ extra cost. However, if the trusted public directory

is used, then we come back to the old problem—maintaining a trusted public directory in highly dynamic environment, such as ad hoc networks. In conclusion, there remains many challenges in applying A-GDH.2 to ad hoc networks.

Chapter 3

Secure Code Group Communication Mechanism (SCGCM)

In the previous chapter, some mainstream group key establishment and management protocols are reviewed. All of them require a group key be established among the group members in order to have secure group communication. When group membership changes, the group key has to be updated or reestablished accordingly. Hence, it may take quite a long time to establish a group key, or even worse it may never successfully establish a group key in extremely dynamic situations. In this chapter, a novel and fully distributed Secure Code Group Communication Mechanism (SCGCM) is presented. The design goal of SCGCM is to find a mechanism to achieve secure and efficient group communication, which not only is easy and flexible to deal with complex context of ad hoc networks, but also adapts well to general group collaboration environments.

Section 3.1 presents some notations and definitions that will be used in this chapter. After briefly introducing SCGCM in Section 3.2, we present the Basic Algorithm of SCGCM and prove the correctness of SCGCM encryption algorithm in Section

3.3. We then extend the Basic Algorithm of SCGCM to two extension algorithms—SCGCM(EX_I) and SCGCM(EX_II) which are presented in Section 3.4 and Section 3.5 respectively. SCGCM(EX_I) is designed for large groups with large amount of information exchanged, while SCGCM(EX_II) aims at improving the efficiency of data transmission using the block cipher technology. Moreover, Section 3.6 discusses the characteristics of SCGCM and compares it with other solutions previously reviewed.

3.1 Notations and Definitions

In this section, the notations that will be used in Chapter 3 and 4 are listed in Table 3.1. As well, some definitions are given.

Definition 3.1.1. The *Cryptoperiod* of a code (or key) is the time period over which it is valid for use by legitimate parties [25].

Definition 3.1.2. A code is in *bipolar form*, if the binary value of ‘1’ is represented as ‘+1’ and ‘0’ is represented as ‘-1’.

Definition 3.1.3. We say that a message msg is *scrambled* with a code C , if the original ‘1’ of msg is substituted by C , and ‘0’ of msg is substituted by $\overline{C}$.

Definition 3.1.4. The resulting message encrypted using the encryption protocol of SCGCM is called *amalgamated message*.

Definition 3.1.5. The codes used for secure group communication and/or code update in SCGCM are called *Secure Codes*. There are various codes such as Data Encryption Secure Code (DESC), Data Decryption Secure Code (DDSC), Code Exchange Encryption Secure Code (CEESC), and Code Exchange Decryption Secure

Table 3.1: Notations

N	the number of group members
M_i	a group member
$\cdot$	operator of <i>normalized inner product</i>
CE_{ij}	a bipolar formed CESC assigned by M_i to M_j .
L_i	the length of CE_{ij} .
C_{ij}	a bipolar formed Secure Code assigned by M_i to M_j .
C_i	M_i 's DESC set. $C_i = \{C_{ij} i, j \in [1, N], i \neq j\}$
D_i	M_i 's DDSC set. $D_i = \{C_{ji} i, j \in [1, N], i \neq j\}$. Note $C_{ij} \in C_i$, and $C_{ij} \in D_j$
l_i	the length of a DESC of M_i .
K_{ij}	a priori knowledge—the key used by M_i to encrypt a CESC to M_j in the initial Secure Code Exchange. It can be M_j 's public key or a symmetric key shared between M_i and M_j .
$E_{K_{ij}}(C_{ij})$	C_{ij} is encrypted with K_{ij} using public-key encryption or symmetric-key encryption depending on K_{ij} .
msg_{ij}	the message sent by M_i to M_j .
$S_{C_{ij}}(msg_{ij})$	msg_{ij} is scrambled with C_{ij} .
MSG_i	the amalgamated message sent from M_i to M_j . $MSG_i = \sum_{j=1, i \neq j}^N S_{C_{ij}}(msg_{ij})$
$D_{C_{ij}}(MSG_i)$	M_j decrypts MSG_i using C_{ij} .
ACK_i	the amalgamated secure code confirmation sent by M_i .
ack_{ij}	M_i 's secure code confirmation to M_j .
$ID C'_{ij}$	the subgroup ID and the subgroup DSC that M_i assigns to M_j .
IV_{ij}	the Initialization Vector used by member M_i to M_j

Code (CEDSC). The DESC and the DDSC are jointly referred to as *Data Secure Code (DSC)*. Similarly, the CEESC and the CEDSC are called *Code Exchange Secure Code (CESC)*. The main difference between CESC and DSC is that CESC has longer *cryptoperiod* than DSC.

Definition 3.1.6. We say that a set of codes is a set of *encryption secure codes*, if the codes are mutually orthogonal, pseudo random, and of the same length. If they are used to encrypt data, they are called *Data Encryption Secure Codes (DESC)*. If they are used to distribute and update DSC, they are called *Code Exchange Encryption Secure Codes (CEESC)*. DESC and CEESC should be totally independent.

Definition 3.1.7. We call a code assigned by member M_j to member M_i as a *Decryption Secure Code* for M_i in that M_i can decrypt an amalgamated message sent by M_j using that code. If the code is used to decrypt data, it is called *Data Decryption Secure Code (DDSC)*. If it is used to distribute and update Data Secure Code, it is called *Code Exchange Decryption Secure Code (CEDSC)*.

3.2 SCGCM Overview

The design of the Secure Code Group Communication Mechanism is based on the orthogonal property of codes. The basic idea is closely related to *Spread Spectrum* technology. Spread-spectrum systems were originally developed for military applications, to provide antijam and low probability of intercept by using a spreading code or pseudo-noise (PN) sequence to spread a signal over a large frequency band and transmitting it with a low power per unit bandwidth [38]. Based on spread spectrum technology, code division multiple access (CDMA) exploits the correlation of

spreading codes to accomplish efficient channel allocation for multi-users. However, its applications are confined to the physical layer.

We adopt a similar idea of spread spectrum technology and apply the orthogonal property of codes to secure group communication in the application layer and obtain an easy and practical mechanism for secure group communication.

In the following section, we will present the assumptions of SCGCM and outline the fundamental procedures of SCGCM—Data Secure Codes exchange and SCGCM group communication.

3.2.1 Assumptions

It is assumed that initial group membership is known and each group member has a secure code generator. In addition, it is assumed that a change in group membership can be detected either through group communication or the underlying communication networks. However, details of the detection are out of the scope of this thesis.

Another assumption is that each pair of group members have *priori* knowledge of each other before the start of group communication. In other words, each group member either has the public key or a shared secret key of the other members. A priori knowledge can also be pre-exchanged and valid (eg. within cryptoperiod) CESC. In this case, the initial secure codes exchange merely includes the exchange of DSCs. This rule is held in all SCGCM algorithms.

Remark 3.2.1. If CESC is the a priori knowledge, the communication and computation overheads of the initial exchange of secure codes will be much smaller than those with other kinds of a priori knowledge. However, in order to achieve the maximum and the

most casual group communication, we relax the restrictions to any types of *a priori* knowledge here.

Recall that in Chapter 2 (Section 2.2.2) we have discussed one of the security requirements of Group Key Management (GKM) is key authentication; *a priori* knowledge is the prerequisite for key authentication. Since most secure group communications are conducted between entities who know (or are able to know) each other before engaging in group communication, the assumption of *a priori* knowledge generally is met. In some applications, such as ad hoc group collaborations where *a priori* knowledge may not exist among group members, the *pre-authentication* techniques discussed in Section 2.3 can be used to fulfill the assumption.

3.2.2 Overview of Data Secure Codes Exchange

Before carrying out secure group communication, group members have to interchange their data secure codes.

Two types of secure codes are involved in SCGCM. Recall that in order to resist a *known-key attack*, it was recommended in Section 2.2.2 that a Traffic-encrypting key (such as DSCs) be updated or distributed using a specific Key-encrypting key (such as CESC). Consequently, each member need to generate a set of CESC that is independent of the set of DESCs.

Initially, each member has two sets of secure codes (CEESCs and DESCs) generated offline. Group members have to exchange their CEESCs first in case that the CESC are not pre-exchanged. Then, they exchange their DSCs using the corresponding CESC. As a result, each member has two extra sets of decryption secure codes (CEDSCs and DDSCs) consisting of the secure codes generated by each of the

other members.

3.2.3 Overview of SCGCM Secure Group Communication

After group members have exchanged their secure codes, they are ready for secure group communication. First of all, a sender scrambles a copy of the original message for each recipient using the corresponding DESC. Next, it amalgamates them as one message such that the sender can broadcast/multicast the resulting message to all recipients.

When a recipient receives the amalgamated message, it decrypts the message simply by correlating the amalgamated message with the Data Decryption Secure Code assigned by the sender. At the same time, the messages for other recipients are opaque to it.

3.3 SCGCM Basic Algorithm

The details of the Basic Algorithm of SCGCM are given in this section. Section 3.3.1 describes the procedure for Secure Code Exchange. Next, Section 3.3.2 presents the mechanism for accomplishing secure group communication using SCGCM encryption algorithm while Section 3.3.3 proves the correctness of the encryption algorithm of SCGCM. Finally, the management of Secure Codes which includes periodic secure code update, join protocol and leave protocol are presented in Section 3.3.4, 3.3.5, and 3.3.6 respectively.

3.3.1 Initial Secure Code Exchange

Initially, each member generates two sets of secure codes offline—one is the Code Exchange Encryption Secure Code (CEESC) set and the other is the Data Encryption Secure Code (DESC) set. All of these codes are stored in bipolar form.

Member List	CEESC List	CEDSC List	DESC List	DDSC List
M_1	CE_{1i}	CE_{1i}	C_{1i}	C_{1i}
...	...	...	...	...
M_{i-1}	CE_{i-1i}	CE_{i-1i}	C_{i-1i}	C_{i-1i}
M_{i+1}	CE_{i+1i}	CE_{i+1i}	C_{i+1i}	C_{i+1i}
...	...	...	...	...
M_N	CE_{Ni}	CE_{Ni}	C_{Ni}	C_{Ni}
Unassigned CEESC & DESC for new group members				

CEESC: Code Exchange Encryption Secure Code
 CEDSC: Code Exchange Decryption Secure Code
 DESC: Data Encryption Secure Code
 DDSC: Data Decryption Secure Code

Figure 3.1: An Example of M_i 's Secure Code Table

After the secure codes have been exchanged, every member constructs a secure code table. Figure 3.1 shows an example of a secure code table that is stored by a member M_i . The table consists of two parts—assigned secure codes and unassigned secure codes. Assigned secure codes consist of a Member List, a Code Exchange Encryption Secure Code (CEESC) List, a Code Exchange Decryption Secure Code

(CEDSC) List, a Data Encryption Secure Code (DESC) List and a Data Decryption Secure Code (DDSC) List, while unassigned secure codes consist of unassigned CEESCs and unassigned DESCs which can be used for new members.

