國立交通大學

資訊科學與工程研究所

碩 士 論 文

在無線隨意網路上基於可自我驗證簽名系統的安全的群組金鑰協議協定

A Secure Group Key Agreement Protocol with Self-Certified Signature in Mobile Ad Hoc Networks

研 究 生：曾智暘

指導教授：曾文貴　教授

中 華 民 國 九 十 八 年 六 月

在無線隨意網路上基於可自我驗證簽名系統的安全的群組金鑰協議協定
A Secure Group Key Agreement Protocol with Self-Certified Signature in Mobile Ad Hoc Networks

研 究 生：曾智暘　　Student：Chih-Yang Tseng

指導教授：曾文貴　　Advisor：Wen-Guey Tzeng

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十八年六月

Abstract

A group key agreement protocol allows participants to cooperatively establish a common session key which is used to encrypt or decrypt transmitted messages among them over an open network environment. In this paper we propose a secure group key agreement protocol based on the self-certified signature scheme which is suitable for the mobile ad hoc network environment. Under the DDH assumption, the proposed protocol is demonstrated to be secure against passive adversaries. Under the random oracle model and DL assumption, the proposed protocol is demonstrated to be secure against active adversaries. Besides, our protocol provides forward secrecy and withstands known-key attacks.

中文摘要

群組金鑰協議協定能讓一個群組中的參與者合作建立一個可在開放網路環境中加密或解密訊息的對話金鑰。在本論文中我們提出了一個基於可自我驗證簽名系統, 可適用於無線行動隨意網路上的安全的群組金鑰協議協定。在DDH 假設下, 此協定可抵抗被動的攻擊者。而在 random oracle 模型與 DL 假設下, 此協定可抵抗主動的攻擊者。此外, 我們設計的協定還提供了向前安全性以及能夠抵抗已知金鑰的攻擊。

誌 謝

在此感謝我的指導老師曾文貴教授, 在我碩士班的學習過程中帶領著我進入了密碼學與資訊安全的領域。老師不只在學業與研究上使我獲益良多, 更在生活以及言行舉止方面給予我許多鼓勵與指導。另外, 我要感謝口試委員, 交大資工蔡錫鈞教授、謝續平教授, 以及中研院資科所呂及人教授, 給予我許多論文上的建議與指導, 使得內容更臻完善。除此之外, 我要感謝實驗室的成康學長、煥宗學長以及孝盈學姊給予我的建議與指導, 實驗室同學宣佐、怡翔、翎恩、宏達、慈潔與學弟們在這段期間所給予的幫助以及精神上的支持與鼓勵。最後, 我要感謝我的家人, 在我碩士班的這段時間, 無論是在精神上或物質上都給我最大的支持, 讓我可以在無後顧之憂的情況下完成學業。在此, 謹以此文獻給所有該感謝的。

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recently, mobile computing and wireless communication technology have significant advances. Mobile devices have gained more powerful communication and computation capability and sufficient memory resources. A mobile ad hoc network (MANET) is formed with some mobile devices and this kind of networks differentiate themselves from traditionally existing networks by the fact that they rely on *no fixed infrastructures* [15]. Instead, nodes rely on each other to keep the network connected. Mobile nodes that are within each other's transmission range can communicate directly via wireless links, while those that are far apart rely on other nodes to relay messages as routers [24]. Because there is no fixed infrastructure in an ad hoc network, all network functions have to be performed by the nodes in it.

Each mobile node in a MANET can move unrestrictedly so that the topology of the network changes very rapidly. Node mobility and wireless connectivity allow nodes to spontaneously join and leave the network. This characteristic is called *dynamic topology*. Usually mobile nodes are small hand-held devices which cannot be locked up in a secure region and therefore they are easily compromised by either being lost or stolen. Due to this reason, MANETs cannot rely on any form of central authority to avoid a single point failure. A MANET which does not rely on any form of offline trusted third party (TTP) is *self-organized*; however, a distributed online TTP can exist [15].

Authentication is an important issue in MANETs. Any nodes in the network have to be authenticated so that no others can impersonate them. Without a robust authentication mechanism in place, the remaining security objectives, for example confidentiality, data integrity, and non-repudiation, cannot be achieved. However, it is not easy to authenticate each node without the help of a trusted authority [15]. In the public key infrastructure (PKI), each user contains a certificate which binds the public key to the user identity. Each user can obtain the other's public key by either querying its certificate or accessing to the certificate directory in the certification authority (CA). Then it checks the validity of the certificate and begins secure communication. In the identity-based (ID-based) infrastructure, the private key generation (PKG) authority is responsible for the generation of a user's private key. After the private key is generated, PKG have to send it to the corresponding user via a secure channel. The identity of each user is the public key and can be used for encryption or signature verification.

In 1991, Girault introduced self-certified public keys (SCK) [12] which can be regarded as intermediate between the PKI and the ID-based infrastructure. In SCK approach, a user chooses his private key, computes the corresponding public key, and sends the public key to the authority. The authority then computes certificate parameters for the user, which are computationally unforgeable given the public key and the identity of the user. A verifier can use the identity and the public keys to compute the certificate parameters. In SCK approach, the validity of a SCK is verified only after a successful communication which is called implicit authentication. Compared to the PKI, it can provide an implicit validation of public keys. Compared to identity-based framework, there is no key escrow problems and secure channels problems in SCK approach.

As stated above, a trusted authority is still necessary in authentication and the authority should not suffer from single point failure. An intuitive and practical approach is to distribute the trust to each node in the network to form an online distributed authority. The most commonly used is threshold cryptography. An (t, n)-threshold cryptography scheme allows n parties to share the ability to perform a cryptographic operation, so that any t parties can perform this operation jointly, whereas it is infeasible for at most $t - 1$ parties to do so, even by collusion [24].

Due to the characteristic of no fixed infrastructure and it is easy to capture any messages transmitted over a wireless network, it is usually necessary to establish a common group key in MANETs to secure group communication. A common group key can be used to encrypt all the communication in a MANET so that only nodes in the MANET can use the corresponding decryption key to decrypt the messages they receive. Besides, due to the characteristic of dynamic topology, *Join* and *Leave* procedures are needed to cope with join or leave of nodes. The establishment of a group key in MANETs is similar to that of a so-called conference key in traditional networks with infrastructure, however, a proper and secure authentication scheme should be considered.

There are two types of group key establishment protocols which allow participants to establish a common session key for encrypting their communication over an insecure network. The first type is group key distribution protocol in which a chairman selects a key and distributes it to the participants. The group key established in this type of protocols is pre-distributed and fixed, and compromise of the a priori secret of any participant breaches the security of all past group keys, thus failing to provide forward secrecy. The second type is group key agreement protocol in which a group key is computed by all participants without a chairman. The group key established in this type of protocols is dynamic and different for each session, and no one can predetermine the group key.

In this paper, we propose a secure group key agreement protocol in MANETs. In this protocol we use a self-certified signature scheme for entity authentication. Since the terms "group key" and "conference key" have the same purpose, we will use them alternately in the following sections.

