Name: Bo Zhu
Degree: Doctor of Philosophy
Dept: Department of Computer Science
Thesis Title: *Security and Privacy for Large Ad-Hoc Networks*

Abstract

In this dissertation, we propose the study of algorithms and protocols for key management and anonymous secure routing in large ad hoc networks. Existing research efforts in key management can only handle very limited number of nodes and are vulnerable to active attacks. Therefore, one aim of this dissertation is to investigate scalable key management protocols for large ad hoc networks by exploiting hierarchical network structures, threshold digital signature schemes, and verifiable secret sharing techniques. Previous research in anonymous secure routing merely provides *weak location privacy* and *route anonymity*. The other aim of this dissertation is to design routing protocols with strong anonymity properties without compromising the underlying security requirements. Furthermore, this dissertation systematically proposes and evaluates methods that might be able to improve the efficiency of anonymous routing protocols in ad hoc networks.

Keywords: *ad hoc networks, key management, certification services, active attacks, secure routing, privacy*

SECURITY AND PRIVACY FOR LARGE AD-HOC NETWORKS

BO ZHU

NATIONAL UNIVERSITY OF SINGAPORE

2006

SECURITY AND PRIVACY FOR LARGE AD-HOC NETWORKS

BO ZHU

(B.Eng. and M.Eng., Wuhan University, M.Sc., National University of Singapore)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2006

Acknowledgments

First and foremost, I wish to express my deepest gratitude to my supervisors, Dr. Mohan S. Kankanhalli, Dr. Feng Bao, and Dr. Robert H. Deng for their outstanding support and guidance. They have created a stable and stimulating environment in which I was able to fully develop my ideas and means of research; this, I highly appreciate. They have a profound affect on my life and any future success I may have is due in large part to their efforts.

I am thankful to former and present members of ICSD-I2R (InfoComm Security Department, Institute for Infocomm Research) for their support and encouragement, Guilin Wang, Yongdong Wu, Tieyan Li, Huafei Zhu, Jianyin Zhou, Hongjun Wu, and Di Ma. I am also thankful to other ICSD students for creating a great working environment and for useful advices and cooperativeness, Zhiguo Wan, Yanjiang Yang, Gang Yao, Xiaoxi Han, and José Antonio Onieva González.

I would also like to thank my friends in National University of Singapore and Singapore Management University, Jun He, Yinjiu Li, Shuhong Wang, Yuanfang Li, Ning Shao, and Liang Yu.

I would like to thank my family for their seemingly inexhaustible well of faith and support. I wish to thank my mother for giving me strength for persevere, and my father for the drive to be successful. I would like to thank my sister and brother who have always been there for me.

Finally, I want to thank Rong for her love, patience and support. She has listened to my rants, put up with my moods, encouraged me during my failures, and celebrated my successes. More than any other single person, Rong has made the completion of this degree possible.

Table of Contents

List of Tables

List of Figures

Abbreviation List

ACK	Acknowledgement
AKM	the Autonomous Key Management scheme
CA	Certificate Authority
CRL	Certificate Revocation List
DLP	Discrete Logarithm Problem
DoS	Denial-of-Service
EASE	the Efficient Anonymity and Security-Enabled routing protocol
GPK	Global Public Key
GSK	Global Secret Key
GTC	Global Trust Coefficient
MAC	Medium Access Control
ORS	Overall Region Size
PKI	Public Key Infrastructure
RERR	Route Error
RREP	Route Response
RREQ	Route Request
RRPR	Route Repair
RTC	Regional Trust Coefficient
RUPD	Route Update
TA	Trusted Authority
TPK	Temporary one-time Public Key
TSK	Temporary one-time Private Key
TTP	Trusted Third Party
VSS	Verifiable Secret Sharing

Summary

In this dissertation, we propose the study of algorithms and protocols for key management and anonymous secure routing in large ad hoc networks. In view of the fact that existing research efforts in key management can only handle very limited number of nodes and are vulnerable to active attacks, one aim of this dissertation is to investigate scalable key management protocols for large ad hoc networks by exploiting hierarchical network structures, threshold digital signature schemes, and verifiable secret sharing techniques. Most previous work in anonymous secure routing only satisfies a part of the requirements for anonymity, and are vulnerable to specific attacks. In addition, they are inefficient due to the high communication and computation costs involved during the route discovery/maintenance process. The other aim of this dissertation is to design efficient routing protocols with strong anonymity properties without compromising the underlying security requirements.

In this dissertation, we focus on two major research areas in ad hoc network security: key management and secure routing. In the area of key management, we propose an *Autonomous Key Management* (**AKM**) scheme for large ad hoc networks, and two algorithms for certification services which are independent of AKM and that are resistant to active attacks. Comprehensive simulation results show that AKM incurs low computation overhead in handling network dynamic events (such as join and leave) and our second algorithm for certification services is much faster than those in [1–3]. In the area

of secure ad hoc routing, we first identify the requirements for anonymity and security. We then propose a new anonymous routing protocol, named the *Efficient Anonymity and Security-Enabled* (**EASE**) routing protocol. Detailed analysis shows that EASE achieves both anonymity and security properties, as defined in our requirements. A major challenge in designing anonymous routing protocols is to reduce the computation and communication costs. To overcome this challenge, EASE has been designed to require neither asymmetric nor symmetric encryption/decryption while forwarding flooding route requests. Moreover, it provides a local repair mechanism to fix broken parts of a route without compromising security and anonymity. We analyze and measure the effectiveness of EASE in various network and mobility settings and show that, compared with previous proposals, EASE is more efficient in highly dynamic networking settings. This dissertation also systematically evaluates methods that might be able to improve the efficiency of anonymous routing protocols in ad hoc networks.

In conclusion, extensive simulation results and analysis in this dissertation show that, we provide a comprehensive solution for key management in ad hoc networks. At the architecture level, AKM is proposed to enhance the scalability, flexibility and adaptivity of key management. At the algorithm level, two algorithms are proposed to defend active attacks against certification services. Besides that, we design an anonymous secure routing protocol which offers enhanced features in the sense of anonymity, security, and efficiency, when compared to previous work.

CHAPTER 1

Introduction

Since their emergence in the 1970s, wireless networks have become increasingly popular in the computing arena. This is particularly true in the past decade, which has seen wireless networks being adapted to enable mobility. There are currently two variations of mobile wireless networks. The first is known as the infrastructure-based network (i.e., a network with fixed and wired gateways), shown in Figure 1.1. The bridges for these networks are known as base stations. Typical applications of this type of network include office *Wireless Local Area Networks* (**WLANs**).

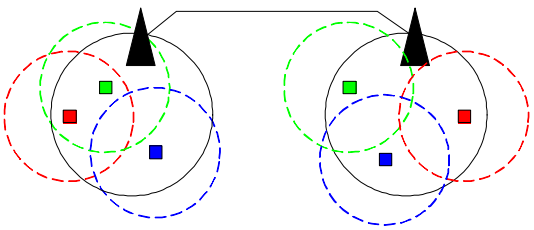


Figure 1.1: Infrastructure-based network

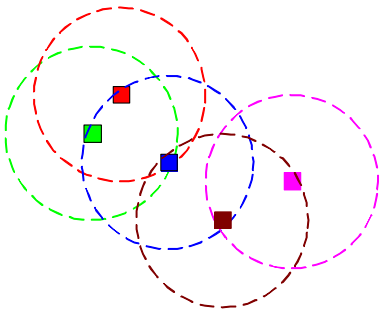


Figure 1.2: Ad hoc network

The second type of mobile wireless network is the infrastructureless mobile network, commonly known as ad hoc networks (shown in Figure 1.2). Ad hoc networks are wireless multi-hop packet networks without any fixed infrastructure. All nodes are capable of movement and can be connected dynamically in an arbitrary manner. Nodes in these networks function as routers which discover and maintain routes to other nodes in the

network. Advantages of such systems are "rapid deployment, robustness, flexibility and inherent support for mobility" [4].