The initial secure code exchange of the Basic Algorithm consists of two stages: (1) Code Exchange Secure Code (CESC) exchange and (2) Data Secure Code (DSC) exchange. There are two steps in each stage: the transmission and confirmation of secure codes.

Code Exchange Secure Code (CESC) exchange

Figure 3.2: Code Exchange Secure Code (CESC) exchange

Figure 3.2 illustrates the procedure for CESC exchange. As shown, each member

Figure 3.3: Code Exchange Secure Code (CESC) transmission

sends the CESC s vertically to all other group members and then confirms CESC exchange horizontally to all other group members after a pre-determined time.

(1) Code Exchange Secure Code (CESC) Transmission Each member broadcasts a concatenated message comprised of encrypted secure codes to all other group members.

$$M_i \rightarrow M_j : \{E_{K_{ij}}(C_{ij}) | i, j \in [1, N], i \neq j\}$$

Figure 3.3 shows that a member M_i broadcasts the concatenated and encrypted secure codes to all other members. M_i firstly looks up its credential database for possible encryption means and encryption key K_{ij} , and then encrypts C_{ij} with K_{ij} . After M_i finishes encrypting the secure codes for all other group members, it concatenates these encrypted secure codes together and broadcasts it.

Remark 3.3.1. Broadcast is used here to reduce the number of transmissions of a sender. However, if a member needs to distribute its secure codes to a small number of parties, then unicast is preferable. There is a tradeoff between the communication overhead of a sender and volume of traffic generated on the network. Moreover, if the network supports multicast transport, then multicast can be used instead of broadcast. In this chapter, *broadcast* can be replaced with *multicast* if possible. However,

broadcast is employed instead of unicast as previously noted.

Remark 3.3.2. Chapter 4 of [20] reviews and presents alternatives for broadcasting session keys. One or more may be employed here to reduce the size of a CESC transmission. Please refer to [20] for additional details.

(2) Code Exchange Secure Code (CESC) Confirmation is intended to reassure a group member that the other parties possess the CESC it has assigned and transmitted. Although this step is optional, it facilitates the management of secure codes.

After a member M_i sends out its CESC, it waits for a pre-determined time (TIMEOUT) to receive the CESC sent by other members. Then, it prepares acknowledgements to all other parties; each acknowledgement ack_{ij} is scrambled with the corresponding CE_{ij} . A possible acknowledgment¹ can be “Hi, I have received your CESC”, or “Sorry I have not received your CESC yet”. M_i amalgamates all acknowledgements as one message and broadcasts it.

At the end of the process (i.e after receiving the acknowledgement or experiencing a timeout), a member can determine precisely the members who have and have not correctly received its secure code. If necessary, it will retransmit the secure codes to those who have not yet received them.

Considering the result of secure code exchange between two arbitrary parties M_i and M_j , there are two exceptional cases.

¹It is also possible to use one-way hash of a secure code as a proof and acknowledgement for possession of the code.

- (1). M_i receives the code from M_j , but M_j does not receive the code sent by M_i .

In this case, ack_{ij} is “Hi, I have received your CESC”; while ack_{ji} is “Sorry I have not received your CESC yet”. Since M_i has CE_{ji} , it can decrypt ack_{ji} and learn that M_j has not received its code yet, therefore it will send it again and wait for the acknowledgement. On the other hand, since M_j does not have CE_{ij} , it will not understand ack_{ij} , therefore it has to wait until it either receives the CE_{ij} or a timeout occurs. If M_j successfully receives the retransmitted CE_{ij} , it should acknowledge it to M_i . If M_i does not receive the ack_{ji} within a certain time, it means the code exchange between M_i and M_j fails so that neither M_i nor M_j needs to maintain the codes and they should delete the codes from their tables.

Remark 3.3.3. A nonce should be used for each CESC exchange so that a member can distinguish between an obsoleted CESC and a fresh one.

- (2). Neither M_i nor M_j receives the other party's code.

Since M_i can not decrypt ack_{ji} and M_j can not understand ack_{ij} either, they will have to wait until a timeout occurs to re-exchange their codes.

Remark 3.3.4. The confirmation of CESC exchange uses the protocol of Secure Group Communication presented in Section 3.3.2.

Data Secure Code (DSC) exchange

After group members have exchanged their CESC, they start to exchange their Data Secure Codes. Both the transmission and confirmation of Data Secure Code

are accomplished using the protocol of the Secure Group Communication presented in Section 3.3.2.

Remark 3.3.5. Each member independently makes decision as to whether or not to exchange codes with some parties. This flexibility can be useful for some multi-party collaborations where there are multiple senders, but not all of the group members are senders.

3.3.2 SCGCM Secure Group Communication

A sender carries out secure group communication by scrambling a given message to a given recipient, amalgamating these scrambled messages into one message and broadcasting it to all members. Messages for different recipients can either be the same or different. It is also possible that some of the messages are the same and some of them are not. Therefore, this mechanism achieves flexible multiparty secure communication amongst any combination of group members.

The encryption and decryption protocols of SCGCM are presented in the following sections.

SCGCM Encryption

Suppose a member M_i wants to send a message to other group members. The messages for different recipients can be the same or different. The procedure for encrypting the messages is as follows:

- (1). Transforming messages

M_i transforms each distinct message into a binary (0,1) form. The same messages are only transformed once.

(2). Scrambling messages

M_i scrambles the resulting binary messages from the previous step with the corresponding Data Encryption Secure Codes (DESCs).

(3). Amalgamating messages

After M_i has scrambled the messages for all recipients, it adds them up bit by bit. The resulting amalgamated message is denoted as MSG_i .

Remark 3.3.6. Note that if not all of the scrambled messages are of the same length, the short messages can be padded with 0. The length of the resulting amalgamated message MSG_i is:

$$l_i \times \max\{\text{length}(msg_{ij}) | j \in [1, N], i \neq j\}$$

It is important to note that there is a spreading factor of SCGCM in encoding group communication. However, we can reduce it by assigning multiple (say x) DSCs to a member at a time. In other words, we can scramble x messages to a member at a time², since the length of the resulting amalgamated message is independent of the number of scrambled messages contained in it, thereby the total bandwidth for group communication being reduced.

² x messages can be x different messages or a message is divided into x small messages.

SCGCM Decryption

When a recipient M_j receives an amalgamated message MSG_i from M_i , it retrieves msg_{ij} by:

- (1). Computing the normalized inner product of MSG_i and the Data Decryption Secure Code (DDSC) C_{ij} .
- (2). Checking each bit of the result from the above step:
 - If a bit is '1', then the original bit in this position is '1'.
 - If a bit is '-1', then the original bit in this position is '0'.
 - Otherwise, if a bit is '0', then there is no message to M_j in this position.

Due to the orthogonal property of the secure code, it has been established that only the receiver who has the corresponding code can extract its message from the amalgamated message. At the same time, the receiver who does not know the decryption secure codes of other members is not able to interpret the messages for others. In the following section, we are going to prove the correctness of the encryption algorithm.

3.3.3 Proof of the correctness of SCGCM Encryption Algorithm

Here, we will prove that the encryption and decryption protocols of SCGCM are correct. In particular, we want to prove the correctness of SCGCM from two aspects—(1) using the decryption protocol of SCGCM, any recipient can correctly retrieve

its message that was encrypted with the encryption protocol of SCGCM; and (2) the recipient can not access the messages intended for other recipients through the decryption protocol.

In order to prove the correctness, let's consider each bit of the encrypted message (i.e. the amalgamated message). Without losing generality, if we can prove that an arbitrary recipient can correctly retrieve *an arbitrary one bit* message from an amalgamated one bit message without accessing the bit messages for other recipients, then we prove that the recipient can correctly decrypt an arbitrary *multiple bits* message without accessing other recipients' message.

Proof. Before proving the correctness, let's review the properties of orthogonal codes. Take an arbitrary member M_i , $i \in [1..N]$ for example, its encryption secure codes, i.e. $\{C_{ix} | x \in [1, N]\}$, are pair-wisely orthogonal, therefore by definition have the following properties [40]:

$$C_{ij} \cdot C_{ik} = \frac{1}{l_i} \sum_{s=1}^{l_i} (C_{ijs} \times C_{iks}) = 0 \quad (\text{Prop. 1})$$

$$C_{ij} \cdot \overline{C_{ik}} = \frac{1}{l_i} \sum_{s=1}^{l_i} (C_{ijs} \times \overline{C_{iks}}) = 0 \quad (\text{Prop. 2})$$

$$C_{ij} \cdot C_{ij} = \frac{1}{l_i} \sum_{s=1}^{l_i} (C_{ijs}^2) = 1 \quad (\text{Prop. 3})$$

$$C_{ij} \cdot \overline{C_{ij}} = \frac{1}{l_i} \sum_{s=1}^{l_i} (-C_{ijs}^2) = -1 \quad (\text{Prop. 4})$$

Where $i, j, k \in [1, N]$, and $i \neq j \neq k$

Consider a one-bit amalgamated message sent by M_i which is denoted as MSG_i .

When M_k receives MSG_i , it computes

$$\begin{aligned}
 MSG_i \cdot C_{ik} &= \left(\sum_{j=1, i \neq j}^N S_{C_{ij}}(msg_{ij}) \right) \cdot C_{ik} \\
 &= \left(\sum_{j \neq k}^N S_{C_{ij}}(msg_{ij}) + \sum_{j=k}^N S_{C_{ij}}(msg_{ij}) \right) \cdot C_{ik} \\
 &= \left(\sum_{j \neq k}^N (C_{ij}, \overline{C_{ij}}, \text{ or } 0) + S_{C_{ik}}(msg_{ik}) \right) \cdot C_{ik} \quad (\text{by Prop. 1 and Prop. 2}) \\
 &= (S_{C_{ik}}(msg_{ik})) \cdot C_{ik} \tag{3.3.1}
 \end{aligned}$$

As demonstrated, according to Prop. 1 and Prop. 2, the messages for other recipients are opaque to M_k . Therefore, it is not possible for M_k to discover the messages for other recipients using the decryption protocol.