1.1 Related Work

In 1976, Diffie and Hellman [10] proposed a key establishment protocol which enabled two participants to establish a common key using their own secret information and some other publicly exchanged information. After that, there have been efforts to extend two-party Diffie-Hellman key exchange to group setting [13, 8, 19]. Ingemaresson et al. [13] proposed the first conference key establishment protocol which is denoted *conference key distribution system*, CKDS. In this system, the nodes who want to participate in the conference are connected in a ring so that node i always sends messages to node $i+1$ and node $n-1$ sends messages to node 0. Figure 1.1 shows this CKDS setting. Based on this setting, the authors proposed a CKDS of order 2 in the beginning and then generalized it to order $j, 2 < j \leq n$. In their CKDS of order 2, the session key is a symmetric function of degree two which has the form $g^{\sum_{0 \leq i,j \leq n-1, i \neq j} r_i r_j}$, where $r_i, i \in [1, n]$, is a secret value chosen

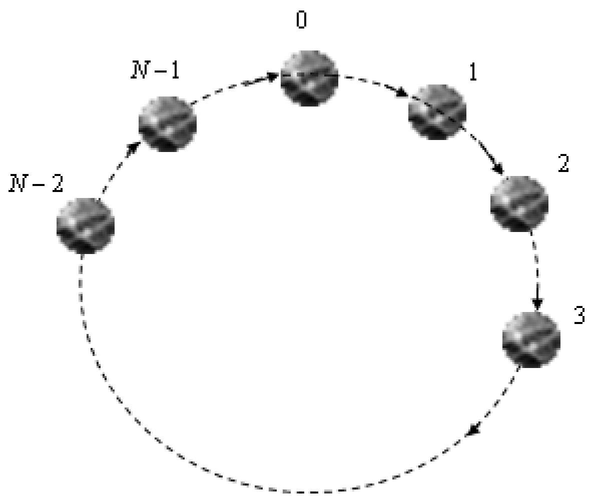

Figure 1.1: CKDS setup

randomly by node i. However, this particular system is insecure because the information exchanged by the nodes makes it possible for a passive adversary to compute the key. Besides, their system needs $n-1$ rounds for all participants to establish a group key and hence it is very inefficient for large scale participants.

In [19], Steiner et al. extended the Diffie-Hellman key exchange to n-party case naturally. These key agreement protocols are GDH.1, GDH.2, and GDH.3. The main feature of these protocols is that they are all asynchronous protocols. This means that any participants can send messages whenever they like. The main drawback of GDH.1 is its relatively large number of rounds. It requires $2(n-1)$ rounds for n participants. In order to reduce the number of rounds in GDH.1, the authors modified the protocol as GDH.2. An example of four participants is shown in Figure 1.2. The number marked on each edge stands for the order of the rounds. In GDH.2 it only requires n rounds to compute the common group key. However, in GDH.1

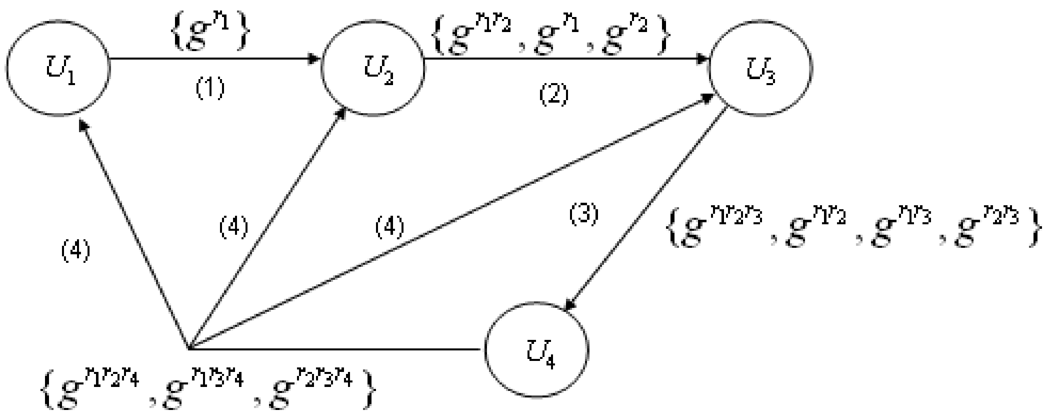
U_1
$\{g^{r_1}\}$
(1)
U_2
$\{g^{r_1r_2}, g^{r_1}, g^{r_2}\}$
(2)
U_3
(3)
$\{g^{r_1r_2r_3}, g^{r_1r_2}, g^{r_1r_3}, g^{r_2r_3}\}$
(4)
(4)
(4)
U_4
$\{g^{r_1r_2r_4}, g^{r_1r_3r_4}, g^{r_2r_3r_4}\}$

Figure 1.2: Message flow in GDH.2

and GDH.2 each participant U_i requires $i+1$ exponentiations, and the computational burden increases as the group size grows. In order to reduce the computational overhead of each participant, the authors proposed GDH.3. In GDH.3 it only requires constant exponentiations for all participants and has the same communicational efficiency as in GDH.2. The security of these three protocols is based on DH assumption.

Although [19] deals with node join and node leave operations, it does not consider entity authentication. So it is not secure against active adversaries. In [1], Ateniese et al. modified GDH.2 in [19] and proposed authenticated GDH.2 (A-GDH.2) and strong authenticated GDH.2 (SA-GDH.2). In their schemes, before the protocol execution each pair of participants has to establish a DH key which is used to provide entity authentication. In A-GDH.2, each participant U_i computes a DH key with U_n, thus each pair of (U_i, U_n) can authenticate each other. However, if U_n is not trusted, participants may not agree on the same group key. An example of four participants is shown in Figure 1.3. The number marked on each edge stands for the order of the

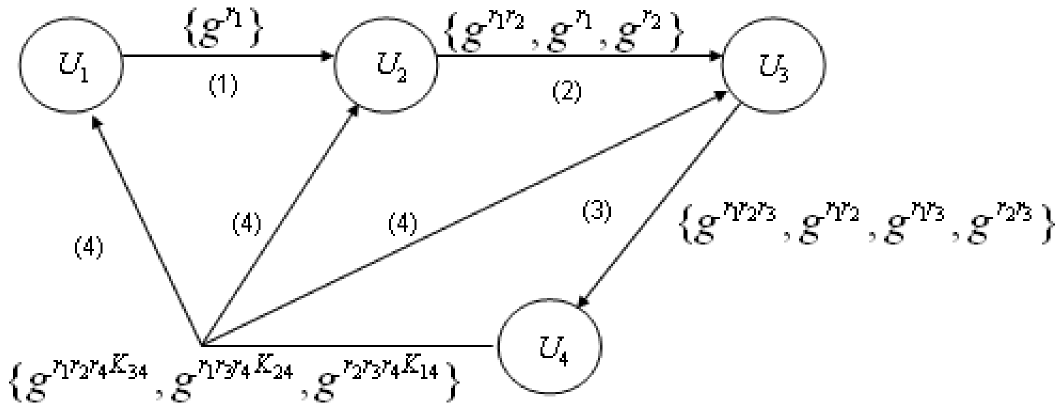
U_1
$\{g^{r_1}\}$
(1)
U_2
$\{g^{r_1 r_2}, g^{r_1}, g^{r_2}\}$
(2)
U_3
(3)
$\{g^{r_1 r_2 r_3}, g^{r_1 r_2}, g^{r_1 r_3}, g^{r_2 r_3}\}$
(4)
(4)
(4)
U_4
$\{g^{r_1 r_2 r_4 K_{34}}, g^{r_1 r_3 r_4 K_{24}}, g^{r_2 r_3 r_4 K_{14}}\}$

Figure 1.3: Message flow in A-GDH.2

rounds. Essentially Figure 1.3 is the same as Figure 1.2 except that the last round messages contain long-term keys K_{in} in the exponents. SA-GDH.2 makes an arbitrary pair of $(U_i, U_j), i \neq j$, authenticate each other. Both A-GDH.2 and SA-GDH.2 provide perfect forward secrecy (PFS) and resistance to passive known-key attacks. PFS is a property that ensures that compromise of a long-term key cannot result in the compromise of past session keys and it is important in an authenticated group key agreement protocol. The member join and member leave procedures are also provided in A-GDH.2 and SA-GDH.2, and they are similar to those in GDH.2 except that each message in the broadcast flow contains long-term keys in the exponent.

