NAIST-IS-DD0561033

Doctoral Dissertation

Constructing Efficient Infrastructures for Secure Communication with Various Autonomy

Hisashi Mohri

February 7, 2008

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Hisashi Mohri

Thesis Committee:
Professor Hiroyuki Seki
Professor Minoru Ito
Professor Minoru Okada
Associate Professor Yuichi Kaji

Constructing Efficient Infrastructures for Secure Communication with Various Autonomy[*]

Hisashi Mohri

Abstract

With the growth of the number of services offered on the computer network, system management techniques for assuring the security become more important. For example, sharing a cryptographic key is the very first step to realize secure communication over an untrusted network infrastructure. In ordinary networks, since we can assume a trusted third party, a central problem is how to design secure services on top of security operation techniques such as Public-Key Infrastructure (PKI). On the other hand, we should investigate a key agreement scheme itself in case of as ad-hoc networks and sensor networks. In ad-hoc networks, we should investigate self-organizing key agreement schemes without trusted third party because of node mobility and lack of trusted third party. Moreover, in sensor networks, it is very difficult to use a public-key cryptosystem itself due to the restriction of computational resources of sensor nodes. Hence, we should investigate key predistribution schemes, which are key agreement schemes without communication overhead for key exchange protocol between two nodes. In this thesis, efficient and robust infrastructures for secure communication in autonomous computer networks are investigated.

Trust management is a security infrastructure based on PKI. Various trust management models were proposed in the literature, however, there has been no trust management model that can explicitly represent a system whose internal

[*] Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0561033, February 7, 2008.

state changes. In Chapter 2, a trust management model that can represent a system with internal states is proposed. First, a policy specification language is presented to define the behavior of the system. Also, the verification problem is defined as the problem to decide whether the behavior of a system with a given policy satisfies a given verification property, and a verification method for the problem is also proposed. Experimental results on the verification of a sample problem are presented, using an implemented verification system and Prolog.

To adapt security infrastructure for ad-hoc networks, using a web-of-trust type PKI systems is a promising method. In a web-of-trust type PKI system, each node can issue certificates to others in a self-organized manner. In Chapter 3, new distributed algorithms for solving the certificate-chain discovery problem for a web-of-trust type PKI are proposed. The algorithm consists of a certificate searching algorithm and a certificate collecting algorithm. The former algorithm uses a distributed algorithm for constructing a spanning tree in the weighted directed graph that models a web-of-trust. Also, communication costs of the proposed methods and an existing method are evaluated by numerical analysis and computer simulation, and it is shown that the costs of the proposed methods is less than the cost of the existing method.

A practical solution to adapt a security infrastructure for sensor networks is key predistribution schemes in which cryptographic keys are predistributed in each sensor node before it is deployed. In Chapter 4, a new method and two extended methods in which keys are assigned according to a basic algebraic geometry are proposed. The proposed methods associate each node with a line over a two-dimensional finite plane, and manage keys so that two nodes can agree a key if and only if the associated lines intersect with each other. The performance of the proposed methods is computed analytically, and the results show that the proposed methods are more efficient and robust than the random key predistribution scheme.

Keywords:

information security, public-key infrastructure, trust management, key predistribution scheme, ad-hoc network, sensor network

種々の自律度に応じた効率のよいセキュア通信基盤に関する研究*

毛利　寿志

内容梗概

計算機ネットワーク上で提供されるサービスが拡大されるに従い，セキュリティ保護等を目的としたシステム管理基盤技術がますます重要となっている．その一例として，暗号鍵共有がある．暗号鍵共有は，信頼できないネットワーク上でのセキュアな通信を行うために必要不可欠な手法であり，現在までに広く研究されているセキュリティ基盤技術である．一般的なネットワークでは，すでに実用的な公開鍵暗号技術などが研究されており，鍵共有方式を包括するセキュリティ基盤技術の構築が問題となっている．この場合，信頼できる第三者を仮定したセキュリティ技術の導入が最適である．一方，アドホックネットワークやセンサネットワークなどのように，セキュリティ基盤技術の構築そのものが問題となっているようなネットワークも存在する．アドホックネットワークでは，ノードの移動性及び特定の信頼できる第三者機関の不在などの制約により，既存の公開鍵暗号基盤をそのまま利用できないという問題点が指摘されている．この場合は，信頼できる第三者を仮定せず，ネットワーク内の当事者同士で鍵共有を行う方式が必要となる．また，センサネットワークでは，センサノードの計算，通信，メモリ資源の制約により，公開鍵暗号そのものの使用ができないことが指摘されている．そのため，ネットワーク施行後に行われる鍵共有プロトコルは用いず，事前に各センサノードに暗号鍵を埋め込んでおく方式が適している．本研究では，従来のネットワークと二種類の新しいネットワーク（アドホックネットワーク，センサネットワーク）に関し，それぞれの特徴や制約に沿った効率が良く，かつ頑健性のあるセキュリティ基盤技術を提案する．

[*] 奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文，NAIST-IS-DD0561033, 2008年2月7日.

信用管理 (Trust Management) とは，公開鍵暗号基盤 (Public-Key Infrastructure，PKI と略) に基づいたアクセス制御技術である．第 2 章では，システムの内部状態を導入した信用管理モデルを提案する．まず，システムの振る舞いを定義するために，ポリシ記述言語を提案する．また，本研究では，与えられたポリシが与えられた検証項目を満たすかどうかを決定する問題として検証問題を定義し，モデル検査手法を用いてこの問題を解く手法を提案する．さらに，ある具体例について，Prolog を用いたモデル検査手法の実装法を提案し，検証に要する時間を示す．

アドホックネットワークにおける鍵共有手法に関する研究として，ネットワークに参加している各ユーザが各自で公開鍵証明書を発行し合うような，web-of-trust 型信頼モデルが注目されている．第 3 章では，アドホックネットワーク上の web-of-trust 型信頼モデルにおいて，効率の良い証明書連鎖発見アルゴリズムを提案する．提案アルゴリズムは，信頼モデル上で証明書連鎖を探索する段階と，探索により見つけた証明書連鎖を収集する段階から成る．探索段階では，web-of-trust 型信頼モデルを表す有向グラフ上で生成木を構成する分散アルゴリズムを利用している．また，提案手法と既存手法の通信コストを数値解析，及び計算機シミュレーションによって比較し，提案手法の方が既存手法より少ないコストで問題を解くことができることを示す．

センサネットワークにおける鍵共有方式に関する研究として，鍵事前格納方式が注目されている．鍵事前格納方式では，センサネットワークの施行前に各センサノードに事前に鍵を複数組み込んでおき，各センサノードを配布してセンサネットワークを構築する．第 4 章では，各ノードへの鍵事前格納について，代数幾何に基づいた新しい方式を提案する．有限二次平面上の各格子点にそれぞれ異なる鍵を割り当て，ある一直線上にある全ての鍵を一つのノードに格納するような方式を提案する．さらに，提案手法の効率性，頑健性を数値解析によって評価し，提案手法の方が既存手法より効率が良く頑健であることを示す．

キーワード

情報セキュリティ, 公開鍵暗号基盤, 信用管理, 鍵事前格納方式, アドホックネットワーク, センサネットワーク

List of Publications

1. Publications Related to the Thesis

1.1. Journal Papers

(1) Hisashi Mohri, Ikuya Yasuda, Yoshiaki Takata, and Hiroyuki Seki: "New Certificate Chain Discovery Methods for Trust Establishment in Ad Hoc Networks and Their Evaluation," IPSJ Journal, **49**, 1, pp.362–374, Jan. 2008. (Chapter 3)

(2) Hisashi Mohri, Ritsuko Matsumoto, and Yuichi Kaji: "Key Predistribution Schemes for Sensor Networks Using Finite Plane Geometry," IEICE Transactions on Information and Systems, **E91-D**, 5, May 2008, to appear. (Chapter 4)

1.2 International Conferences (Reviewed)

(3) Hisashi Mouri, Yoshiaki Takata, and Hiroyuki Seki: "A Formal Model for Stateful Trust Management Systems," In 9th IASTED International Conference on Software Engineering and Applications (SEA2005), 467-030, pp.87–92, AZ, USA, Nov. 2005. (Chapter 2)

(4) Hisashi Mohri, Ritsuko Matsumoto, and Yuichi Kaji: "Key Predistribution Schemes for Sensor Networks Using Lines and Points over a Finite Geometry," Third Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON2006), poster presentation, VA, USA, Sept. 2006. (Chapter 4)

(5) Hisashi Mohri, Ikuya Yasuda, Yoshiaki Takata, and Hiroyuki Seki: "Certificate Chain Discovery in Web of Trust for Ad Hoc Networks," The 2007 IEEE International Symposium on Ubisafe Computing (UbiSafe-07), 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW), **2**, pp.479–485, Ontario, Canada, May 2007. (Chapter 3)

1.3 Workshops

(6) Hisashi Mouri, Yoshiaki Takata, Hiroyuki Seki: "A Stateful Trust Management Model," IEICE General Conference, A-7-2, March 2004. (in Japanese)

(7) Hisashi Mouri, Yoshiaki Takata, and Hiroyuki Seki: "A Formal Model for Stateful Trust Management Systems," Technical Report of IEICE, SS2005-20, pp.13–18, June 2005.

(8) Ritsuko Matsumoto, Hisashi Mohri, and Yuichi Kaji: "A Pre-Distribution Scheme of Keys for Sensor Nodes Using Small Key Pools," Symposium on Cryptography and Information Security (SCIS), 3D4-4, Jan. 2006. (in Japanese)

(9) Ritsuko Matsumoto, Hisashi Mohri, and Yuichi Kaji: "Key Predistribution Schemes for Sensor Networks Using Lines and Points over a Finite Geometry," Technical Report of IEICE, ISEC2006-85, pp.97–104, Sept. 2006.

(10) Ikuya Yasuda, Hisashi Mohri, Yoshiaki Takata, and Hiroyuki Seki: "PKI Certificate Chain Finding Problem for Ad Hoc Networks," Technical Report of IPSJ, 2006-MBL-39(5), pp.31–38, Nov. 2006. (in Japanese)

(11) Ritsuko Matsumoto, Hisashi Mohri, and Yuichi Kaji: "Key Predistribution Schemes for Sensor Networks Using Finite Plane Geometry," Symposium on Cryptography and Information Security (SCIS), 3F2-5, Jan. 2007.

2. Other Publications

2.1. International Conference (Reviwed)

(12) Jun Noda, Mie Takahashi, Itaru Hosomi, Hisashi Mouri, Yoshiaki Takata and Hiroyuki Seki: "Integrating Presence Inference into Trust Management for Ubiquitous Systems," Proceedings of 11th ACM Symposium on Access Control Models and Technologies (SACMAT2006), pp.59–68, CA, USA, June 2006.

2.2. Workshops

(13) Jun Noda, Hisashi Mouri, Yoshiaki Takata, Hiroyuki Seki, Daigo Taguchi, Mie Takahashi, and Itaru Hosomi: “A New Architecture for Trust Management with User Presence,” Technical Report of IEICE, SS2005-50, pp.13–18, Oct. 2005.

(14) Yuichi Nino, Yuichi Kaji, Hisashi Mohri, and Daigo Taguchi: “A Proposal on Key Sharing Method for Access Control in P2P Network,” Symposium on Cryptography and Information Security (SCIS), 3D4-5, Jan. 2006. (in Japanese)

(15) Hisashi Mohri, Jun Noda, Yuichi Nino, and Yuichi Kaji: “A Key Management Scheme for Sensor Networks Utilizing Multiple Attribute Partitions,” Symposium on Cryptography and Information Security (SCIS), 3F2-4, Jan. 2007. (in Japanese)

(16) Hisashi Mohri and Ryo Nojima: “Private Evaluation of a Joint Graph,” The 3rd workshop on Information and Communication System Security, ICSS2006-24, Feb. 2007.

(17) Jun Noda, Yuichi Kaji, Hisashi Mohri, Yuichi Nino, and Toshiyasu Nakao: “Development and Evaluation of the Key Management Scheme Using Some Attributes Associated with Sensors,” Multimedia, Distributed, Cooperative, and Mobile Symposium (DICOMO), 3C-4, July 2007. (in Japanese)

2.3. Technical Report

(18) Yoshiaki Takata, Jun Noda, Mie Takahashi, Hisashi Mouri, Itaru Hosomi, Daigo Taguchi, and Hiroyuki Seki: “A Presence-aware Trust Management System,” Technical Report NAIST-IS-TR2005006, Nara Institute of Science and Technology, Oct. 2005.

3. Award

(1) Paper Award, The 2007 Multimedia, Distributed, Cooperative, and Mobile Symposium (DICOMO 2007), July 2007.

Acknowledgements

During the course of this work, I have received help from many people.

First, and foremost, I would like to thank my supervisor, Professor Hiroyuki Seki for giving me a chance to study computer science, and providing me with continuous support, encouragement and guidance for the work. Moreover, the professor accommodated me with an excellent environment in Seki Lab. to study information security and formal method. I would like to thank Professor Minoru Ito and Professor Minoru Okada for providing me with beneficial comments to improve this thesis. I am very grateful to Associate Professor Yuichi Kaji for his continuous support and guidance of the work. His wealth of knowledge and open mind greatly help me understanding and investigating information security.

I would like to thank Dr. Toshimitsu Masuzawa, Professor at Osaka University, for helpful comments about distributed algorithms. I also would like to thank Associate Professor Keiichi Yasumoto for helpful comments about ad-hoc networks. Dr. Yoshiaki Takata, Lecturer at Kochi University of Technology, has provided me guidance in the work since he was an assistant professor at Nara Institute of Science and Technology. I am very grateful to him for his continuous support and guidance.

I would like to thank Assistant Professor Yoshitaka Nakamura for helpful comments about ad-hoc and sensor networks and Dr. Isao Yagi for helpful comments about the work.

I am grateful to Miss Ikuya Yasuda and Miss Ritsuko Matsumoto, collaborators of the work, for their contributions and efforts to achieve good experimental results in the work. Also, I would like to thank Mr. Takahito Sakamoto for introducing related work about security techniques for sensor networks.

Finally, I would like to thank all the members of Seki Laboratory for giving me much encouragement and help.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the growth of the number of services offered on the computer network, system management techniques for assuring the security become more important. For example, sharing a cryptographic key is the very first step to realize secure communication over an untrusted networks infrastructure. Key agreement schemes include the following three types [1]:

1. the trusted-server scheme where nodes in a network exchange a key through a trusted third party and the performance of a key agreement scheme depends on the trusted third party,

2. the self-enforcing scheme where nodes in a network exchange a key without trusted third parties,

3. and the key predistribution schemes where keys are stored in each node before it is deployed.

Because which scheme we should use depends on the type of a network, no single security infrastructure can adapt all types of networks. In this thesis, security infrastructures in autonomous computer networks are focused. In ordinary networks, since we can assume a trusted third party, a central problem is how to design secure services on top of security operation techniques such as *Public-Key Infrastructure (PKI)* [2, 3]. On the other hand, we should investigate a key agreement scheme itself in case of as ad-hoc networks and sensor networks. In ad-hoc networks, because of node mobility and lack of trusted third party, we cannot use conventional security operation techniques such as PKI. Thus, we should investigate self-organizing key agreement schemes without trusted third party. To make matters worse, in sensor networks, it is very difficult to use a public-key cryptosystem due to the restriction of computational resources of sensor nodes. Thus, we should investigate key predistribution schemes which are key agreement schemes without communication overhead for key exchange protocol between two nodes. Taking these observations into consideration, efficient infrastructures for secure communication are investigated in this thesis.

Trust Management Access control is one of security techniques, in which only an authorized user can access a specific resource or receive a specific service. In a usual access control system, all the users and their rights have to be registered beforehand. However, in ubiquitous environment which offers a service to the general public, it is impossible to set the access rights of all users beforehand. Recently, *trust management* [4], which is a system management technique based on PKI, is extensively investigated. In a trust management system, a user requiring a service indicates *digital certificates* [2, 3] (or certificates for short) based on PKI. The system verifies the indicated certificates and assigns a kind of trust (e.g., a role in Role-Based Access Control (RBAC)) to the user. A module that assigns a trust is called a trust establishment module (TE), and the communication between a user and TE for assigning a trust is called trust negotiation [5, 6]. Based on the trust assigned to the user, the system determines whether it authorizes the user to access a specific resource or receive a specific service.

A typical model for trust management was proposed by Herzberg, et al. [7]. A policy specification language TPL (Trust Policy Language) based on first-order predicate logic was also proposed. Generally, an internal state of a system may change during a trust negotiation and the internal state may affect the next trust establishment. For example, consider a service which authorizes access rights for resources to a user who indicates a certificate issued by a trusted customer. In this service, it is desirable that a trusted customer who introduces more users is treated better (i.e., is given access rights to more resources). This service can be realized in such way that the current "prestige" of a trusted customer is stored in the system and is updated when the trusted customer introduces other users during trust establishment of the users. However, no trust management model has been proposed which can explicitly represent a system whose internal state changes as a result of the trust establishment.

In Chapter 2, a formal model for trust management systems with internal states (or *stateful trust management systems*) is introduced and a verification method for the model is proposed. First, a policy specification language to define the behavior of a system is proposed. A policy specification is a finite set of rules of definite Horn clause form. These rules define next-state predicates and output predicates from current-state predicates and input predicates. Also, formal verification for the proposed model is discussed. The verification problem is defined as the problem to decide for a given policy and a verification property whether the behavior of the system defined by the policy satisfies the verification property. Experimental results on the verification of a few problems are presented by using an implemented verification system and Prolog.

Certificate-Chain Discovery for Ad-Hoc Networks An ad-hoc wireless network is a formed network that can be de-formed on-the-fly without the need for any system administration [8]. Unfortunately, these characteristics prevent us from applying traditional security techniques to ad-hoc networks. In particular, the conventional PKI systems cannot be applied to ad-hoc networks [9, 10].

PKI is one of the most useful security techniques and is a security infrastructure in which we can authenticate a public key by using certificates. In public-key cryptosystems, we have to obtain other users' public keys to securely communicate with those users. In PKI systems, we can verify that pk_v is the public key of v by using certain certificates. Conventional PKI systems require several trusted authorities to issue certificates, store certificates, and so on.

One of the problems in adopting conventional PKI systems in ad-hoc networks is that we cannot assume a trusted certificate authority (to manage digital certificates) and a centralized repository (to store digital certificates securely). Moreover, we cannot assign the tasks of a trusted certificate authority or a centralized repository to any node in an ad-hoc network. If we did so, the PKI could not work because the node may move out of the network. Hence, to adapt security infrastructure for ad-hoc network, using a *web-of-trust-type PKI system* is a promising method. In a web-of-trust-type PKI system adopted in Pretty Good Privacy (PGP) [11], each node can issue certificates to others in a self-organizing manner. Some authors have proposed web-of-trust-type PKI systems for ad-hoc networks where each node has a distributed repository instead of a centralized repository [12, 13, 14, 15, 16, 17]. However, these systems suffer from the common problem that a node must discover a *certificate-chain* if the node does not have enough certificates for key authentication[1]. It is not trivial to discover a certificate-chain in distributed repositories.

In Chapter 3, the certificate-chain discovery problem in ad-hoc networks is investigated. First, a formal model for PKI in ad-hoc networks is given. A web-of-trust-type PKI system is defined as a weighted directed graph where an edge represents the direct trust relation guaranteed by a certificate between the nodes and the weight of the edge represents the number of hops in the lower (physical) layer. Next, a new distributed algorithm and its modification for solving the certificate-chain discovery problem are proposed based on the model. The algorithm consists of a certificate searching algorithm and a certificate collecting algorithm. The former algorithm uses a distributed algorithm for constructing a spanning tree in the weighted directed graph. Finally, communication costs of the proposed methods and an existing method are evaluated by numerical analysis and computer simulation, and it is shown that the costs of the proposed methods is less than the cost of the existing method.

Key Predistribution Scheme for Sensor Network Requirement for a security infrastructure in sensor networks is different from that in ad-hoc networks while the two networks are similar. A critical issue in sensor networks is electricity consumption. Because usual energy of sensor nodes is battery, reducing electricity consumption helps increase the sensor network's overall lifetime. For example, because energy consumed for computation and communication is so expensive, the security infrastructure in sensor networks should be low computation and communication overhead. Moreover, electricity consumption is proportional to memory size in the node. Hence, the security infrastructure with low memory size helps to reduce energy consumed.

[1] A set of certificates for authenticating a node's public key is called a certificate-chain [18].

Many known key agreement schemes commonly make use of techniques of public-key cryptography or similar methods, and require terminals to perform rather large and complicated computation. Consequently, it is widely considered that those techniques for key agreement are not suitable for a sensor network which consists of resource-restricted sensor nodes. A primitive but practical solution for the key agreement in sensor networks is to predistribute keys to sensor nodes before they are deployed. The random key distribution is an implementation of key predistribution proposed by Eschenauer [19]. In the random key distribution, a key manager determines a set of keys beforehand. The set of keys are sometimes called a key pool. Keys which are embedded in a sensor node is randomly chosen by the key manager before the node is deployed to a field. Two sensor nodes can agree a cryptographic key if and only if they happen to have keys in common. Several extensions of this simple scheme have been investigated.