On the other hand, there are three possible values for msg_{ik} — ‘1’, ‘0’, or nothing. That is, the scrambled message $S_{C_{ik}}(msg_{ik})$ will be $C_{ik}, \overline{C_{ik}}$, or ‘0’ accordingly. Next, we will prove that the recipient M_k can correctly restore msg_{ik} in all cases.

(1). M_i sends ‘1’ to M_k .

In this case, $S_{C_{ik}}(msg_{ik}) = C_{ik}$, therefore Equation 3.3.1 reduces to

$$C_{ik} \cdot C_{ik} = 1 \quad (\text{by Prop. 3})$$

i.e. M_k restores the original bit, which is ‘1’ in binary form.

(2). M_i sends ‘0’ to M_k .

In this case, $S_{C_{ik}}(msg_{ik}) = \overline{C_{ik}}$, therefore Equation 3.3.1 reduces to

$$\overline{C_{ik}} \cdot C_{ik} = -1 \quad (\text{by Prop. 4})$$

i.e. M_k restores the original bit, which is ‘0’ in binary form.

(3). M_i sends nothing to M_k .

In this case, $S_{C_{ik}}(msg_{ik}) = 0$, therefore Equation 3.3.1 reduces to

$$0 \cdot C_{ik} = 0$$

i.e. M_k will find that there is no information for it in this bit.

Based on the previous discussion, we deduce that an arbitrary recipient M_k can correctly recover a *one-bit* message sent by an arbitrary sender M_i . In addition, we deduce that M_k can not access the messages for other recipients using the decryption protocol.

Since M_k is able to decrypt a message bit by bit correctly, it can decrypt all the bits of the message thereby the whole message sent by M_i correctly provided that the sequence of the bits is kept correct. Therefore, it is proved that the encryption algorithm presented here is correct. Proof is complete. $\square$

3.3.4 Secure Code Update

The periodic update of the secure codes is very important in the management of secure codes. The protocol of Secure Code Update includes the update of the Data Secure Codes (DSCs) and the Code Exchange Secure Codes (CESCs).

Data Secure Code (DSC) update

It is essential to have data secure codes with short *cryptoperiod* to limit the exposure of the data secure codes, and thereby preserve the strength of the mechanism. The periodic update of the DSCs is accomplished using the protocol of SCGCM Secure

Group Communication with the corresponding CEESCs as the encryption codes. Sometimes, it is desirable to reduce the frequency of the generation of secure codes. Hence, a DSC can be used for more than one session as long as the security of the code is not risked.

Code Exchange Secure Code (CESC) update

The update of the CESC is accomplished using the protocol of SCGCM Secure Group Communication with previous CEESCs as the encryption codes. One motivation of using chained secure codes update is that since the CESC is less likely to be compromised because they are used much less frequently than the DSCs. Another motivation is that the impact of a compromised code can be confined to a certain period of time thereby *forward secrecy* being preserved (i.e. past sessions are kept secret). Using the previous CESC to update a CESC is much safer than using the long-term key (e.g. the symmetric shared key or the public key) directly because if the long-term key is compromised, all past sessions will be disclosed and no *forward secrecy* will be kept at all.

Figure 3.4 illustrates the impacts of a compromised CESC. As illustrated, a CESC is used to update several DSCs and a CESC is updated using its previous CESC. Suppose that an attacker compromises $CESC_i$ using crptoanalysis at time t_i , past sessions remain protected up to this point since the past CESC (i.e. from $CESC_1$ to $CESC_{i-1}$) are irrelevant to the compromised code $CESC_i$. Suppose at time t_j , the compromised CESC is detected, and new CESC is updated using the protocol of CESC transmission presented in Section 3.3.1, i.e. they are updated using the long term symmetric shared key or the public key. As a result, the communications after

Figure 3.4: The Impacts of a compromised CESC

t_j will be secure while the DSCs, the CESC and the data traffic transmitted during t_i to t_{j-1} (within the shadowed box) are compromised.

Also note that SCGCM employs a fully distributed secure code management scheme so that the compromise of a CESC will not affect the secure group communications with other members. For example, suppose at time t_p , M_i 's $CESC_{ij}$ is compromised, all other CESC (i.e. $CESC_{ix}$ and $CESC_{xi}$, where $x \in [1, N]$ and $x \neq j$) will not be effected, provided that it is computationally infeasible to deduce other random orthogonal codes from one code. Therefore, M_i can continue to have secure communication with other members.

3.3.5 SCGCM(BA)—Join Protocol

When a new member M_i requests to join the group, the new member is required to assign a Code Exchange Secure Code (CESC) and a Data Secure Code (DSC) to each of the other members that it wants to communicate with in a secure manner.

This procedure is the same as the protocol of Secure Code Exchange presented in Section 3.3.1.

Here, two special cases are possible. In the first case, there is no requirement for M_i to communicate with some of the group members securely. However, those group members may need to send information securely to M_i . In this case, the new member M_i does not need to assign the secure codes to those group members. Instead, those group members have to assign the secure codes to M_i . Hence, one-way secure communication from those group members to M_i is achieved.

The second case is the reverse of the first one, i.e. M_i wants to securely talk with some group members, but those group members are not required to talk to M_i in a secure manner. In this case, only M_i is required to send secure codes to those members. As a result, one-way secure communication from M_i to those group members is possible.

When multiple members make requests to join, the join protocol is still applicable. Since the secure code management is fully distributed, the join protocol can add these new members to group communication without interrupting the current communication with other members. Moreover, since the secure codes assigned to the new members are independent to the codes assigned to other members, backward access secrecy is achieved.

3.3.6 SCGCM(BA)—Leave Protocol

The purpose of the Leave Protocol is to prevent the leaving member or the evicted member from participating in further group communication. If a member M_j leaves the group or is evicted of the group, we can simply delete all the codes related to

M_j . In other words, each of the remaining members deletes the row corresponding to member M_j from its secure code tables.

Suppose that it is computationally expensive to derive other orthogonal codes from a known code in a set of random orthogonal codes. Thus, the leave protocol is adequately secure to prevent the departing member from accessing future group communication. For example, when M_j leaves the group, all related codes, i.e

$$CESC_{xj}, DSC_{xj}, CESC_{jx}, DSC_{jx}, \text{ where } x \in [1, N]$$

are revoked. Since it is hard to deduce

$$CESC_{xy} \text{ (or } DSC_{xy}), \text{ where } y \in [1, N] \text{ and } y \neq j$$

from $CESC_{xj}$ (or DSC_{xj}), M_j cannot access further communication between other group members.

When multiple members must be removed at the same time, the same leave protocol can be used. Each remaining group member merely deletes appropriate row from its secure code table.

3.4 SCGCM Extension Algorithm I

The Basic Algorithm discusses a general case of group communication where messages for different recipients may not be same. In this section, we consider a common kind of group communications where a sender usually sends the same message to all or part of group members.

Basic Algorithm can accomplish this kind of secure group communication with no problem, however it may not be optimal. This is because a group message has to be

scrambled individually for different recipients. As a result, the number of scrambling operations is linear to the number of recipients.

However, compared with conventional encryption algorithms, such as Data Encryption Standard (DES), which requires 16 rounds of encryption function—each round includes expansion, XOR, S-box, permutation and XOR with the left part of a message, the cost of scrambling a message for a recipient with SCGCM is almost 5% of DES. Moreover, SCGCM decryption is even trivial in comparison with DES decryption, which contains 16 rounds of reverse DES encryption. Hence, for a group that has not too many group members, the Basic Algorithm of SCGCM is enough.

When a group is very large or group messages are particularly large, it is necessary to optimize the Basic Algorithm. SCGCM Extension Algorithm I (SCGCM(EX_I)) is designed for this purpose.

The general idea of SCGCM(EX_I) is that a user divides a set of group members into several subgroups. Each member is assigned a *subgroup DDSC* in addition to the decryption secure codes (CEDSCs and DDSCs) as discussed previously. Please note that subgroup DESCs and individual DESCs are mutually orthogonal. In other words, they are chosen from the same set of secure codes.

One motivation of dividing a large group into several small subgroups is to decrease the instability of the whole group. On one hand, if the whole group is taken as one group, then any member joining/leaving the group will cause the group code to be changed accordingly. On the other hand, if we divide a large group into several small subgroups, then the effects of a member's joining or leaving are restricted to the subgroup(s) with which it is associated.

Moreover, when considering the cost comparison of the SCGCM encryption algorithm with DES, SCGCM is as efficient as traditional shared group key solutions (suppose they use DES) provided that there are less than twenty subgroups.

The criteria for the creation of subgroups could be based on the user's knowledge such as organizational structure, or future communication with potential recipients. In this way, the number of times that data must be scrambled by a member is reduced to a constant—equal to the number of subgroups.

As each member individually creates its subgroups, the number of subgroups created by each member will tend to vary. Each member records the subgroup information in its *extended secure code table*.

3.4.1 SCGCM(EX_I)—Secure Code Exchange

The protocol of Secure Code Exchange of the Extension Algorithm I is similar to that of the Basic Algorithm, except for the fact that after CESC's are interchanged, a Subgroup ID, a *subgroup DSC* as well as an individual DSC are encrypted and exchanged during the DSC exchange stage.

After the secure codes have been exchanged, an extended secure code table is obtained. Figure 3.5 shows an example of the member M_i 's extended secure code table. Two new columns—"Subgroup Info" and "Assigned Subgroup Info" have been added to its *secure code table*. While "Subgroup Info" contains the subgroup ID and the subgroup DSC that the table owner M_i has assigned to the member identified in a row, the "Assigned Subgroup ID" contains the subgroup ID and the subgroup DSC that this member has assigned to M_i .