In [6], Bresson et al. developed the first formal security model for an authenticated conference key agreement protocol. They model instances of players via oracles available to the adversary through queries. The queries are available to use at any time to allow modeling attacks involving multiple instances of players activated concurrently and simultaneously by the adversary. The two modes of corruptions, which are weak-corruption model and strong-corruption model, are modeled. In the weak-corruption model, a corruption only reveals the long-lived key (LL-key) of player U. However in the strong-corruption model, a corruption not only reveals the LL-key of U but also all internal data that his instances did not explicitly erase. In order to model these two modes of corruption, they consider the presence of two cryptographic devices, secure coprocessors and smart cards, which are made available to the adversary through queries. Before their work, there had been little formal security analysis [9, 18]. They modified the protocols in [7] and [5] to obtain a protocol which is referred to as AKE1$^+$ secure against strong corruptions. Their security theorem does not need a random oracle assumption [3] and thus holds in the standard model.

The above protocols mentioned are generalized from the Diffie-Hellman key exchange protocol. Burmester and Desmedt [8] presented a notable result. They proposed several conference key agreement protocols based on various types of network connections. In their broadcast channel based scheme, the session key is a cyclic function (of the indices of the users) of degree two which has the form $g^{r_1r_2+r_2r_3+\cdots+r_nr_1}$, where $r_i \in [1, n]$ is a secret value chosen randomly by node i. Their unauthenticated conference key agreement protocol based on broadcast channel requires only two rounds and is very communicational efficient. Based on the Diffie-Hellman (DH) assumption, their protocols are proven secure against a passive adversary. For data origin authentication they use an interactive public key authentication scheme which is proven secure based on discrete logarithm (DL) assumption. Combining the interactive data origin authentication scheme with the unauthenticated conference key agreement protocol makes the conference key agreement protocol provably secure against malicious active adversaries based on the DH assumption. However, [13] and [8] did not provide member join and member leave procedures.

In [2], Becker and Wille derived a lower bound of only one round for multi-party contributory key agreement protocols. However, no generalized Diffie-Hellman key exchange is able to meet this bound. They left as an open problem whether any contributory key agreement scheme can meet this bound. There are some group key agreement protocols [4, 23] proposed after Becker and Wille, and they only require constant rounds.

In [23], the authors proposed two group key agreement protocols which are proved to have *fault-tolerance, correctness, fairness, authentication, and no useful information leakage* and require only one round. The main feature of [23] is that non-interactive proof systems (NIPVS) and non-interactive proof systems with authentication (NIAPVS) are used. In their proposed protocols, fault-tolerance and authentication can be achieved by using either NIPVS and digital signature or NIAPVS. Their protocols are secure against passive and active adversaries under the random oracle model. They also modified **Protocol 3** in [8] to obtain a protocol with fault-tolerance.

In [4], Boyd and Nieto pointed out that in [23] the session token (ST) is used to prevent replay attacks and unless the ST is agreed beforehand their protocols cannot be completed in one round. They proposed a protocol which requires only one round on the side. Their protocol is proved secure in Bellare and Rogaway's model [3]. However, both [23] and [4] do not provide forward secrecy.

In [22] the author proposed a conference key agreement protocol with both forward secrecy and fault tolerance. Before this work, [14] proposed a protocol with forward secrecy and [21] proposed a protocol with fault tolerance. However, [14] and [21] do not provide both forward secrecy and fault tolerance.

The remainder of this paper is organized as follows. Chapter 2 introduces some tools we use in our protocol. Chapter 3 describes the problem and the security requirements of a group key agreement protocol. Chapter 4 gives an overview and describes details of our secure group key agreement protocol. The security analysis and the performance analysis are discussed in Chapter 5 and Chapter 6. Finally, we give a conclusion and future works in Chapter 7.

Chapter 2

Preliminaries

In this chapter we introduce some tools used in our protocol.

2.1 (t, n)-threshold Secret Sharing Scheme

The goal of a secret sharing scheme is to share a secret s among n parties so that each one P_i keep a share s_i for $1 \leq i \leq n$. A coalition of some of the parties is able to recover s.

Shamir proposed a (t, n)-threshold secret sharing scheme in 1979 [16]. In their scheme a threshold t is set. Each t of the n shares are sufficient to recover s. It is impossible to do so from $t - 1$ or fewer shares. The scheme comprises two parts : *secret sharing* and *secret recovery.*

- Secret sharing :

  Given a secret $s \in \mathbb{Z}_q$, a threshold t, and a number n, randomly select a $(t - 1)$-degree polynomial

$$f(x) = \sum_{i=1}^{t-1} a_i x^i + s \pmod{q}$$

and compute

$$s_i = f(i) \pmod q$$

as a share of P_i for $1 \leq i \leq n$.

- Secret recovery :

  Given t shares (i, s_i), s can be recovered by

$$s = \sum_{i=1}^{t}(s_i \cdot \prod_{1 \leq j \leq t, j \neq i} \frac{j}{j-i}) \pmod q$$

  from the Lagrange's interpolation formula.

2.2 Bilinear Map

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of order q for some large prime q. Let $\hat{e}$ be a mapping $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. $\hat{e}$ satisfies :

1. Bilinear:

   $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$, $a, b \in \mathbb{Z}_q$.

2. Non-degenerate:

   $\hat{e}$ does not map all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$. Since $\mathbb{G}_1, \mathbb{G}_2$ are groups of prime order this implies that if P is a generator of $\mathbb{G}_1$ then $\hat{e}(P, P)$ is a generator of $\mathbb{G}_2$.

3. Computable:

   There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

2.3 Proof of Equal Logarithm

In our group key agreement protocol we use non-interactive proof of knowledge of the common logarithm r of $y_1 = g_1^r \in \mathbb{Z}_p^*$ and $y_2 = g_2^r \in \mathbb{Z}_p^*$. It is easy to convert an interactive proof into a non-interactive one. Thus, we first give the interactive version of the proof.

$ProofLogEq(g_1, y_1, g_2, y_2)$:

1. Prover chooses $s \in_R \mathbb{Z}_q$ and computes $a = (a_1, a_2) = (g_1^s, g_2^s)$. Prover sends a to Verifier.

2. Verifier chooses $c \in_R \mathbb{Z}_q$ and sends c to Prover.

3. Prover computes $b = s - cr$ and sends b to Verifier.

4. Verifier accepts if and only if $a_1 = g_1^b y_1^c$ and $a_2 = g_2^b y_2^c$.

- **Completeness**

  If Prover knows r and both Prover and Verifier are honest, then $a_1 = g_1^b y_1^c$ and $a_2 = g_2^b y_2^c$, and Verifier will accept.

- **Soundness**

  If any cheating prover can convince Verifier with a probability greater than $1/q$, it can find (a_1, a_2) such that two challenges c and c' can be answered. This results in computing r.