1.1 Applications of Ad Hoc Networks

Some applications of ad hoc network technology include industrial and commercial applications involving cooperative mobile data exchange. Example applications are emergency search-and-rescue operations [5], meetings or conventions in which persons wish to quickly share information, or data acquisition in inhospitable terrains [6, 7]. There are also existing and future military networking requirements for robust data services within mobile wireless communication networks. Many of these networks consist of highly dynamic autonomous topology segments. In addition, the developing technologies of "wearable" computing and communications [8] provide applications for ad hoc networks as well. When properly combined with satellite-based information delivery [9], ad hoc networks can provide an extremely flexible method for establishing communications for fire/safety/rescue operations or other scenarios requiring rapidly deployable communications with survivable, efficient dynamic networking. The unique characteristics of the satellite bring it several advantages over ground-based nodes. For example, the high altitude of a satellite enable it to communicate with ground nodes within a very large area. Moreover, with the spot beam technology, satellites can support localized communications between disjoint sets of ground nodes.

In the future, ad hoc networks will likely form the outermost region of the internetwork, where a wired backbone connects both the fixed local area networks and the

mobile (both the fixed infrastructure and the ad hoc) networks. Whereas the base stations of a fixed infrastructure networks are directly connected to the core, ad hoc networks are typically connected through a satellite link or a terrestrial switch (fixed wired connection point, or mobile radio link). This vision, however, requires still some further developments in ad hoc networking.

1.2 Characteristics of Ad Hoc Networks

Compared to conventional networks, ad hoc networks have several salient characteristics:

No central management Ad hoc networks are autonomously formed with a large number of heterogeneous nodes (ranging in complexity from sensors to palmtops and fully functional laptops and routers) without the aid of any pre-existing communication infrastructure. Therefore, tasks are distributed over and carried out by groups of collaborating nodes. This lack of infrastructure introduces problems in routing, name resolution, service discovery and security [10,11]. The trust in the prior context of security can be transferred and transformed when providing security services. There is also absence of an online server that has its influence on security.

Dynamic topology and membership Nodes are free to move arbitrarily. Thus, the network topology, which is typically multi-hop, may change randomly and rapidly at unpredictable times, and may consist of both bidirectional and unidirectional links. In addition, nodes may join or leave a group or a region with freedom.

Bandwidth-constrained variable capacity links Wireless links will continue to have significantly lower capacity than their hardwired counterparts. In addition, the

realized throughput of wireless communications, after accounting for the effects of multiple access, fading, noise, and interference conditions, etc., is often much less than a radio's maximum transmission rate. One effect of the relatively low to moderate link capacities is that congestion is typically the norm rather than the exception, i.e. aggregate application demand will likely approach or exceed network capacity frequently. As the ad hoc network is often simply an extension of the fixed network infrastructure, mobile ad hoc users will demand similar services. These demands will continue to increase as multimedia computing and collaborative networking applications rise.

Energy-constrained operation Some or all of the nodes in an ad hoc network may rely on batteries or other exhaustible means for their energy. For these nodes, the most important system design criteria for optimization may be energy conservation.

Limited physical security Ad hoc networks are generally more prone to physical security threats than are fixed-cable networks. The increased possibility of eavesdropping, spoofing, and *Denial-of-Service* (**DoS**) attacks should be carefully considered. Existing link security techniques are often applied within wireless networks to reduce security threats. As a benefit, the decentralized nature of network control in ad hoc networks provides additional robustness against the single point of failure problem of more centralized approaches.

These characteristics create a set of underlying assumptions and performance concerns for protocol design, which extend beyond those guiding the design of routing, security, and many other aspects of a network within the higher-speed, relatively static topology of the fixed Internet.

1.3 Security Threats in Ad Hoc Networks

Compared to the wired networks, ad hoc network is much more vulnerable to security attacks. This is mainly due to its features of open medium, dynamic topology, cooperative algorithms, lack of centralized monitoring and management point.

1.3.1 Types of Attacks

Attacks against ad hoc networks can be divided into two groups: passive attacks and active attacks. Passive attacks involve only eavesdropping of data. Active attacks involve actions performed by adversaries, for instance the replay, fabrication, modification and deletion of exchanged data. External attacks are typically active attacks that are targeted e.g. to cause congestion, propagate incorrect routing information, prevent services from working properly or shut down them completely. External attacks can typically be prevented by using standard security mechanisms, such as firewalls, encryption, and so on. Internal attacks are typically more severe attacks, since malicious insider nodes already belong to the network as an authorized party and are thus protected with the security mechanisms the network and its services offer. Thus, such malicious insiders who may even operate in a group may use the standard security means to actually protect their attacks. These kind of malicious parties are called compromised nodes, as their actions compromise the security of the whole ad hoc network.

1.3.2 Denial of Service

The denial of service threat either produced by an unintentional failure or a malicious action forms a severe security risk in any distributed system. The denial of service attack

has many forms. The classical way is to flood any centralized resource so that it no longer operates correctly or crashes, but in ad hoc networks this may not be applicable due to the distribution of responsibility. Distributed denial of service attack is a more severe threat. If the attackers have enough computing power and bandwidth to operate with, smaller ad hoc networks can be crashed or congested rather easily. There are, however, more serious threats to ad hoc networks. For example, compromised nodes may be able to reconfigure the routing protocol or any part of it so that they send routing information very frequently, thus causing congestion or very rarely, thus preventing nodes to gain new information about the changed topology of the network. In the worst case, the adversary is able to change routing protocol to operate arbitrarily or perhaps even in the (invalid) way the attacker wants. If the compromised nodes and the changes to the routing protocol are not detected, the consequences are severe, as from the viewpoint of the nodes the network may seem to operate normally.

1.3.3 Impersonation

Impersonation attacks form a serious security risk at all levels of ad hoc networking. If proper authentication of parties is not supported, compromised nodes in the network layer may be able to join the network undetected or send false routing information masquerading as some other trusted node. Within network management, the attacker could gain access to the configuration system as a super user. At the service level, a malicious party could have its public key certified even without proper credentials. Thus, impersonation attacks are related to all critical operations in ad hoc networks. Impersonation threats are mitigated by applying strong authentication mechanisms in contexts where

a party has to be able to trust the origin of data it has received or stored. Most often this means in every layer the application of digital signature or keyed fingerprints over routing messages, configuration or status information or exchanged payload data of the services in use. Digital signatures implemented with public-key cryptography are as such a problematic issue within ad hoc networks, as they require an efficient and secure key management service and relatively large computation power. Thus, in many cases, more efficient solutions like the use of keyed hash functions or a priori negotiated and certified keys and session identifiers are needed. They do not, however, remove the demand for secure key management or proper confidentiality protection mechanisms.