Member List	CEESC List	CEDSC List	DESC List	Subgrp Info	DDSC List	Ass. Subgrp Info
M_1	CE_{i1}	CE_{1i}	C_{i1}	$(ID C')_{i1}$	C_{1i}	$(ID C')_{1i}$
...	...	...	...	...	...	...
M_{i-1}	$CE_{i,i-1}$	$CE_{i-1,i}$	$C_{i,i-1}$	$(ID C')_{i,i-1}$	$C_{i-1,i}$	$(ID C')_{i-1,i}$
M_{i+1}	$CE_{i,i+1}$	$CE_{i+1,i}$	$C_{i,i+1}$	$(ID C')_{i,i+1}$	$C_{i+1,i}$	$(ID C')_{i+1,i}$
...	...	...	...	...	...	...
M_N	CE_{iN}	CE_{Ni}	C_{iN}	$(ID C')_{iN}$	C_{Ni}	$(ID C')_{Ni}$
Unassigned CEESC & DESC for new group members						

CEESC: Code Exchange Encryption Secure Code

CEDSC: Code Exchange Decryption Secure Code

DESC: Data Encryption Secure Code

DDSC: Data Decryption Secure Code

Subgrp Info: The subgroup ID and subgroup code that table's owner M_i assigns to the row member

Ass. Subgrp Info: The subgroup ID and subgroup code that row member assigns to the table's owner M_i

Figure 3.5: An Example of M_i 's Extended Secure Code Table

Source Address	Destination Address	Message Indicator	Subgroup IDs	Member IDs	Amalgamated Message
----------------	---------------------	-------------------	--------------	------------	---------------------

Message Indicator:

01	includes subgroup message only
10	includes member message only
11	includes both subgroup message and member message

Destination Address can be a broadcast address,
or a multicast address

Figure 3.6: SCGCM(EX_I)—The Format of a Message sent by a Member

3.4.2 SCGCM(EX_I)—Encryption

A unique characteristic of the Extension Algorithm I is that the resulting amalgamated message can contain pure subgroup messages or pure individual messages or a mixture of them. It requires a message header called "Message Indicator" to indicate the composition of an amalgamated message.

Figure 3.6 shows the format of a message using Extension Algorithm I. Here, the Destination Address can be a broadcast address or a multicast address (if applicable) for the whole group. In addition, "Message Indicator" indicates the contents of the amalgamated message. If an amalgamated message contains only subgroup messages, the "Message Indicator" is set to 01, the "Member ID" section is empty, and the "Subgroup IDs" section contains all subgroup IDs of the recipients. To send pure individual messages, the "Message Indicator" is set to 10, the "Member IDs" section contains all member IDs of the recipients, and the "Subgroup IDs" section is empty. Otherwise, for both subgroup and individual member messages, the "Message

Indicator” is 11, the “Member IDs” section contains all member IDs of the recipients and the “Subgroup IDs” section contains all subgroup IDs of the recipients.

3.4.3 SCGCM(EX_I)—Decryption

The decryption operation is similar to the one used in the Basic Algorithm. However, depending on the values of “Message Indicator”, a recipient can obtain different types of messages.

When a recipient receives an amalgamated message, it will check the “Message Indicator”. If it is 01, it means that the payload message consists of pure subgroup messages. According to the subgroup IDs listed in the “Subgroup IDs” section, a recipient will determine if there is a (sub)group message for it and, if so, use the corresponding subgroup DDSC to decrypt the message. If the “Message Indicator” is 10, it means that the payload message consists of pure member messages. Using the “Member IDs” section, a recipient will discover whether or not there is a message for it and will decrypt the message using the appropriate individual DDSC. Finally, a value of 11 indicates that the payload contains a mixed message. In this case, a recipient will have to check both the “Subgroup IDs” section and the “Member IDs” section to see if there are zero, one or two messages for it. If necessary, it will use the corresponding code(s) to retrieve the message(s).

3.4.4 SCGCM(EX_I)—Secure Code Update

Like the Secure Code Update protocol of the Basic algorithm, the Secure code update of SCGCM Extension algorithm I consists of a CESC and a DSC update. The CESC

update of SCGCM(EX_I) is the same as the update process used in the Basic Algorithm. However, the Data Secure Code update includes both subgroup DSC as well as individual DSC update. They are updated using the same corresponding CESC.

3.4.5 SCGCM(EX_I)—Join Protocol

When a new member M_i wants to join a group, it assigns a CESC, an individual DSC, and a subgroup DSC to all the members that it will communicate securely in the future. Here, it uses a priori knowledge to encrypt the CESC to other members thereby achieving implicitly key authentication with each of the other members. Afterwards, it uses SCGCM encryption to encrypt the DSCs (subgroup and individual) using the corresponding CESC before broadcasting the amalgamated message. The procedure is the same as the protocol of Secure Code Exchange presented in the section on Basic Algorithm.

In the meantime, the current group members who want to send secure messages to M_i will assign a CESC and an individual DSC to M_i . However, there are two options for assigning a subgroup DSC to M_i .

- **Option 1**—Assign a subgroup DSC to M_i immediately. Since the DSC of the subgroup to which M_i is going to join should be updated, an amalgamated subgroup DSC update message is constructed. While the message intended for M_i contains a subgroup ID and a subgroup DSC, an equivalent message sent to other members contains merely a new subgroup DSC. This option is suitable to less dynamic cases where new members join the group infrequently and/or the size of the affected subgroup is large.

- **Option 2**—Do not assign a subgroup DSC to M_i until the next update of subgroup DSCs. In the interim, the individual DSC is used to encrypt a message for M_i . This option is more suited to highly dynamic environment and/or the size of the affected subgroup is small.

Remark 3.4.1. Suppose that it is the originator of a message who is primarily concerned about security. The sender selects and records an appropriate option to be used for a subgroup according to the characteristics of the communication of the subgroup. This facilitates group management, especially when group membership changes. The same measures can be employed in the Leave Protocol which is discussed in the next section.

As for multiple members joining the group, each existing member associates them with the appropriate subgroups. Of course, if multiple new members are to be placed in a new subgroup, then a new subgroup is established and a new subgroup DSC is assigned to them. At the same time, each of the new members authenticates other members and broadcasts its secure codes as with the case of one member joining the group.

3.4.6 SCGCM(EX_I)—Leave Protocol

This protocol is similar to the Leave Protocol of the Basic Algorithm. When a member M_i leaves or is evicted, each of the remaining members deletes the row of secure codes, associated with M_i , from its extended secure code table. In addition, the code of the affected subgroup (i.e. subgroup to which M_i belonged to) should also be updated. There are two options:

- **Option 1**—Update the subgroup code immediately. A member broadcasts the new subgroup DSC to the affected subgroup members.
- **Option 2**—Delay the subgroup secure code update until the next update of the secure codes. In the interim, the members of the affected subgroup can use their individual DSCs to securely communicate with each other. This option reduces the frequency and the number of updates of the subgroup DSCs. Sometimes, this option is more appropriate for dynamic group collaboration within the context of wireless ad hoc networks, because it is not necessary to update the subgroup code immediately after a member leaves. In addition, because the size of a subgroup is much smaller than that of the whole group, the cost of using individual DSCs will not be too expensive.

If multiple members need to be removed from a group and most of them, if not all, belong to one subgroup, then option 1 is more appropriate; otherwise, a different option is used according to the management criteria of each subgroup.

3.5 SCGCM Extension Algorithm II

In the previous section, we presented the first extension algorithm of SCGCM. Another extension algorithm (SCGCM(EX_II)), which employs the block cipher technology will be presented in this section.

Consider the length of the resulting amalgamated message of SCGCM encryption, it is proportional to the length of the longest individual message. In other words, a long individual message will cause the resulting amalgamated message to be very long

and thus occupy a lot of bandwidth. Moreover, if errors occur during transmission, the amalgamated message will have to be resent. A sender can divide a message into several blocks to prevent this problem. One advantage is that if errors do occur, only the corrupted blocks are retransmitted. Another advantage is that the recipients with short messages do not need to wait for the complete transmission of the whole (long) message. They can decrypt the message block by block. Suppose that an *End of Message* flag is added at the end of each message, and that there is a *Nonce* that is appended for each message sent by a group member. Then, it is a simple matter for a recipient to determine the end of its message. Consequently, the efficiency of data transmission can be improved.

3.5.1 SCGCM(EX_II)—Secure Code Exchange

In addition to the exchange of the DSCs, group members have to exchange their Initialization Vectors (IV_{ij}) as well. However, if they have established an agreement before secure communication, then the Secure Code Exchange protocol is the same as that used in the Basic Algorithm ³.

Remark 3.5.1. According to [25], an Initialization Vector (IV) usually does not need to be secret, therefore a member can use an IV for all group communications. Or else, all group members can simply use a pre-determined IV. However, in order to present the algorithm in a general manner, IV_{ij} instead of IV is used.

³Note that SCGCM(EX_II) can also be used with Extension algorithm I. We merely use the Basic Algorithm for an example here.

3.5.2 SCGCM(EX_II)—Encryption

Before encrypting the messages, a sender M_i divides all the messages that it prepares to send into blocks—each of which has b bits. The last block may need to be padded with 0 from the right.

Remark 3.5.2. The length of the messages for different recipients may be different. However, this will not affect the operation of the block cipher since short messages can be padded with '0'. For discussion purposes, we assume that all messages are divided into the same number of blocks.

In order to avoid a situation where a recipient, who has a short message, wastes its time receiving information destined to other recipients, an “End of Message” flag is used at the end of its message. After all messages are prepared, they are scrambled and amalgamated block by block using the encryption protocol of SCGCM.

Remark 3.5.3. There is a *nonce* associated with each message sent by a member, which is used to uniquely identify the message and the sequence of the message sent by that member and as a result thwart the *reply attack*. In addition, a subsequence number is used to identify each block message so that a recipient can tell if two block messages belong to the same message or not.

The block mode can be Electronic CodeBook (ECB) mode or Cipher Block Chaining (CBC) mode. Although the final cipher message is an amalgamated message in SCGCM, when considering each individually scrambled message ECB mode of SCGCM, the same plaintext block is still encrypted to the same ciphertext block with the same secure code with ECB mode in SCGCM. Therefore, it is vulnerable to the collusion of the compromised group members. On the other hand, since CBC

mode has the property of chaining dependency, it may prove beneficial in some applications where error detection (or data integrity) is required. SCGCM(EX.II) in CBC mode is presented as follows.