To obtain a non-interactive proof, we use the Fiat-Shamir heuristic [11]. Let $h : \{0,1\}^* \rightarrow \mathbb{Z}_q$ be a collision-resistant hash function. We replace the challenge $c \in \mathbb{Z}_q$ from Verifier with $c := h(g_1||y_1||g_2||y_2||a_1||a_2)$. So the non-interactive proof $ProofLogEq(g_1, y_1, g_2, y_2) = (c, b)$ is obtained. Verifier accepts if and only if $c = h(g_1||y_1||g_2||y_2||g_1^b y_1^c||g_2^b y_2^c)$.

Chapter 3

Background and Security Model

In this chapter, we describe the group key agreement problem and the security requirement of a group key agreement protocol. We also consider the possible adversaries who want to obtain the information of the group key and the assumption in our protocol.

3.1 Definition of the Problem

The goal of a group key agreement protocol is to establish a group key securely. Each participant cannot obtain others' secret values. Besides, each message transmitted has to be authenticated so that the participants can know who sends that message. Finally, each participant computes the same group key.

3.2 Assumption

Since we use (t, n)-threshold secret sharing scheme to share a master key s, any t participants can collude with each other to recover s. In this paper, we assume that at most $t-1$ participants can collude with each other.

3.3 Adversary Model

In this paper, we consider two types of adversaries : passive adversary and active adversary.

- **Passive Adversary**
  A passive adversary can obtain information about the established group key by only eavesdropping on messages transmitted over the network. This kind of adversaries is also called *eavesdropper*.
- **Active Adversary**
  Active adversaries can be classified into two kinds, one is *impersonator* and the other is *malicious participant*.
  - The purpose of an impersonator is to impersonate a legitimate participant. If the impersonator successfully impersonate a legitimate participant, the others cannot distinguish a legitimate message from a fake one.
  - A malicious participant is a member of the group but he does not follow the protocol. He can send some error messages to disrupt the key establishment.

We will show that our protocol can defeat passive and active adversaries.

3.4 Security Requirement

- **Correctness**

  Since the group key computed in a group key agreement protocol is determined by all of the participants, the protocol fails as long as anyone does not follow the protocol steps. So it is necessary to detect those who does not follow the protocol steps. After excluding those malicious participants, a common group key can be evaluated by each remaining honest participant.

- **Privacy**

  Any eavesdroppers and malicious participants cannot obtain information about the group key by observing messages transmitted over the network. Besides, the information about the private values of each participant cannot leak out.

- **Authentication**

  To defeat an impersonator, authentication mechanism is necessary. Although an impersonator can use messages transmitted by legitimate participants to produce new legitimate messages, he still cannot impersonate a legitimate participant if he cannot convince others.

- **Robustness**

  To prevent a malicious participant, robustness is necessary. A malicious participant has to be excluded so that he cannot disrupt the key agreement protocol. After excluding those malicious participants, the common group key can be evaluated by each remaining honest participant.

3.5 Notations

We summarize the notations used in our protocol in Table 3.1.

Table 3.1: Notations

Notation	Description
U_i	group participant with index i
q	large prime number
$\mathbb{G}_1$, $\mathbb{G}_2$	bilinear group of order q
$\hat{e}$	bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$
P	generator of $\mathbb{G}_1$
H	hash function, $H : \mathbb{G}_1 \times \{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{G}_1$
F	hash function, $F : \{0,1\}^* \times \mathbb{G}_2 \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$
s	master key of PKG, $s \in_R \mathbb{Z}_q$
P_{PKG}	public key of PKG, $P_{PKG} = sP$
$Params$	system parameters, $Params = \{q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, H, F\}$
ID_i	identity of U_i, $ID_i \in \{0,1\}^*$
x_i	private key of U_i, $x_i \in \mathbb{Z}_q$
Y_i	public key of U_i, $Y_i = x_i P$
d_i	private key of U_i computed by PKG, $d_i = sH(P_{PKG}, ID_i, Y_i) = sH_i$
Sig_M	signature of message M
IG	BDH parameter generator, $\{q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P\} \leftarrow IG(1^k)$

Chapter 4

Our Scheme

In this chapter, we describe our group key agreement protocol in MANET. We first give an overview and then describe details of the protocol.

4.1 Overview

In this paper we propose a group key agreement protocol which is based on the self-certified signature scheme in [17]. We modify the scheme to be distributed and hence it is suitable to MANETs.

The purpose of the signature scheme is to provide authentication of participants. In the beginning, there is a private key generator (*PKG*). The *PKG* computes and sends the shares of the master key to the initial participants via secure channels and each one has to register a pair of public/private key which is used for signature. When a participant want to join the group for the first time, the participants in the group have to cooperatively compute a share of the master key and a pair of public/private key for him. When a participant want to leave the group, the others have to perform key evolving procedure to refresh their shares of the master key. We note that a share of the master key has to be computed for a join participant every time.

The group key agreement scheme used in our protocol is modified from [22]. We combine the group key agreement scheme with the self-certified signature scheme to obtain our group key agreement protocol in MANETs.

4.2 Self-Certified Signature Scheme

The self-certified signature scheme is composed of six phases which are *Key Generation*, *Registration*, *Signature*, *Verification*, *Join*, and *Leave*.

4.2.1 Key Generation Phase

In this phase the system parameters are generated. The PKG selects a private number as a master key and computes the corresponding public key. Each initial participant selects a private value and computes the corresponding public value. The phase is shown as follows:

Key Generation:

1. $\{q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P\} \leftarrow IG(1^k)$.
2. Choose two hash functions $H : \mathbb{G}_1 \times \{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{G}_1$ and $F : \{0,1\}^* \times \mathbb{G}_2 \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$. The system parameters are $Params = \{q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, H, F\}$.
3. PKG selects a master key $s \in_R \mathbb{Z}_q^*$ and sets $P_{PKG} = sP$ as its public key.
4. Each initial participant U_i with identity $ID_i \in \{0,1\}^*$ selects $x_i \in_R \mathbb{Z}_q^*$ as its private key and sets $Y_i = x_iP$ as public. The public key of the participant is

$$\langle P_{PKG}, ID_i, Y_i, Params \rangle.$$

4.2.2 Registration Phase

In this phase each participant registers to the PKG for his private key and a share of the master key. Here we use verifiable threshold secret sharing scheme to share the master key. The public key of each participant can be used to establish a secret channel to send the share securely. When a message is encrypted by the public key of a particular participant, only those who know the corresponding private key can decrypt it. The phase is shown as follows:

Registration:

1. Each initial participant U_i authenticates himself to PKG and then sends (ID_i, Y_i) to PKG.
2. PKG randomly selects a degree-$(t-1)$ polynomial $f(x) = \sum_{k=1}^{t-1} a_k x^k + s$ and computes the share s_i of s for the participant
$$s_i = f(i)$$
3. PKG computes $H_i = H(P_{PKG}, ID_i, Y_i)$ and sets the private value $d_i = sH_i$. Then PKG selects a number $r \in_R \mathbb{Z}_q^*$ and computes $U = rP, V = d_i + rY_i, W = s_i + rY_i$, and $a_1P, a_2P, \ldots, a_{t-1}P$. Finally PKG sends $(U, V, W, a_1P, a_2P, \ldots, a_{t-1}P)$ to the participant publicly.
4. The participant recovers $d_i = V - x_iU$ and $s_i = W - x_iU$. Then he verifies d_i and s_i by checking whether the following equations

hold

$$\begin{aligned} H_i &= H(P_{PKG}, ID_i, Y_i) \\ \hat{e}(d_i, P) &= \hat{e}(H_i, P_{PKG}) \\ s_i &= P\sum_{k=1}^{t-1}(a_k P)i^k + P_{PKG}. \end{aligned}$$

If the checking passes, the participant obtains his private key $\langle x_i, d_i \rangle$ and the share s_i of the master key s.