The primal objective of Chapter 4 is to construct a better implementation of the key predistribution scheme than the random key distribution scheme of [19]. As for the measure of "goodness", three probabilities are investigated. To evaluate the performance of the schemes, three quantitative measures to evaluate key predistribution schemes are considered; the node *memory size*, the *connectivity* and the *key-survivability*. The node memory size (or *memory size* for simplicity) is the number of keys which a node needs to remember, the *connectivity* is the probability that randomly chosen two sensor nodes have one or more keys in common, and the *key-survivability* is the probability that the key which has been agreed by two nodes stays secure even if an intruder mounts the *node capture attack*, in which an intruder captures sensor nodes deployed in a field, and retrieves keys embedded in the nodes. In the random key distribution scheme, these probabilities are determined just by coincidence.

The goal of Chapter 4 is to actively control keys to be embedded in sensor nodes so that all of memory size, the connectivity and the key-survivability are larger than the random key distribution scheme. A new key predistribution scheme and two extended methods are proposed. In the proposed methods, keys are assigned according to a basic geometric geometry. The proposed methods associate each node with a line over a two-dimensional finite plane, and manage keys so that two nodes can agree a key if and only if the associated lines intersect with each other. Two randomly chosen lines intersect with each other unless they are parallel, thus two nodes succeed in key agreement with high probability. Then, the performance of the proposed methods is computed analytically, and it is shown that the proposed scheme realizes better trade-off points than the random key distribution scheme. Also, memory size of the basic scheme and the random key predistribution scheme where the same connectivity and key-survivability are also compared, and it is shown that memory usage of the basic scheme is about 23% of that of the random key distribution scheme.

Chapter 2

A Formal Model for Stateful Trust Management Systems

2.1. Introduction

Recently, trust management [4], which is a system management technique based on Public-Key Infrastructure (PKI), is extensively investigated. In a trust management system, a user requiring a service indicates digital certificates (or certificates for short) based on PKI. The system verifies the indicated certificates and assigns a kind of trust (e.g., a role in Role-Based Access Control (RBAC) [20]) to the user. A module that assigns a trust is called a trust establishment module (TE), and the communication between a user and TE for assigning a trust is called trust negotiation [6]. Based on the trust assigned to the user, the system determines whether it authorizes the user to access a specific resource or receive a specific service. A typical model for trust management was proposed by Herzberg, et al. [7]. In their model, a trust management system consists of a trust establishment module and a RBAC module (Figure 2.1). A policy specification language TPL (Trust Policy Language) based on first-order predicate logic was also proposed. Cassandra [21, 22] is another policy specification language based on first-order logic. Computational complexity of access control with Cassandra has been investigated in [21, 22]. As well as [7, 21, 22], a few trust management systems and models have been proposed [5, 23].

Generally, an internal state of a system may change during a trust negotiation

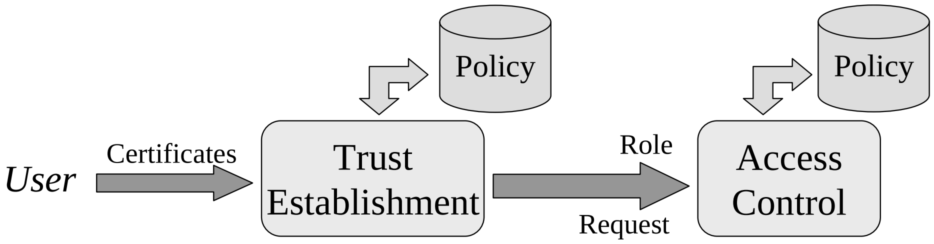

Figure 2.1. A trust management model

and the internal state may affect the next trust establishment. For example, consider a service that authorizes access rights for resources to a user who indicates a certificate issued by a trusted customer. In this service, it may be desirable that a trusted customer who introduces more users is treated better (i.e., is given access rights for more resources). This service can be realized in such a way that the current "prestige" of a trusted customer is stored in the system and is updated when the trusted customer introduces other users during trust establishment of the users. However, no trust management model has been proposed that can explicitly represent a system whose internal state changes as a result of the trust establishment. For example, one may think the above-mentioned policy "if a certificate is indicated, then the prestige of the trusted customer is increased by one." can be specified as a first-order formula as follows. Let $cert(X, Y)$ be a predicate which denotes that there is a certificate issued by X for Y, and $point(X, P)$ be a predicate which denotes that the prestige of X is P.

$$\begin{aligned} &point(X, 0). \\ point(X, P+1) \quad &:- \quad cert(X, Y), point(X, P). \end{aligned} \tag{2.1}$$

However, by the formula (2.1), once $cert(X, Y)$ becomes true for some Y, $point(X, P)$ becomes true for every $P \geq 0$, i.e., the prestige of user X increases infinitely. This is because both sides of the formula (2.1) represent the truth values of them in a same interpretation, not representing a state transition.

In this chapter, a policy specification language to define the behavior of stateful trust management systems is proposed. A policy specification is a finite set of rules of definite Horn clause form. These rules define next-state predicates and output predicates from current-state predicates and input predicates. Current-state and next-state predicates represent internal states of the system, and output

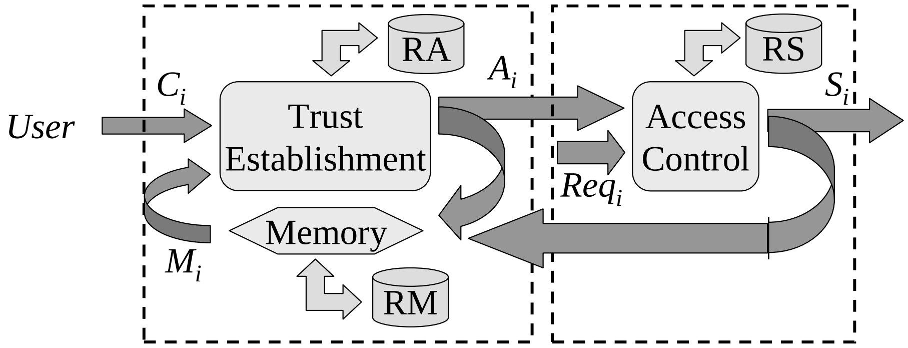

Figure 2.2. Proposed model

predicates represent the result of trust establishment and access control. Also, an example of a stateful trust management system by using the policy specification language is given, and a verification method for the model is investigated. The verification problem is defined as the problem to decide for a given policy and a verification property whether the behavior of the system defined by the policy satisfies the verification property. Assuming that a policy specifies a finite state system, a semi-automatic verification method is proposed: For a given policy and a verification property written in LTL [24], the author constructs a Prolog program that searches the state space defined by the policy according to the verification property and outputs 'yes' if the system satisfies the verification property and outputs a counterexample otherwise. A verification result based on the proposed method is also reported.

2.2. Proposed Model

2.2.1 Scheme for Stateful Trust Management Systems

Figure 2.2 shows an overview of the proposed model. Similar to the trust management model of Herzberg, et al. [7], a user who is requesting a service indicates certificates to the trust establishment module. A system has a storage (or memory) for storing information on the interaction history between the user and the system. This is one of the main features of the proposed model because most of the previous trust management models such as KeyNote [25] and TPL [7] are based on a subclass of first-order predicate logic which has "stateless" or "memory-less" semantics. This information stored in the memory is used at the next trust negotiation, and the information is updated as a result of the trust establishment. Based on the output of the trust establishment and the request from the user, the access control module decides whether the user can be provided with the requested service. One application of the proposed model is the reputation system in which the interaction history is used for estimating the user's (or other's) reputation [26]. The proposed model can store any kind of information according to a given policy and application of the proposed model is not restricted to reputation systems. A time period from indication of certificates and service request to completion of trust establishment and update of the memory is called a *round*. Let C_i, M_i, A_i, Req_i and S_i represent the followings in the i-th round:

$$\begin{aligned} C_i &: \text{ certificates indicated to the system,} \\ M_i &: \text{ contents of the memory (i.e., internal state),} \\ A_i &: \text{ output of the trust establishment,} \\ Req_i &: \text{ service requests from the user,} \\ S_i &: \text{ output of the access control.} \end{aligned}$$

Let RA be a trust establishment policy, RM be a policy for determining how the memory is updated (i.e., internal state transition), RS be a policy for determining whether the service is permitted. Relation among them can be represented as follows (see Figure 2.2):

$$A_i = RA(M_i, C_i) \tag{2.2}$$

$$M_{i+1} = RM(A_i, S_i, M_i) \tag{2.3}$$

$$S_i = RS(A_i, Req_i) \tag{2.4}$$

For example, equation (2.2) means that A_i is computed by applying RA to M_i and C_i. In ubiquitous computing environments, it is important to take context [27] into consideration in order to provide a service that fits the user's physical and/or logical situation, and a trust management system should consider context [28]. In this chapter, we assume for simplicity that context is given as certificates issued by a context server, i.e., context is included in C_i. Hereafter, a triple (RA, RS, RM) is simply called *a policy*.

2.2.2 Policy Predicates

Predicates used in a policy are classified into four categories. Some predicates are shown as examples for each category.

1. Predicates for C_i and Req_i (input predicates)
   $cert(User1, User2)$: There is a certificate issued by $User1$ for $User2$ in C_i.

2. Predicates for A_i and S_i (output predicates)
   $custom(User)$, $intro(User)$, $prefer(User)$: $User$ is identified as a customer, introducer, preferred customer, respectively, in round i as a result of trust establishment or access control.

3. Predicate for M_i (current-state predicates)
   $point(User, Point)$: $User$'s prestige is scored as $Point$ in the current state.

4. Predicates for M_{i+1} (next-state predicates)
   Each predicate in this category has the same name as a predicate in category 3 followed by a prime (') symbol.
   $point'(User, Point)$: User's prestige is scored as $Point$ in the next state.

2.2.3 Syntax and Semantics of Policy

For a policy (RA, RS, RM), each of RA, RS and RM is specified by a set of definite Horn clauses called *inference rules* (or *rules* for short). We put the following restriction on the syntax of policy specification.

- The head of each rule in RA and RS should be a predicate in category 2.
- RM is divided into the initial state definition part ($RMinit$) and the transition definition part ($RMtrans$).
  - Each rule in $RMinit$ should be a fact (i.e., an inference rule with empty assumptions) on a predicate in category 3.
  - The head of each rule in $RMtrans$ should be a predicate in category 4.

These restrictions embody the relation represented in equations (2.2), (2.3) and (2.4).

Next, the semantics of a policy specification is described. In every rule of a policy, the author interprets every predicate $p(\ldots)$ as if it has an extra argument representing a round i like $p(i, \ldots)$. That is, $p(i, \ldots)$ represents the truth of $p(\ldots)$ in round i. Each rule is interpreted as follows:

- Each rule $p_0(\ldots) :- p_1(\ldots), \ldots, p_n(\ldots)$ specified in RA or RS is interpreted as $p_0(I, \ldots) :- p_1(I, \ldots), \ldots, p_n(I, \ldots)$ using a new variable I that represents an arbitrary round. That is, RA and RS define the relationship among the predicates within a single round I.

- $RMtrans$ defines the truth of the predicates in category 4 in round i (equivalently, the truth of the predicates in category 3 in round $i + 1$ in term of predicates in categories 1 to 3 in round i). Thus, each rule $p_0(\ldots) :- p_1(\ldots), \ldots, p_n(\ldots)$ in $RMtrans$ is interpreted as $p_0(I + 1, \ldots) :- p_1(I, \ldots), \ldots, p_n(I, \ldots)$.

- $RMinit$ defines the truth of the predicate in category 3 in round 1 (the initial round). Thus, each rule $p(\ldots)$ in $RMinit$ is interpreted as $p(1, \ldots)$.

2.3. An Example of Policy Specification

In this section, “introducer gains benefits” system is introduced as an example which should be defined as a stateful trust management system in its nature. In the following, a policy specification is shown and the behavior of the system is explained.

“Introducer gains benefits” system is the system that a shop s gives prestige to an introducer a when a introduces another user b to s. When we say that a introduces b to s, it means that a issues to b a certificate which gives access rights to s. The existence of the certificate is represented by the predicate $cert(a, b)$. In the system, shop s has the following policy.

i. The shop authorizes the user a as a customer.

ii. The shop authorizes a user introduced by a customer as a customer.

iii. When (ii) applies, the introducer's prestige increases by one.

iv. The shop authorizes the customer whose prestige is more than or equal to five as a preferred customer.

In this thesis, the following policy predicates are used for specifying a policy (RA, RS, RM) for this example.

$$
\begin{aligned}
\mathit{custom}(X) &: \text{user } X \text{ is authorized as a customer} \\
\mathit{intro}(X) &: \text{user } X \text{ introduces another user to the shop} \\
\mathit{prefer}(X) &: \text{user } X \text{ is authorized as a preferred customer} \\
\mathit{point}(X, P) &: \text{the } X\text{'s prestige is } P
\end{aligned}
$$

Example 2.1. *RA, RS and RM of the above system are specified by using Prolog-like syntax. A name whose first character is a capital letter or an underscore (_) represents a variable. Otherwise, a name represents a predicate or constant.*

```
[RA,RS]
custom(a).
custom(Y) :- cert(X,Y), custom(X).
intro(X)  :- cert(X,_), custom(X).
prefer(X) :- custom(X), point(X,P), P >= 5.
[RM trans]
point'(X,P) :- intro(X), point(X,Q), P = Q+1.
point'(X,P) :- not intro(X), point(X,P).
[RM init]
point(X,0).
```

When customer a introduces user b to the shop s, $\mathit{custom}(b)$ and $\mathit{intro}(a)$ become true. Moreover, the a's prestige increases by one and $\mathit{point}(a, 1)$ becomes true since $\mathit{point}(a, 0)$ was true in the previous round.

Example 2.2. *The semantics of the policy in Example 2.1 is given by the following set of definite Horn clauses according to Section 2.2.3.*

```
[RA,RS]
custom(I,a).
custom(I,Y) :- cert(I,X,Y), custom(I,X).
```

```
intro(I,X)  :- cert(I,X,_), custom(I,X).
prefer(I,X) :- custom(I,X), point(I,X,P), P >= 5.
[RM trans]
point(I+1,X,P) :- intro(I,X), point(I,X,Q), P = Q+1.
point(I+1,X,P) :- not intro(I,X), point(I,X,P).
[RM init]
point(1,X,0).
```

2.4. Formal Verification

It is desirable to have a method for verifying whether there is no contradiction among RA, RS and RM, whether the policy will not cause any problematic system behavior, etc. In this section, the verification problem is defined as the problem to decide whether the behavior of a system with a given policy satisfies a given verification property, and a verification method for the problem is proposed.

2.4.1 Verification Problem

Input of the verification problem is a pair of a policy to be verified and a verification property that the policy should satisfy. A verification property is specified by an LTL (linear temporal logic) formula [24]. Given a policy (RA, RS, RM), a transition system (Kripke structure) is induced to define the verification problem by satisfaction relation in LTL (Section 2.4.1).

Definition 2.1. ***The verification problem***

Input *A policy* (RA, RS, RM) *and an LTL formula* ψ

Output *Does the transition system* S *induced by* (RA, RS, RM) *satisfy* ψ*?*

Induced Transition System The transition system $S = (Q, I, R)$ induced by (RA, RS, RM) is defined as follows. A request Req_i from a user is assumed to be a member of the set C_i of input certificates in round i for simplicity. A set C of input certificates is called an *input* for short.

- Q is a set of states defined by (M, C) where M is a memory content and C is an input.

- I is a set of initial states defined by
  $I = \{(M_1, C) \mid M_1$ is the initial content of the memory and C is an input$\}$.
- R is a transition relation defined by $((M_i, C_i), (M_{i+1}, C)) \in R$ if and only if M_{i+1} is the memory content defined by (2.2), (2.3) and (2.4) in section 2.2.1 for M_i and C_i.

Note that a labeled transition system (RA, RS, RM) is translated to an equivalent Kripke structure $S = (Q, I, R)$ by shifting an input C into the second component of a state in Q. This transition system is in general an infinite state system. However, in this section, only a finite state system is considered, i.e., we assume that the domain of each predicate of categories 2 and 3 in Section 2.2.2 is finite.

Verification Property A verification property is specified by an arbitrary LTL formula where an atomic proposition is a ground instance of a policy predicate of categories 2 and 3. The following LTL formula (2.5) is a verification property which means that "if $cert(a, b)$ is true infinitely often, then $prefer(a)$ will eventually become true." :

$$\Box\Diamond cert(a, b) \to \Diamond prefer(a) \tag{2.5}$$

2.4.2 Verification Method Using Prolog

Assuming that a policy specifies a finite state system and a verification property is written in LTL, an automatic verification method is proposed. The author examined the following LTL model checking methods for a finite state system specified by Horn clauses.

- **Method 1 : SPIN** In this method, the system verified by using a model checker SPIN [29]. However, this method needs a transformation from definite Horn clauses to a program in Promela, the modeling language of SPIN, and such a transformation is not trivial in general.
- **Method 2 : SPIN with Prolog** In this method, I give a Promela program that executes an external Prolog interpreter to compute the next state for each state transition.

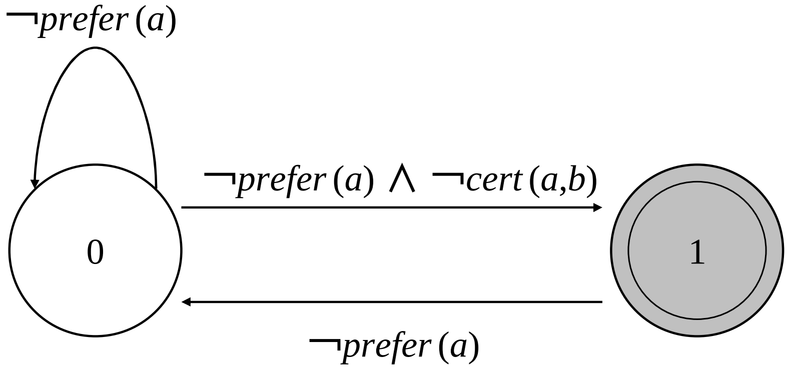

Figure 2.3. An example of a never process

- **Method 3 : Prolog** In this method, model checking is performed by a Prolog program.

Neither method 2 nor 3 needs transformation of Horn clauses. In method 3, we have to implement a model checking algorithm as a Prolog program. Methods 2 and 3 ware implemented in this thesis, and method 3 was more efficient than method 2 (shown in Section 2.4.3).

An implementation of method 3 is as follows. First, Büchi automaton BA representing the negation of the given LTL formula is constructed and is transformed into a Prolog program. Next, the product Büchi automaton P of BA and the state transition system S induced by (RA, RS, RM) is constructed. Finally, a model checking algorithm is executed on P, which finds an execution sequence of S that satisfies the negation of the verification property. The detail of the Büchi automaton BA, the product Büchi automaton P and the model checking algorithm is described in the following section.

2.4.3 Verification Example

This section describes the details of the conversion of an input of the verification problem into a Prolog program, taking the "introducer gains benefits" system in Section 2.3 as an example.

Input to the Verification Problem

- **Representation of *RA*, *RS* and *RM*** Each predicate of categories 1–4 in Section 2.2.2 is extend so that it has arguments representing the state components M_i and C_i. For example, the predicate $cert(M, C, X, Y)$ represents the truth of $cert(X, Y)$ when the state is (M, C). Each predicate of category 4 is defined as a predicate distinct from each predicate of category 3. In this thesis, underscore (_) is used instead of prime (’) symbol. The predicate $initial(M)$ represents $RMinit$, and is true if and only if M is the initial content of the memory. By this conversion, the following program is obtained from Example 2.1.

  Example 2.3. ***A program obtained from Example*** 2.1

```
[RA,RS]
custom1(_M, _C, a, _).

custom1(M,C,Y,L) :- cert(M,C,X,Y),
                   not(member(Y,L)),
                   custom1(M,C,X,[Y|L]).
custom(M,C,Y)   :- custom1(M,C,Y,[]).
intro(M,C,X)    :- cert(M,C,X,_),
                   custom(M,C,X).
prefer(M,C,X)   :- custom(M,C,X),
                   point(M,C,X,P), P >= 5.
[RM trans]
point_(M,C,X,P) :- intro(M,C,X),
                   point(M,C,X,P),
                   P is Q+1,
                   Q < 5, !.
point_(M,C,X,P) :- point(M,C,X,P).
[RM init]
initial([0,0]).
```

  Note that the definition of custom is modified to avoid an infinite recursion caused by the top-down evaluation strategy of Prolog. Also note that the first rule of $RMtrans$ is modified so that each user’s prestige is more than or equal to zero and less than or equal to five to make the state space finite.