Figure 3.7: SCGCM(EX.II)—Encryption in Cipher Block Chaining (CBC) mode

Figure 3.7 illustrates how member M_i encrypts the messages in CBC mode. It divides every message into blocks, for example, msg_{i1} is divided to m blocks, i.e.

$$msg_{i1} = \{msg_{i1_1}, msg_{i1_2}, \dots, msg_{i1_m}\}$$

The first block message msg_{i1_1} is XORed with IV_{i1} , and then it is scrambled with C_{i1} . All the resulting scrambled block messages are added together to obtain the first block of the amalgamated message which is also used as an input to the XOR operation of the next block. I.e. the first block of the amalgamated message is

$$MSG_{i_1} = \sum_{j=1, i \neq j} S_{C_{ij}} (msg_{ij_1} \oplus IV_{ij}) \quad (3.5.1)$$

and the k^{th} block of the message is

$$MSG_{i_k} = \sum_{j=1, i \neq j} S_{C_{ij}} (msg_{ij_k} \oplus MSG_{i_{k-1}}) \quad (3.5.2)$$

3.5.3 SCGCM(EX_II)—Decryption

After a recipient M_j receives a message, it first decrypts each cipher block using the decryption protocol of SCGCM. Then, it *XORs* the intermediate results with previous cipher block so as to recover the original block message. To decrypt the first cipher block, IV_{ij} is used instead of the previous cipher block, i.e. the k^{th} block message, where $k \in [1, m]$ is:

$$msg_{ij_k} = \begin{cases} (MSG_{i_1}) \cdot C_{ij} \oplus IV_{ij} & \text{if } k = 1 \\ (MSG_{i_k}) \cdot C_{ij} \oplus MSG_{i_{k-1}} & \text{if } k \neq 1, k \in [1, m] \end{cases}$$

Figure 3.8 illustrates how M_1 decrypts a message sent by M_i .

3.5.4 Proof of the Correctness of Extension Algorithm II

In this section, we are going to prove that the SCGCM Extension Algorithm II can correctly decrypt a message that was encrypted using the encryption protocol of SCGCM(EX_II). Note that the definition of the correctness here is the same as that in Section 3.3.3.

Figure 3.8: SCGCM(EX.II)—Decryption in Cipher Block Chaining (CBC) mode

Proof. In order to prove the correctness, we consider an arbitrary secure communication between any two arbitrary members M_i and M_j . Without losing generality, we suppose M_i is the sender and M_j is one of the recipients.

When M_j receives a block message MSG_{i_k} sent by M_i , it decrypts the message:

- (1). If $k = 1$, then it computes

$$\begin{aligned}
 & D_{c_{ij}}(MSG_{i_1}) \oplus IV_{ij} \\
 &= \left(\left(\sum_{p=1, i \neq p} S_{c_{ip}}(msg_{ip_1} \oplus IV_{ip}) \right) \cdot C_{ij} \right) \oplus IV_{ij} \quad (3.5.3)
 \end{aligned}$$

Recall that in Section 3.3.3 we proved that a member could correctly retrieve the message for it by computing the normalized inner product of its decryption secure code and an amalgamated message, i.e.

$$\left(\sum_{p=1, i \neq p} S_{c_{ip}}(msg_{ip_1} \oplus IV_{ip}) \right) \cdot C_{ij} = (msg_{ij_1} \oplus IV_{ij})$$

Hence, Equation 3.5.3 reduces to

$$(msg_{ij_1} \oplus IV_{ij}) \oplus IV_{ij} = msg_{ij_1} \quad (3.5.4)$$

(2). If $k \neq 1$, then M_j computes

$$\begin{aligned} & D_{C_{ij}}(MSG_{i_k}) \oplus MSG_{i_{k-1}} \\ &= \left(\left(\sum_{p=1, i \neq p} S_{C_{ip}}(msg_{ip_k} \oplus MSG_{i_{k-1}}) \right) \cdot C_{ij} \right) \oplus MSG_{i_{k-1}} \\ &= (msg_{ij_k} \oplus MSG_{i_{k-1}}) \oplus MSG_{i_{k-1}} \\ &= msg_{ij_k} \end{aligned} \quad (3.5.5)$$

Equation 3.5.4 and Equation 3.5.5 prove that an arbitrary recipient can correctly recover an amalgamated message sent by an arbitrary sender who encrypts the message using the encryption protocol of SCGCM(EX_II).

Proof is complete. □

The ECB mode of SCGCM is similar to the CBC mode with one key exception. The previous amalgamated block message does not serve as input to the next message block during encryption and decryption.

Proof of the SCGCM(EX_II) in ECB mode is trivial since the encryption and decryption of a block message in ECB mode is exactly the same as the SCGCM encryption algorithm—the only difference is that the length of a message is trimmed to the length of a block. Recall that when we proved the correctness of the encryption algorithm of SCGCM in Section 3.3.3, an arbitrary recipient could recover an arbitrary multiple-bits message. Therefore, the correctness of the SCGCM(EX_II) in ECB mode is trivially proven.

3.6 Discussion of SCGCM

It is important to note that the security of SCGCM systems depends on available computer resources. Understanding their security limitations can only be the subject of further investigations involving simulations.

In this section, we compare the SCGCM algorithms (i.e. SCGCM(BA) and SCGCM(EX_I)) with other solutions and discuss the characteristics of SCGCM algorithms.

3.6.1 Comparison with other solutions

Besides those solutions reviewed in Chapter 2, a Simple Solution (SS) is used for comparison as well. In Simple Solution, each member acts as a key controller to generate and distribute a key to each of the other group members. Hence, any combination of members can securely communicate using the corresponding (*Encryption and Decryption*) individual keys.

We compare SCGCM with other solutions in all aspects including the storage overhead and the computation and communication overhead involved in group initialization, member leaving, member joining and data communication. In addition, the effects of the overhead on group communication and scalability as well as some improvement measures are examined.

Among the six algorithms in Table 3.2, only SS, SCGCM(BA), and SCGCM(EX_I) are fully distributed and provide *Source Authentication* for group communication. They also support *arbitrary subgroup communication*, while Tree Key Graph (TKG)

Table 3.2: Comparison of SCGCM With Other Solutions

Parameters	SS	SCGCM (BA)	SCGCM (EX_I)	A-GDH.2	TKG	OFT
Central. Ctrl. ¹	No	No	No	Yes [†]	Yes	Yes
Subgrp. Comm. ²	Yes [†]	Yes [†]	Yes [†]	No	Yes	Yes
Key Confirm.	Yes	Yes	Yes	No	No	No
Being Delayed	No	No	No	Yes	No	No
Storage(Ctrl)	N/A	N/A	N/A	$O(N)$	$O(N)$	$O(N)$
Storage (Member) ³	$O(N)$	$O(N)$	$O(N)$	$O(1)$ or $O(N)$	$O(\log_k N)$	$O(\log_2 N)$
Msg. Length	$O(N)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Group Initialization						
Bits/unicast ⁴	0	0	0	$O(N)$	0	0
Bits/Broadcast ⁵	$O(N)$	$O(N)^*$	$O(N)^*$	$O(N)$	$O(N \log_k N)$	$O(N \log_2 N)$
Max. Encrypt. ⁶	$O(N)$	$O(N)^*$	$O(N)^*$	$O(N)$	$O(N \log_k N)^{\dagger}$	$O(N)^{\dagger}$
Max. Decrypt.	$O(N)$	$O(N)$	$O(N)$	$O(1)$	$O(\log_k N)$	$O(\log_2 N)$
Join Protocol						
Bits/unicast	$O(1)$	$O(1)$	$O(1)$	$O(N)$	$O(\log_k N)$	$O(\log_2 N)$
Bits/Broadcast	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(\log_k N)$	$O(\log_2 N)$
Encrypt. (New Member)	$O(N)$	$O(N)$	$O(N)$	$O(N)$	0	0
Decrypt. (New Member)	$O(N)$	$O(N)$	$O(N)$	0	$O(\log_k N)$	$O(\log_2 N)$
Max. Encrypt. (Other Member) ⁷	$O(1)$	$O(1)$	$O(1)$	$O(N)$	$O(k \log_k N)^{\dagger}$ $O(\log_k N)$	$O(\log_2 N)^{\dagger}$ $O(\log_2 N)$
MAX. Decrypt. (Other Member)	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(\log_k N)$	$O(\log_2 N)$
Leave Protocol						
Bits/Broadcast	0	0	[1] : $O(1)$ [2] : 0	$O(N)$	$O(k \log_k N)$	$O(\log_2 N)$
Max. Encrypt. ⁸	0	0	[1] : $O(1)$ [2] : 0	$O(N)$	$O(k \log_k N)^{\dagger}$	$O(\log_2 N)^{\dagger}$
Max. Decrypt.	0	0	[1] : $O(1)$ [2] : 0	$O(1)$	$O(\log_k N)$	$O(\log_2 N)$

1. Key controller of A-GDH.2 can be dynamic.

2. SS, SCGCMs accomplish arbitrary group members communication.

3. To substitute a controller, a member has to store the previous broadcast in A-GDH.2.

4. Suppose all protocol use concatenate broadcast instead of unicast

5, 6. If CESC's in SCGCMs are pre-exchanged, then broadcast bandwidth will become $O(1)$.

Also the computation costs are greatly reduced, see Remark 3.6.1.

6, 7 and 8. † indicates encryption done by the controller; others are done by group members

and One-way Function Tree (OFT) only provide limited secure subgroup communication that is restricted by the key graph (or the tree). Not only does A-GDH.2 not support subgroup communication but the group key agreement procedure can also be delayed by any slow members.

In TKG and OFT, the storage cost of the key controller is $O(N)$ and of each member is $O(\log N)$. For A-GDH.2, it is $O(N)$ for the key controller and $O(1)$ for normal group members. However, for the member who could become a key controller⁴, it is $O(N)$. For fully distributed systems such as SS, SCGCM(BA), and SCGCM(EX_I), the storage cost of each member is $O(N)$.

In TKG, OFT and A-GDH.2, the storage cost of the key controller is much more than other members, therefore the key controller can be a bottleneck for systems scalability. To compare with that, the storage overhead of a member in SCGCM is equally $O(N)$. When the size of the group expands, members with inferior storage capabilities may not be able to properly communicate with all other members. However, due to the flexibility of SCGCM, communication between other members will be least affected by the inferior member. At the same time, the inferior member can choose to merely communicate with the most wanted members.

To illustrate how severe it can be to have $O(N)$ storage in SCGCM algorithms, let us consider the number of bits a secure code will have. According to [30], the code length for commercial Spread Spectrum systems is 10 to 100 bits, while the code length for military systems is 1000 to 1 million bits. As we do not have experimental results indicating the number of optimal bits to be used with SCGCM, high end

⁴The main reason is that in A-GDH.2, if the key controller is compromised or leaves, a new key controller must be selected. The new controller needs to store previous broadcast message. Details can be found in Chapter 2.

commercial Spread Spectrum systems (code length is 100 bits, i.e. 12.5 bytes) are used as an example. If 1000 group members want to communicate simultaneously, it will require 50KB(12.5×4) for each member to store the codes in SCGCM(BA) and 75KB(12.5×6) in SCGCM(EX_I). Moreover, if we use data compression technology, the storage cost can be further decreased. In contrast, the available memory for current wireless devices, such as a Palm, ranges from 8MB to 16MB, while memory for a Pocket PC ranges from 16MB to 32MB. Thus, storage will not be a big issue for SCGCM algorithms.