1.3.4 Disclosure

Any communication must be protected from eavesdropping, whenever confidential information is exchanged. Also critical data that the nodes store must be protected from unauthorized access. In ad hoc networks, such information can include almost anything, e.g. specific status details of a node, location of nodes, private or secret keys, passwords, and so on. Sometimes, control data is more critical information in respect of the security than the actual exchanged data. For instance, the routing directives in packet headers, such as the identity or location of the nodes, can sometimes be more valuable than the application-level messages. This applies especially in critical military applications.

1.3.5 Wormhole Attack

The wormhole attack [12] is a severe threat against ad hoc routing protocols that is particularly challenging to detect and prevent. In a wormhole attack, a malicious node

can record packets (or bits) at one location in the network and tunnel them to another location through a private network shared with a colluding malicious node. Most existing ad hoc routing protocols, without some mechanism to defend against the wormhole attack would be unable to find consistent routes to any destination which can severely disrupting communication. A dangerous attack can be launched silently and successfully, if a wormhole attacker tunnels all packets through the wormhole honestly and reliably since no harm seems to be done: the attacker actually seems to provide a useful service in connecting the network more efficiently. However, when an attacker forwards only routing control messages and not data packets, communication may be severely damaged. As an example, when used against an on-demand routing protocol such as *Dynamic Source Routing* (**DSR**) [13,14], a powerful application of the wormhole attack can be mounted by tunnelling each RREQ message directly to the destination target node of the request. Under some extreme condition, i.e. two colluding attackers are one hop from the sender and the receiver respectively, this attack prevents routes more than two hops long from being discovered because RREP messages would arrive to the source faster than any other replies or, worse, RREQ messages arriving from other nodes next to the destination than the attacker would be discarded since they would have already been seen. In [12], Hu et al. propose a general mechanism, called packet leashes, for detecting and thus defending against wormhole attacks.

1.4 Criteria for Securing Ad Hoc Networks

To secure an ad hoc network, we consider the following criteria: authentication, availability, confidentiality, integrity, and non-repudiation.

1.4.1 Authentication

From a security point of view, the ad-hoc network has a fundamental problem: there are no online servers [10, 11]. Instead, usually a centralized administrator takes care of authorization mechanism. However, the interaction with the administrator can be expensive and time-consuming. The mechanism can be implemented by access control lists or capabilities like signed certificates. The length of valid period of access control lists and capabilities has influence on convenience. Some trade-off between them has to be made.

1.4.2 Availability

Availability means that the service is offered to the user when it is requested. It is a central issue in ad hoc networks that must operate in dynamic and unpredictable conditions. The network nodes may be idle or even be shut down once for a while. Thus, the ad hoc network cannot make any assumptions about availability of specific nodes at any given time. For commercial applications using ad hoc networks, availability is often the most important issue from the viewpoint of the clients. The routing protocol must guarantee the robustness of the routing fabric so that the connectivity of the network is maintained even when threatened by rapid changes in topology or attackers. Finally, many ad hoc networking protocols are applied in conditions where the topology must

scale up and down efficiently, e.g. due to network partitions or merges. The scalability requirements also directly affect various security services, such as key management.

1.4.3 Confidentiality

Confidentiality ensures that certain information is never disclosed to unauthorized entities. Network transmission of sensitive information, such as strategic or tactical military information, requires confidentiality. Leakage of such information to adversaries could have devastating consequences. Routing information must also remain confidential in certain cases, because the information might be valuable for enemies to identify and to locate their targets in a battlefield.

1.4.4 Integrity

Integrity is close to authenticity, but it means that no node has been maliciously changed [10]. In case of integrity, there are similarities between that of ad-hoc network and that of more conventional systems. Integrity has to be preserved. As certificate and identities are not secret, it is possible to clone a device. By securing the public key and letting only the device to know his private key, it is possible to use the certificate in a sensible way. The device has to be tamper-proof, so that his private key cannot be read out and used in a fake device. The best way to avoid attacks is to try to make the node tamper-proof, which is not simple at all. In [15], Anderson and Kuhn state the importance and necessity of design security systems more carefully. The systems have to be tested also against hostile attackers. The physical tamper-evidence mechanisms are often more suitable. The tampering itself can be more than changing a code or key. It can also be

a direct attack to a device.

1.4.5 Non-repudiation

Non-repudiation ensures that the origin of a message cannot deny having sent the message. It is useful for detection and isolation of compromised nodes. When node A receives an erroneous message from node B, non-repudiation allows A to accuse B using this message and to convince other nodes that B is compromised.

1.5 Motivations and Problem Definition of Our Work

The research areas in security for ad hoc networks include key management, secure routing, intrusion detection, availability, fairness, secure sensor networks, and lightweight cryptographic protocols. In this dissertation, we concentrate on two major research fields: key management, and secure routing, more specifically, anonymous secure routing.

1.5.1 Key Management

Among all the security issues in ad hoc networks, key management is the most crucial, because key management is the essential assumption of many other security services. For instance, many secure routing protocols, such as ARAN [16] and SRP [17], assume that a pair of private and public keys and a certificate signed by a *Trusted Third Party* (**TTP**) have been assigned to nodes. The following are key management issues that have been addressed in our work.

Scalability Current research work in key management [1–3, 18–21] can only handle

limited number of nodes. When the number of nodes increases, most of them become either inefficient or insecure.

Flexibility and Adaptivity Since there is no clear line of defense in ad hoc networks, we cannot classify nodes in advance according to the risks they face. Even worse, in some cases, for example soldiers in the battle field, not only do different parts of the network face different degrees of risk, but also such risks may change rapidly because of the dynamic property of ad hoc networks. Therefore, flexibility and adaptivity are two crucial properties that should be considered when we design a key management scheme for ad hoc networks.

Efficiency A major difference between ad hoc networks and wired network is that, in the former, nodes normally have a very limited power supply and computation capability (e.g. CPU and memory). As a result, any protocol that requires high computation cannot be of practical use.

Security and Robustness There are two types of possible attacks on certification services in ad hoc networks: passive attacks and active attacks. In passive attacks, adversaries drop other nodes' requests of assigning or renewing certificates silently. In active attacks, in contrast, adversaries may return a fake reply (e.g. an invalid partial certificate) to the node requesting certification service. Most current approaches for certification services [1–3, 19, 20] mainly focus on passive attacks, and are inefficient in cases where a malicious node launches active attacks. Unfortunately, active attacks are feasible, especially in the military field. Because of the poor physical security in ad hoc networks, adversaries may take over some nodes in the network. By launching active attacks, adversaries can disable the certification services of the whole network without

being caught, with only a very small portion of nodes.

1.5.2 Anonymous Secure Routing

Current research on secure routing in ad hoc networks mainly focuses on confidentiality, integrity, authentication, availability, and fairness, and only recently have researchers begun to pay attention to anonymity [22–25]. Anonymity should be an important part of the overall solution for truly secure ad hoc networks, especially in certain privacy-vital environments. In our dissertation, we address the following issues when designing a new routing protocol for certain privacy-vital environments, e.g. in a battle field.

Privacy We want to protect the privacy of nodes involved in the routing protocol. The privacy information includes the identity and location of the parties in communication, as well as the route information and motion pattern of the source or the destination. Accordingly, we divide the requirements on privacy into three parts: *identity anonymity*, *location privacy*, and *route anonymity*, which are explained in detail in Section 4.1.

Security and Robustness We want to ensure that adversaries cannot obtain the contents of communication (i.e., confidentiality), or disable the communication (i.e., availability and integrity).