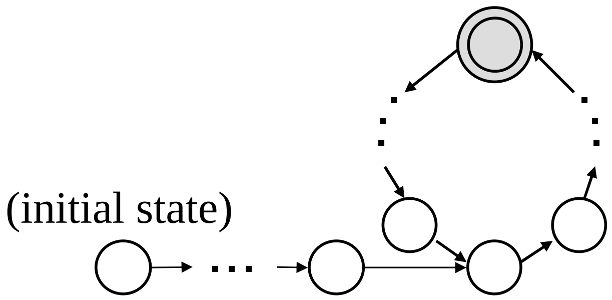

Figure 2.4. Transition sequence

- **Representation of states** For simplicity, we assume that categories 1 and 3 contain exactly one predicate, respectively. Let $p : D_1 \times D_2 \times \cdots \times D_n \rightarrow \{true, false\}$ be the predicate in category 1. A state component C is represented by a list of truth values whose length is $|D_1| \times |D_2| \times \cdots \times |D_n|$. Each element of the list corresponds to each element $(d_1, d_2, \ldots, d_n)$ of $D_1 \times \cdots \times D_n$ and represents the truth value of $p(d_1, d_2, \ldots, d_n)$. For example, if p is $cert(User1, User2)$ and the domain of both $User1$ and $User2$ is $\{a, b\}$, then the state component is represented as a 4-tuple of truth values where the elements of the tuple represent the truth values of $cert(a, a)$, $cert(a, b)$, $cert(b, a)$, and $cert(b, b)$, respectively. In this thesis, 0 and 1 are used for denoting false and true, respectively.

  An input part C of state (M, C) is represented by a list consisting of 0 and 1. For example, the list [0,1,0,0] represents that $cert(a, b)$ is true, and the list [0,1,1,0] represents that $cert(a, b)$ and $cert(b, a)$ are true. A memory part M of state (M, C) is represented in the same way as C. In the "introducer gains benefits" system, for each user u there exists exactly one value of P such that the predicate $point(u, P)$ is true. In such a case, the state component M can be represented by a list of the values P that makes $point(u, P)$ true for each $u \in D_1$. For example, the list [3,2] denotes that both $point(a, 3)$ and $point(b, 2)$ are true.

- **Definition of input predicates** The truth of the predicate of category

Table 2.1. Verification time

# of users	Restriction	# of state transitions of S	Verification Time[†]	
			SPIN+Prolog	Prolog
3	$cert(v, v)$ is fixed to be false	884,736	17m.16s.	5s.
3	$cert(v, v)$ is not fixed	56,623,104	16h.54m.35s.	1m.16s.
4	$cert(v, v)$ is fixed to be false	3,623,878,656	–	1h.48m.39s.

[†] SWI-Prolog v.5.0.6 and SPIN v.4.2.5, on Solaris8 (UltraSPARCIIe(500MHz), 256MB RAM)

1 depends only on C, and is defined by the following facts.

$$\begin{array}{ll} cert(M, [1, \_, \_, \_], a, a), & cert(M, [\_, 1, \_, \_], a, b), \\ cert(M, [\_, \_, 1, \_], b, a), & cert(M, [\_, \_, \_, 1], b, b). \end{array}$$

Similarly, the truth of the predicate of category 3 depends only on M, and is defined by the following facts.

$$point([P, \_], C, a, P), \quad point([\_, P], C, b, P).$$

- **Description of transition relation** The predicate $trans(M_i, C, M_{i+1})$ which represents the relation between (M_i, C) and M_{i+1} is defined as follows. The internal state $M_{i+1} = [Pa, Pb]$ is defined by $point\_$ which is the predicate of category 4.

$$\begin{array}{lll} trans(M, C, [Pa, Pb]) & :- & valid\_cert\_set(C), \\ & & point\_(M, C, a, Pa), \\ & & point\_(M, C, b, Pb). \end{array}$$

The predicate $valid\_cert\_set(C)$ is true if and only if C is a valid input for the trust management system. This predicate is defined as follows.

$$\begin{aligned} \mathit{valid\_cert\_set}([AA, AB, BA, BB]) \quad :- \quad & (AA = 0 \; ; \; AA = 1), \\ & (AB = 0 \; ; \; AB = 1), \\ & (BA = 0 \; ; \; BA = 1), \\ & (BB = 0 \; ; \; BB = 1). \end{aligned}$$

where ";" is the disjunction operator, i.e., the predicate *valid_cert_set*(C) is true if and only if C is a list of 0 and 1 whose length is four.

- **Description of verification property** Büchi automaton BA represents the negation of a verification property, i.e., it accepts any execution sequence that does not satisfy the verification property. BA, sometimes called a never process, is constructed from a given verification property written in LTL by using an automatic tool such as SPIN. Figure 2.3 shows an example of a never process generated from the verification property given by formula (5). This never process can be represented in Prolog as follows.

Example 2.4. *Prolog specification of a Büchi automaton BA in Figure 2.3.*

```
ba_trans(M, C, 0, 0) :- not(prefer(M, C, a)).
ba_trans(M, C, 0, 1) :- not(prefer(M, C, a)),
                        cert(M, C, a, b).
ba_trans(M, C, 1, 0) :- not(prefer(M, C, a)).
ba_initial(0).
ba_accept(1).
```

Büchi automaton P is the product automaton constructed from a Büchi automaton BA and a transition system S induced by (RA, RS, RM) in Section 2.4.1. Given BA and S, P is defined by the following Prolog program.

```
p_trans(M, C, Q, M2, Q2) :- trans(M, C, M2),
                            ba_trans(M, C, Q, Q2).
p_initial(M, Q) :- initial(M), ba_initial(Q).
p_accept(_, Q)  :- ba_accept(Q).
```

Verification Algorithm The Büchi automaton P accepts an infinite sequence π if and only if π is an execution sequence of S and does not satisfy the verification property. The language of P is not empty if and only if there exists a sequence consisting of a simple path from the initial state followed by a simple cycle including an accepting state (Figure 2.4). To find this path and cycle, the nested depth-first search [29] used in SPIN is implemented, as a Prolog program. The main part of the program is shown in Appendix A.

Verification Result The above-mentioned Prolog program using SWI-Prolog [30] was executed, and it successfully halted without counterexample, which means the policy satisfies formula (2.5). The same verification using the method 2 was also performed and it was finished in the same result. For each method, the running times for two cases are shown in Table 2.1, where the set of users is either $\{a, b, c\}$ or $\{a, b, c, d\}$. As shown in Table 2.1, method 3 was much faster than method 2. The predicate $cert(v, v)$ for each $v \in \{a, b, c, d\}$ represents the existence of a certificate for a user issued by him- or herself. If we assume that such a certificate, which is useless in practice, cannot be indicated, then the running time is much reduced.

The verification time is propotional to the number of state transitions of S. The number of state transitions of S in this example equals $2^{2n^2} \times m^n$ where n is the number of users and m is the size of the range of the prestige ($m = 6$ in this example): As described above, the size of the range of C is 2^{n^2} and that of M is m^n. In this example policy, the next state component M_{i+1} is uniquely determined by M_i and C_i, and thus the number of state transitions equals the product of the size of the range of M_i, the one of C_i, and the one of C_{i+1}. When we assume that the self-issued certificates cannot be indicated, the size of the range of C reduces to $2^{n(n-1)}$.

2.5. Conclusion of Chapter 2

In this chapter, a formal model for stateful trust management systems which can represent a system with internal states was proposed. Also, an LTL model checking method for the model was proposed. In this verification method, only a finite state system is considered, and a verifier can choose an arbitrary LTL formula as a verification property.

Chapter 3

New Certificate Chain Discovery Methods for Trust Establishment in Ad Hoc Networks and Their Evaluation

3.1. Introduction

In this chapter, certificate-chain discovery problem in ad-hoc networks is investigated. This chapter is focused on a *web-of-trust*-type PKI system considered in Pretty Good Privacy (PGP) [11]. In such a system, each node can issue certificates to others in a self-organizing manner, so it is suitable for adapting to ad-hoc networks. Some authors have proposed web-of-trust-type systems for ad-hoc networks where each node has a distributed certificate repository [12, 13, 14, 15, 16, 17]. However, these systems suffer from the common problem that a node must discover a certificate-chain if the node does not have enough certificates for key authentication.

A set of certificates for authenticating a node's public key is called a *certificate-chain* [18]. In this thesis, the node that authenticates a public key is called the *source node*, and the node whose public key will be verified by the source node is called the *destination node* (See the following section for the definitions of source node, destination node, and certificate-chain). Even if the source node does not directly sign the public key, the source node is able to verify a public key by discovering a certificate-chain from the source node to the destination node because the trust relation represented by certificates is *transitive*. First, the source node trusts the nodes whose public keys are signed by the source node because the source node can verify the certificates using her public key. Next, the source node trusts nodes whose public keys are signed by the already trusted nodes because the source node can verify the certificates using the already trusted nodes' public keys. In conventional PKI systems, we can find such a certificate-chain from the set of certificates in a trusted repository. However, it is not trivial to discover a certificate-chain in distributed repositories.

In this chapter, new distributed algorithms for solving the certificate-chain discovery problem in ad-hoc networks are proposed. After stating the certificate-chain discovery problem, the author models a web-of-trust-type PKI system, shows that solving the problem can be reduced to finding a path between two nodes in the graph, and proposes a new distributed algorithm for solving the problem. In this thesis, the certificate-chain discovery is divided into the *certificate searching phase* and the *certificate collecting phase*. Then, a search method is proposed based on a distributed algorithm for constructing a spanning tree, and a method for collecting all certificates in the discovered certificate-chain is also proposed. The whole algorithm will be called a *basic scheme* in this chapter. Also, a modification scheme of the basic scheme is also proposed. Furthermore, a measure of communication cost is proposed, and according to the measure, the proposed methods are compared with the existing method. For large-size networks, the author derives formulae that approximately represent the communication costs of the existing and proposed methods and numerically evaluates the formulae. For moderate-size networks, communication costs are evaluated by computer simulation on a randomly generated unit disk graph as a routing graph and a random Hamilton graph as a trust model.

Certificate :

pk_v is the public key of v.

sk_u (issuer's signature)

v	: A user
pk_v	: The public key of v
u	: The issuer (the signer of the certificate)
sk_u	: The secret key of u (to sign)
pk_u	: The public key of u (to verify)

Figure 3.1. A certificate for v issued by u

3.2. Problem Statement

A certificate C is a data structure including a public key, ID of the owner of the public key, and the signature by a signer. For a certificate C, the owner of a public key is called the *user of* C and its signer is called the *issuer of* C (See Figure 3.1). For simplicity, a certificate of which issuer and user are u and v respectively is written as $\langle u, v\rangle$.

This thesis focuses a web-of-trust-type PKI system where every node v has a repository and stores only the certificate $\langle u, v\rangle$ for some u and $\langle v, w\rangle$ for some w (see Figure 3.1) [13, 14, 15, 16, 17]. That is, each node v stores certificates such that v is an issuer or a user of the certificates in the PKI system, and each node does not have a list of trusted nodes. The procedure for constructing such a PKI system in an ad-hoc network is as follows [12, 13].

(Step 1) Creating cryptographic keys: Every node creates a secret key and corresponding public key using a public key cryptosystem. Though there are many public key cryptosystems, which one is most suitable for ad-hoc networks is not considered in this thesis.

(Step 2) Creating a certificate: If a node v would like another node u to issue certificate for v, v sends the public key pk_v to u. After receiving the key, u creates the certificate $u \rightarrow v$ and sends the certificate to v. And then, each node stores the certificate in each repository. In this step, u has to get the v's public key pk_v without PKI mechanisms. To do this step securely, for example, we may use a side channel such as an infrared channel at the time of a physical encounter.

This system enables us to reduce communication cost for obtaining certificates to be stored in a repository in advance. Moreover, this method reduces the cost of the certificate revocation phase. In this method, a certificate is only held by its issuer and user (the owner of the public key in the certificate). When the issuer (or the user) wants to revoke the certificate, she only has to send revocation information to the user (or the issuer) without heavy computation and communication (e.g., using a *certificate revocation list* [20]). In this thesis, "trust" represents a trust relation between the issuer and the user of a certificate. That is, "an issuer trusts a user to be honest and to correctly authenticate the owner of a public key before signing it [31]." By a certificate-chain and the trust relation in a certificate, a source node can verify the public key of a destination node even if the source node does not issue the certificate in which user is the destination node.

However, there is a new problem of discovering a path of certificates based on the trust relationship in distributed repositories to verify whether a public key is correct. this problem is called the *certificate-chain discovery problem in ad-hoc networks* (Definition 1). Assume that a node u wants to communicate with another node v. If u does not directly trust v, u has to find a certificate-chain from u to v. A *certificate-chain from u to v* is a sequence of certificates $\langle u_0, u_1\rangle, \langle u_1, u_2\rangle, \ldots, \langle u_{l-1}, u_l\rangle$ $(l \geq 1)$ such that $u = u_0$, $v = u_l$, and the user of the certificate $\langle u_{i-1}, u_i\rangle$ is the issuer of the next certificate $\langle u_i, u_{i+1}\rangle$. u_0 can verify the public key pk_{u_1} by the certificate $\langle u_0, u_1\rangle$. Also, u_0 can verify the public key pk_{u_2} by using the certificate $\langle u_1, u_2\rangle$ and the verified public key pk_{u_1}. By performing this verification repeatedly, u_0 can get the public key pk_{u_l}. We say u_0 and u_l are the *source node* and the *destination node* of the certificate-chain, respectively. The source node u can verify the public key pk_{u_1} based on a technique of the public-key cryptography (digital signature) and the trust relation ("u trusts u_1"). Moreover, u can verify the public key pk_v based on the certificate-chain and the trust relation ("u_i trusts u_{i+1}"). If we do not assume the trust relation, the source node u has to consider another method to verify a certificate in which u is neither the issuer nor the user. If u cannot find a certificate-chain from u to v, u cannot verify the public key pk_v. Therefore, the trust relation is necessary to consider certificate-chain-discovery methods.

Definition 3.1. ***Certificate-chain discovery problem in ad-hoc networks*** *Assume that we are given a web-of-trust-type PKI system where every node v has a repository and stores certificates in which v is the issuer or the user. Also assume that we are given a source node and a destination node, then, find a certificate-chain from the source node to the destination node and collect all certificates in the certificate-chain.*

3.3. Related Work

Some authors have proposed PKI systems [16, 17] that limit the issuing of certificates for simplifying the problem in Definition 3.1. However, such methods have the problem that a node cannot always issue certificates based on the trust relation among the nodes.

Kitada, et al. proposed a public key management scheme for ad-hoc networks [13, 14, 15]. Their proposed scheme is able to reduce the communication cost in the certificate revocation phase more than the method proposed by Capkun, et al. [12]. Kitada, et al. also proposed the Ad hoc Simultaneous Nodes Search (ASNS) protocol to resolve the problem stated in Definition 3.1. ASNS finds a certificate-chain as follows.

- The source node broadcasts a search packet p to nodes that the source node directly trusts.
- If a node v receives a packet p, v modifies and sends p as follows.
  - The node v adds its own certificate to the packet p, rewrites the address of p to the nodes that v directly trusts, and broadcasts p to the nodes that v directly trusts.
  - If v directly trusts the destination node, v adds its own certificate to p, rewrites the address of p to the destination node, and sends p to the destination node.

  - If v is the destination node of p, v adds its own certificate to p and sends p to the source node.

- If a node receives more than one packet sent by an identical source node, the node processes only the first packet as above and discards all other packets.

Because each node processes only the first packet, the number of packets per search is proportional to the number of certificates.

However, ASNS has the following shortcomings. In distributed networks such as ad-hoc networks, the protocol is completed, not when the destination node is discovered, but when all nodes in the network receive the packet. Thus, ASNS may have a heavy communication cost because of broadcasting packets with certificates.

3.4. Proposed Method

In this section, a web-of-trust-type PKI system is formally defined to make the certificate-chain discovery problem clear. Based on this model, the problem is divided into two phases. Finally, a new distributed algorithm to solve the problem is proposed.

3.4.1 Web-of-Trust in Ad Hoc Networks

Definition 3.2. ***Trust model***
A model of a web-of-trust-type PKI in ad-hoc networks is a weighted directed graph $N = (V, E, \phi)$, where

- *V is a set of nodes,*
- *E is a set of directed edges, and*
- *ϕ is a weight function that maps each directed edge to a non-negative integer.*

A node v in V represents a node in ad-hoc networks. An edge $u \rightarrow v$ in E represents a certificate from u to v. The weight $\phi(\langle u, v\rangle)$ of an edge $\langle u, v\rangle$

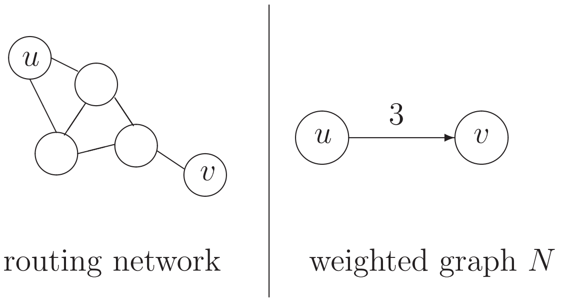

Figure 3.2. Relation between routing network and weighted graph N

represents the number of hops from node u to node v (see Figure 3.2)[1]. For simplicity, we assume that the set V in a trust model equals the set of nodes in the corresponding ad-hoc network.

3.4.2 Basic Scheme

As described in Section 3.3, certificates are added to a search packet in the Kitada method. Thus, all nodes receive a search packet with a number of certificates whether or not a node needs the certificates. In this thesis, the certificate-chain discovery problem is divided into the certificate searching phase and the certificate collecting phase, and a new algorithm for each phase is proposed.

Certificate Searching Phase We assume that each node knows only edges adjacent to the node in N. The problem in this phase is to find a certificate-chain from a given source node to a given destination node. Note that we need not find all certificate-chains; finding one certificate-chain is sufficient for authentication.

To solve the problem in the certificate searching phase, the author uses a distributed algorithm for constructing a spanning tree where the root node is the source node. We can use any distributed algorithm for constructing a spanning tree in a directed graph. The communication complexity of standard algorithms for constructing a spanning tree is $O(|E|)$, where $|E|$ is the number of elements of E [32].

[1] Note that an edge $\langle u, v \rangle$ in a trust model does not mean that node u can directly communicate with node v.

Kitada, et al. showed that the Kitada method can find a certificate-chain in a usual ad-hoc network with topology change by computer simulation [13]. This is because there are some certificate-chain from a given source node to a given destination node. Assume that a given source node u wants to find a certificate-chain to a given destination node v. Usually, there are more than one certificate-chains from u to v in a trust model. In the certificate searching phase, a spanning tree on the trust model is constructd. When an intermediate node in a certificate-chain does not receive a search packet from u, another certificate-chain is found by the distributed algorithm constructing a spanning tree. Hence the probability that the distributed algorithm cannot find any certificate-chain from u to v in the trust model is low.

Certificate Collecting Phase When the certificate searching phase is completed, each node knows which node is the parent in the constructed tree. However, no nodes, including the source node, know about the entire tree while the source node needs to obtain all certificates in a certificate-chain. We can reduce this problem to the problem of collecting all certificates in a path from the source node to the destination node in the tree because there must be such a path in the spanning tree. To solve the problem in this phase, the following method is proposed.

- The destination node sends a packet to the parent node.
- Each intermediate node that received the packet adds its own certificate to the packet and sends it to its parent node.

This process is repeated until the packet reaches the source node. When this process is completed, the source node obtains all certificates in a certificate-chain. Note that the certificate-chain is not a path in the routing graph, but a path in the trust model. Therefore the collecting method can solve the problem with some topology changes. That is, the source node can collect all certificates in the certificate-chain by using the method if every intermediate node is in the ad-hoc network.

3.5. Evaluation

In this section, the communication cost is defined, the costs of the basic scheme and the Kitada method are analyzed, and the costs of the two methods are compared.

3.5.1 Preliminaries

Definition of the Communication Cost In Kitada's work [13], communication cost is defined as the number of packets. This definition does not consider the size of a certificate and the number of certificates in a packet. This definition is not realistic because a packet that includes several heavy certificates is counted as "one packet". Thus, the author defines a more realistic communication cost as follows.

Definition 3.3. ***The communication cost***
Let e be any edge in the directed graph N. The communication cost is defined as follows.

$$\sum_{edge\ e} \{total\ bit\ size\ on\ e \times \phi(e)\}.$$

That is, the communication cost is defined as the total message bits by taking the size of a certificate and the number of certificates in the packet into account.

Cost Tree To estimate the communication cost of the two methods, a *cost tree* is considered in this thesis. A cost tree is a balanced tree such that the root node is a source node, the number of nodes is $2|V|$ where $|V|$ is the number of nodes in a trust model, and the degree of a node is m where m is the average number of certificates issued by the node. This tree represents search packet flows from a given source node to all other nodes on the trust model.

In both of the two methods, a source node u broadcasts a search packet p to nodes that u directly trusts in the certificate searching phase. When a node receives more than one packet sent by an identical source node u, all the packets except the first one are discarded. Even after a destination node has been found, the other nodes do not know it and thus the distributed algorithm for constructing a spanning tree on the trust model does not halt until all the packets are discarded. Therefore the total number of nodes in the cost tree is equal to $2|V|$.