As for group initialization, the number of computations are fairly equivalent in SS, SCGCM(BA), and SCGCM(EX_I), i.e. each member has $O(N)$ computations including encryption and decryption. In contrast, with A-GDH.2, TKG, and OFT, the key controller has more computations than end users—the computations of the key controller are $O(N)$, $O(N \log_k N)$ and $O(N)$ respectively; while the computations of their end users are $O(N)$, $O(\log_k N)$, and $O(\log_2 N)$ respectively⁵.

On the other hand, the communication overhead of SS, SCGCM(BA), SCGCM(EX_I), and A-GDH.2 in group initialization is $O(N)$ bits per broadcast, whereas that of TKG and OFT is $O(N \log_k N)$ and $O(N \log_2 N)$ bits per broadcast. While there are N broadcasts in SS, SCGCM(BA), and SCGCM(EX_I) for fully meshed N -to- N communication, only m broadcasts are needed for m -to- N communication, where m is the number of senders.

Remark 3.6.1. The overhead of the initial code exchange seems to be very large for large fully meshed groups. However, if all group members have pre-exchanged their

⁵The actual number of computations of a user in A-GDH.2 depends on that user's position among the ring. However, the maximum number of computations is $O(N)$.

CESCs, then the initial secure code exchange is able to employ SCGCM encryption algorithm, and thereby the bandwidth taken by each member for code exchange is merely $O(1)$. Meanwhile, the computation cost of each member is also reduced One twentieth due to the lightness of the SCGCM encryption algorithm. Note that the group size of a fully meshed N -to- N group communication generally is not very big, thus the overhead of the initial secure code exchange in SCGCM will not be a significant bottleneck for the scalability.

When a member joins, there is a unicast message with $O(1)$ bits from each existing member to the new member, and a broadcast message with $O(N)$ bits from the new member to all existing members in SS, SCGCM(BA), and SCGCM(EX_I). In AGDH.2, TKG, and OFT, however, there is a unicast message with $O(N)$, $O(\log_k N)$, and $O(\log_2 N)$ bits respectively and a broadcast message with $O(N)$, $O(\log_k N)$, and $O(\log_2 N)$ bits respectively. As for TKG and OFT, there are only logarithm computations for the new member whereas in all other algorithms, there are $O(N)$ computations. However, in SS, SCGCM(BA), and SCGCM(EX_I), there are merely $O(1)$ computations for existing group members, while TKG and OFT require logarithm computations. In some applications, it may be desirable to minimize the impact of changes to group membership on the existing members. Furthermore, if we suppose that the new member has pre-exchanged the CESCes with other members, the computation and the communication overhead of the new member will be greatly decreased. See Remark 3.6.1.

When a group member leaves, SS, SCGCM(BA), and SCGCM(EX_I) have trivial computation and communication overhead. One reason is that a group key is not used in SS and SCGCM(BA), thus group communication will not be affected

by the departing member. In this case, the computation of an existing member is almost zero⁶. In SCGCM(EX_I), where the whole group is divided into subgroups, the impact of a member leaving the group is restricted to its subgroup. When a member leaves, the computation overhead of an existing member is zero if the related subgroup code is not updated immediately, or $O(1)$ if all remaining members update the related subgroup code immediately. Usually, it is not necessary to update the subgroup code immediately because SCGCM(BA) can be used temporarily for secure subgroup communication until the secure codes are updated according to a pre-established schedule. The only inconvenience is the need for multiple data scrambling operations within the affected subgroup. However, the relatively smaller size of the subgroup reduces the number of data scrambling operations. As a result, group communication will not be interrupted by departing members. In contrast, there are $O(N)$ computations and communications in A-GDH.2 and logarithm computations and communications in TKG and OFT.

From the previous discussion, we can see that the first three algorithms (i.e. Simple Solution (SS), SCGCM(BA), and SCGCM(EX_I)) demonstrate excellent characteristics for dynamic group communication. However, one major disadvantage of SS is that the bandwidth required for group communication is linearly related to the number of group members which makes it unsuitable for group communication. In contrast, the bandwidth required for group communication in SCGCM is not linear to the group size. Instead, it depends on the length of the DSC.

⁶Here, changes of group membership are assumed to be detected by the underlying networks. However, the analysis of the related cost is out of scope of this thesis.

3.6.2 Characteristics of SCGCM algorithms

The characteristics of SCGCM algorithms are summarized as follow:

- Source authentication

They support source authentication for group communication because a recipient can only decrypt a message sent by a specific sender using the specific secure code that was assigned by the sender.

- Confidentiality

They provide data communication confidentiality using data scrambling and data amalgamating technology. In addition, they provide forward access secrecy. Since the secure codes used by each member are pseudo-random and independent, when a member leaves and the related secure codes are deleted, the departing member will not access future communication between the group members. Similarly, they provide backward access secrecy. When a new member joins, existing group members will assign new secure codes to the new member and the new member cannot deduce the secure codes of other members using these newly assigned secure codes. Therefore, it will be unable to access the communication which had commenced prior to its membership.

- Efficiency

SCGCMs employ orthogonal properties of pseudo random codes. The encryption and decryption operations are very simple. Twenty SCGCM encryptions are equivalent to one DES encryption. SCGCM decryption is even trivial in comparison to DES decryption. One advantage is that it consumes less battery, a desirable characteristic for wireless devices.

- Support highly dynamic environment

Recall the analysis of the performance of SCGCM algorithms. Since the secure code management of SCGCM algorithms is fully distributed, and the relationship between group members is loose, there is little impact when membership changes. Thus, the algorithms are very efficient in order to support highly dynamic environment.

- Scalability

The impeding factors of scalability in previous GKM schemes do not exist in fully distributed systems such as SS and SCGCM. By using the orthogonal property of secure codes, the message length transmitted in SCGCM will not increase with the increase of group members (suppose the length of the secure code is constant) thereby the scalability of SCGCM being greatly improved.

- Flexibility and Generality

SCGCM achieves secure communications between arbitrary members and provides both one-way and two-way secure communication within a group at the same time. Moreover, a unique feature of SCGCM is that it is able to send both group message and individual message at the same time. Furthermore, the applications of SCGCM are independent of the underlying communication networks. In fact, SCGCM works well in large group communications in wired networks, such as pay per view, video conference, etc.

3.6.3 Drawbacks of SCGCM algorithms

A drawback of the mechanism is that there is a spreading factor associated with group communication in SCGCM. The spread factor is linear in the length of the secure encryption codes. Take the codes used in high end commercial Spread Spectrum systems for example (code length is 100 bits [30]), the spread factor can be 100.

One possible way to reduce the spreading factor is to assign a member multiple (say x) DSCs at a time. Since the size of an amalgamated message is not linear to the number of messages, we can scramble x messages⁷ to a member at a time, therefore the bandwidth of the resulting amalgamated message is reduced by a factor of x . Of course, the storage cost has to be increased accordingly. There is a tradeoff between storage cost and bandwidth consumption. Moreover, note that assigning x secure codes to a member will result in x secure codes being disclosed when that member leaves the group, therefore the upper bound of x is restricted by the security strength of SCGCM, the storage capabilities of a group member as well as the security requirements of different applications.

⁷ x messages can be x different messages or a message is divided into x small messages.

Chapter 4

Case Studies of SCGCM Applications

In this chapter we present potential applications of SCGCM. As we have mentioned before, possible applications of SCGCM exist not only in the context of wireless ad hoc networks, but also in wired networks such as the Internet. Hence, applications of SCGCM fall into two categories: one-to-many and many-to-many. While the mechanism of applying SCGCM to one-to-many communication such as Pay-per-view and News subscription is described in the following section, the application of SCGCM to many-to-many communication such as Video conference and Ad Hoc Collaboration is addressed in section 4.2.

Please note that the examples provided in this chapter have been specifically chosen in order to illustrate the basic operations of SCGCM and thus does not constitute the full range of potential applications.

4.1 One-to-many communication

In this section, we present two different types of one-to-many communication scenarios: Pay Per View and News Subscription.

4.1.1 Pay Per View

Suppose there are N members: M_0 is the central server, and M_i , ($0 < i < N$) are subscribers. In this case, only the server is concerned about the secrecy of the message being transmitted. Therefore, it needs to have a random secure code generator and to distribute the Data Secure Codes (DSCs) and/or the Code Exchange Secure Codes (CESCs) to its subscribers. Note that since the server is able to decide how to configure the security mechanism, it can assign a CESC to a subscriber when the subscriber registers the service. On the other hand, the subscribers only need to decrypt the amalgamated message.

Given that the information transmitted in Pay Per View is generally very large and the number of subscribers is massive, it will be more efficient to use SCGCM(EX_I). In addition, the central server is less likely to send a program to only one subscriber, thus the use of an individual Data Secure Code (DSC) almost has no meaning. The use of CESCs and Subgroup DSCs alone should be enough.

With SCGCM(EX_I), the server dynamically creates subgroups amongst subscribers according to the program that they request. It is obvious that the membership of a subgroup will change frequently because a given subscriber may order different movies each time thereby becoming a member of different subgroups. In fact, the server can only determine the membership of a subgroup after receiving the

subscribers' requests. If we consider the transmission of a movie as one session, the subgroup DSCs should be updated in every session.

Subgroup DSCs can be assigned when a subscriber makes a request for a program. If the subscriber discontinues the service, then all the related secure codes are deleted. Similarly, if a new subscriber joins, it is provided with a CESC upon registration. A subgroup DSC will be assigned to it when the subscriber orders a program.

CESCs are periodically updated using the protocol of secure code update of SCGCM(EX_I), and subgroup DSCs are deleted after each session. New subgroup DSCs are regenerated and distributed when a new session begins.

Let g denote the number of subgroups; that is, there are g programs which have been ordered. The computational cost for the server to deliver g programs to its subscribers is one SCGCM encryption with g scrambles. The length of the transmitted (amalgamated) message is equal to the length of the longest scrambled information, and is independent of the number of recipients and programs. While the storage cost for the server is N CESC and g subgroup DSCs, the cost for a recipient is merely one CESC and one subgroup DSC.