Efficiency Current research efforts in anonymous routing are based on flooding route requests, and thus the route discovery is a very costly process. Besides that, cryptographic operations are employed on a hop-by-hop basis. Therefore, any new anonymous routing protocol is expected to be both communicationally and computationally efficient.

1.5.3 Thesis Statement

As a summary, we write the thesis statement of our dissertation as follows:

For key management in ad hoc networks, we propose a novel hierarchical scheme based on threshold cryptography that provides the scalability, flexibility, and adaptivity; in addition, *Verifiable Secret Sharing* (**VSS**) [26, 27], including both the standard schemes and their variants, is employed to defend active attacks against certification services. To overcome the challenges in anonymous secure routing, a new routing protocol is designed to efficiently protect the privacy and security of the nodes involved and the route found in the route discovery process. Our improvements are mainly based on the following ideas: (1) minimize the usages of identity information to protect the anonymity; (2) minimize the usages of cryptographic operations to improve the efficiency and lower the effects of DoS attacks; (3) ensure that the lengths of the meaningful contents of routing packets are invariant to thwart traffic analysis.

1.6 Outline of Dissertation

The dissertation is organized as follows. In Chapter 2, we introduce the background information in cryptography used in our work, and present a comprehensive review of the research work in two major research fields of ad hoc network security: key management and secure routing. In Chapter 3, we propose the *Autonomous Key Management* (**AKM**) scheme for large ad hoc networks and two algorithms for certification services, which can be used independent of AKM. Afterwards, in Chapter 4, we first rigorously define requirements for anonymity and security properties of ad hoc networks, and then

propose the *Efficient Anonymity and Security-Enabled* (**EASE**) protocol that can provide additional properties for anonymity, and at the same time ensure the security of discovered routes against various passive and active attacks. In Chapter 5, we conclude the dissertation.

CHAPTER 2

Related Work

In this chapter, we first introduce the background information in cryptography, including the Shamir secret sharing scheme, proactive security, and verifiable secret sharing. They are fundamental tools in our solutions for key management. Afterwards, related work in key management in ad hoc networks are presented. Following that, we review the anonymity-related research aiming at the Internet, before presenting research work on secure routing in ad hoc networks.

2.1 Background in Cryptography

Efficient threshold schemes can be very helpful in the management of cryptographic keys. In order to protect data, we can encrypt it, but in order to protect the encryption key, we need a different method (further encryptions change the problem rather than solve it). The most secure key management scheme keeps the key in a single, well-guarded location (a computer, a human brain, or a safe). This scheme is highly unreliable since a single misfortune (a computer breakdown, sudden death, or sabotage) can make the information inaccessible. An obvious solution is to store multiple copies of the key at different locations, but this increases the danger of security breaches (computer penetration betrayal, or human errors). To solve this problem, threshold signature was

proposed [28, 29].

2.1.1 Shamir Polynomial Scheme

The first secret sharing scheme was proposed by Shamir in 1979 [30]. This scheme is also called a *(n, k)-threshold scheme*, since it has the following properties: (1) any k or more users can reconstruct the secret from their shares; (2) for $k-1$ or fewer users it is impossible to reconstruct the secret. The integer k is called the *threshold*. The details of the scheme are shown as follows.

Let $p > n$ be a large prime, let $c_1, \cdots, c_n \in Z_p$ be the user identifiers, let k be the threshold, and let $S \in Z_p$ be the secret. A trusted authority chooses random elements $a_1, \cdots, a_{k-1} \in Z_p$ and sets up the polynomial

$$f(x) = a_{k-1}x^{k-1} + \cdots + a_1 x + S \in Z_p[x]$$

of degree of $k-1$, for a k-threshold scheme. The shares are obtained by

$$S_i = f(c_i) \pmod{p} \quad for\ 1 \leq i \leq n$$

and then distributed to the users.

The Shamir threshold scheme has several other nice properties. For instance, it is easy to add new users without changing the shares of the existing users. The trusted authority just chooses a nonzero identifier $c_{n+1} \in Z_p$ that has not been used before and assigns the share $S_{n+1} = f(c_{n+1}) \pmod{p}$. This does not affect the existing shares. Similarly, we can implement various levels of control. If a user is higher up in the

hierarchy, he/she can be provided with multiple shares corresponding to several different user identifiers. This gives more weight to this person in privileged coalitions or in blocking decisions. On the other hand, the Shamir threshold scheme can be used only once. As soon as the members of a privileged coalition have disclosed their shares to recover the secret, these shares are compromised.

2.1.2 Proactive Security

Distribution of the key based on secret sharing makes it harder for an adversary to expose the secret key, but does not remove this risk. Common mode failures, for example those flaws that exist in the implementation of the protocol or the operating system being run on all servers, imply that "breaking into several machines may not be much harder than breaking into one" [31]. Therefore, it is realistic to assume that even a distributed secret key can be exposed. Proactive schemes, e.g. schemes proposed by Feldman [26], Frankel and Desmedt [32], Jarecki [33], Herzberg et al. [34,35], and Frankel et al. [36,37], address this to some extent and make the adversary's task even harder, requiring all of the break-ins to occur within a limited time frame. These schemes are summarized below.

A proactive threshold cryptography scheme uses share refreshing, which enables users to compute new shares from old ones in collaboration without disclosing the service private key, i.e. the shared secret, to any user. The new shares constitute a new (n, k) sharing of the service private key. After refreshing, users remove the old shares and use the new ones to generate partial signatures. Because the new shares are independent of the old ones, the adversary cannot combine old shares with new shares to recover

the private key of the service. Thus, the adversary is challenged to compromise k users between periodic refreshing.

Share refreshing relies on the following homomorphic property. If $(s_{11},\ s_{12}, \cdots,\ s_{1n})$ is an $(n,\ k)$ sharing of k_1 and $(s_{21},\ s_{22}, \cdots,\ s_{2n})$ is an $(n,\ k)$ sharing of k_2, then $(s_{11} + s_{21},\ s_{12} + s_{22}, \cdots,\ s_{1n} + s_{2n})$ is an $(n,\ k)$ sharing of $k_1 + k_2$. If k_2 is 0, then we get a new $(n,\ k)$ sharing of k_1.

Given n users, let $(s_1, s_2, \cdots, s_n)$ be an $(n,\ k)$ sharing of the private key of the service, with user i having s_i. Assuming all users are correct, share refreshing proceeds as follows: first, each user randomly generates $(s_{i1}, s_{i2}, \cdots, s_{in})$, an $(n,\ k)$ sharing of 0. We call these newly generated s_{ij}'s subshares. Then, every subshare s_{ij} is distributed to user j through a secure link. When user j gets the subshares $s_{1j}, s_{2j}, \cdots, s_{nj}$, it can compute a new share from these subshares and its old share ($s'_j = s_j + \sum_{i=1}^{n} s_{ij}$). Figure 2.1 illustrates the share refreshing process.