4.2.3 Signature Phase

When a participant sends a message out, he has to compute the signature of the message. The signature is computed as follows:

Sign:

1. To sign a message M, the signer U_i selects a random number $k \in \mathbb{Z}_q^*$ and computes

$$\begin{aligned} H_i &= H(P_{PKG}, ID_i, Y_i) \\ r &= \hat{e}(kH_i, P) \\ f_{i,M} &= F(M, r, H_i) \\ V_{i,M} &= kH_i + f_{i,M}x_iH_i + f_{i,M}^2 d_i. \end{aligned}$$

$Sig_M = (f_{i,M}, V_{i,M})$ is a signature of M.

4.2.4 Verification Phase

When a participant receives a signature of a message, he has to verify the signature. The verification is as follows:

Verify:

1. The verifier computes

$$H_i = H(P_{PKG}, ID_i, Y_i)$$
$$r' = \hat{e}(V_{i,M}, P)\hat{e}(-H_i, f_{i,M}Y_i + f_{i,M}^2 P_{PKG}).$$

   then checks the equation

$$f_{i,M} = F(M, r', H_i).$$

4.2.5 Join Phase

When a new member wants to join, t participants are randomly chosen to compute jointly the share of the master key and a pair of public/private key for him. The process is as follows:

Join: Assume new member U_{n+1} wants to join.

1. U_{n+1} selects $x_{n+1} \in \mathbb{Z}_q^*$ as its private value , then broadcasts $Y_{n+1} = x_{n+1}P$ and ID_{n+1}.
2. Randomly select t participants $\{U_1, U_2, \ldots, U_t\}$. Each U_l randomly selects a degree-$(t-1)$ polynomial $h_l(x) = \sum_{k=1}^{t-1} a_{l,k}x^k + s_l(n+1)$, where $s_l(n+1) = f(l) \cdot \prod_{1 \le k \ne l \le t} \frac{(n+1)-k}{l-k}$. Then U_l selects a number $r_l \in_R \mathbb{Z}_q^*$ and sends $Z_{lj} = h_l(j) + r_l Y_j$ and $r_l P$ to U_j.
3. When U_l receives Z_{jl} and $r_j P$ from the other $t-1$ participants, he recovers $h_j(l) = Z_{jl} - x_l(r_j P)$ and computes $F(l) = \sum_{k=1}^{t} h_k(l)$. Then he sends $F(l)$, $s_l P$, and $s_l H_{n+1}$ to U_{n+1}.

4. U_{n+1} computes

$$f(n+1) = \sum_{k=1}^{t} F(k) \cdot \prod_{1 \le k \ne j \le t} \frac{j}{j-k}$$

$$P_{PKG} = \sum_{k=1}^{t} s_k P \cdot \prod_{1 \le k \ne j \le t} \frac{j}{j-k}$$

$$d_{n+1} = \sum_{k=1}^{t} s_k H_{n+1} \cdot \prod_{1 \le k \ne j \le t} \frac{j}{j-k}$$

In this process, each participant U_l in the list randomly selects a degree-$(t-1)$ polynomial with the constant $s_l(n+1) = f(l) \cdot \prod_{1 \le k \ne l \le t} \frac{(n+1)-k}{l-k}$. Since $f(n+1) = \sum_{j=1}^{t} f(j) \cdot \prod_{1 \le k \ne j \le t} \frac{(n+1)-k}{j-k}$, we only need to recover the constant of $F(l)$ which is $\sum_{k=1}^{t} F(k) \cdot \prod_{1 \le k \ne j \le t} \frac{j}{j-k}$. So $f(n+1)$ can be computed correctly.

4.2.6 Leave Phase

When a participant wants to leave, the others have to refresh their shares of the master key. The process is as follows:

Leave:

1. Each participant U_l who still in the network randomly selects a degree-$(t-1)$ polynomial $r_l(x)$ with constant 0 and broadcasts $r_l(j), 1 \le j \le n$.
2. When U_l receives $r_j(l)$ from the other participants, he computes $f'(l) = f(l) + \sum_{j=1}^{n} r_j(l)$ which is his new share of the master key s.

In this process, each remaining participant U_l randomly selects a degree-$(t-1)$ polynomial $r_l(x)$ with constant 0. Since $f'(x) = f(x) + \sum_{j=1}^{n} r_j(x)$, U_l computes the correct new share $f'(l)$ of the master key.

4.3 Group Key Agreement

Now we describe the group key agreement protocol. The protocol consists of three phases which are *Initialization*, *Join*, and *Leave*. In these three phases, signatures are computed for broadcast messages each time. For simplicity, we do not write out signatures.

4.3.1 Initialization Phase

In this phase the initial group key is established. The phase is shown as follows:

Initialization:

1. Each U_i selects $r_i \in_R \mathbb{Z}_q^*$ and broadcasts $w_i = g^{r_i} \bmod p$.
2. When U_i receives w_{i-1}, w_{i+1}, it computes

$$\begin{aligned} b_i =& w_{i+1}/w_{i-1} \bmod p \\ z_i =& b_i^{r_i} \bmod p \\ c_i =& h(g||w_i||b_i||z_i||g^{s_i}||b_i^{s_i}) \\ a_i =& s_i - c_i r_i \bmod q \end{aligned}$$

   in which $s_i \in_R \mathbb{Z}_q^*$. Then broadcast z_i, a_i, b_i, and c_i.
3. When U_i receives z_j, a_j, b_j, and c_j from the others, it checks the following equations

$$\begin{aligned} b_j =& w_{j+1}/w_{j-1} \bmod p \\ c_j =& h(g||w_j||b_j||z_j||g^{a_j} \cdot w_j^{c_j}||b_j^{a_j} \cdot z_j^{c_j}). \end{aligned}$$

If the checking passes, U_i can use them to compute the group session key

$$\begin{aligned} K_i =& w_{i-1}^{nr_i} z_i^{n-1} z_{i+1}^{n-2} \dots z_{i-2} \bmod p \\ =& g^{r_1 r_2 + r_2 r_3 + \dots + r_n r_1} \bmod p. \end{aligned}$$

It is notable that we use the non-interactive proof system to prove that U_i knows r_i and the exponent of z_i is exactly r_i. This is necessary so that no one can forge z_i based on the Diffie-Hellman (DH) assumption.

4.3.2 Join Phase

When a new member wants to join, a new group key is established. However, the old group key cannot be known by the new member. The phase is shown as follows:

Join: Assume U_{n+1} wants to join.

1. U_{n+1} selects $r_{n+1} \in_R \mathbb{Z}_q^*$ and broadcasts $w_{n+1} = g^{r_{n+1}} \bmod p$.
2. • U_1 selects $r'_1 \in_R \mathbb{Z}_q^*$ and computes

$$\begin{aligned} w'_1 =& g^{r'_1} \bmod p \\ b =& w_{n+1} w_2 \\ K^* =& K \cdot b^{r'_1} \cdot w_n^{-r_1} \bmod p \\ c_1 =& h(g || w'_1 || b || b^{r'_1} || g^s || b^s) \\ a_1 =& s - c_1 r'_1 \bmod q \\ c_2 =& h(g || w_1 || w_n^{-1} || (w_n^{-1})^{r_1} || g^s || (w_n^{-1})^s) \\ a_2 =& s - c_2 r_1 \bmod q, \end{aligned}$$

in which $s \in_R \mathbb{Z}_q^*$. Then broadcast w'_1, K^*, $E_K(b^{r'_1})$, $E_K(w_n^{-r_1})$, c_1, a_1, c_2, a_2.