There is a spreading factor of SCGCM in encoding data messages. To reduce the spreading factor (i.e. the bandwidth consumption), x subgroup DSCs are assigned to a subgroup at a time. In other words, an originated message can be divided into x small messages and be scrambled with each of the x subgroup DSCs. Let l denote the length of a subgroup DSC, L the length of a CESC, c a constant. If $x = l$, then the spreading factor is completely eliminated. The storage cost of a member is increased from $L + l + c$ bytes to $L + lx + c$ bytes while that of the server is increased from $NL + gl + c$ bytes to $NL + xgl + c$ bytes. (Take $x = l = 12.5$ bytes and $g = 20$

for example, the storage cost of a member is increased by 1.25KB and of the server is increased by 25KB.). On the other hand, the computation overhead is hardly added with this optimization. This can be illustrated by an example. Let l_m denote the length of a message, scrambling a l_m bytes message with a (subgroup) DSC has roughly the same computational cost as scrambling x ($\frac{l_m}{x}$ bytes) messages with x different (subgroup) DSCs.

Furthermore, in order to improve the efficiency of the mechanism, we can apply SCGCM(EX-II) to SCGCM(EX-I), in which case, the original messages are divided into blocks. Possible block cipher modes we can use are ECB and CBC. Since the security concern in Pay Per View is confidentiality instead of integrity, there may not be rigorous requirements for errors or falsification during movie (video) transmission. Consequently, it may be more beneficial to use ECB mode as CBC mode will propagate errors.

4.1.2 News Subscription

Electronically delivered news, stock quotations, and journals, etc. is becoming more and more popular with the development of Internet and wireless technology. For news subscription, only the service provider is concerned about data security, while subscribers are more concerned with the accuracy and reliability of information, thus it requires source authenticity.

The groups in news subscription is not as dynamic as those in Pay Per View. Therefore, the cryptoperiod of a Subgroup DSC can be longer more than one session. This is a typical application of SCGCM(EX-I).

The server can divide subscribers into subgroups according to their subscription. During the initial secure code exchange stage, CESC's and individual DSC's are distributed from the server to the subscribers. Subgroup DSC's can be distributed subsequently after the subgroup has been determined. Should a member leave within the cryptoperiod of the subgroup DSC, SCGCM(EX_I) offers two options. In the case where the size of the effected subgroup is small and/or the group is highly dynamic, it may prove beneficial to use *Option 2* of SCGCM(EX_I), i.e. to delay the subgroup DSC update until the secure codes are updated. One advantage is that it reduces the number of subgroup code updates and decreases the impacts of a (sub)group's instability. In addition, it reduces the usage of bandwidth. Otherwise, if group membership does not change frequently and/or the subgroup size is very large, we should choose *Option 1* of SCGCM(EX_I). In brief, the selection of a given option is merely a tradeoff between the communication overhead and the computation overhead.

The storage cost of the server is $2N + g$ (where g is the number of subgroups) and of a subscriber is $O(1)$ which includes a CESC, a DSC and a subgroup DSC. When a new subscriber is added to the group, it is authenticated and is assigned a CESC. In addition, a subgroup DSC and an individual DSC are subsequently assigned to the new subscriber which involves $O(1)$ communication overhead. When a subscriber leaves or is evicted, there is trivial computation and communication overhead if using *Option 2*; and $O(1)$ computation and communication overhead if using *Option 1*.

4.2 Many-to-many communication

In this section, two kinds of many-to-many communication scenarios are given. Section 4.2.1 discusses the application of SCGCM algorithms to Video Conference followed by a discussion of the application of the same algorithms to ad hoc collaborations in Section 4.2.2.

4.2.1 Video Conference

This is a typical many-to-many communication scenario. A video conference is a live connection between people in separate locations for the purpose of communication, usually involving audio and often text as well as video. The most basic video conference provides the transmission of static images and text between two locations while the most sophisticated ones provide the transmission of full-motion video images and high-quality audio between multiple locations[12]. Practical applications of video conference include Distance learning, Telemedicine, Remote business meeting, and Remote Research & Development collaboration.

As every participant of a video conference is concerned with the security of the communication, each is equipped with a SCGCM module. Since most of video conference applications involve interactive audio and video transmissions, real time transmission is required. Moreover, it is assumed that all members have the ability to receive the messages sent by other members thereby the relationship between group members being fully meshed (N-to-N). Therefore, group members should pre-exchange their CESC's before the conference so as to reduce the overhead of the initial exchange of secure codes.

There are two possible applications of SCGCM depending on the size of the group.

(1). Small Group

Consider a small Video Conference where the number of participants is less than twenty. When the encryption algorithm of SCGCM(BA) was compared with DES in Chapter 3 (Section 3.4), it was established that twenty SCGCM encryptions are equivalent to one DES encryption. Therefore, it is sufficient to merely use SCGCM(BA) for a small group with less than twenty members. The advantage is that there is no need to maintain the subgroup codes.

Since real-time voice and image transmissions are required for video conference, SCGCM(EX_II) should be used. With block cipher technology, the encryption and transmission work can be pipelined thereby improving the efficiency of the systems. According to the recommendation of RFC1889 [32], CBC mode in SCGCM should be used here.

(2). Large Group

When the number of participants is larger than twenty, it is more appropriate to use SCGCM(EX_I) algorithm.

According to the size of a subgroup, there are two cases:

- Case 1: if the number of members in a subgroup is less than 20. It is better to use the Option 2 of SCGCM(EX_I), i.e to delay the update of the subgroup DSCs. The price is that we have to maintain the individual DSCs as well.

- Case 2: if the number of members in a subgroup is more than 20. If the membership of the group does not change frequently, Option 1 of SCGCM(EX.I), i.e to update the subgroup code immediately when group membership changes, is obviously a better choice.

However, if the changes of group membership are frequent, the computation resources are abundant, and the stabilities of the systems are most important, Option 2 of SCGCM(EX.I) is worth considering.

4.2.2 Ad Hoc Collaboration

There are two kinds of ad hoc collaborations. One is non-prearranged collaboration between non-strangers where the participants may know or are able to discover each other (using PKI or PGP) before collaboration, but they cannot predict the exact time that they are going to collaborate. This kind of group collaboration is loose and informal such that a participant can join or leave the collaboration frequently. Another kind of ad hoc collaboration is where the participants of the group may be strangers, i.e. they do not have priori knowledge. The underlying networks for ad hoc collaboration may be pure ad hoc networks, mixed wired or wireless networks. We are going to study ad hoc group collaboration in the following two sections according to these two definitions.

Ad Hoc Collaboration between non-strangers

One example of this type of group collaboration is evident in disaster recovery situations (e.g earthquake, flood etc.) where network infrastructures have been damaged.

Suppose an ad hoc communication network is established between some of the group members where the constituent nodes may contain other intermediate nodes. It is necessary to prevent the unrelated parties from accessing group communication. Please note that under disaster recovery situations, it is normal that some participants are delayed in joining the networks (collaboration) while others leave the group suddenly.

SCGCM can be readily applied to this situation because it is fully distributed and very flexible such that group collaborations are hardly affected by any delayed participant or frequent joining/leaving members. Initially, group participants exchange their secure codes using the prior knowledge. It may take a relatively long time to initialize the collaboration (i.e. exchange their secure codes), however the overhead for later maintenance is small. Moreover, the collaboration between multiparty can be successfully carried out no matter how dynamic the situation can be. Either SCGCM(BA) or SCGCM(EX-I) (option 1 or option 2) is chosen depending on the number of participants and/or different applications.

Ad Hoc Collaboration between strangers

Collaboration between strangers often takes place in an ad hoc manner. Ad hoc conference is one such example where people attending a conference want to share some documents and facilities (e.g. printer, projector). The collaborating parties including printers etc. can be total strangers.

The challenge in this case is to identify a mechanism for establishing necessary prior knowledge. The *pre-authentication* techniques, such as “Imprint technology [34]” or “location-limited channel [6]”, or the key management service [46] are possible

means of establishing the prior knowledge. SCGCM(BA) or SCGCM(EX_I) can then be used depending on the type of collaborations. The guidelines for how to apply SCGCM are similar to what we have discussed previously.

Chapter 5

Conclusions

In this thesis, secure group communication issues especially in ad hoc networks, are examined. Apart from the security requirements of group communication, frequent changes in group membership require that group management scheme be efficient in order to support the highly dynamic situations. Group communication in ad hoc networks is even more challenge due to the characteristics (easy-to fail and node mobility) of the underlying ad hoc networks. Therefore, a novel Secure Code Group Communication Mechanism (SCGCM) has been developed to address these requirements. SCGCM is a simple and practical solution for secure group communication, especially in ad hoc networks. To my knowledge, it is the first general secure mechanism that is well suited to various group communication scenarios.

An important issue to be investigated is the security provided by SCGCM systems. This clearly depends on user and adversary capabilities. The SCGCM systems should be well-studied to determine which attacks are relevant. Existing attacks to CDMA systems can be good references for attack studies of SCGCM. Based on that, simulation or mathematical proof of the lower bound of computational effort required by the best known methods to defeat a system, is needed.

To aid the simulation or implementation in the future, we recommend a toolkit—Ensemble, which is developed at Cornell University and at the Hebrew University of Jerusalem [42]. Ensemble is a highly modular and re-configurable toolkit that provides a library of protocols that can be used for quickly building complex distributed applications.