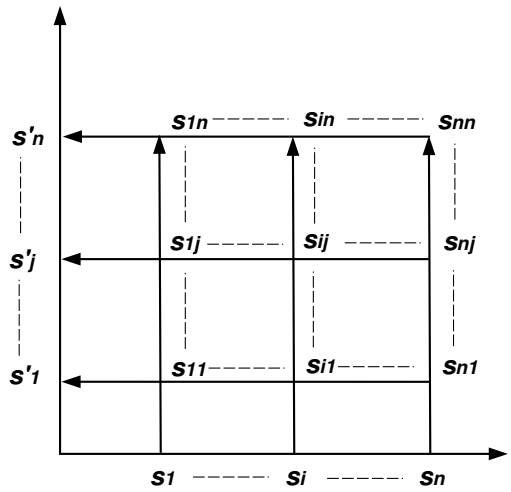


Figure 2.1: Share refreshing process based on homomorphic property

2.1.3 Verifiable Secret Sharing Schemes

There are two weaknesses in the Shamir secret sharing scheme.

Honesty of the dealer If the dealer distributes erroneous subshares to some or all the users, how can users verify whether the subshares received are correct?

Honesty of users During the share recovery process, if some compromised users provide false subshares, how can other users detect them?

Share refreshing must tolerate missing subshares and erroneous subshares from compromised users. A compromised user may not send any subshares. However, as long as correct users agree on the set of subshares to use, they can generate new shares using only subshares generated from k users in an (n, k) secret sharing scheme.

To detect incorrect subshares, several VSS schemes [26, 27, 38–44] were proposed. A verifiable secret sharing scheme generates extra public information for each (sub)share using a one-way function. The public information can testify the correctness of the corresponding (sub)shares without disclosing the (sub)shares.

Verifiable secret sharing schemes can be divided into two categories: non-interactive and interactive. Non-interactive VSS schemes by Feldman [26] and Pedersen [27] assume that the dealer broadcasts a non-interactive zero-knowledge proof to the shareholders, and only the dealer may be faulty. Interactive VSS schemes assume that either the dealer or some of the shareholders may be faulty, and include multiple rounds of communication between the dealer and the shareholders to identify faulty participants; representative examples include schemes by Chor et al. [38], Benaloh [39], Gennaro and Micali [40, 41], Goldreich et al. [42], and Rabin and Ben-Or [43, 44]. The scheme of Gennaro and Micali

[40, 41], and that of Rabin and Ben-Or [43, 44], are information-theoretically secure.

In this section, we present the scheme proposed by Pedersen [27] to show the procedure for achieving verifiable secret sharing. Firstly, we introduce the commitment scheme used in the Pedersen's VSS scheme [27]. Let g and h be elements of G_q, such that nobody knows $log_g h$. These elements can either be chosen by a trusted center, when the system is initialized, or by (some of) the participants using a coin-flipping protocol. The committer commits himself to an $s \in \mathbb{Z}_q$ by choosing $t \in \mathbb{Z}_q$, at random and computing

$$E(s,\ t) = g^s h^t \tag{2.1}$$

Such a commitment can later be opened by revealing s and t. It is proved in [27] that $E(s,\ t)$ reveals no information about s, and that the committer cannot open a commitment to s as $s' \neq s$ unless he can find $log_g(h)$.

Assume that a dealer, D, has a secret $s \in \mathbb{Z}_q$ and wants to distribute it among n parties, $P_1, \ldots,\ P_n$, such that any k of the shareholders can find s if necessary, but less than k shareholders get no (Shannon) information about s (a $(k,\ n)$-threshold scheme). Pedersen extend the Shamir scheme with a *Verification Protocol* (**VP**) that satisfies the following two requirements:

1. If the dealer follows the distribution protocol and if the dealer and P_i both follow VP, then P_i accepts with probability 1.
2. For all subset S_1 and S_2 of $1, \ldots,\ n$ of size k such that all parties $(P_i)_{i \in S_1}$ and $(P_i)_{i \in S_2}$, have accepted their shares in VP the following holds except with negli-

gible probability in $|q|$: If s_i is the secret computed by the participants in S_i (for $i = 1,\ 2$) then $s_1 = s_2$.

A share is called correct, if it is accepted in VP. These two requirements guarantees that the dealer will be almost always (with negligible probability in $|q|$) caught when trying to cheat, i.e. distributing inconsistent shares.

Detailed steps of the VSS scheme proposed in [27] are presented as follows. By the fact that $\mathbb{Z}_q$ is a field, the dealer can distribute $s \in \mathbb{Z}_q$ as follows.

1. D publishes a commitment to s: $E_0 = E(s,\ t)$ for a randomly chosen $t \in \mathbb{Z}_q$.
2. D chooses $F \in \mathbb{Z}_q[x]$ of degree at most $k-1$ satisfying $F(0) = s$, and computes $s_i = F(i)$ for $i = 1, \ldots,\ n$. Let $F(x) = s + F_1 x + \ldots + F_{k-1}x^{k-1}$. D chooses $G_1, \ldots, G_{k-1} \in \mathbb{Z}_q$ at random and use G_i when committing to F_i for $i = 1, \ldots, k-1$ D broadcasts $E_i = E(F_i,\ G_i)$ for $i = 1, \ldots,\ k-1$.
3. Let $G(x) = t + G_1 x + \ldots + G_{k-1}x^{k-1}$ and let $t_i = G(i)$ for $i = 1, \ldots,\ n$. Then D sends $(s_i,\ t_i)$ secretly to P_i for $i = 1, \ldots,\ n$.

When P_i has received his share $(s_i,\ t_i)$ he verifies that

$$E(s_i,\ t_i) = \prod_{j=0}^{k-1} E_j^{ij} \tag{2.2}$$

Let $S \subset 1, \ldots,\ n$ be a set of k participants such that Equation 2.2 holds for these k parties. Then these k parties can find a pair $(s',\ t')$ such that $E_0 = g^{s'}h^{t'}$. Also, the members in S can find the secret s using the following formula:

$$s = \sum_{i \in S} a_i s_i \quad where \quad a_i = \prod_{j \in S,\ j \neq i} \frac{i}{i-j} \tag{2.3}$$

2.2 Key Management in Ad Hoc Networks

In general, security goals in ad hoc networks are achieved through cryptographic mechanisms such as public key encryption and digital signature. For example, the basic assumption adopted by some secure routing protocols such as SRP [17] is the existence of an a priori security association between all the communicating nodes of the network. The limitations introduced by this approach range from the need of a managed environment, such as a common authority that pre-charges all the ad hoc nodes with a secret key shared by every pair of communicating nodes, to scalability problems.

Other secure routing protocols, for example Ariadne [45], rely on an initialization phase during which a well known TTP issues public key certificates used to authenticate (together with the private key of each certificate holder) hash chain elements that will be subsequently used to provide a low cost (in terms of CPU usage) authentication service. In this case, the use of such a secure protocol is not limited to the managed environment, and the open environment can be targeted. In other words, the authority that manages the mobile nodes in the ad hoc network may not be the one that provides the initial authentication setup. However, the bootstrap phase requires an external infrastructure, which has to also be available during the lifetime of the ad hoc network to provide revocation services. To solve this problem, several schemes [1–3,18–21,46–50] have been proposed to distribute certificate-based public-key in a secure way. We classify them into three categories: partially distributed key management, fully distributed key management, and self-organizing public key management. In partially distributed key management, the privilege of assigning a certificate is shared within a group of trusted

servers, while in fully distributed key management, it is shared within all the users in the network. As to self-organizing public key management, every user in the network acts as a *Certificate Authority* (**CA**) and issues public-key certificates to other users.