Though we assume a cost tree is balanced, the distributed algorithms in both methods may not construct a balanced tree because an ad-hoc network in the real world is asynchronous. If we assume an unbalanced cost tree instead, the height of the unbalanced tree is larger than a balanced tree with the same number of nodes. Thus, the proposed method outperforms the Kitada method for an unbalanced tree more than a balanced tree because certificates are attached to search packets. Therefore we assume a balanced tree as the cost tree so that comparison of the two methods is not disadvantageous to the Kitada Method. The number of nodes $|V|$ and the number of edges $|E|$ (i.e., the number of certificates) in a trust model can be represented by using the height k of a cost tree and the average number m of the node degree in the cost tree as follows:

$$\begin{aligned} 2|V| &= \frac{m^k - 1}{m - 1}, \\ |E| &= m \times |V|. \end{aligned}$$

3.5.2 Analysis of the Kitada Method

In this section, the Kitada method is examined. Though they analyzed their method [13], it was based on a communication cost that did not consider the size of a packet. The author first divides the method into two phases and analyzes each of the two phases by using Definition 3.3 to compare the method and the proposed method.

We can consider ASNS to be a distributed algorithm constructing a spanning tree in which certificates are added to a search packet. The length of a certificate-chain is the number of edges from the source node to the destination node. On the other hand, the height of the spanning tree must be at least the length of the chain because the chain is also a path in the tree, and the height may be longer than the chain because the distributed algorithm does not halt even if a chain is discovered. That is, the following relation holds between the length and the height:

$$
\begin{aligned}
& \text{(the length of a certificate chain in a spanning tree)} \\
\leq\ & \text{(the height of the tree).}
\end{aligned}
$$

We assume that the length of a certificate chain equals the height of the tree, i.e., the communication cost is estimated based on the upper bound of the length. This assumption was also used in Section 3.5.3.

Certificate Searching Phase In this phase, the source node broadcasts a search packet to all nodes that the source node directly trusts. When a node receives the packet, the node adds its own certificate to the packet and broadcasts it to all nodes that the node directly trusts. A packet is transmitted until a node receives the same packet twice. Then, the communication cost in this phase $S_1(k)$ is given by the following equation, where n is the average number of hops, m is the average number of degrees of nodes in N, $Cert$ is the size of a certificate, k is the height of the constructed spanning tree, and $Cert\_req$ is the packet size of a certificate search packet.

$$S_1(k) = n \sum_{i=1}^{k} \{Cert(i-1) + Cert\_req\} m^i.$$

Certificate Collecting Phase When the destination node receives a packet with certificates, the node adds its own certificate to the packet and sends it back to the source node. The destination node sends back k certificates to the source node because the number of certificates in this packet is equal to the length of a certificate-chain. Therefore, the cost $C_1(k)$ is given by the following equation, where $Cert\_rpl$ is the size of a replying packet:

$$C_1(k) = n(k \times Cert + Cert\_rpl). \tag{3.1}$$

3.5.3 Analysis of the Basic Scheme

Certificate Searching Phase In the basic scheme, a source node constructs a spanning tree using any distributed algorithm, and no certificates are added to a packet. Therefore, the cost $S_2(k)$ is as follows:

$$S_2(k) = n \sum_{i=1}^{k} Cert\_req \times m^i.$$

Certificate Collecting Phase When the destination node receives a search packet, each node in the tree knows which node is the parent node. The destination node sends the packet to the parent node, and each intermediate node receiving the packet adds its own certificate and sends it to the parent node. Thus, the source node receives a packet with $(k-1)$ certificates. The cost $C_2(k)$ is as follows:

$$C_2(k) = n \sum_{i=1}^{k} \{Cert(i-1) + Cert\_rpl\}.$$

3.5.4 Comparison between the Two Methods

Complexity Analysis The fraction of the communication cost of the two methods is analyzed. The total cost of the Kitada method is equal to $(S_1(k) + C_1(k))$ and the total cost of the basic scheme is equal to $(S_2(k) + C_2(k))$:

$$\begin{aligned} \frac{S_1(k) + C_1(k)}{S_2(k) + C_2(k)} &= \frac{O(k \cdot Cert \cdot m^{k+2})}{O(m^{k+1})} \\ &= O(k \cdot Cert \cdot m). \end{aligned}$$

This tells us that the cost of the Kitada method is $O(k \cdot Cert \cdot m)$ times the cost of the basic method.

Numerical Analysis The author also compares the costs by numerical analysis. Let be $n = 4$ and $m = 4$. Kitada, et al. estimated that the average number of hops (n) for realistic ad-hoc networks is at most around four. In the Kitada's paper, they empirically showed that a trust model is possibly not strongly connected if $m < 4$. Hence only $m \geq 4$ are considered. Note that since a trust model is given independently of the trust establishment algorithm, we should compare the proposed method with the Kitada method with an identical value for m. Because the ratio of the cost of the Kitada method to ours is in proportion to m as shown in the above paragraph, $m = 4$ is the most advantageous setting for the Kitada method. We assume that a trust model is strongly connected to evaluate the entire phase of the two methods. In a realistic world, a trust model may not be strongly connected. However, if a trust model is not strongly connected, there is no certificate chain between a pair of two nodes in different connected components. Therefore, if we considered such a trust model, we cannot evaluate the cost of the certificate collecting phase. Note that the costs of the certificate searching phase in the two methods do not depend on whether the destination node is found or not because the both methods are distributed algorithms.

In the RSA public key cryptosystems, the size of a public key is 1024bits and the size of each cipher text (or signature) is more than 1024bits. Therefore the size of a certificate is more than 2048bits[2]. Therefore, we let $Cert = 2050$.

Also, we let $Cert\_req = 100$ and $Cert\_rpl = 100$. We assume the size of each ID as 10bits. These 10bit length IDs can identify $2^{10} = 1024$ nodes, which is large enough for an ordinary ad-hoc network. Each search packet $Cert\_req$ includes a source node, destination node, and pairs of the next node on the trust model and the next hop on the routing graph [13]. The number of pairs of the next node and the next hop is not a constant and depends on the number of certificates issued by each node (the degree of a node in a trust model). According to Kitada's estimation that the average degree of nodes to construct a web-of-trust is four [13], we assume that the average number of pairs of the next node and the next hop in a search packet is four. Therefore, the average size of a search packet is $10 + 10 + 4 \times (10 + 10) = 100$. On the other hand, each reply packet $Cert\_rpl$ may be less than 100bits. However, the basic scheme outperforms the Kitada method for less than 100bits more than it does for 100bits because $Cert$ divided by $Cert\_rpl$ is larger. Therefore, we assume that $Cert\_rpl$ is also 100bits so that the comparison of the two methods is not disadvantageous to the Kitada Method.

Figure 3.3 shows the costs of the searching phase. The cost of the basic scheme is lower than the cost of the Kitada method. This is because the searching phase

[2] In the real world, a certificate includes ID of the issuer and owner of a public key, a timestamp of expiration date and so on, and the size of a certificate may become larger than 2050bits.

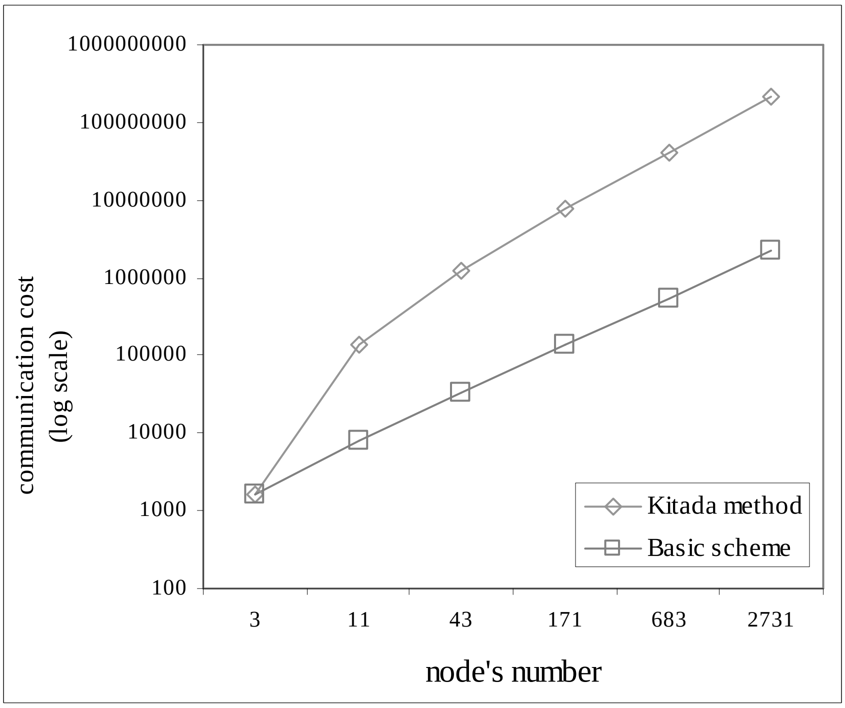

Figure 3.3. Basic scheme vs. the Kitada method (searching cost)

in the basic scheme broadcasts search packets without adding certificates. Figure 3.4 shows the graph of the cost of the collecting phase. In the collecting phase of the Kitada method, the destination node sends packets to the source node. On the other hand, we have to collect certificates from the destination node to the source node while sending back the packet along with the certificate-chain in the proposed method. Thus, the cost of the basic scheme is higher than the Kitada method. Finally, Figure 3.5 compares the total costs of the two methods. We can see that the basic scheme has a lower cost than the Kitada method.

As a comparison in a more realistic environment, the author also compared the proposed method and the Kitada method by computer simulation in Section 3.7. Moreover, according to the Kitada's definition of the communication cost, the author analyzed the packet numbers of the proposed method and the Kitada method by numerical analysis in Section 3.8.2.

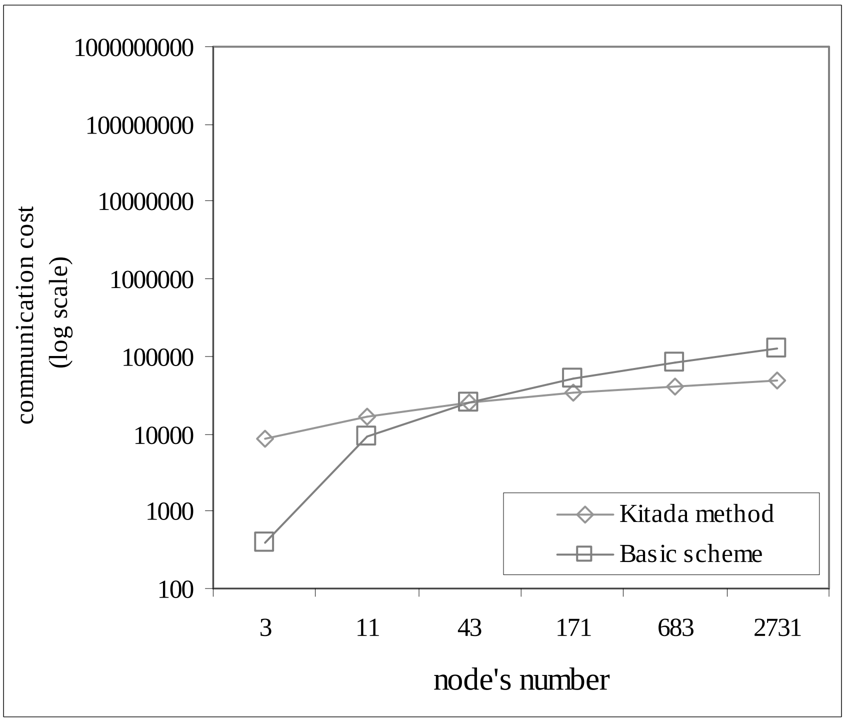

Figure 3.4. Basic scheme vs. the Kitada method (collecting cost)

3.6. A Modification Scheme of the Basic Scheme

The basic scheme has a disadvantage on the cost of the collecting phase. To reduce the cost, the collecting phase of the basic scheme is revised.

3.6.1 Modification Scheme

In the basic scheme, the source node obtains all certificates in a certificate-chain by making each intermediate node add its certificate to the replying packet. This scheme requires extra cost because the packet from the destination node runs through the whole chain, expanded with the added certificates. To avoid this overhead, the phase is modified as follows:

- The destination node sends a packet to the parent node in the certificate-chain.

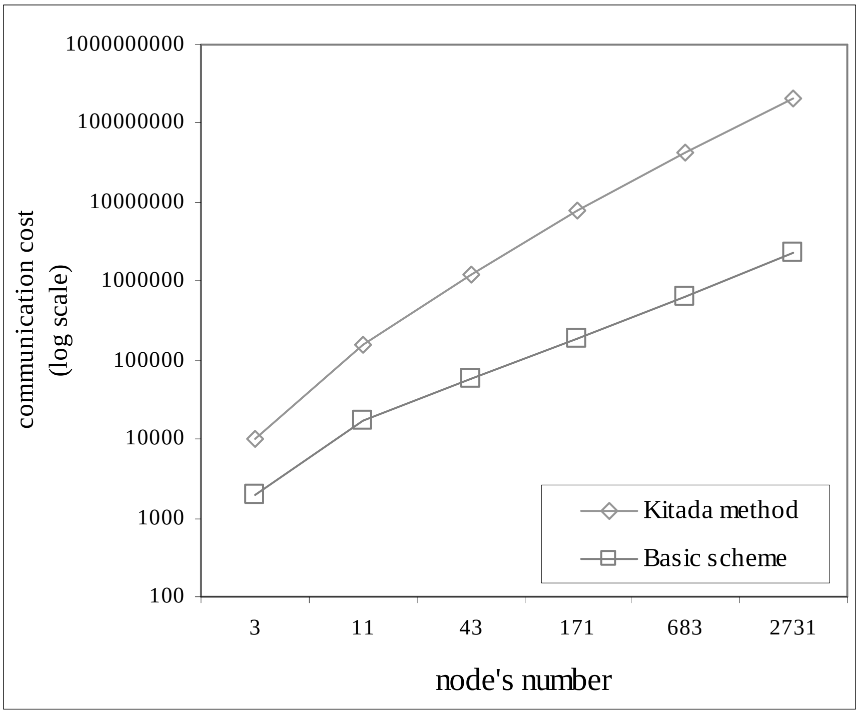

Figure 3.5. Basic scheme vs. the Kitada method (total cost)

- Each intermediate node receiving the packet sends its certificate directly to the source node and also sends the packet to the parent node (because no node knows whether the node itself is on the certificate-chain to the destination node).

The whole algorithm is called a *modification scheme* in this chapter. As same as the basic scheme, the source node can collect all certificates in the certificate-chain by using the method if every intermediate node is in the ad-hoc network.

3.6.2 Evaluation of the Modification Scheme

After this modification, the cost $S_3(k)$ of the certificate searching phase is the same as $S_2(k)$, and the cost $C_3(k)$ of the collecting phase is as follows.

$$C_3(k) = (k-1) \times n \times (Cert + Cert\_rpl). \tag{3.2}$$

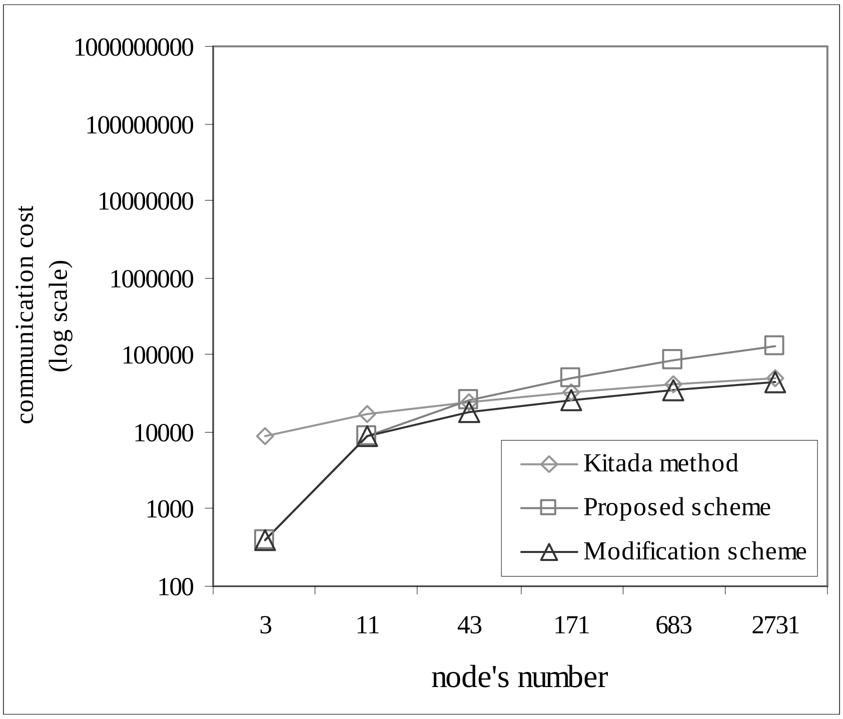

Figure 3.6. Comparison of collecting costs

In the modification scheme, the source node can collect all certificates in the certificate-chain if every intermediate node is in the ad-hoc network.

The author also investigated another modification in his previous work [33]. An idea of the other modification is to use a distributed algorithm for constructing a shortest path tree such as the distributed Bellman-Ford algorithm in the searching phase. However, the upper bound of the computation time for the distributed Bellman-Ford algorithm is $O(V \cdot E)$ [32] and the one for the standard distributed Dijkstra algorithms is $O(V^2)$ [34]. Because these costs are much higher than the costs of algorithms for spanning trees, the method using a distributed algorithm for constructing a shortest path tree require more total cost than the basic scheme and the Kitada method.

Numerical Analysis The above methods are compared by numerical analysis. We let $n = 4$, $m = 4$, $Cert = 2050$, and $Cert\_req = Cert\_rpl = 100$.

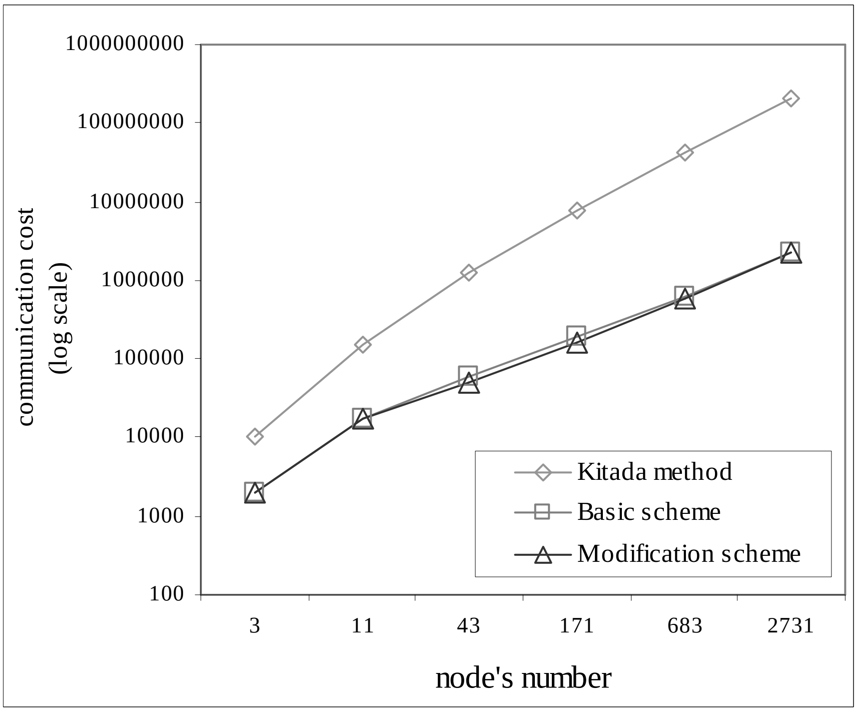

Figure 3.7. Comparison of total costs

The cost of the modification scheme is the same as the cost of the basic scheme and is lower than the Kitada method (see Figure 3.3).

Figure 3.6 shows a graph of the cost of the collecting phase. The modification method also has a lower cost than the Kitada method in this phase.

In Figure 3.7, the total costs of all the above methods are compared. From this comparison, we obtain the following result:

$$\begin{aligned} & S_1(k) + C_1(k) && \text{(the Kitada method)} \\ > \ & S_2(k) + C_2(k) && \text{(the basic scheme)} \\ > \ & S_3(k) + C_3(k) && \text{(the modification scheme).} \end{aligned}$$

Note that for (3.1) and (3.2), $C_1(k) \leq C_3(k)$ if and only if $k \geq \frac{Cert}{Cert\_rpl} + 2$. We assumed that $Cert = 2050$ and $Cert\_rpl = 100$ and hence $C_1(k) < C_3(k)$ if $k > 22$. Figure 3.6 conforms to this result. The number of nodes in a network is already more than 100,000 when $k = 9$ and is out of range in Figure 3.6. The result of the numerical analysis is summarized in Table 3.1.