Bibliography

- [1] A. Aggarwal, M. Kapoor, L. Ramachandran, and A. Sarkar, *Clustering algorithms for wireless ad hoc networks*, In Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (Boston, MA USA), August 2000, pp. 54–63.
- [2] Y. Amir, Y. Kim, C. Nita-Rotaru, J. Schultz, J. Stanton, and G. Tsudik, *Exploring Robustness in Group Key Agreement*, Technical Report CNDS 2000-4, Department of Computer Science, Johns Hopkins University, 2000, <http://citeseer.nj.nec.com/amir00exploring.html>.
- [3] N. Asokan and P. Ginzboorg, *Key Agreement in Ad-hoc Networks*, Computer Communications **23** (2000), no. 17, 1627–1637.
- [4] G. Ateniese, M. Steiner, and G. Tsudik, *New Multiparty Authentication Services and Key Agreement Protocols*, IEEE Journal of Selected Areas in Communications **18** (2000), no. 4, 628–639.
- [5] D. Balenson, D. McGraw, and A. Sherman, *Key Management for large Dynamic Groups: One-Way Function trees and Amortized Initialization*, IETF Internet draft (work in progress) (1999).
- [6] D. Balfanz, D.K. Smetters, P. Stewart, and H. C.Wong, *Talking to Strangers: Authentication in Ad-Hoc Wireless Networks*, Network and

- Distributed System Security Symposium 2002 (paper accepted), 2002, <http://www.isoc.org/isoc/conferences/ndss/02/proceedings/papers/balfan.pdf>.
- [7] M. Baugher, R. Canetti, and L. Dondeti, *Group Key Management Architecture*, IETF Internet draft (work in progress) (2001).
 - [8] D. Blankenbeckler, *An Introduction to Bluetooth*, 2001, <http://www.wirelessdevnet.com/channels/bluetooth/features/bluetooth.html>.
 - [9] R. Bruno, M. Conti, and E. Gregori, *WLAN Technologies for Mobile Ad Hoc Networks*, In Proceedings of the 34th Hawaii International Conference on System Sciences, 2001, <http://dlib2.computer.org/conferen/hicss/0981/pdf/09819003.pdf>.
 - [10] M. Burmester and Y. Desmedt, *A Secure and Efficient Conference Key Distribution System*, In Proceedings of Advances in Cryptology - EUROCRYPT'94, Lecture Notes in Computer Science, 1994, pp. 275–286.
 - [11] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Oinkas, *Multicast Security: A Taxonomy and Some Efficient Constructions*, In Proceedings of INFOCOM '99, vol. 2, 1999, pp. 708–716.
 - [12] Video Conference, <http://www.whatis.com/>.
 - [13] G. D. Crescenzo and O. Lornievskais, *Efficient Multicast Encryption Schemes*, Preprint (2002).
 - [14] Short description of the standard (802.11), 1999, <http://grouper.ieee.org/groups/802/11/main.html>.
 - [15] A. Fiat and M. Naor, *Broadcast Encryption*, In Proceedings of Advances in Cryptology-CRYPTO'93, vol. 773, Lecture Notes in Computer Science, 1994, pp. 480–491.

- [16] T. Hardjono, B. Cain, and I. Monga, *Intra-Domain Group Key Management Protocol*, IETF Internet draft (work in progress) (2000).
- [17] H. Harney and C. Muckenhirn, *GKMP Specification*, RFC 2093 (1997).
- [18] H. Harney and C. Muckenhirn, *Group Key Management Protocol (GKMP) Architecture*, RFC 2094 (1997).
- [19] D.B. Johnson and D.A. Maltz, *Dynamic Source Routing in Ad Hoc Wireless Networks*, Mobile Computing (T. Imielinski and H. Korth, eds.), Kluwer Academic Publishers, 1996, pp. 153–181.
- [20] M. Just, *Methods of Multi-Party Cryptographic Key Establishment*, Master's thesis, Carleton University, August 1994.
- [21] M. Just and S. Vaudenay, *Authenticated Multi-Party Key Agreement*, In Proceedings of Advances in Cryptology – ASIACRYPT '96, Lecture Notes in Computer Science, Springer-Verlag, 1996, pp. 36–49.
- [22] C. Law and Kai-Yeung Siu, *A Bluetooth Scatternet Formation Algorithm*, In Proceedings Global Telecommunications Conference 2001 GLOBECOM'01, vol. 5, November 2001, pp. 2864–2869.
- [23] S.J. Lee, W. Su, and M. Gerla, *On-Demand Multicast Routing Protocol*, In Proceedings of IEEE Wireless Communications and Networking Conference—WCNC'99 (1999), 1298–1302.
- [24] D.L. Lough, T.K. Blankenship, and K.J. Krizman, *A Short Tutorial on Wireless LANs and IEEE 802.11*, 1997, <http://computer.org/students/looking/summer97/ieee802.htm>.
- [25] A.J. Menezes, P. V. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996, <http://www.cacr.math.uwaterloo.ca/hac>.

- [26] M.J. Moyer, J. Rao, and P. Rohatgi, *A Survey of Security Issues in Multicast Communications*, IEEE Network **13** (1999), no. 6, 12–23.
- [27] Specification of the Bluetooth System (version 1.1), February 2001, <http://www.bluetooth.com>.
- [28] C.E. Perkins, E.M. Royer, and S.R. Das, *Ad hoc On-Demand Distance Vector (AODV) Routing*, IETF Internet draft (2001).
- [29] R.L. Pickholtz, D.L. Schilling, and L.B. Milstein, *Theory of Spread-Spectrum Communications—A Tutorial*, Transactions on Communications **30** (1982), no. 5, 855–884.
- [30] R. Roberts, *Introduction to Spread Spectrum*, <http://www.sss-mag.com/ss.html>.
- [31] T. Salonidis, Pravin Bhagwat, and Leandros Tassiulas, *Proximity awareness and fast connection establishment in Bluetooth*, Technical Report TR-2001-10, University of Maryland, College Park, MD, 2001.
- [32] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson and, *RTP: A Transport Protocol for Real-Time Applications*, RFC 1889 (1996).
- [33] Group Security, <http://www.irtf.org/charters/gsec.html>.
- [34] F. Stajano and R.J. Anderson, *The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks*, In Proceedings of the 7th Security Protocols Workshop, vol. 1796, Lecture Notes in Computer Science, Springer Verlag, 1999, pp. 172–194.
- [35] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 2nd ed., Prentice Hall, 1999.

- [36] D. Steer, L. Strawczynski, W. Diffie, and W. Wiener, *A Secure Audio Teleconference System*, In Proceedings of Advances in Cryptology - CRYPTO'88, Lecture Notes in Computer Science, Springer Verlag, September 1988, pp. 520–528.
- [37] M. Steiner, G. Tsudik, and M. Waidner, *Diffie-Hellman Key Distribution Extended to Groups*, In Proceedings of the 3rd ACM Conference on Computer and Communications Security, ACM Press, March 1996, pp. 31–37.
- [38] G.L. Stuber, *Principles of Mobile Communication*, ch. 9, pp. 456–513, Kluwer Academic Publishers, 1996.
- [39] G. Tan, A. Miu, J. Gutttag, and H. Balakrishnan, *Forming Scatternets from Bluetooth Personal Area Networks*, Technical Report MIT-LCS-TR-826, MIT Laboratory for Computer Science, 2001.
- [40] A. Tanenbaum, *Computer Networks*, 3rd ed., Prentice Hall, 1996.
- [41] G. Tsudik, *Research Issues in Reliable and Secure Wireless/Mobile Networks*, 1997, <http://www.isi.edu/workshop/public.html/wmcw97/tsudik.html>.
- [42] Cornell University and Hebrew University of Jerusalem, *The ensemble distributed communication system*, <http://www.cs.cornell.edu/Info/Projects/Ensemble/>.
- [43] D. Wallner, E. Harder, and R. Agee, *Key Management for Multicast: Issues and Architectures*, RFC 2627 (1999).
- [44] C. Wong, M. Gouda, and S. Lam, *Secure Group Communication Using Key Graphs*, In Proceedings of the ACM SIGCOMM'98, September 1998, pp. 68–79.
- [45] Workshop on “Mobile Ad Hoc Collaboration”, *CHI2002 Conference*, April 2002, <http://www.cs.uoregon.edu/research/wearable/chi2002ws>.
- [46] L. Zhou and Z. Haas, *Securing Ad Hoc Networks*, IEEE Network **13** (1999), no. 6, 24–30.

Index

- A Priori knowledge, 16
- A priori knowledge, 42
- A-GDH.2, *see* Authenticated Group Diffie-Hellman
- Authenticated Group Diffie-Hellman, 32–36, 75
- Backward access secrecy, 1, 26, 31, 57, 79
- Basic Service Set, 11
- BSS, *see* Basic Service Set
- CBC, *see* Cipher Block Chaining
- CEDSC, *see* Code Exchange Decryption Secure Code
- CEESC, *see* Code Exchange Encryption Secure Code
- CESC, *see* Code Exchange Secure Code
- Cipher Block Chaining, 68, 85
- Code Exchange Decryption Secure Code, 40
- Code Exchange Encryption Secure Code, 44
- Code Exchange Secure Code, 40
- Cryptoperiod, 17, 18, 38, 40, 86
- Data Decryption Secure Code, 45, 51
- Data Encryption Secure Code, 38, 44, 50
- Data Secure Code, 40, 60
- DDSC, *see* Data Decryption Secure Code
- DESC, *see* Data Encryption Secure Code
- DPG, *see* Dynamic Peer Groups
- DSC, *see* Data Secure Code
- Dynamic Peer Groups, 22, 32
- ECB, *see* Electronic CodeBook
- Electronic CodeBook, 68, 85
- ESS, *see* Extended Service Set
- Extended Service Set, 11
- Forward access secrecy, 1, 26, 31, 79
- Forward secrecy, *see* Perfect Forward Secrecy
- GKM, *see* Group Key Management
- Group Key Management, 1–3, 13–15, 17, 21, 42
- GSEC, *see* Group Security

- IBSS, *see* Independent Basic Service Set
- IETF, *see* Internet Engineering Task Force
- Independent Basic Service Set, 11
- Initialization Vector, 67
- Internet Engineering Task Force, 2
- Internet Research Task Force, 2
 - Group Security, 2, 4
- IRTF, *see* Internet Research Task Force
- IV, *see* Initialization Vector
- KEK, *see* Key-encrypting key
- Key authentication, 15
 - explicit key authentication, 16
 - implicit key authentication, 16, 22
- Key confirmation, 16
- Key Encryption Key, *see* Key-encrypting Key
- Key Graph, 22
 - Tree Key Graph, 22, 24–26, 28, 75
- Key-encrypting key, 17, 18, 42
- Known-key attack, 16, 17, 42
- Nonce, 19, 48, 67, 68
- OFT, *see* One-way Function Tree
- One-way Function Tree, 28, 30, 31, 75
- Perfect Forward Secrecy, 17, 55
- PFS, *see* Perfect Forward Secrecy
- PGP, *see* Pretty Good Privacy
- PKI, *see* Public Key Infrastructure
- Pre-authentication, 42
- Priori knowledge, 41
- SCGCM, *see* Secure Code Group Communication Mechanism
 - Basic Algorithm, 43, 57, 73, 78
 - encryption algorithm, 60
 - Extension Algorithm I, 58, 60, 63–65, 73, 78, 83
 - Extension Algorithm II, 66–68, 70, 85
- SCGCM(BA), *see* SCGCM Basic Algorithm
- SCGCM(EX.I), *see* SCGCM Extension Algorithm I
- SCGCM(EX.II), *see* SCGCM Extension Algorithm II
- Secure Code Group Communication Mechanism, 4, 5, 37, 40, 59, 73
- Secure Multicast Group, 2
- Simple Solution, 73
- SMuG, *see* Secure Multicast Group
- SS, *see* Simple Solution
- TEK, *see* Traffic-encrypting key
- TKG, *see* Tree Key Graph
- Traffic-encrypting key, 18