- U_i, $2 \leq i \leq n$, broadcast $w_i = g^{r_i} \bmod p$.

3. When each participant receives $E_K(b^{r'_1})$ and $E_K(w_n^{-r_1})$ from U_1, it decrypts and gets $b^{r'_1}$ and $w_n^{-r_1}$. Then it checks the following equations

$$\begin{aligned} c_1 =& h(g||w'_1||b||b^{r'_1}||g^{a_1} \cdot (w'_1)^{c_1}||b^{a_1} \cdot (b^{r'_1})^{c_1}) \\ c_2 =& h(g||w_1||w_n^{-1}||(w_n^{-1})^{r_1}||g^{a_2} \cdot w_1^{c_2}||(w_n^{-1})^{a_2} \cdot (w_n^{-r_1})^{c_2}) \\ K^* =& K \cdot b_1^{r'_1} \cdot w_n^{-r_1} \bmod p, \end{aligned}$$

   then checks whether w_i, $2 \leq i \leq n$, are the same as those received previously. If the checking passes, then each node updates w_1 by $g^{r_1+r'_1} = w_1 w'_1 \bmod p$.

4. U_n computes $k = w_{n+1}^{r_n} \bmod p$ and broadcasts k.

5. When U_{n+1} receives k, it checks whether $k = w_n^{r_{n+1}} \bmod p$.

6. Each participant computes the new session key

$$K' = K^* \cdot k \bmod p.$$

It is notable that $b^{r'_1}$ and $w_n^{-r_1}$ in step 2 are broadcast in encrypted form so that U_{n+1} cannot compute K from K^* by $K = K^* \cdot (b^{r'_1})^{-1} \cdot (w_n^{-r_1})^{-1} \bmod p$.

4.3.3 Leave Phase

Leave: Assume U_l wants to leave and the remaining participants form a group $G' = G - \{U_l\}$ in which all participants are re-indexed from 1 to $n' = n - 1$.

1. Each participant in $\{U_i\}$, $1 \leq i \leq n$ and $i \bmod 2 = 1$, selects $r_i \in_R \mathbb{Z}_q^*$ and broadcasts $w_i = g^{r_i} \bmod p$.

2. Each participant in G' computes

$$\begin{aligned} b_i =& w_{i+1}/w_{i-1} \bmod p \\ z_i =& b_i^{r_i} \bmod p \\ c_i =& h(g||w_i||b_i||z_i||g^{s_i}||b_i^{s_i}) \\ a_i =& s_i - c_i r_i \bmod q \end{aligned}$$

   in which $s_i \in_R \mathbb{Z}_q^*$. Then broadcast z_i, a_i, b_i, and c_i.

3. When U_i receives b_j, z_j, a_j, and c_j from the others, it checks the following equations

$$\begin{aligned} b_j =& w_{j+1}/w_{j-1} \bmod p \\ c_j =& h(g||w_j||b_j||z_j||g^{a_j} \cdot w_j^{c_j}||b_j^{a_j} \cdot z_j^{c_j}). \end{aligned}$$

   If the checking passes, U_i can use them to compute the session key

$$\begin{aligned} K_i' =& w_{i-1}^{n' r_i} z_i^{n'-1} z_{i+1}^{n'-2} \ldots z_{i-2} \bmod p \\ =& g^{r_1 r_2 + r_2 r_3 + \cdots + r_{n'} r_1} \bmod p. \end{aligned}$$

Chapter 5

Security Analysis

In this chapter, we discuss the security of the proposed group key agreement protocol.

The signature scheme had been proved existential unforgeability against adaptive chosen message attacks (EUF-CMA) in the random oracle model under the difficulty of the computational Diffie-Hellman (CDH) problem by using the Forking Lemma [17]. It is proved secure against Type 1 and Type 2 adversaries. A Type 1 adversary is an uncertified user who wants to impersonate a victim by using public keys of its choice along with the identity of the victim. However, a Type 1 adversary cannot query the private key of the challenge public key from the *PKG*. A Type 2 adversary is a malicious *PKG* which wants to impersonate a victim with a given public key. However, a Type 2 adversary cannot access the corresponding private key chosen by the victim. Since in our scheme the *PKG* is distributed to the initial participants and we assume that at most $(t-1)$ participants can collude with each other, we do not consider Type 2 adversaries.

5.1 Security against Passive Adversary

We show that an eavesdropper cannot efficiently distinguish a group key constructed from our protocol and a random value under the decisional Diffie-Hellman (DDH) assumption.

Theorem 1. *Under the decisional Diffie-Hellman assumption,* $(w_i, z_i, K = g^{r_1r_2+r_2r_3+\cdots+r_nr_1}$ *mod* $p)$ *and* $(w_i, z_i, R^n/(z_1^n \prod_{j=2}^{n} z_j^{j-1}))$*, for* $1 \le i \le n$*, are computationally indistinguishable, where* R *is a random value in* $\mathbb{Z}_p^*$*.*

Proof. We prove this theorem by contradiction. Assume that there exists an algorithm A that can efficiently distinguish (w_i, z_i, K) and $(w_i, z_i, R^n/(z_1^n \prod_{j=2}^{n} z_j^{j-1}))$, for $1 \le i \le n$. We can use algorithm A to construct another algorithm A' that can efficiently distinguish $(y_1, y_2, g^{x_1x_2} \bmod p)$ and (y_1, y_2, R), where $y_1 = g^{x_1} \bmod p$, $y_2 = g^{x_2} \bmod p$, and $x_1, x_2 \in \mathbb{Z}_q^*$. The values y_1, y_2, and R are the input of algorithm A'. We let $w_1 = y_1$, $w_2 = y_2$ without loss of generality. Then algoritm A' computes the following values :

$$\begin{aligned} w_3 &= w_1 \cdot g^{s_1} \bmod p \\ w_4 &= w_2 \cdot g^{s_2} \bmod p \\ &\cdot \\ &\cdot \\ &\cdot \\ w_{n-1} &= w_{n-3} \cdot g^{s_{n-3}} \bmod p \\ w_n &= w_{n-2} \cdot g^{s_{n-2}} \bmod p \end{aligned}$$

where $s_i \in \mathbb{Z}_q^*$ is a random value, for $1 \le i \le n-2$. Let $w_i = g^{r_i} \bmod p$, for $1 \le i \le n$. A' can compute $z_2 = (w_3/w_1)^{r_2} \bmod p = y_2^{s_1} \bmod p$ and $z_i = (w_{i+1}/w_{i-1})^{r_i} \bmod p = w_i^{s_{i-1}} \bmod p$ for $3 \le i \le n-1$.