2.2.1 Partially Distributed Key Management

In [18], Zhou and Haas focus on the establishment of a secure key management service in an ad hoc networking environment. They propose the use of threshold cryptography [29, 51] to distribute trust, i.e. the functionality of generating a certificate, among a set of servers. An (n, t+1) threshold cryptography scheme is employed. The private key k of the service is divided into n shares $(s_1, s_2, \cdots, s_n)$, each of which is assigned to one server. Each server i also has a public/private key pair K_i/k_i and knows the public keys of all nodes. Figure 2.2 illustrates how the service is configured.

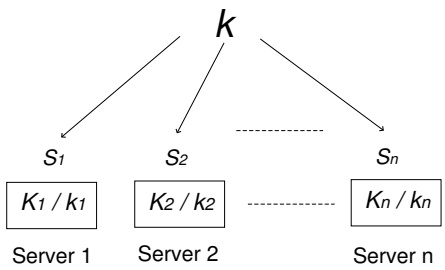


Figure 2.2: Secret sharing of the private key k

For the service to sign a certificate, each server generates a partial signature for the certificate using its private key share and submits the partial signature to a combiner. With $t + 1$ correct partial signatures, the combiner is able to compute the signature for the certificate. However, compromised servers (there are at most t of them) cannot

generate correctly signed certificates by themselves, because they can generate at most t partial signatures.

Figure 2.3 shows how servers generate a signature using a (3, 2) threshold signature scheme. Given a service consisting of 3 servers. Let K/k be the public/private key pair of the service. Using a (3, 2) threshold cryptography scheme, each server i gets a share s_i of the private key k. For a message m, server i can generate a partial signature $PS(m, s_i)$ using its share s_i. Correct servers 1 and 3 both generate partial signatures and forward the signatures to a combiner c. Even though server 2 fails to submit a partial signature, c is able to generate the signature $(m)_k$ of m signed by service private key k.

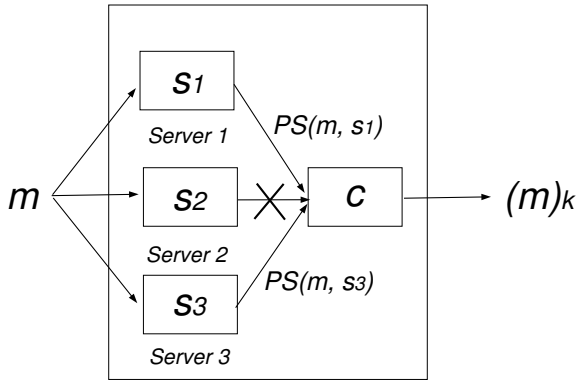


Figure 2.3: A (3, 2) threshold signature scheme

The focus of their work is to maximize the security of the shared secret in the presence of possible compromises of the secret share holders. It assumes a small group of servers with rich connectivity. However, the key establishment issue of normal nodes is not addressed. Furthermore, the authors propose to use proactive schemes [33–37] of share refreshing and to adapt to changes in the network in a scalable way.

In [46], Yi and Kravets propose a scheme named Mobile Certificate Authority (MOCA). Similar to [18], this scheme limits the candidates who hold a share of the secret key to a subset of users denoted as MOCA. However, it leads to difficulties, such as judging the level of security, choosing MOCAs, and ensuring that MOCAs are distributed uniformly, etc. Besides that, they assume the existence of some nodes that are more trustworthy, computationally more powerful, and physically more secure. Therefore, the scheme is inappropriate for purely ad hoc networks. They also propose a new pattern of communication, termed as "Manycast" [46], between a client and MOCAs. The pattern is based on the assumption that the client knows the location and identities of the MOCA nodes, and thus is not suitable for purely ad hoc networks. Furthermore, the authors ignore potential passive and active attacks from malicious users, which may weaken the benefit of employing MOCAs to save the packet overhead.

2.2.2 Fully Distributed Key Management

Based on the process of generating shared key, fully distributed key management can be divided into two types: shared key generation based on RSA [52] and shared key generation based on *Discrete Logarithm Problem* (**DLP**) [53]. Besides that, researchers also try to make use of ID-based cryptography [54–56], which can be combined with these two types of fully distributed solutions, in key management.

2.2.2.1 Shared Key Generation Based on RSA

In [1–3], Kong et al. also use threshold secret sharing to distribute the functions of the CA but extend it to normal nodes. In the proposed scheme, any two communicating

nodes can establish a temporary trust relationship via globally verifiable certificates. With a scalable threshold sharing of the certificate-signing key, certification services (issuing, renewal and revocation) are distributed among every node in the network. A single node holds just a share of the complete certificate-signing key. While no single node has the power of providing full certification services, multiple nodes in a network locality can collaboratively provide such services. It minimizes the effort and complexity for mobile clients to locate and contact the service providers.

The authors propose a localized trust model to characterize the localized nature of security concerns in large ad hoc wireless networks. When applying such a trust model, an entity is trusted if any k trusted entities claim so: these k trusted entities are typically the neighboring nodes of the entity. A locally trusted entity is globally accepted and a locally distrusted entity is regarded untrustworthy anywhere. k is a system wide parameter that sets the global acceptance criteria and should be honored by each entity in the system.

The basic assumptions necessary for their scheme to function properly are that: (1) each node has a unique nonzero identifier; (2) each node has some one-hop discovery mechanism; (3) each node has at least k one-hop legitimate neighboring nodes, or the network has a minimum density of well-behaving nodes; (4) each node is equipped with some detection mechanism to identify misbehaving nodes among its one-hop neighborhood; (5) the mobility is characterized by a maximum node moving speed.

The two main issues presented in [1–3] are certification services and system maintenance (i.e. the initialization and updates of secret shares).

Certification services include certificate renewal, certificate revocation and *Certificate*

Revocation List (**CRL**) maintenance.

There are two different methods provided to assign a new certificate or to renew certificates. For example, node i sends a request for a new certificate. As shown in Figure 2.4, in the first method, node i selects k nearby nodes which form a group B in advance, and sends its request within group B. Each node receiving the request will generate a partial certificate and returns it to node i. Finally, node i combines all the k partial certificates to generate the new certificate. The second method is similar to the first one, except that the member of group B is not decided at the beginning. Instead, node i first broadcasts its request, and then selects any k replies from all the replies which form group B. The process of the second method is shown in Figure 2.5.

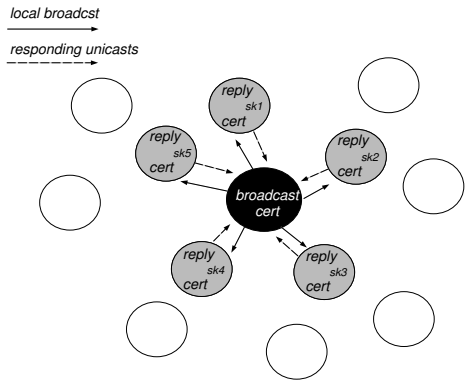


Figure 2.4: The first method of certificate renewal

As to certificate revocation, it is based on the accusations from other nodes. If node i receives more than k accusations on node j, it adds node j to its CRL database.