Table 3.1. The proposed schemes vs. the Kitada method

Method Name	**Search**	**Collect**	**Total**
Basic	✓		✓
Modification	✓	✓†	✓

Note : "✓" means that the cost is lower than the Kitada method.

† $C_1(k) > C_3(k)$ if and only if $k < \frac{Cert}{Cert\_rpl} + 2$.

3.7. Simulation Results

The numerical analysis in Section 3.6 showed that the modification scheme requires the least communication cost among the existing and proposed methods. In this section, the author compares the modification scheme and the Kitada method more precisely for moderate-size networks. The author describes the simulation scenarios, and then shows the average weight in the trust model and the communication costs of the two methods. All simulations were performed with a simulator implemented in Java with a java library of graph algorithms and optimization [35].

3.7.1 Simulation Scenario

The following simulation scenarios are used.

- The number of nodes $|V|$ are 20, 40, 60, 80, and 100.
- The number of certificate $|E|$ is $4 \times |V|$.
- The power range of each node is 100.
- The simulation regions are as follows.
  - 100×100 ($20 \leq |V| \leq 100$),
  - 200×200 ($20 \leq |V| \leq 100$),
  - 300×300 ($20 \leq |V| \leq 100$),
  - 400×400 ($40 \leq |V| \leq 100$),

  - 500×500 ($60 \leq |V| \leq 100$),
  - 600×600 ($n = 100$).

- The routing graph is a *unit disc graph.* Formally, a unit disc graph is the intersection graph of a set of unit diameter closed disks in the plane [36]. Generally, a unit disc graph is not always connected, which means that an ad-hoc network itself is not formed. Because this section focuses on the cost of the two methods, connected unit disc graphs are used.

- The trust model is a *random Hamilton graph* where each weight of an edge $\langle u, v \rangle$ is defined as the shortest path length from i to j in the unit disc graph (see Definition 3.2). Kitada, et al. assume a trust model as a strongly connected graph in simulation scenario, so a random Hamilton graph is used as an example of such graphs.

- The packet size is 100bits,

- The certificate size is 2050bits.

3.7.2 Results

Average Number of Hops Figure 3.8 shows the average number of hops in a unit disc graph. The number of hops is evaluated by computing the average of the shortest hops between arbitrary pairs of distinct nodes in a unit disc graph. When the simulation region is 500×500, the number of hops nearly equals four, which matches the number of hops ($m = 4$) adopted in the numerical analysis (see Section 3.5.4 and 3.6.2).

Comparison of the Two Methods Figure 3.9–3.13 show the communication costs of the modification scheme and the Kitada method by simulations. The modification scheme outperforms the Kitada method also in the simulation results. Below detailed comparisons are given.

Figure 3.9 and 3.10 show the costs of the searching phase. The cost of the modification scheme is much lower than the cost of the Kitada method. In the searching phase of both methods, a source node constructs a spanning tree. For

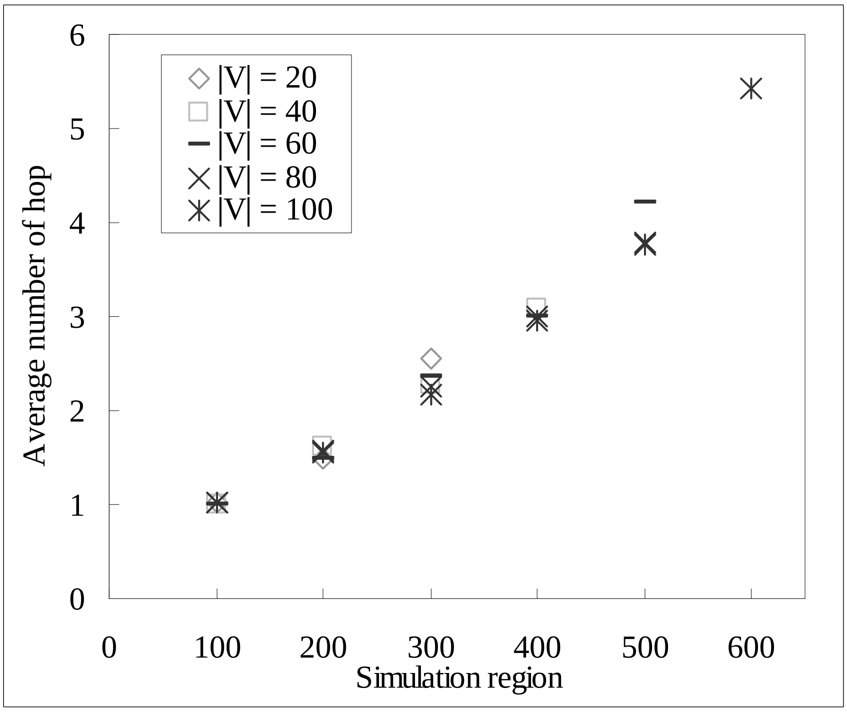

Figure 3.8. Average number of hops in a unit disc graph

simplicity, a shortest path tree is used for the constructed spanning tree. A constructed spanning tree is not always a shortest path tree because of node failure or non-uniform communication delay. However, the probability of not constructing a shortest path tree is not high. Furthermore, whether we use a shortest path tree or not does not largely affect the comparison result of the two methods. Figure 3.11 and 3.12 show the costs of the collection phase. Also for this phase, the costs are evaluated by using a shortest path tree. Figure 3.13 shows the total costs of the two methods. The average of the total cost is evaluated as the sum of the average searching cost and the average collecting cost. The simulation results showed that the total cost of the modification scheme is less than 10% of the cost of the Kitada method.

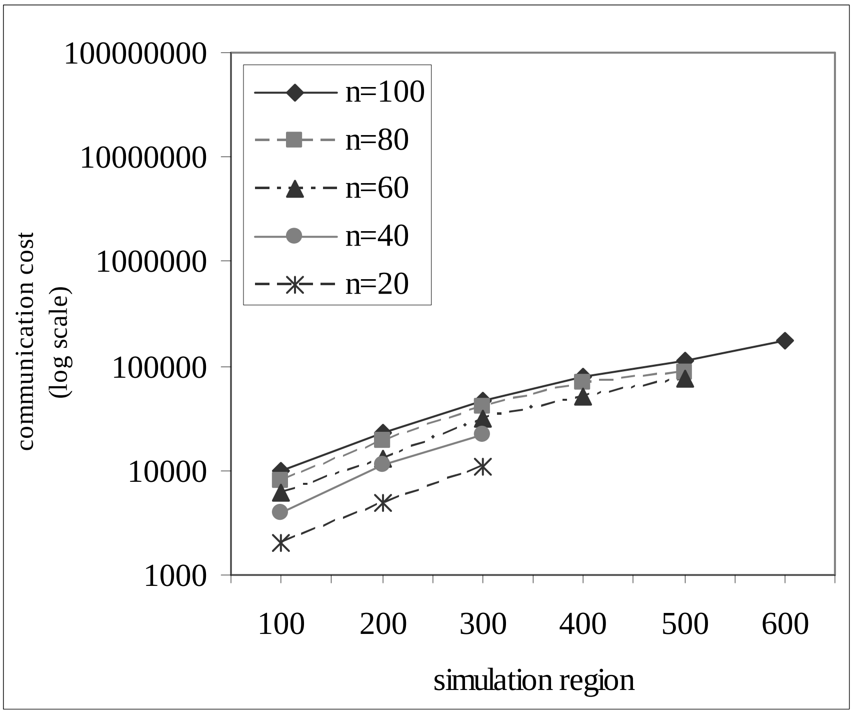

Figure 3.9. Average searching cost of the Modification scheme

3.8. Discussion

3.8.1 Security Consideration

In this section, the robustness of the proposed methods to some known attacks are discussed.

Sybil attack In a distributed network without a trusted third party maintaining identities (ID), a malicious node can have not only a legitimate ID but also one or more counterfeit IDs. This attack is called the *Sybil attack* [37]. If a node succeeds in the Sybil attack, it can improperly raise its ranking in a reputation system [38] by voting itself using counterfeit IDs, for example.

In this thesis, two variations of the Sybil attack are considered. One is that a malicious node obtains one or more counterfeit IDs but it has only one (legitimate) public key (and its corresponding secret key). The other

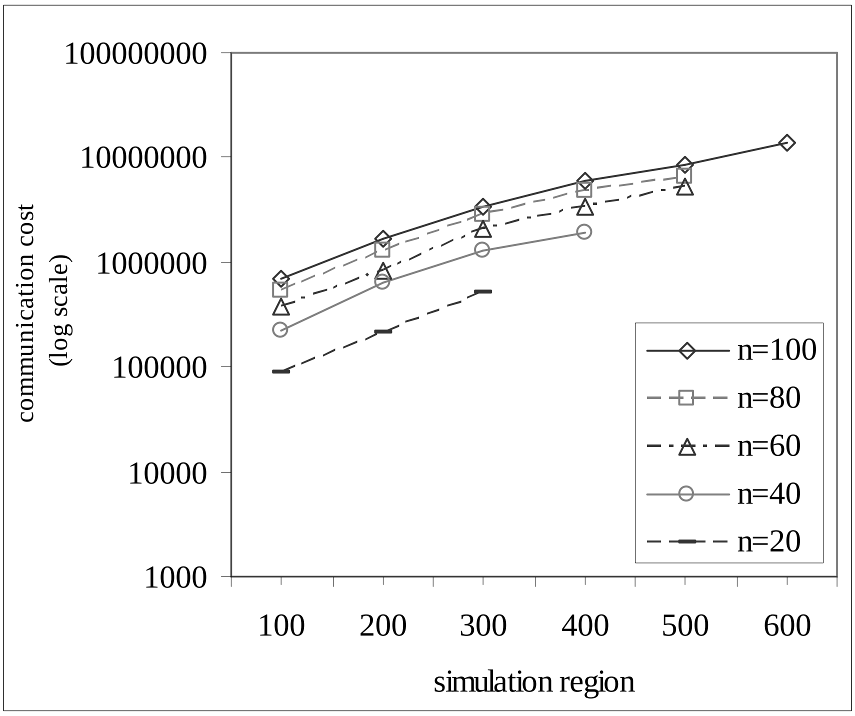

Figure 3.10. Average searching cost of the Kitada method

is that a malicious node obtains both of counterfeit IDs and public keys. The former case is easy to treat. Remember that a certificate binds the ID and the public key of a user. Hence, if we find two certificates that bind different IDs with the same public key, we know that the user of the public key is malicious. For the latter case, it is generally impossible to find out whether or not the Sybil attack occurs since no node can always know the physical relation between a public key and its owner. However, the proposed methods can find a certificate-chain correctly as long as every node, including the malicious node of the Sybil attack, faithfully relays packets. Also, the public-key exchange and the creation of a certificate are usually done through a side channel (e.g., over an infrared channel at the time of a physical encounter) before constructing an ad-hoc network [12]. Therefore the threat of the latter type of Sybil attack is not serious.

Man-in-the-middle attack When nodes *A* and *B* exchange their public keys

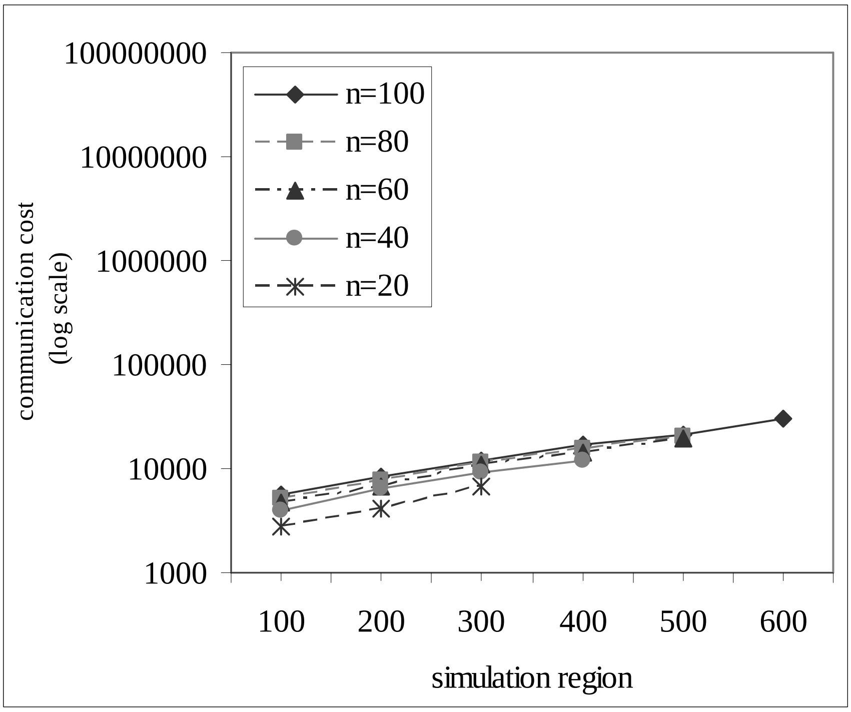

Figure 3.11. Average collecting cost of the Modification scheme

pk_A and pk_B via a key exchange protocol, a malicious intermediate node M may be able to give A pk_M instead of pk_B and give B pk_M instead of pk_A. If this attack succeeds, M can eavesdrop on communications between A and B. However, in conventional PKI systems, a certificate $\langle u, v\rangle$ is issued if u trusts v to be honest and has correctly authenticated v as the owner of its public key before signing it [31]. Moreover, in web-of-trust-type PKI systems, the source node can verify the public key of the destination node by a certificate-chain. Therefore, web-of-trust-type PKI systems including the proposed methods can prevent the man-in-the middle attack.

Also, when the public key exchange between A and B is completed, this attack would be foiled with public keys because the man-in-the-middle would not have the private key to be able to decrypt messages encrypted under A's public key [39].

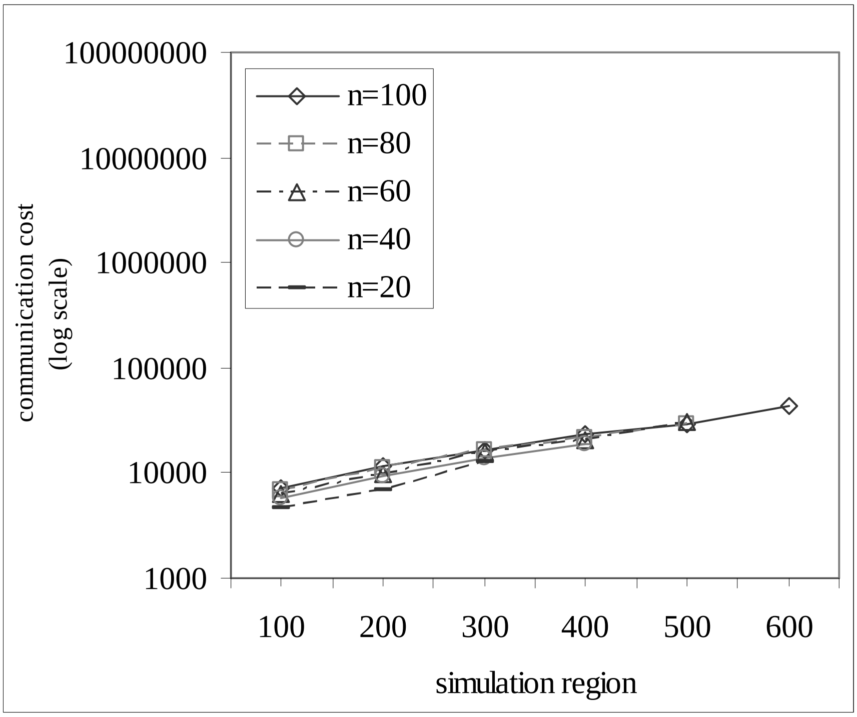

Figure 3.12. Average collecting cost of the Kitada method

Denial of service attack Denial of service (DoS) attacks disable routing protocols of ad-hoc networks, and some authors have investigated efficient DoS attacks for ad-hoc networks [40]. The main aim in this chapter is to get the public key of another node securely and to prevent the eavesdroppings of malicious nodes, so the resilience against DoS attacks is out of the scope of this thesis. However, a perfectly reliable routing on ad-hoc networks are not assumed in this thesis, i.e., the proposed methods work even if a routing is unreliable.

No single security system can prevent all kinds of attacks. Generally, we combine several security techniques against these attacks. The main aim of this chapter is to provide a secure *End-to-End encryption* [39]. In the End-to-End encryption, the sender of a message and its receiver should have a shared key or the sender should have the public key of the receiver. Because a message is encrypted by a shared key between the sender and the receiver, even an intermediate host

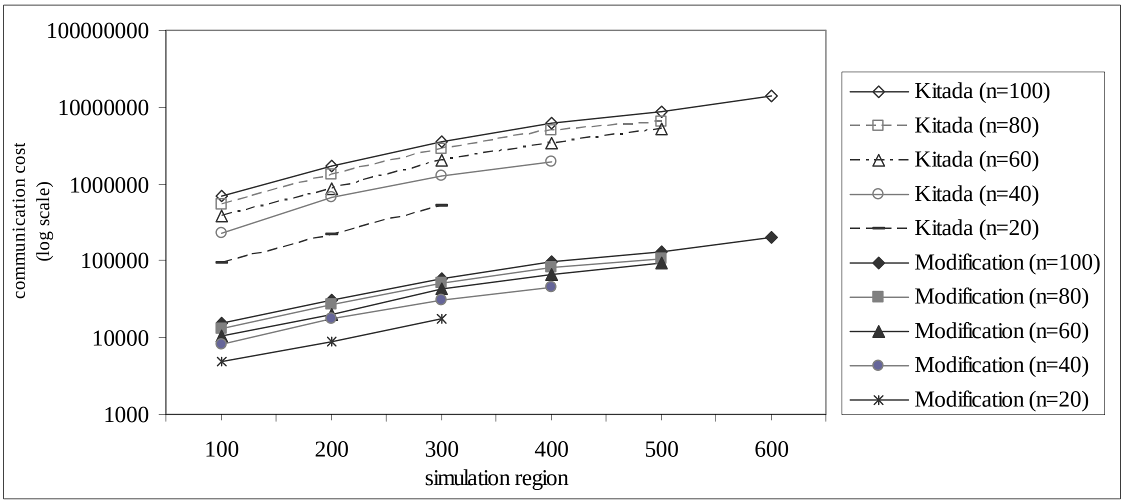

Figure 3.13. Modification scheme vs. the Kitada method (total cost)

cannot decrypt it. This encryption scheme does not depend on routing protocols, so we usually can use some key management and exchange protocol on routing protocols. On the other hand, there is *Link encryption* [39] such that the sender and the receiver and each intermediate node should share a key. Because every block of a packet, not only the data part but also the header part, is encrypted in the Link encryption, each intermediate node has to decrypt the packet to check the header part and encrypt it before sending it to next node. In this encryption, each intermediate node can read not only the header part but also the data part because each intermediate node has the shared key and can decrypt the packet. In this chapter, the author investigates a secure End-to-End encryption scheme in ad-hoc networks, where we should use keys and key management schemes distinct from those in the Link encryption for security. Though the Link encryption is also important, this thesis do not consider it.

3.8.2 Analysis of Packet Numbers of the Three Methods

The communication costs of the basic scheme, the modification scheme and the Kitada method are compared by numerical analysis in Section 3.5. In this thesis, the cost is defined as the total bit size transmitted while a certificate-chain discovery method is performed. In [13] Kitada, et al. estimated the number of packets. Though the cost defined in this thesis is more realistic estimation than that defined in [13], some readers may be interested in analysis of the total packet numbers of the three methods. Thus, the total packet numbers of the three methods are analyzed in this section. The number of packet depends on the *Maximum Transmission Unit* (MTU) that decides the maximum quantity of data per transmission. Therefore, in this thesis, the definition of the number of packets in [13] are generalized, and the number of packets of each methods is analyzed according to the generalized definition.

We let MTU be size of the MTU[3]. By using MTU, each number of packets of the three methods is represented as follows.

Analysis of the Kitada Method

$$\begin{aligned} S'_1(k) &= n\sum_{i=1}^{k}\left\lceil\frac{Cert(i-1)+Cert\_req}{MTU}\right\rceil \times m^i, \\ C'_1(k) &= n(k \times \left\lceil\frac{Cert+Cert\_rpl}{MTU}\right\rceil). \end{aligned}$$

Analysis of the Basic Scheme

$$\begin{aligned} S'_2(k) &= n\sum_{i=1}^{k}\left\lceil\frac{Cert\_req}{MTU}\right\rceil \times m^i, \\ C'_2(k) &= n\sum_{i=1}^{k}\left\lceil\frac{Cert(i-1)+Cert\_rpl}{MTU}\right\rceil. \end{aligned}$$

Analysis of the Modification Scheme

$$\begin{aligned} S'_3(k) &= S'_2(k) \\ C'_3(k) &= (k-1) \times n \times \left\lceil\frac{Cert}{MTU}\right\rceil + k \times n \times \left\lceil\frac{Cert\_rpl}{MTU}\right\rceil. \end{aligned}$$

[3] Note that the size represents the number of bit in this thesis though the number of byte is used generally.