If n is even, then

$$\begin{aligned} z_1 =& (w_2/w_n)^{r_1} \bmod p = y_1^{-(t_2+t_4+\cdots+t_{n-2})} \bmod p \\ z_n =& (w_1/w_{n-1})^{r_n} \bmod p = w_n^{-(t_1+t_3+\cdots+t_{n-3})} \bmod p. \end{aligned}$$

If n is odd, then

$$\begin{aligned} z_1 =& (w_2/w_n)^{r_1} \bmod p = (y_2/y_1)y_1^{-(t_1+t_3+\cdots+t_{n-2})} \bmod p \\ z_n =& (w_1/w_{n-1})^{r_n} \bmod p = (y_1/y_2)w_n^{-(t_2+t_4+\cdots+t_{n-3})} \bmod p. \end{aligned}$$

So if algorithm A' has constructed w_i, z_i for $1 \le i \le n$, and computes $R^n/(z_1^n \prod_{j=2}^n z_j^{j-1})$, A' can call A with these values. If A can efficiently distinguish (w_i, z_i, K) and $(w_i, z_i, R^n/(z_1^n \prod_{j=2}^n z_j^{j-1}))$ for $1 \le i \le n$, then A' can use A to efficiently distinguish $(y_1, y_2, g^{x_1x_2} \bmod p)$ and (y_1, y_2, R), which is a contradiction for the DDH assumption. $\square$

In the *Leave* phase, U_l can still receive the broadcast messages from the group G' after it leaves. All it can do is to use the original group key K and the broadcast messages from G' to attempt to compute the new group key $K' = g^{r'_1r_2+r_2r'_3+\cdots+r_nr'_1} \bmod p$. However, U_l cannot know any information about the new group key K' under the CDH assumption. So the group key agreement protocol is secure against a passive adversary.

5.2 Security against Active adversary

In the registration phase, the *PKG* computes the partial private key $d_i = sH_i$ for U_i. We notice that H_i can be computed by any participants, however, an adversary cannot recover s and obtain d_i under the DL assumption for elliptic curves. In the join phase, any t participants are chosen to collaboratively compute the share of the master key and a pair of public/private key for a new member. By our assumption that at most $t-1$ participants can collude with each other, an adversary cannot obtain d_i from these t participants as well.

A participant joins the group successfully means that his identity has been authenticated by the others. Since the adversary cannot obtain d_i, he cannot impersonate U_i.

To demonstrate that the group key agreement protocol is secure against an active adversary, we have to argue two things :

1. w_i is computed and broadcast by the legal participant U_i.
2. z_i is the form of $(w_{i+1}/w_{w-1})^{r_i}$ and is computed by the participant who knows the secret value r_i.

Theorem 2. *Under the random oracle model and the discrete logarithm (DL) assumption, any malicious adversary A with d_i cannot compute the valid w_i of any participant U_i.*

Proof. Assume that there exists a malicious adversary A which can impersonate U_i to sign w_i with a non-negligible probability ε. The hash function F is a truly random function under the random oracle model, that is, $F(w_i, r, H_i)$ is independent of w_i, r, and H_i, where $H_i = H(P_{PKG}, ID_i, Y_i)$, $r = \hat{e}(kH_i, P)$, and $k \in \mathbb{Z}_q^*$ is a random number selected by U_i. A can compute two valid signatures (f_{i,w_i}, V_{i,w_i}) and (f'_{i,w_i}, V'_{i,w_i}) of w_i with a non-negligible probability, where f_{i,w_i} and f'_{i,w_i} are two hash values of $F(w_i, r, H_i)$ under the random oracle model, $V_{i,w_i} = kH_i + f_{i,w_i}x_iH_i + f_{i,w_i}^2 d_i$, and $V'_{i,w_i} = kH_i + f'_{i,w_i}x_iH_i + (f'_{i,w_i})^2 d_i$. A can compute x_i by solving the two equations

$$\begin{aligned} V_{i,w_i} =& kH_i + f_{i,w_i}x_iH_i + f_{i,w_i}^2 d_i \\ V'_{i,w_i} =& kH_i + f'_{i,w_i}x_iH_i + (f'_{i,w_i})^2 d_i \end{aligned}$$

and obtain

$$x_i = \frac{(V'_{i,w_i} - V_{i,w_i}) - ((f'_{i,w_i})^2 - f^2_{i,w_i})d_i}{(f'_{i,w_i} - f_{i,w_i})H_i},$$

which is a contradiction. □

Theorem 3. *Under the random oracle model and the discrete logarithm (DL) assumption for elliptic curves, any malicious adversary A without knowing r_i or with a fake r_i cannot generate the right form of $(w_{i+1}/w_{i-1})^{r_i}$.*

Proof. Assume that there exists a malicious adversary A which can impersonate U_i to generate c'_i and a'_i used to verify X_i with a non-negligible probability ε. Under the random oracle model, A can generate two valid tuples (z_i, c_i, a_i) and (z_i, c'_i, a'_i) such that

$$g^{a_i} = g^{s_i} - w_i^{c_i}$$
$$g^{a'_i} = g^{s_i} - w_i^{c'_i}$$

and

$$b_i^{a_i} = b_i^{s_i}/z_i^{c_i}$$
$$b_i^{a'_i} = b_i^{s_i}/z_i^{c'_i}.$$

Then

$$r_i = \log_P w_i = \log_{b_i} z_i = \frac{a'_i - a_i}{c_i - c'_i},$$

which is a contradiction. □

By Theorem 2, only the legal participant U_i with the secret x_i can compute the valid w_i. Since an impersonator does not know x_i, it cannot compute the valid w_i. So the key agreement protocol is secure against an impersonator. Even the malicious impersonator can be detected in the verification step of *Initialization*, *Join*, and *Leave* phase. By Theorem 3, z_i is of the form $(w_{i+1}/w_{i-1})^{r_i}$ and is computed by the legal participant who know r_i. Therefore, the key agreement protocol is secure against an active adversary.

5.3 Other Security Properties

Our proposed protocol also provides forward secrecy and withstand known-key attacks. We prove these two properties in the following theorems.

Theorem 4. *Our protocol provides forward secrecy under the DL assumption.*

Proof. Let $U = \{U_i\}_{i=1}^{N}$ be the set of participants who have established a group key K at some past time τ. Suppose that the secret key (x_i, d_i) of some participant $U_i \in U$ is compromised by an adversary A at time $\tau + 1$. Since (x_i, d_i) is only used to signature of message and the computation of K is only related to $w_i = g^{r_i}$ and $Sig_{w_i} = (f_{i,w_i}, V_{i,w_i})$, learning r_i from (x_i, d_i) is impossible. Also, learning r_i from w_i and Sig_{w_i} is infeasible under the DL assumption. Since r_i is randomly chosen by U_i, it is obvious that A with (x_i, d_i) cannot learn K at time τ. Therefore, our protocol provides forward secrecy under the DL assuption. □

Theorem 5. *Our protocol is secure against known-key attacks under the random oracle model.*

Proof. Suppose that an adversary knows the group key K and attempts to learn other established group keys. We note that $K = g^{r_1r_2+r_2r_3+\cdots+r_nr_1}$ does not contain any information about (x_i, d_i). Since each r_i is chosen randomly by U_i at each time τ, all established group key are independent of each other. So the adversary cannot learn other group key and cannot impersonate any participant at time $\tau+1$. Therefore, our protocol is secure against known-key attacks under the random oracle model. $\square$

Chapter 6

Performance Analysis

In this chapter we discuss the performance of each phase in our proposed protocol.

6.1 Initialization

In this phase we only need 2 rounds to establish a common group key when no malicious participants are detected. If any malicious participants U_j are detected in step 3, U_{j-1} and U_{j+1} have to re-compute the broadcast messages in step 2. So one additional round is required.