In [1–3], a dealer is required to initialize the first k nodes. After that, it destroys all the secrets and quits. These k nodes then cooperate to distribute new shares to nearby nodes. As more and more nodes are initialized, they further serve to initialize their

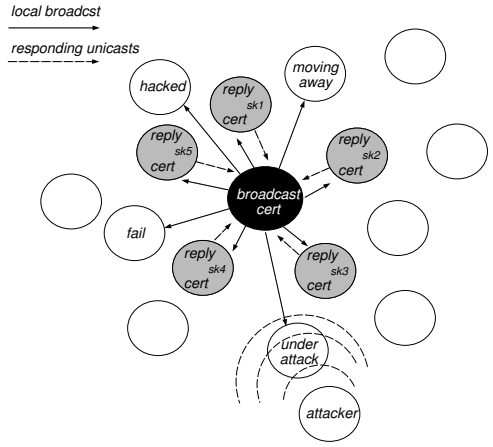


Figure 2.5: The second method of certificate renewal

neighboring nodes, thus generating a diffusion process.

The process of self-initialization and share updates, in fact, is similar to the first method of certification renewal, but this time each serving node returns a partial share instead of a partial certificate. To prevent the secret shares held by each serving node from leaking to other nodes including the requesting node, a complete shuffling scheme is executed. The process is shown in Figure 2.6.

One of the two major weaknesses of this scheme is that it is difficult to set an appropriate threshold k, which is a globally fixed parameter that is honored by each entity in the system. This scheme assumes that each node has *at least* k one-hop legitimate neighboring nodes and the deal needs to initialize k nodes at the beginning. If k is set to a large number, many nodes may have problems in localizing certification service and the initialization of secret shares. On the other hand, if k is too small, the probability of global private key being compromised is quite high. It is also inefficient to change the threshold according to the size of the system, since in the proactive

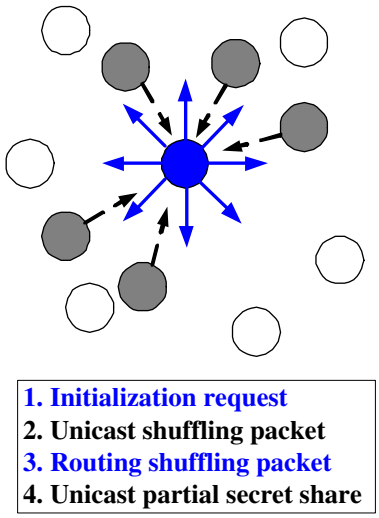


Figure 2.6: The process of self-initialization of secret shares

secret sharing scheme the computational complexity of changing the configuration of the threshold scheme is very high. Another serious problem is that, due to the inability of distinguishing adversaries who provide invalid partial certificates from honest nodes, both of their algorithms for certificate renewal are vulnerable to active attacks, such as send invalid partial certificates which result in the failure of renewing or assigning a certificate. In addition, such algorithm is infeasible for a large ad hoc network, because the time needed to initialize all the nodes could be so long that it results in asynchronism in the secret scheme. In particular, the first k initializing nodes may have to refresh their secret shares before all the nodes in the network have obtained their secret shares.

In [19], Lehane et al. present a similar scheme based on shared RSA key generation, and thus share the pros and cons of the scheme proposed by Kong et al. [1–3]. In addition, according to their empirical results, the efficiency of their protocol is not good.

2.2.2.2 Shared Key Generation Based on Discrete Logarithm Problem

One recent work by Narasimha et al. [47, 48] point out that the algorithm proposed by Kong et al. [1–3] is vulnerable to active attacks, and propose a Threshold DSA signature scheme. Their scheme is a standard VSS process based on DLP, and thus is very costly. In addition, the standard VSS process requires almost twice as many as neighbors involved in the certification service within one round. More specifically, given that, there are $k-1$ adversaries in the network, the Threshold DSA signature scheme [47, 48] requires helps from $2k-1$ neighbors.

2.2.2.3 ID-based Key Management

In [20], Khalili et al. try to combine the ideas of ID-based [54–56] and threshold cryptography [29, 51]. Their scheme avoids the need for users to generate their own public keys and distribute these keys throughout the network, since the user's identity acts as her public key. Besides that, users only need to propagate their identities instead of the certificates. This can lead to huge savings in bandwidth. One major weakness of this scheme is that it is vulnerable to active attacks. More specifically, malicious members of the network can provide newly-joining members with a false master public key, perhaps one for which the malicious member holds the corresponding master secret key.

2.2.3 Self-organizing Public Key Management for Mobile Ad Hoc Networks

In [21, 49, 50], Čapkun et al. propose a public-key management system for fully self-organizing mobile ad hoc networks in which there is no infrastructure, no central au-

thority, no centralized trusted third party, no central server, no secret share dealer, even in the initialization phase. In [57], this self-organizing trust relationship is used to build a secure routing protocol out of an incomplete set of security associations. In [58], Čapkun et al. show that mobility can enhance security in mobile ad hoc networks based on the fully self-organizing security associations.

Their approach is similar to PGP [59,60] in the sense that users issue certificates for each other based on their personal acquaintances. However, in their proposed system, certificates are stored and distributed by the users themselves, unlike in PGP, where this task can be performed by online servers (called certificate directories). Figure 2.7 illustrates a simple example of how PGP works. Bob has issued a certificate to Chris thus stating that pk_{Chris} really is the public key belonging to Chris. Alice has also issued a certificate to Bob, indicating that pk_{Bob} is really the public key belonging to Bob. Alice also trusts Bob not to issue any false certificates, thus Alice will trust any certificates issued by him. Therefore having $cert_{Alice,Bob}$ and $cert_{Bob,Chris}$, Alice can verify that pk_{Chris} is authentic. She can then securely communicate with Chris even though they have never met.

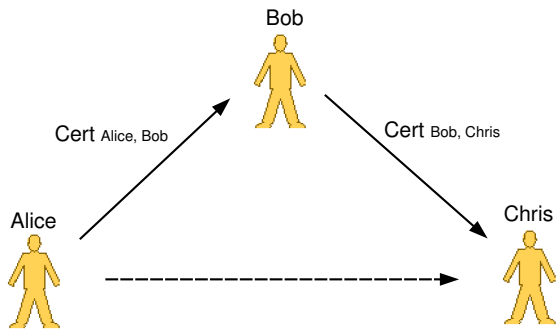


Figure 2.7: An example of a certificate chain

2.2.3.1 System Overview

The proposed self-organizing public-key management system works in two phases, as shown in Figure 2.8.

1. Initialization: users construct their local certificate repositories.
2. When two users want to verify the public keys of each other, they merge their local certificate repositories and try to find appropriate certificate chains within the merged repository that make the verification possible.

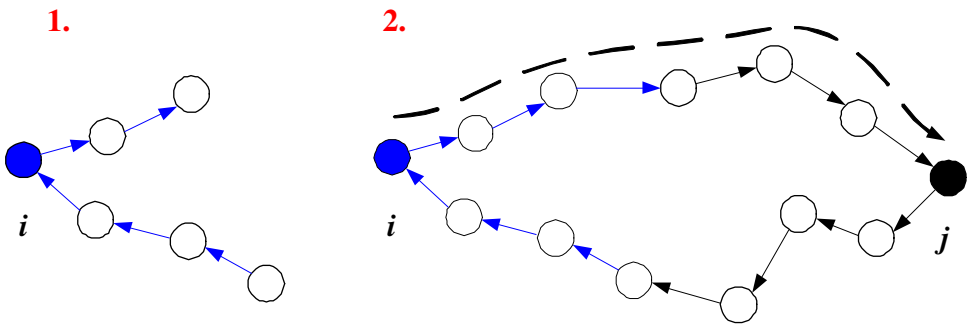


Figure 2.8: Two phases of how the system works

The success of this approach very much depends on the construction of the local certificate repositories and on the characteristics of the certificate graphs. In a certificate graph, vertices represent public keys of the users, and the edges represent public-key certificates issued by the users. Each user stores a local repository of public-key certificates, which can be considered as a subgraph. These certificates can be placed in three categories: (1) the certificates issued by the user (outgoing edges); (2) the list of certificates that others issued for the user (incoming edges); (3) an additional set of certificates chosen according to some algorithm.