Comparison among the Three Methods Packet numbers of the three methods are compared by numerical analysis. We let $n = 4$, $m = 4$, $Cert = 2,050$, and $Cert\_req = Cert\_rpl = 100$. The largest MTU value allowed by Ethernet is 1500byte (= 12000 bit) packet while the smallest MTU value is 128byte (= 1,024 bit). Therefore, This thesis compare the number of messages of the three methods with $MTU = 12,000$ and $MTU = 1,024$.

Table 3.2 shows packet numbers of the searching phase. The packet numbers of the proposed schemes are smaller than the packet number of the Kitada method with $MTU = 12,000$ while the packet numbers of the three methods are same value for each k with $MTU = 1,024$. Generally, there is a following relation between the Kitada method and the proposed schemes.

$$\begin{cases} S'_1(k) = S'_2(k) & \text{if } \left\lceil \frac{(k-1)\times Cert}{MTU} \right\rceil = 1 \\ S'_1(k) \gg S'_2(k) & \text{otherwise} \end{cases}$$

Thus $S'_1(k) \geq S'_2(k)$ for any MTU and k.

In Table 3.3, the packet numbers of the collecting phase are shown. The packet numbers of the proposed schemes are larger than that of the Kitada method.

Finally, in Table 3.4, the total packet numbers of the three methods are compared. The packet numbers of the proposed schemes are smaller than that of the Kitada method with $MTU = 12,000$ since $|S'_1(k) - S'_2(k)|$ is much bigger than $|C'_1(k) - C'_2(k)|$ (or $|C'_1(k) - C'_3(k)|$). On the other hand, with $MTU = 12,000$, the packet numbers of the proposed schemes are slightly bigger than that of the Kitada method. That is, the result of the comparison of the three methods depends on the value of $\lceil \frac{(k-1)\times Cert}{MTU} \rceil$. If $\lceil \frac{(k-1)\times Cert}{MTU} \rceil > 1$, the packet numbers of the proposed schemes are smaller than that of the Kitada method. On the other hand, if we assume sufficient amounts of the MTU value or a small size certificate so that we can include many certificates in one packet, the total packet number of the Kitada method is smaller than that of the proposed schemes. However, in a realistic world, an ad-hoc network consists of heterogeneous mobile devices with several MTU values. Also, every node has to agree a protocol to separate certificates contained in one packet before they form an ad-hoc network. Therefore, even if we assume $MTU = 12,000$ or a small size certificate, the Kitada method may not be better choice than the proposed methods.

3.9. Conclusion of Chapter 3

In this chapter, the author modeled web-of-trust-type PKI systems, formally defined the certificate-chain discovery problem, and proposed a new distributed algorithm as well as a modification for solving the problem. Furthermore, a measure for the communication cost is proposed, and according to the measure, the proposed algorithms are compared with the Kitada method by numerical analysis. To evaluate the performance of the modification scheme in a more realistic environment, it and the Kitada method are evaluated by simulation. The simulation results showed that the modification scheme requires a lower cost than the Kitada method.

Unfortunately, existing routing protocols for ad-hoc networks are unable to catch up with frequent link changes [8]. These protocols minimize the effect of the dynamic change of the topology caused by nodes' mobility by reducing time, communication, and round complexity. The proposed methods also address node mobility by reducing such complexities as existing routing protocols do.

Table 3.2. Comparison of packet numbers in certificate searching phase

k	$\|V\|$	$MTU = 1,024$		$MTU = 12,000$	
		$S'_1(k)$	$S'_2(k) = S'_3(k)$	$S'_1(k)$	$S'_2(k) = S'_3(k)$
1	3	16	16	16	16
2	11	208	80	80	80
3	43	1,488	336	336	336
4	171	8,656	1,360	1,360	1,360
5	683	45,520	5,456	5,456	5,456
6	2,731	225,744	21,840	21,840	21,840

Table 3.3. Comparison of packet numbers in certificate collecting phase

k	$\|V\|$	$MTU = 1,024$			$MTU = 12,000$		
		$C'_1(k)$	$C'_2(k)$	$C'_3(k)$	$C'_1(k)$	$C'_2(k)$	$C'_3(k)$
1	3	12	4	4	4	4	4
2	11	20	16	20	4	8	12
3	43	28	36	36	4	12	20
4	171	36	64	52	4	16	28
5	683	44	100	68	4	20	36
6	2,731	52	144	84	8	24	44

Table 3.4. Comparison of the total packet numbers

k	$\|V\|$	$MTU = 1,024$			$MTU = 12,000$		
		$S'_1(k) + C'_1(k)$	$S'_2(k) + C'_2(k)$	$S'_3(k) + C'_3(k)$	$S'_1(k) + C'_1(k)$	$S'_2(k) + C'_2(k)$	$S'_3(k) + C'_3(k)$
1	3	28	20	20	20	20	20
2	11	228	96	100	84	88	92
3	43	1,516	372	372	340	348	356
4	171	8,692	1,424	1,412	1,364	1,376	1,388
5	683	45,564	5,556	5,524	5,460	5,476	5,492
6	2,731	225,796	21,984	21,924	21,848	21,864	21,884

Chapter 4

Key Predistribution Schemes for Sensor Networks Using Finite Plane Geometry

4.1. Introduction

In this chapter, key predistribution schemes for two neighbor sensor nodes (nodes that can directly communicate with each other) are investigated. The primal objective of this chapter is to construct a better implementation of the key predistribution scheme than the random key distribution scheme of [19]. Three quantitative measures to evaluate key predistribution schemes are considered; the node *memory size*, the *connectivity* and the *key-survivability.* The node memory size (or *memory size* for simplicity) is the number of keys that a node needs to remember, the *connectivity* is the probability that randomly chosen two sensor nodes have one or more keys in common, and the *key-survivability* is the probability that the key which has been agreed by two nodes stays secure even if an intruder mounts the *node capture attack*, in which an intruder captures sensor nodes deployed in a field, and retrieves keys embedded in the nodes. There are obvious tradeoff relations between the three quantitative measures. For example, if we try to increase the connectivity without changing the memory size, then the key-survivability decreases accordingly in general. A key predistribution scheme needs to provide good tradeoff points with respect to the three measures, and it is preferred that the tradeoff points are controllable. The random key distribution scheme is, however, not good with respect to these issues, because the scheme is so simple that we have little way to control its performance.

In this chapter, the author considers to actively control keys to be embedded in sensor nodes so that both the connectivity and the key-survivability are larger than the random key distribution scheme. For this sake, a method using simple geometric properties of lines and points is considered. The proposed methods associate each node with a line over a two-dimensional finite plane, and manage keys so that two nodes can agree a key if and only if the associated lines intersect with each other. Two randomly chosen lines intersect with each other unless they are parallel, thus two nodes succeed in key agreement with high probability. A scheme that is directly based on this idea is proposed (called a *basic scheme*). The basic scheme can realize high connectivity with small memory size, but it is difficult to flexibly control the tradeoff points in general. To overcome this problem, two extensions of the basic scheme are also proposed. The extended schemes have more flexibility than the basic scheme, though they are not as advantageous as the basic scheme compared to the random key distribution scheme. The basic and extended schemes is analytically evaluated, and the result is compared with that of the random key distribution scheme.

4.2. Related Work

A naive scheme for key agreement in sensor networks is to use a global key. The key manager determines a unique “global key”, and distributes the global key to all the sensor nodes. A node can agree a key with any other node in the system and therefore the connectivity is 1. On the other hand, the key-survivability of the scheme is quite poor because an intruder can obtain all the secret of the system by capturing just one node. Another straightforward scheme is to assign different keys for different pairs of nodes, and provide a sensor node with the keys which the node commits to. This scheme gives high key-survivability, but the memory size increases as the network size (the number of nodes in the network) increases. Usually a sensor network consists of many sensor nodes which have small memory size, and therefore this scheme is not suitable for sensor networks.

In the random key distribution scheme [19], as briefly introduced in Section 1, the connectivity and the key-survivability are controlled by two parameters; the size of the key pool K and the node memory size m (the number of keys stored in a node). We can show easily that the connectivity and the key-survivability (under the situation that c nodes are captured) of the random key distribution scheme are given as

$$\begin{aligned} p_{\text{con}} &= 1 - \prod_{i=0}^{m-1} \frac{|K| - m - i}{|K| - i}, \\ p_{\text{sur}_c} &= (1 - \frac{m}{|K|})^c, \end{aligned}$$

respectively. The random key distribution scheme is sometimes used as a "building block" of more sophisticated key predistribution schemes. For example, Chan, et al. considered to improve the trade-off points of the random key distribution scheme by making use of multiple keys which are shared by two nodes [41], but they basically used exactly the same random key distribution scheme in its key distribution phase. The scheme proposed by Du, et al. [1] also used the random key distribution scheme partly.

There are several studies which take different approaches from the random key scheme. One major approach among them is to make use of the deploy location of sensor nodes. In the random key approach, it is implicitly assumed that we cannot predict where a sensor node is deployed. In some applications, however, this is not the case. We may be able to statistically predict the deploy location of each sensor node. Such statistical information is helpful to realize secure and efficient key agreement, as investigated in [42, 43, 44, 45]. The relation between the location-based and the location-free approaches will be discussed in Section 4.5.2.

There also exist studies in which much emphasis is devoted to establishing *path-keys*. Assume that two nodes, say a and b, fail to agree a cryptographic key. In this case, we can ask other nodes to help a and b setting up a key: Determine a sequence of nodes $a = n_0, n_1, \dots, n_{r-1}, n_r = b$ so that n_i and n_{i+1} with $0 \le i \le r-1$ can agree a key, and ask the intermediate nodes n_i to relay secret information which enables key agreement between a and b. The key established between a and b in this way is called a *path-key*. Liu proposed in [46] a scheme which makes use of bivariable polynomials. The scheme has relatively low connectivity, but high probability that randomly chosen two nodes succeed to agree a path-key. Du, et al. considered similar approach [47], but they used the Blom's scheme [48] instead of bivariable polynomials. To the authors' understanding, a path-key cannot be an alternative to the *link-key* which is directly agreed between two nodes. Computing a path-key requires sensor nodes to pay large computational and communication overheads, and its security will be seriously damaged if there are selfish or dishonest nodes in the network. Consequently it is quite misleading to compare the connectivity wit respect to link-keys and the connectivity with respect to path-keys.

Other well-known studies on the key management in sensor networks include SPINS by Perrig, et al. [49]. SPINS is a collection of protocols for key management between sensor nodes and a base station, and does not intend for the key agreement between two sensor nodes. LEAP is another widely known scheme for key management in sensor networks [50]. LEAP is a general mechanism which allows a group of nodes to agree a key, but the security of LEAP strongly depends on an uncommon assumption; a node is equipped with a precise timer, a (global) master key, and a special mechanism which erases the master key at a certain timing. The node makes use of the master key to share cryptographic keys with other nodes, and then erases the master key to avoid troubles in case the node is captured. If the key-erasing mechanism does not work as expected and an intruder happen to capture a node in which the master key remains not-erased, then the entire system might be paralyzed completely.

Light mechanisms for key agreement have been studied for long years. For example, Matsumoto and Imai have proposed the concept of the Key Predistribution Scheme (KPS) in late 80s [51], mainly intended for applications in smart-cards. The KPS itself is a general concept, and [51] gives an implementation of KPS which utilizes matrix computation. The concept of KPS is also effective for sensor networks, but concrete implementations of KPS which is suitable for sensor networks is still an open problem. Gong and Wheeler have proposed a key management scheme that is based on algebraic geometry [52]. The idea to use lines and point is similar to this work, though, the used techniques and the obtained results of [52] are quite different from ours. Some characteristics of the scheme in [52] are problematic in sensor network applications. For example, the scheme in [52] cannot support a large network that consists of nodes with quite small memory size.

4.3. Proposed Scheme

4.3.1 Preliminary

Let p be a prime number, $Z_p = \{0, \ldots, p-1\}$ and $Z_p^2 = \{(x,y)|x, y \in Z_p\}$. We call Z_p^2 a *plane* and elements in Z_p^2 *points*. For a and b in Z_p, a *line* is a collection of points defined as $l(a,b) = \{(x,y)|x \in Z_p, y = ax + b \pmod{p}\}$. We can easily show the following lemma.

Lemma 4.1. *Consider two lines $l(a_1, b_1)$ and $l(a_2, b_2)$:*

1. *If $a_1 = a_2$ and $b_1 \neq b_2$, then $l(a_1, b_1) \cap l(a_2, b_2) = \emptyset$ (we say that the two lines are* parallel*).*

2. *If $a_1 \neq a_2$, then $l(a_1, b_1) \cap l(a_2, b_2)$ contains exactly one point in Z_p^2 (we say that the two lines* intersect*, and the point is an* intersection point*).*

Proof. Let $a_1 = a_2$ and $b_1 \neq b_2$, and assume without loss of generality that $b_1 \leq b_2$. Suppose that there is (x, y) such that $(x, y) \in l(a_1, b_1) \cap l(a_2, b_2)$, so $a_1 x + b_1 \equiv a_1 x + b_2 \bmod p$. This implies that $b_2 - b_1 \equiv 0 \bmod p$. Because $(b_2 - b_1) \in Z_p$, $b_2 - b_1 \equiv 0$ if and only if $b_2 = b_1$. This contradicts our assumption, and (1) is proved. Next, let $a_1 \neq a_2$, and assume without loss of generality that $a_1 < a_2$. For $x \in Z_p$, define as $d_x = (a_2 x + b_2) - (a_1 x + b_1) \pmod{p}$. To prove (2), I show that $d_0, \ldots, d_{p-1}$ is the permutation of $0, \ldots, p-1$. It is clear that $d_x \in Z_p$ for any $x \in Z_p$, and it suffices to show that $d_0, \ldots, d_{p-1}$ are all different. Suppose for contradiction that $d_x = d_{x'}$ for $x < x'$, then

$$\begin{aligned} & (a_2 x + b_2) - (a_1 x + b_1) \\ \equiv \; & (a_2 x' + b_2) - (a_1 x' + b_1) \bmod p. \end{aligned}$$

Thus, $(a_2 - a_1)(x' - x) \equiv 0 \bmod p$. However, the product of $(a_2 - a_1)$ and $(x' - x)$ cannot be divisible by p, because p is prime, $(a_2 - a_1) \in Z_p$ and $(x' - x) \in Z_p$.

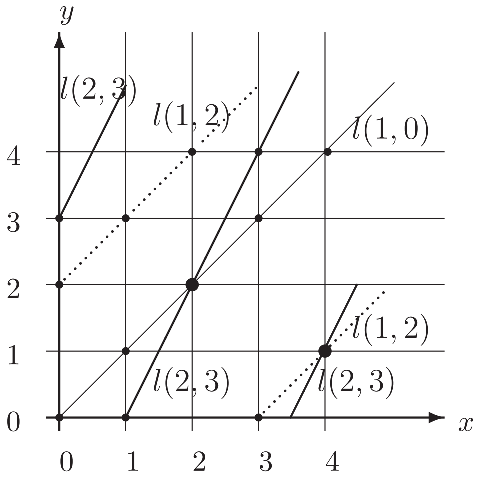

Figure 4.1. Lines and points over finite geometry

Therefore, no value appears twice in $d_0, \ldots, d_{p-1}$. The intersection point of the two lines is $(x, a_1x + b_1)$ for x such that $d_x = 0$. □

Example 4.1. *Consider a plane Z_5^2. Figure 4.1 shows three lines $l(1,0)$, $l(2,3)$, and $l(1,2)$ over Z_5^2. The lines consist of the following points:*

$$\begin{aligned}
l(1,0) &= \{(0,0),(1,1),(2,2),(3,3),(4,4)\}, \\
l(2,3) &= \{(0,3),(1,0),(2,2),(3,4),(4,1)\}, \\
l(1,2) &= \{(0,2),(1,3),(2,4),(3,0),(4,1)\}.
\end{aligned}$$

The line $l(1,0)$ intersects with the line $l(2,3)$ at the point $(2,2)$, and the line $l(2,3)$ intersects with the line $l(1,2)$ at the point $(4,1)$. The line $l(1,0)$ is parallel to the line $l(1,2)$.

Let $\mathcal{L}$ be the class of all lines, then $\mathcal{L}$ contains p^2 lines in total. Let l be a line in $\mathcal{L}$, then there are p lines (including l itself) which are parallel to l, and all the other $p^2 - p = p(p-1)$ lines intersect with l. Thus if we choose another line from $\mathcal{L}$ randomly, then the line intersect with l with probability $p(p-1)/p^2 = (p-1)/p$, which approaches to 1 as p increases.

4.3.2 Basic Scheme

To make the later discussion clear, a key predistribution scheme is described by a four-tuple (K, N, m, k), where K, the *key pool*, is a collection of keys, N is the set of nodes, m is the number of keys which are assigned to a sensor node, and k is a *key assignment function* mapping N to a subset of K with cardinality m. For example, in the random key distribution scheme [19], k is a random function which associates a node with a randomly chosen subset of K.

Now a new key predistribution scheme is defined and called *the basic scheme* in this chapter. First, choose a prime number p, and associate each point in Z_p^2 with a randomly chosen cryptographic key. The key that is assigned to a point π is denoted as $k(\pi)$. The key pool K is defined as $K = \{k(\pi)|\pi \in Z_p^2\}$. For a node $n \in N$, let $l(n)$ be a line which is chosen randomly from $\mathcal{L}$. The key assignment function k is then defined as

$$k(n) = \{k(\pi)|\pi \in l(n)\}.$$

Intuitively, a node is associated with a randomly chosen line in $\mathcal{L}$ and the node has keys that are on the line. Because a line contains p points, a node is assigned with p keys. The obtained key predistribution scheme is thus (K, N, p, k).

It is easily understood that two sensor nodes n_1 and n_2 share a key in common if and only if lines $l(n_1)$ and $l(n_2)$ intersect. Thus the connectivity of the basic scheme is

$$p_{\mathrm{con}} = (p-1)/p. \tag{4.1}$$

To discuss the key-survivability, assume that $l(n_1)$ and $l(n_2)$ intersect at a point π, and n_1 and n_2 share the key $k(\pi)$. The number of lines which pass through π is p, and therefore if we choose a line from $\mathcal{L}$ randomly and uniformly, then the chosen line passes through π with probability $p/p^2 = 1/p$. This probability coincides with the probability that a sensor node which is captured by an intruder happen to have $k(\pi)$. Consider that an intruder have captured c sensor nodes. The key $k(\pi)$ stays secure (not known to the intruder) if and only if all the c nodes do not include $k(\pi)$. This happens with probability $(1-1/p)^c$. Thus the key-survivability of the basic scheme under c captured nodes is

$$p_{\mathrm{sur}_c} = (1-1/p)^c.$$

4.3.3 Evaluation of the Basic Scheme

The performance of the basic scheme is evaluated in this section. The basic scheme is intended to replace for the random key distribution scheme in [19], and to be used as a "building block" of other advanced schemes. Thereby the basic scheme and the random key distribution scheme are compared, other advanced schemes are not considered in this section. As noted in Section 4.1, there are tradeoff relations between the memory size, the connectivity and the key-survivability. To make the discussion clear, we meanwhile set the memory size to a constant number, and observe the relation between the connectivity and the key-survivability.

According to some literatures [41, 19], we assume that a node can have about 100 keys. In the basic scheme, the memory size equals to the prime number p, and hence choosing $p = 101$ seems reasonable for this setting. This choice of p makes the connectivity $p_{con} = 0.99$. In the random key distribution scheme, the connectivity is controlled by the size of the key pool $|K|$ and the memory size m. To make $p_{con} = 0.99$ and $m = 101$, the key pool must contain 2311 or less keys. Under the choice of these parameters, the survivabilities of the two schemes under a node capture attack are compared. The numerical result is depicted in Figure 4.2. The x-axis of the graph is the number of compromised nodes, and the y-axis is the key-survivability. As the number of compromised nodes increases, the key-survivability decreases in general. We can see from the figure that the key-survivability of the random key distribution scheme drops rapidly as the number of compromised nodes increase, whereas that of the basic scheme decreases slowly. Compare, for example, the key-survivability of the two schemes when 50 nodes are compromised. The figure shows that only 10% of keys survive in the random key distribution scheme, while about 60% of keys survive in the basic scheme. This means that the basic scheme offers more robustness against node capture attacks than the random key distribution scheme, with the same connectivity and the same number of keys in each node.