Each participant U_i requires 3 exponentiations in step 2. In step 3 each participant U_i requires 4 exponentiations to verify messages received, and n exponentiations to compute the group key. So there are total $5n$ exponentiations.

w_i is broadcast in step 1 and z_i, a_i, b_i, and c_i are broadcast in step 2. Therefore, the message size is $3|p| + 2|q|$.

6.2 Join

In this phase each participant including U_{n+1} have to use K^* and k which are broadcast by U_1 and U_n respectively to compute a new group key. We notice that both K^* and k are computed after receiving w_{n+1}. Since the messages from U_1 and the messages from U_n can be broadcast simultaneously, it requires at least 2 rounds to establish the new group key.

Each participant requires 8 exponentiations in step 3 to check whether $c_1 = h(g||w'_1||b||b^{r'_1}||g^{a_1} \cdot (w'_1)^{c_1}||b^{a_1} \cdot (b^{r'_1})^{c_1})$ and $c_2 = h(g||w_1||w_n^{-1}||(w_n^{-1})^{r_1}||g^{a_2} \cdot w_1^{c_2}||(w_n^{-1})^{a_2} \cdot (w_n^{-r_1})^{c_2})$. U_1 requires 7 exponentions in step 2. U_n requires 2 exponentions.

For each U_i, $1 \leq i \leq n+1$, the commitment w_i is broadcast. In addition, U_1 broadcasts K^*, $E_K(b^{r'_1})$, $E_K(w_n^{-r_1})$, c_1, c_2, a_1, and a_2, and U_n broadcast k.

6.3 Leave

The round number, computation overhead, and communication overhead are the same as in **Initialization** phase except that $U_{i, 1 \leq i \leq n, i \bmod 2 = 0}$ need not broadcast w_i again.

Despite message signature, we compare our protocol with [22] in Table 6.1.

Table 6.1: Performance comparison with [22]

Protocol		Round	Exponentiation	Messages
Protocol[22]	Join	2	4	$4\|p\| + \|q\|$
	Leave	2	4	$4\|p\| + \|q\|$
Our	Join	2	$U_1 : 6$ $U_n, U_{n+1} : 2$ others : 1	$U_1 : 4\|p\| + 4\|q\|$ $U_n : 2\|p\|$ others : $\|p\|$
	Leave	2	$U_{i,i}$ is even : 3 $U_{i,i}$ is odd : 4	$U_{i,i}$ is even : $2\|p\| + 2\|q\|$ $U_{i,i}$ is odd : $3\|p\| + 2\|q\|$

Chapter 7

Conclusions

We have proposed a secure group key agreement protocol based on self-certified public keys in this paper. We use the properties of self-certified public keys to adapt the group key agreement protocol to mobile ad hoc networks. The correctness is specified when we describe our protocol and the protocol has the following security properties :

- **Secure against a passive adversary :**
  We have proved that an eavesdropper cannot efficiently distinguish a group key established and a random value under the DDH assumption in our protocol. Also, a participant U_l who has left the group cannot know any information about the new established group key K'.

- **Secure against an active adversary :**
  We provide two theorems for security against an active adversary. In the first theorem we prove that any malicious adversary A cannot impersonate another U_i to compute the valid commitment w_i. In the second theorem we prove that any malicious adversary A without secret r_i or with a fake secret r_i cannot generate the right form of the

other commitment $(w_{i+1}/w_{i-1})^{r_i}$. Also the malicious impersonator can be detected and be excluded from the group.

- **Forward secrecy**

  Any adversary with private key (x_i, d_i) at time $\tau + 1$ cannot obtain any information about the group key k at time τ.

- **Secure against known-key attacks**

  Any adversary with the group key K at time τ cannot learn other group key and cannot impersonate any participant at time $\tau + 1$.

For the future works, we have two goals to achieve. First, find a computationally symmetric **Join** phase. Second, reduce the computation cost in the **Leave** phase.

Bibliography

[1] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. New multiparty authentication services and key agreement protocols. IEEE Journal of Selected Areas in Communications, Vol. 18, No. 4, pp. 628-639, 2000.

[2] Klaus Becker and Uta Wille. Communication complexity of group key distribution. In *Proceedings of the 5rd ACM Conference on Computer and Communications Security* (CCS'98), pp. 1-6, ACM Press, 1998.

[3] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of Eurocrypt'00*, LNCS 1807, pp. 139-155, 2000.

[4] Colin Boyd and Juan Manuel Gonzalez Nieto. Round-optimal contributory conference key agreement. PKC 2003, LNCS 2567, pp. 161-174, Springer, 2003.

[5] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Provably authenticated group Diffie-Hellman key exchange - the dynamic case. In *Proceedings of Asiacrypt'01*, LNCS 2248, pp. 290-309, 2001.

[6] Emmanuel Bresson, Oliver Chevassut, and David Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In *Advances in Cryptology - Proceedings of Eurocrypt'02*, pp. 321-336, 2002.

[7] Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably authenticated group Diffie-Hellman key exchange. In *ACM CCS'01*, pp. 255-264, 2001.

[8] Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology - Eurocrypt'94*, pp. 275-286, Springer, 1995.

[9] Giovanni Di Crescenzo, Niels Ferguson, Russell Impagliazzo, and Markus Jakobsson. How to forget a secret. In *Proceedings of STACS'99*, LNCS 1563, pp. 500-509, 1999.

[10] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, Vol. IT-22, No. 6, pp. 644-654, 1976.

[11] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. CRYPTO'86, 1987.

[12] Girault. Self-certified public keys. In *Proceedings of the Eurocrypt'91*, pp. 491-497, 1991.

[13] Ingemar Ingemarsson, Donald T. Tang, and C.K.Wong. A conference key distribution system. *IEEE Transactions on Information Theory*, Vol. IT-28, No. 5, pp. 714-720, 1982.

[14] Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. In *Advances in Cryptology - Crypto'03*, pp. 110-125, 2003.

[15] Johann van der Merwe, Dawoud S. Dawoud, and Stephen McDonald. A survey on peer-to-peer key management for mobile ad hoc networks. ACM Computing Surveys (CSUR), Volume 39, Issue 1, 2007.

[16] Adi Shamir. How to share a secret. *Communications of the ACM* 22(11), pp. 612-613, 1979.

[17] Zuhua Shao. Self-certified signature scheme from pairings. Journal of Systems and Software (JSS), Vol. 80, Issue 3, pp. 388-395, 2007.

[18] Victor Shoup and Avi Rubin. Session key distribution using smart cards. In *Proceedings of Eurocrypt'96*, LNCS 1070, pp. 321-331, 1996.

[19] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie-Hellman key distribution extended to group communication. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security* (CCS'96), pp. 31-37, ACM Press, 1996.

[20] Joseph Chee Ming TEO and Chik How TAN. Authenticated dynamic group key agreement for autoconfigurable mobile ad hoc netowrks. IEICE Transactions on Communications 2006 E89-B(9):2480-2492.

[21] W. G. Tzeng. A secure fault-tolerant conference-key agreement protocol. *IEEE Transactions on Computers*, Volume 51, Issue 4, pp. 373-379, 2002.

[22] Y. M. Tseng. A communication-efficient and fault-tolerant conference-key agreement protocol with forward secrecy. Journal of Systems and Software (JSS), Volume 80, Issue 7, pp. 1091-1101, 2007.

[23] W. G. Tzeng and Z. J. Tzeng. Round-efficient conference key agreement protocols with provable security. ASIACRYPT2000, LNCS 1976, pp. 614-627, 2000.

[24] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE network* (special issue on network security), 1999.