To compare the two schemes from another direction, consider how many keys are needed in the random key distribution scheme to achieve the same performance as the basic scheme. Assume that we would like to realize a key agreement scheme with the connectivity $p_{con} = 0.99$ and the key-survivability $p_{sur_c} \geq 0.9$

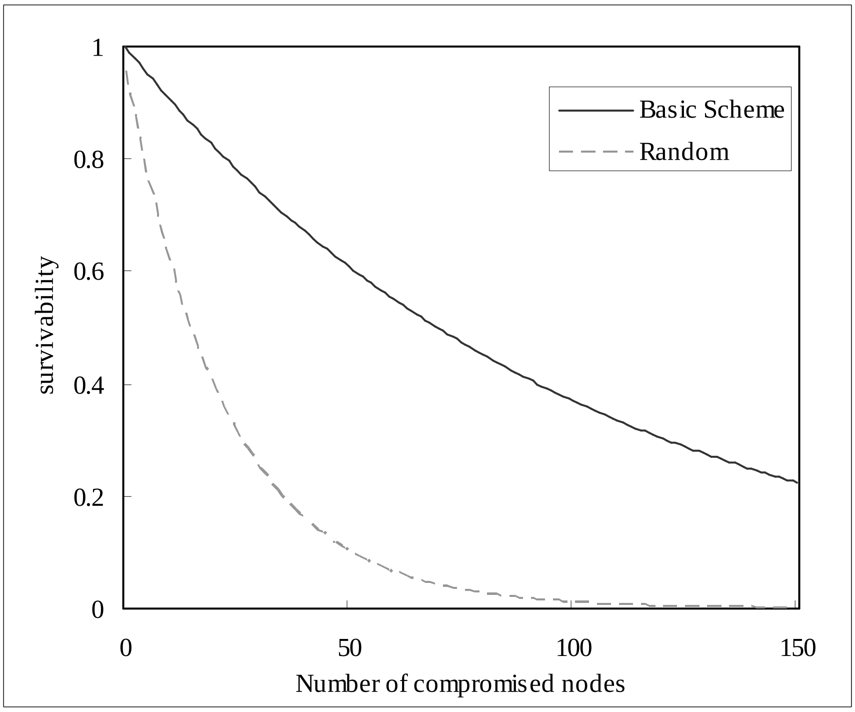

Figure 4.2. The key-survivability of the basic scheme with $p_{con} = 0.99$ and $m = 101$

for up to 10 compromised nodes. In the basic scheme, choosing $p = 101$ achieves the required performance as illustrated in Figure 4.2. In this case, a sensor node needs to have 101 keys. To achieve the same performance by using the random key distribution scheme, we need to choose so that $|K| = 41504$ and $m = 454$. The random key distribution scheme requires a sensor node to have four times more keys than the basic scheme. Other comparisons for other choice of p_{sur_c} and c are presented in Figure 4.3. We can see that the basic scheme requires smaller number of keys than the random key scheme. The number of keys has strong relationship to the memory size of a sensor node, and thus to the manufacturing cost and the energy efficiency of a node. We could see clear advantage of the basic scheme against the random key distribution scheme.

The basic scheme also has an advantage in the communication overhead for the key agreement. In the random key distribution scheme, two nodes need to exchange what keys they do have. This will be done by exchanging the indices of

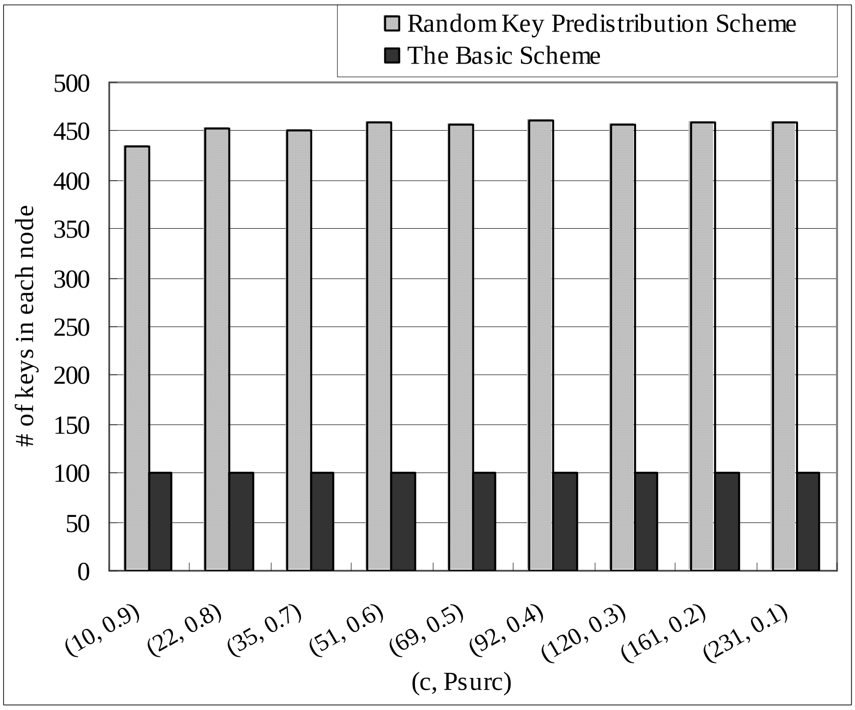

Figure 4.3. The memory size of the basic scheme with $p_{con} = 0.99$

the keys, but the communication cost for this operation should be proportional to the number of keys in the nodes. In the basic scheme, the keys are assigned in a structured and systematic manner, which helps reducing the communication overhead. Remind that a line over a plane is defined by just two parameters, a and b. Once two node, n and n', exchange their parameters, then the nodes can compute the intersection point of $l(n)$ and $l(n)'$ by themselves. The communication for exchanging the parameters can be done with constant amounts of clear-text messages. This is a significant advantage in sensor networks in which energy consumed for computation and communication is so expensive.

4.4. Extended Schemes

The basic scheme is advantageous to the random key distribution scheme. On the other hand, the basic scheme does not have sufficient flexibility to control the connectivity and the key-survivability as described below. The scheme has just one parameter, the prime number p, and choosing p determines the memory size (the number of keys assigned to a node), the connectivity and the key-survivability. In general, the prime number p cannot be very small, which means that the connectivity $p_{con} = (p-1)/p$ should be always close to 1. In a sparse network, keeping p_{con} close to 1 is desirable, while in a non-sparse network, it is sometimes allowed that p_{con} can be much smaller than 1 to obtain more key-survivability. Unfortunately, it is difficult in the basic scheme to make p_{con} small value without affecting other criteria. In this section, extensions of the basic scheme is investigated so that we can have more flexibility and controllability.

4.4.1 Extended Scheme 1

The first type of extensions is to use lines partially. In the basic scheme, a sensor node n is associated with a line $l(n)$, and all keys in $\{k(\pi)|\pi \in l(n)\}$ are assigned to n. Now, instead of assigning all keys on the line, consider to assign a subset of $\{k(\pi)|\pi \in l(n)\}$ to n. Let m be an integer with $m < p$ and define the key assignment function k as

$$k(n) = \{k(\pi)|\pi \in l_m(n)\},$$

where $l_m(n)$ is a randomly chosen subset of $l(n)$ with cardinality m. Thus, the number of keys that a node needs to remember is smaller than the prime number p. This may reduce the memory requirement of sensor nodes, or allow us choosing the prime number p bigger than the memory restriction of sensor nodes.

Two nodes share a key if and only if the lines associated with the nodes intersect, and both of the two nodes have the key at the intersection point of the lines. Thus the connectivity of this extension scheme is

$${p_{\mathrm{con}}}' = p_{\mathrm{con}}(m/p)^2 = (p-1)m^2/p^3,$$

where p_{con} is defined in (4.1). Let $k(\pi)$ be the key shared by the two nodes. A randomly chosen sensor node n contains $k(\pi)$ if and only if the line $l(n)$ passes through π and $l_m(n)$ happens to contain π. The probability that n have $k(\pi)$ is therefore $(p/p^2)(m/p) = m/p^2$. The key-survivability of the extended scheme for c captured nodes is

$${p_{\mathrm{sur}}}'_c = (1 - m/p^2)^c.$$

4.4.2 Extended Scheme 2

The above described extension can be regarded as a combined scheme of the basic scheme and the random key distribution scheme, where the random key distribution scheme is used as an "inner scheme". We can consider another combined scheme in which the random key distribution scheme is used as an "outer scheme". Let $S_0 = (K_0, N, m_0, k_0)$ be a random key predistribution scheme. Assume that K_0 contains r keys and keys in K_0 are numbered from 1 to r. The number associated with a key $k \in K_0$ is denoted by $i(k)$. Also define r basic schemes $S_i = (K_i, N, m, k_i)$ with $1 \leq i \leq r$ where $K_i \cap K_j = \emptyset$ if $i \neq j$. Now combine these schemes and define

$$S_0 \circ \{S_1, \dots, S_r\} = (\bigcup_{i=1}^{r} K_i, N, m_0 m, k'),$$

where

$$k'(n) = \bigcup_{k \in k_0(n)} k_{i(k)}(n).$$

In this extension, r independent key distribution schemes $S_1, \dots, S_r$ are prepared. The "parent scheme" S_0 assigns each node with keys in K_0. If $k \in K_0$ is assigned to a node by S_0, a "child scheme" $S_{i(k)}$ is used to determine m keys for the node n. In other words, S_0 is used to determine which schemes should be used to assign keys for a node. The scheme S_0 assigns m_0 keys to a node, and each S_i with $1 \leq i \leq r$ assigns $m(= p)$ keys if it is chosen by S_0. Therefore a node receives $m_0 m (= m_0 p)$ keys in total.

In this scheme, two nodes share a key if and only if (i) there is a scheme S_i with $1 \leq i \leq r$ that is commonly used by the two nodes, and (ii) the two nodes agree a key by using the scheme S_i. Remark that S_0 is a random key distribution scheme and it can assign, to two nodes, multiple keys in common. In this case, the two nodes use multiple child schemes in common. The probability that two nodes use i child schemes in common is

$$\frac{\binom{r}{i}\binom{r-i}{2(m_0-i)}\binom{2(m_0-i)}{m_0-i}}{\binom{r}{m_0}^2} = \frac{\binom{m_0}{i}\binom{r-m_o}{m_0-i}}{\binom{r}{m_0}}.$$

Two nodes fail to agree a key if all the i key schemes fail to assign common keys.

Therefore the connectivity of this extended scheme is

$$p_{\mathrm{con}}'' = \sum_{i=1}^{m_0} \frac{\binom{m_0}{i}\binom{r-m_o}{m_0-i}}{\binom{r}{m_0}}(1-(1-p_{con})^i). \tag{4.2}$$

where p_{con} is defined in (1). If we use the extended scheme 1 for $S_1, \ldots, S_r$ instead of the basic scheme, then the connectivity changes to a value that is obtained by replacing p_{con} with p'_{con} in (2). To discuss the key-survivability, assume that two nodes are assigned a common scheme S_i, and share a key $k(\pi)$ of S_i. The probability that a node which is captured by an intruder happen to include $k(\pi)$ is $(m_0/r)(1/p) = m_0/rp$. Thus the key-survivability of this scheme under c captured nodes is

$$p_{\mathrm{sur}_c}'' = (1 - m_0/rp)^c.$$

If we use the extended scheme 1 for $S_1, \ldots, S_r$, just replace m_0/rp in the above equation with $m_0 m/rp^2$.

In this extended scheme 2, there is a lot of flexibility for parameter choices. Now, consider that a node can have about 200 keys. In this case, we can arbitrarily choose m_0 and p to accomplish $m_0 p \approx 200$. After we choose m_0 and p, we can change the connectivity and the key-survivability by changing the parameter r. For example, consider three choices $(p, m_0) = (199, 1), (41, 5)$, and $(19, 10)$ which all make $m_0 p \approx 200$. For each choice, we choose $r = 2, 40$ and 145 to make the connectivity $p''_{con} = 0.5$. Figure 4.4 shows the evaluation results of key-survivability for the above three choices of parameters. This shows that the key-survivability gets better as p increases.

4.4.3 Evaluation of the Two Extended Schemes

This section is to compare the extended schemes 1 and 2 with the random key distribution scheme. I set m, the number of keys in a node, to $m = 200$ in this section, and observe how the key-survivability changes for $p_{con} = 0.33, 0.5$ and 0.9. Table 4.1 summarizes parameters that make each scheme achieve the given connectivity.

Figures 4.5–4.7 illustrate the key-survivability for each p_{con}. We can see that the extended schemes show better key-survivability than the random key distribution scheme. The advantage of the proposed schemes is especially significant

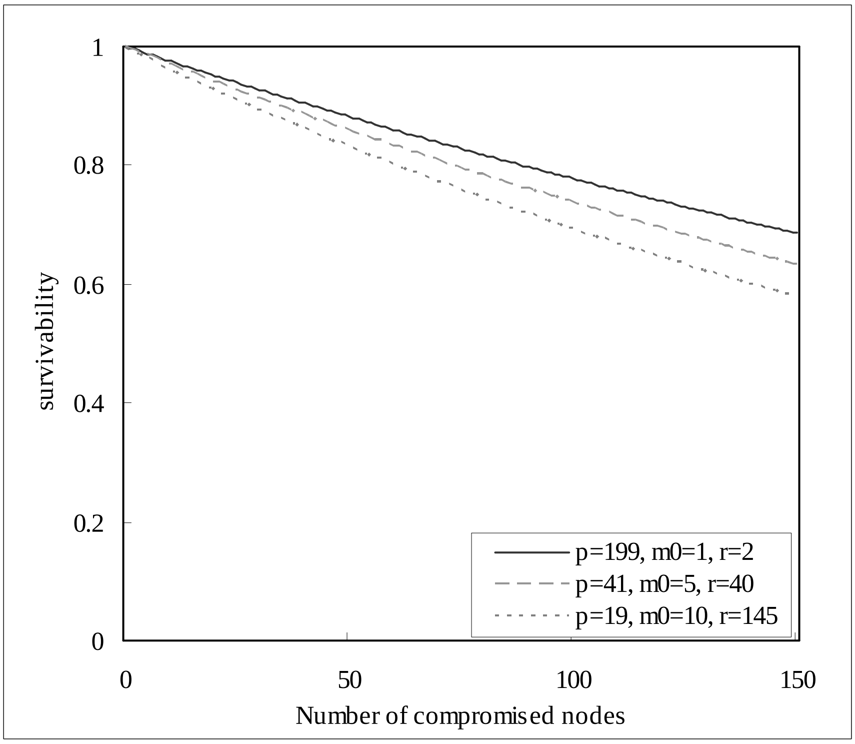

Figure 4.4. The performance of the extended scheme 2 for three choices of parameters with $p_{\mathrm{con}} = 0.5$

when p_{con} is close to 1. This result suggests that the proposed scheme is more favorable when we need high connectivity.

Numerical results show that the extended scheme 1 has small advantage to the extended scheme 2 with respect to the key-survivability. However, the extended scheme 2 has some advantages that are not illustrated in the figures. For example, the extended scheme 2 allows sensor nodes to have two or more keys in common. Hence techniques similar to the q-composite key [41] is easily available in the extended scheme 2, and the key-survivability might be improved by using such additional techniques.

Table 4.1. Parameters to achieve given connectivity

p_{con}	**Extended 1** p	**Extended 2** (p, r, m)	**The Random Key** $\|K\|$
0.33	347	(199,3,1)	99,619
0.50	283	(199,2,1)	58,295
0.90	211	(37,21,6)	18,008

4.5. Discussion

4.5.1 Choice of Parameters

To use the proposed scheme in real applications, we need to determine the values of parameters. Different choice of parameters gives the scheme different characteristics. We would like to find the optimum values of parameters for a given application, but the choice of parameters strongly depends on many aspects which are difficult to quantify.

For example, how the density of sensor nodes affects the choice of parameters is considered. If sensor nodes are deployed sparsely in a field, then we should choose parameters so that the scheme has high connectivity. In a sparse network, a node is expected to have small number of neighbor nodes. If the connectivity is small in this sparse network, then it can happen with considerable probability that a node cannot agree a key with any of its neighbor nodes, and is isolated (in the sense of secure communication) in the network. Such an isolation of a node happens with probability $(1 - p_{con})^d$, where d is the average number of neighbor nodes. To make this probability small, we need to let p_{con} take a value close to 1. This is possible by choosing a large prime number for p in the basic scheme, or by choosing p and m close in the extended scheme 1. (parameter choices of the extended scheme 2 is complicated, but we can determine appropriate parameter values by solving an equation).

The discussion goes differently if sensor nodes are deployed densely. On discussing key agreement, there are two points that we need to remark to compare a sparse network and a dense network. The first point is that, in a dense network, an intruder will be able to find and capture sensor nodes easily, and the

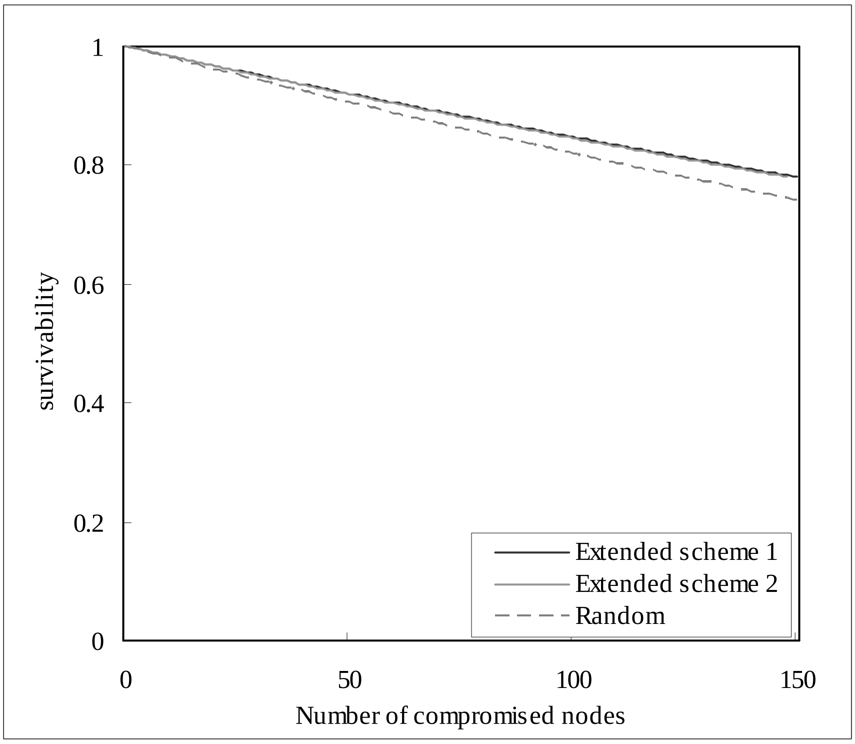

Figure 4.5. The key-survivability of the extended schemes with $p_{\mathrm{con}} = 0.33$

node capture attack is more serious than the sparse network case. The second point is that the cost to reinforce a key agreement scheme, for example by using path-keys [46, 47] or the multipath key reinforcement [41], is smaller and more acceptable in a dense network than in a sparse network. Hence, instead of increasing the connectivity, we may choose parameters so that the scheme has large key-survivability, and to use the reinforce techniques to mitigate the low connectivity.

4.5.2 Relation to the Location-Based Approach

A sensor network is constructed by deploying a number of sensor nodes in a designated area. In many cases, it is difficult to predict (or control) the exact deploy location of each node, and we cannot know in advance which nodes come close to a given node. To realize a key-agreement scheme under this situation, there are two different approaches. In a *location-free* approach, I investigate

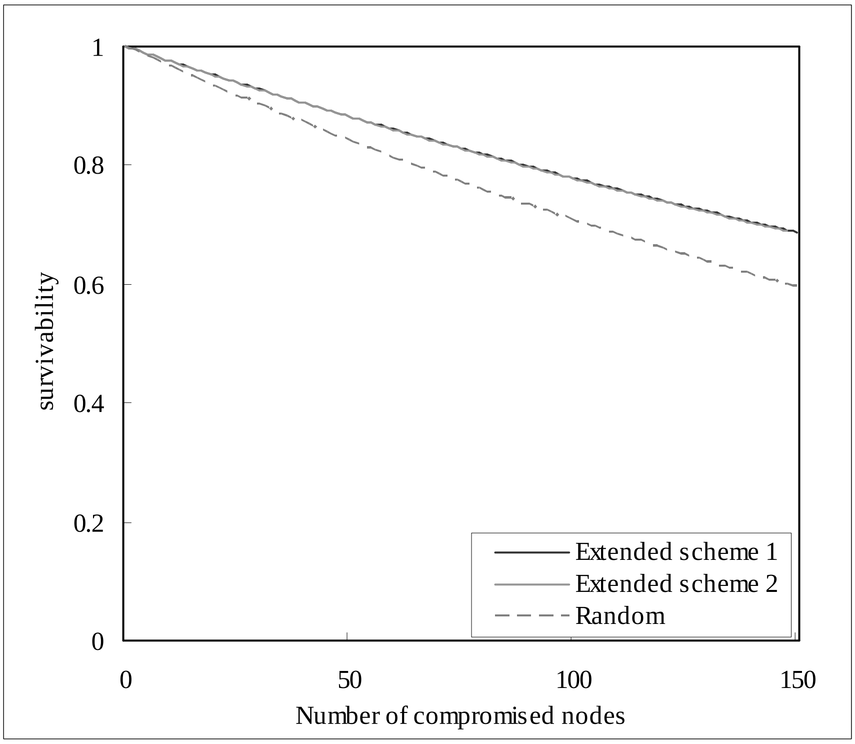

Figure 4.6. The key-survivability of the extended schemes with $p_{\mathrm{con}} = 0.5$

a scheme that shows reasonable performance for "any" deploy scenario. Even if nodes are deployed unintentionally or randomly, the scheme is required to show reasonable performance. The scheme investigated in this thesis follows the location-free approach. In a *location-based* approach, we assume that nodes are deployed under a certain condition, and devise a scheme that is optimized for the assumed deploy scenario. For example, [1] considers a location-based scheme assuming that nodes are deployed by dropping from a helicopter. We cannot predict exactly where a node is settled, but a certain distribution function will be available in this case. By using the distribution function, we can choose keys to be embedded in a node "non-uniformly". In other words, we choose keys so that two nodes which will be deployed closely have keys in common. The idea of the location-based approach is further developed in [43] and [42] for example.

The location-based approach has several advantages against the location-free approach. If nodes are deployed as expected, then a location-based scheme shows better connectivity than a location-free scheme in general. It is also said that, in

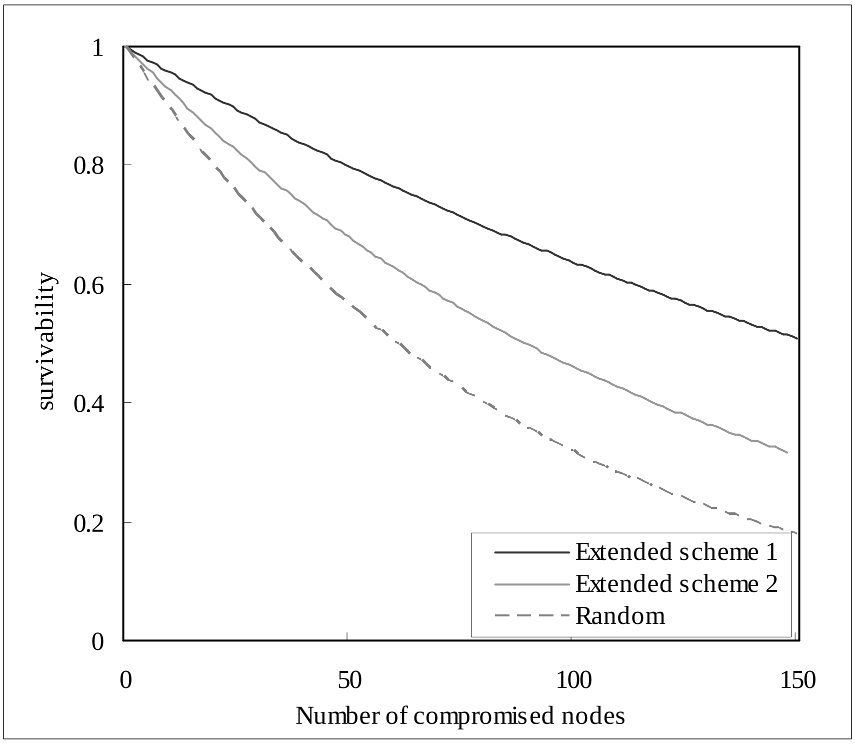

Figure 4.7. The key-survivability of the extended schemes with $p_{\mathrm{con}} = 0.9$

the location-based approach, the affect of a node capture attack is confined in a small local section of the network. The approach is, furthermore, able to support very large scale network.

In spite of these advantages of the location-based approach, it is significant to investigate the location-free aproach. The essential problem of the location-based approach is that the approach is not always available. If we know nothing about the node deployment, then we cannot devise a location-based scheme. Consider for example mobile nodes which autonomously and freely move after they are deployed. Obviously location-based schemes are not available to support such mobile nodes, and we need to consider location-free schemes. In other words, location-based schemes cannot replace for all the location-free schemes, and hence studying good location-free scheme has practical significance.

Comparison of the location-free and location-based approaches suggests possible applications of the proposed scheme. The key agreement scheme investigated in this chapter is intended to be general, multi-purpose and fundamental technology which is available in wide range of applications. However, unfortunately, the proposed scheme is not the very best for all applications: For example, location-based schemes will be more useful than the proposed scheme if we can predict the deploy locations of sensor nodes. The proposed scheme is superior to other schemes if we cannot predict the deploy locations in advance, and if high connectivity is strongly required. For example, consider mobile sensor nodes which are carried by users or equipped on a vehicle. In such applications, nodes will move extensively large area, and the network can be regarded as very sparse. The connectivity must be sufficiently large in this case, because an encounter with other nodes is "precious" in such a sparse network and we should not miss the opportunity. The location-based approach is not available in this case since the move of nodes is not predictable. Within the location-free framework, the proposed scheme show clear advantage to the random key distribution scheme when the connectivity is required to be large. It seems that the proposed scheme is the best choice for such an application.

4.5.3 Robustness of the Proposed Methods

When an intruder compromises a node, the intruder can absorb a line stored in the node. However, threat of the node capture attack against the proposed method is no bigger than that against the random distribution scheme. Remind that the basic scheme associates each lattice point in Z_p^2 with a randomly chosen cryptographic key and does not create a cryptographic key by using information about Z_p^2. We assume that an intruder would like to break a secure link between two nodes without compromising the two nodes. In the proposed methods, the intruder cannot create the key for the secure link if the compromised nodes do not store the key. Thus, information about Z_p^2 is not additional profit for the intruder. On the other hand, two nodes exchange just two parameters a and b in the basic method while the two nodes exchange m indices of keys in the random key distribution scheme. Since the intruder can easily absorb the index of the shared key used for the secure link by eavesdropping whole the network, the intruder may find the shared key in the basic method easier than that in the random key predistribution scheme.

4.6. Conclusion of Chapter 4

New schemes for predistributing cryptographic keys in sensor networks ware proposed. The proposed scheme is a direct scheme with which key agreement is accomplished between two involved nodes only. The scheme is well suited for realizing high connectivity, which is strongly desired in sensor networks in which sensor nodes are deployed very sparsely. It is also notable that the proposed schemes do not require special assumptions nor equipments such as a timer and key diminish mechanisms.

The proposed scheme can replace for the naive random key scheme, achieving higher security (with respect to the key-survivability) while other system parameters unchanged. The random key distribution scheme has been used as a fundamental component for constructing more sophisticated key management scheme, and therefore the proposed scheme can make significant contribution in the wide ranges of key management techniques for sensor networks.

Chapter 5

Conclusion

In this thesis, efficient and robust infrastructures for secure communication in autonomous computer networks were investigated. Because which scheme we should use depends on properties of a network (e.g., node mobility, existance of trusted third party, resource constraint, and so on), no single security infrastructure can adapt all types of networks. According to the properties of the network, the thesis proposed three security infrastructures.

In Chapter 2, a policy specification language to define the behavior of stateful trust management systems that can represent a system with internal states was proposed. A policy specification is a finite set of rules of definite Horn clause form. Also, the verification problem was defined as the problem to decide whether the behavior of a system with a given policy satisfies verification property, and a verification method for the problem was also proposed. Input of the verification problem is a pair of a policy to be verified and and a verification property that the policy should satisfy specified by an LTL formula. The verification methods considered in the thesis are two automatic model checking methods; one consists of SPIN with Prolog (Method 2) and the other consists of only Prolog (Method 3). Model checking methods for the "introducer gains benefits" system (See Section 2.3) ware implemented as an example. The result showed that the Method 3 is more efficient than the Method 2 (See Table 1). Also, I showed that the verification time is propotional to the number of state transitions of S where S is a state transition system induced by the given policy. As a fragment of future work, I will consider a detailed implementation based on the proposed model, e.g., an implementation of the memory and interface among the modules (trust establishment, access control, and memory).

In Chapter 3, Public-Key Infrastructure for ad-hoc networks was investigated. A central problem to adapt Public-Key Infrastructure for ad-hoc networks is the certificate-chain discovery problem. In this thesis, new distributed algorithms for solving the certificate-chain discovery problem for a web-of-trust type PKI are proposed. The certificate searching algorithm uses a distributed algorithm for constructing a spanning tree in the web-of-trust. Also, to evaluate communication overhead of the proposed methods, the communication cost was defined as the total message bits by taking the size of a certificate and the number of certificates in the packet into account. According to the definition, performance evaluations of the existing and proposed methods were conducted by numerical analysis and computer simulation. The results showed that the communication cost of the proposed algorithm is less than ten percent of that of the existing method (See Table 2). The author's future work will include the modeling of web-of-trust-type PKI systems for ad-hoc networks more deeply to construct a new trust model combining the models of Capkun, et al. [12] and Kitada, et al. [13].

In Chapter 4, predistributing cryptographic keys in sensor networks was investigated. A new method and two extended methods in which keys are assigned according to a basic algebraic geometry are proposed. The proposed methods associate each node with a line over a two-dimensional finite plane, and manage keys so that two nodes can agree a key if and only if the associated lines intersect with each other. From comparison the extended scheme 1 and 2, we can see the schemes has defferent advantages. The performance of the proposed methods was computed analytically, and the results show that the proposed methods have better trade-off point than the random key predistribution scheme. The extended scheme 1 has small advantage to the extended scheme 2 with respect to the key-survivability. The extended scheme 2 allows sensor nodes to have two or more keys in common. Hence techniques similar to the q-composite key [41] is easily available in the extended scheme 2, and the key-survivability might be improved by such applications of advanced scheme. It seems that there are a lot of points for improving and extending the basic scheme. For example, lines and points over a two-dimensional plane considered. We can extend the geometry to three or more dimensional space.

References

[1] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney: "A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge," IEEE Conference on Computer Communications (INFOCOM), 2004.

[2] R. Housley W. Polk W. Ford, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 3280, 2002.

[3] L. M. Kornfelder: "Toward a Practical Public-Key Cryptosystem," Bachelor's thesis, Dept. Electrical Eng., Massachusetts Inst. of Technology, 2005.

[4] M. Blaze, J. Feigenbaum, and J. Lacy: "Decentralized Trust Management," IEEE Symposium on Security and Privacy (S&P), pp.164–173, 1996.

[5] K. E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu: "Requirements for Policy Languages for Trust Negotiation," IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY), pp.68–79, 2002.

[6] M. Winslett: "An Introduction to Trust Negotiation," International Conference on Trust Management (iTrust), Lecture Notes in Computer Science, 2692, pp.275–283, 2003.

[7] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid: "Access Control Meets Public Key infrastructure Or: Assigning Roles to Strangers," IEEE Symposium on Security and Privacy (S&P), pp.2–14, 2000.

[8] C. K. Toh: "Ad-Hoc Mobile Wireless Networks: Protocols and Systems," Prentice Hall, 2001.

[9] L. Zhou and Z. J. Hass: "Securing Ad Hoc Networks," IEEE Network, **13**, 6, pp.24–30, 1999.

[10] S. Yu and R. Kravets: "Composite Key Management for Ad Hoc Networks," IEEE Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services (Mobiquitous), pp.52–61, 2004.

[11] P. Zimmermann: "The Official PGP User's Guide," MIT Press, 1995.

[12] S. Capkun, L. Buttyan, and J. P. Hubaux: "Self-Organized Public-Key Management for Mobile Ad Hoc Networks," IEEE Transactions on Mobile Computing, **2**, 2, pp.52–64, 2003.

[13] Y. Kitada, Y. Arakawa, K. Takemori, A. Watanabe, and I. Sasase: "On Demand Distributed Public Key Management Using Routing Information for Wirelss Ad Hoc Netwoks," IEICE Transactions on Information and Systems, **J88-D1**, 10, pp.1571-1583, 2005. (in Japanese)

[14] Y. Kitada, A. Watanabe, K. Takemori, and I. Sasase: "On Demand Distributed Public Key Management for Wireless Ad Hoc Networks," IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim), 2005.

[15] Y. Kitada, A. Watanabe, K. Takemori, and I. Sasase: "On Demand Distributed Public Key Management without Considering Routing Tables for Wireless Ad Hoc Networks," Asia Pacific Symposium on Information Technology (APSITT), pp.375-381, 2005.

[16] X. Li, S. Gordon, and Jill Slay: "On Demand Public Key Management for Wireless Ad Hoc Networks," Australian Telecommunication Networks and Applications Conference (ATNAC), pp.36–43, 2004.

[17] R. Li, J. Li, H. Kameda, and P. Liu: "Localized Public-Key Management for Mobile Ad Hoc Networks," IEEE Global Telecommunications Conference (Globecom), pp.1284–1289, 2004.

[18] D. E. Clarke, J. E. Elien, C. M. Ellison, M. Fredette, A. Morcos, and R. L. Rivest: "Certificate Chain Discovery in SPKI/SDSI," Journal of Computer Security, **4**, 9, pp.285–322, 2001.

[19] L. Eschenauer and V. D. Gligor: "A Key Management Scheme for Distributed Sensor Networks," ACM Conference on Computer and Communications Security (CCS), pp.41–47, 2002.

[20] M. Bishop: "Computer Security: Art and Science," Addison Wesley, 2002.

[21] M. Y. Becker and P. Sewell: "Cassandra: Distributed Access Control Policies with Tunable Expressiveness," IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY), pp.159–168, 2004.

[22] M. Y. Becker and P. Sewell: "Cassandra: Flexible Trust Management, Applied to Lectronic Health Records," IEEE Computer Security Foundations Workshop (CSFW), pp.139–154, 2004.

[23] S. Weeks: "Understanding Trust Management Systems," IEEE Symposium on Security and Privacy (S&P), pp.94–105, 2001.

[24] E. M. Clarke, O. Grumberg, and D. A. Peled: "Model Checking," The MIT Press, 2000.

[25] M. Blaze, J. Feigenbaum, and J. Ioannidis: "The KeyNote Trust-management System Version 2," RFC2704, Sept., 1999.

[26] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara: "Reputation Systems," Communications of the ACM, **43**, 12, pp.45–48, 2000.

[27] A. K. Day: "Understanding and Using Context," Personal and Ubiquitous Computing Journal, **5**, 1, pp.4–7, 2001.

[28] J. Al-Muhtadi, A. Ranganathan, R. Campbell, and M. D. Mickunas: "Cerberus: A Context-Aware Security Scheme for Smart Spaces," IEEE International Conference on Pervasive Computing and Communications (PerCom), pp.489–496, 2003.

[29] G. J. Holzmann: "The SPIN Model Checker," Addison-Wesley, 2004.

[30] SWI-Prolog's Home page, http://www.swi-prolog.org/

[31] U. Maurer: "Modelling a Public-Key Infrastructure," European Symposium on Research in Computer Security (ESORICS), Lecture Notes in Computer Science, 1146, pp.325–350, 1996.

[32] N. A. Lynch: "Distributed Algorithms," Morgan Kaufmann Publishers, 1996.

[33] H. Mohri, I. Yasuda, Y. Takata, and H. Seki: “Certificate Chain Discovery in Web of Trust for Ad Hoc Networks,” IEEE International Symposium on Ubisafe Computing (UbiSafe), 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW), **2**, pp.479–485, 2007.

[34] K. Miura, T. Masuzawa, and N. Tokura: “A Distributed Shortest Paths Algorithm with Distance-Dependent Message Complexities,” IEICE Transactions on Information and Systems, **J77-D1**, 1, pp.21–32, 1994. (in Japanese)

[35] H. T. Lau: “A Java Library of Graph Algorithms and Optimization,” Chapman & Hall/CRC, 2006.

[36] H. Breu and D. G. Kirkpatrick: “Unit Disk Graph Recognition is NP-hard,” Computational Geometry: Theory and Applications, **9**, pp.3–24, 1993.

[37] J. R. Douceur: “The Sybil Attack,” International Workshop on Peer-to-Peer Systems (IPTPS), Lecture Notes in Computer Science, 2429, pp.251–260, 2002.

[38] N. Mezzetti: “A Socially Inspired Reputation Model,” European PKI Workshop: Research and Applications (EuroPKI), Lecture Notes in Computer Science, 3093, pp.191–204, 2004.

[39] C. P. Pfleeger and S. L. Pfleeger: “Security in Computing,” Prentice Hall, 2006.

[40] I. Aad, J. P. Hubaux, and E. W. Knightly: “Denial of Service Resilience in Ad Hoc Networks,” ACM Annual International Conference on Mobile Computing and Networking (MobiCom), pp.202–215, 2002.

[41] H. Chan, A. Perrig, and D. Song: “Random Key Predistribution Schemes for Sensor Networks,” IEEE Symposium on Security and Privacy (S&P), pp.197–213, 2003.

[42] F. Anjun: “Location Dependant Key Management Using Random Key-Predistribution in Sensor Networks,” ACM Workshop on Wireless Security, pp.21–30, 2006.

[43] T. Ito, H. Ohta, N. Matsuda, and T. Yoneda: “A Key-Distribution Scheme for Deployable Sensor Networks Using Probability Density Function of Node Deployment,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, **J89-A**, 12, pp.1034–1043, 2006. (in Japanese)

[44] D. Liu, P. Ning, and W. Du: “Group-Based Key Pre-Distribution in Wireless Sensor Networks,” ACM Workshop on Wireless Security, pp.11–20, 2005.

[45] L. Zhou, J. Ni, and C. V. Ravishanker: “Efficient Key Establishment for Group-Based Wireless Sensor Reployments,” ACM Workshop on Wireless Security (WiSe), pp.1–10, 2005.

[46] D. Liu and P. Ning: “Establishing Pairwise Keys in Distributed Sensor Networks,” ACM Transactions on Information and System Security, **8**, 1, pp.41–77, 2005.

[47] W. Du, J. Deng, Y. S. Han, P. Varshney, J. Katz and A. Khalili: “A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks,” ACM Transactions on Information and System Security, **8**, 2, pp.228–258, 2005.

[48] R. Blom: “An Optimal Class of Symmetric Key Generation Systems,” Advances in Cryptology (CRYPTO), Lecture Notes in Computer Science, 209, pp.335–338, 1984.

[49] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler: “SPINS: Security Protocols for Sensor Networks, ”ACM Journal of Wireless Networks, **8**, 5, pp.521–534, 2002.

[50] S. Zhu, S. Setia, and S. Jajodia: “LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks,” ACM Conference on Computer and Communications Security (CCS), pp.62–72, 2003.

[51] T. Matsumoto and H. Imai: “On The Key Predistribution System: A Practical Solution to The Key Distribuion Problem,” Advances in Crypology (Crypto), Lecture Notes in Computer Science, 293, pp.185–193, 1988.

[52] L. Gong and D. J. Wheeler: "A Matrix Key Distribution Scheme," Journal of Cryptology, **2**, 1, pp.51–59, 1990.

Appendix

A. Verification Program

The top level goal for the model checking algorithm is "go". In this program, I use the predicates *recorda* and *recorded* that are built-in predicates of SWI-Prolog to remenber visited states. These predicates manage lists of terms. By *recorda*$(id, pair(m, q))$, the term $pair(m, q)$ is added into the list named *id*. Subgoal *recorded*$(id, pair(m, q))$ checks whether or not the term $pair(m, q)$ is in the list named *id*. Although this remembering facility can be implemented by the predicate *assert*, it requires recompling of program and is not efficient.

```
/* Model Checking Algorithm */
go :- p_initial(M,Q), dfs(M,Q,[],[]).

/* search for outside */
dfs(M,Q,Stack,InputSeq)
:- recorda(visited,pair(M,Q)),
   recursion_for_each_trans(M,Q,Stack,InputSeq);
   p_accept(M,Q),
   inner_dfs(M,Q,[],[pair(M,Q)|Stack],InputSeq).

/* search for all transition from (M,Q) */
recursion_for_each_trans(M,Q,Stack,InputSeq)
:- p_trans(M,C,Q,M2,Q2),
   not(recorded(visited,pair(M2,Q2))),
   dfs(M2,Q2,[pair(M,Q)|Stack],[C|InputSeq]),
   fail.

/* search for inside */
inner_dfs(M,Q,Stack,OuterStack,InputSeq)
:- recorda(inner,pair(M,Q)),
   inner_recursion_for_each_trans(M,Q,Stack,
    OuterStack,InputSeq).

/* search for all transition from (M,Q) */
inner_recursion_for_each_trans(M,Q,Stack,
OuterStack,InputSeq) :-
   p_trans(M,C,Q,M2,Q2),
   (member(pair(M2,Q2),OuterStack)
    -> (show_counterexample(M,C,Q,M2,Q2,Stack,
        OuterStack,InputSeq), fail)
     ; /* else */ true),
   not(recorded(inner,pair(M2,Q2))),
   inner_dfs(M2,Q2,[pair(M,Q)|Stack],OuterStack,
    [C|InputSeq]),
   fail.

/*  showing counterexample */
show_counterexample(M,C,Q,M2,Q2,Stack,
 [_Ac|OuterStack],InputSeq) :-
   write(’fail: ’),
   write([pair(M2,Q2),pair(M,Q)|Stack]),
    write(OuterStack),
   write([C|InputSeq]), nl.
```