In [21, 49, 50], Čapkun et al. propose several repository construction algorithms and study their performance. The proposed algorithms take into account the characteristics of the certificate graphs in the sense that the choice of the certificates that are stored by each user depends on the connectivity of the user and her certificate graph neighbors.

More precisely, each user stores in her local repository several directed and mutually disjoint paths of certificates. Each path begins at the user herself, and each certificate on the path is chosen from the set of certificates that are connected to the last selected user on the path in such a way that the chosen certificate leads to a user that has the highest number of certificates connected to her (i.e., the highest vertex degree). This algorithm is called the *Maximum Degree Algorithm*, as the local repository construction criterion is the degree of the vertices in the certificate graph.

In a second, more sophisticated algorithm, certificates are selected into the local repositories based on the number of the shortcut certificates connected to the users. Here, a shortcut certificate is defined as a certificate such that when it is removed from the graph, the shortest path between the two users previously connected by this certificate becomes strictly larger than two. This algorithm is called the *Shortcut Hunter Algorithm*.

2.2.3.2 Weaknesses

This method has a few weaknesses. Firstly, before being able to perform key authentication, each user must first build her local certificate repository. Such initialization phase is relatively expensive (in terms of bandwidth and time). Although the authors claim that such operations are performed rarely, other protocols, such as those in [1–3],

can help each node obtain its certificate with less costly initialization process, and such process is also performed once only.

Secondly, the local repositories become obsolete if a large number of certificates are revoked, as then the certificate chains are no longer valid; the same comment applies in the case when the certificate graph changes significantly (e.g., a large number of new users join the system). Authors argue that key revocations occur very rarely as the user-key bindings that are expressed by certificates do not change very frequently. Such claim is not true in ad hoc networks, since the probability of physical compromises of nodes is much higher than in wired networks. And such attack leads to certificate revocations of compromised nodes.

Thirdly, to achieve better assurance about the user-key binding, authentication metrics is used. However, deciding which metric is the most appropriate for applications in ad hoc networks remains an open issue. Even if such a metric exists, the dynamic property of certificate revocation results in very high computation costs in reconstructing local certificate repositories of all the nodes.

Finally, the metric's confidence in someone's honesty can be easily cheated, as any user can create an arbitrary number of public keys and issue many false certificates. Other metrics can also be defined, but currently there is no efficient metric that can be used to determine confidence in user's honesty based on the certificates in the certificate graph.

	Schemes	Existence of online CAs	Existence of offline CAs	Responsibility of a normal node in cert. services
partially distributed key management	[18], [46]	Yes	N/A	None
fully distributed key management	[1–3], [19], [47, 48], [20]	No	Yes	cooperate with other nodes to provide cert. services
self-organizing public key management	[21, 49, 50]	No	No	provide cert. services by itself

Table 2.1: Comparison of key management schemes for ad hoc networks

2.2.4 Summary

In Table 2.1, we summarize previous research work in key management for ad hoc networks, in terms of the existence of online/offline CAs and the responsibility of a normal node in certification services.

[18,46] exploit the idea of secret sharing to achieve robust key management in highly dynamic environments, such as ad hoc networks. They aim at key servers only, and assume the existence of a group of online CAs. Therefore, partially distributed key management is not suitable for purely ad hoc networks. Following efforts [1–3, 19, 20, 47, 48] extend the secret sharing process to normal nodes. Every node in the network holds a share of the global secret key that can be used to sign a new certificate. A certain number (i.e. the threshold of the secret sharing scheme employed) of nodes can cooperate together to assign a certificate to a new node. Current fully distributed key management schemes work fine when the size of the network is small. Unfortunately, it becomes either insecure or inefficient when the size of the network increases, and it is also vulnerable to active attacks. As shown in Table 2.1, self-organizing public key

management [21, 49, 50] has an advantage that it does not require the existence of either online or offline CAs[1]. However, there still exist many problems to be solved, e.g. reduce the cost of the initialization phase and find appropriate authentication metrics, before being employed in practice.

2.3 Privacy and Anonymity on the Internet

Previous research work on privacy and anonymity on the Internet concentrate on two issues: user anonymity and anonymous communication. User anonymity aims at providing the users anonymity while they are using the network by letting them hide their identity from the communicating peers. Research on anonymous communication focuses on providing a communication channel that is immune to traffic analysis so that the communicating parties can be anonymous against the eavesdroppers.

2.3.1 Mix

The basic building block for achieving privacy and anonymity on the Internet was presented by David Chaum in 1981 [61]. This basic building block is called a *mix*. A Mix accepts pieces of encrypted e-mail for delivery from many sources, holds, re-sorts, possibly introduces null mail, rewrites the from address, and possibly sends the mail to another mix for delivery. The end goal is each piece of output mail is equally likely to have come from any original sender. Generically, Mixes need not accept only email, it will become clear from the context what the Mixes in each protocol accept. The

[1]In this table, CA means the certificate authority that is trusted by all the users in the network. Thus, we say there is no CA in self-organizing public key management, although in each node can act as a CA and assign other nodes certificates

following anonymous protocols base their anonymity on his work.

2.3.2 User Anonymity

Anonymizer [62] is a user anonymity solution, which prevents online tracking by blocking the real IP address of the user. Users can achieve anonymity by rerouting their HTTP connections through the Anonymizer, which replaces the information in the connection so that the websites cannot infer the users' identities. This approach has the problem of a centralized trusted entity. The Anonymizer site can track all the anonymous user activities and is also a single point of failure.

In [63], Reiter and Rubin introduce a system called *Crowds* for protecting users' anonymity on the Internet. Crowds extends the idea of Anonymizer by introducing many computers in the communication path between the initiator (web-surfer) and the receiver (the web server). In this manner, no single point of failure compromises the sender's anonymity. Upon receiving one request, each member of Crowds can either submit the request directly to the end server or forward it to another random chosen member based on a certain probability. When the request is eventually submitted, it is submitted by a random member, thus preventing the end server from identifying its true initiator. Crowds provides sender anonymity, not receiver, because the end server is contacted by the last member directly.

WebMIX [64] provide a system similar to Crowds [63] and Hordes [65, 66], but addresses some of the vulnerabilities inherent in Mix-based systems. In particular, a ticket-based authentication system is presented to prevent DoS attacks. In such a system, participants on the network are issued a ticket to use the network. If the participant de-

sires to communicate anonymously, she redeems her ticket and sends her message. This prevents users from flooding the network with traffic, WebMIX also addresses Mixes that drop messages or break messages by changing their contents. Each Mix will digitally sign each message as a sign that it did not tamper with the message. When a Mix cannot verify the digital signature of a given message, it flags a signature error. Each Mix is then forced to verify that it did not tamper with the message by publishing the encrypted message it received and its decrypted version.

2.3.3 Anonymous Communication

In [65, 66], Shields and Levine present a protocol, *Hordes*, for providing anonymous communication in the Internet. Hordes builds on the Crowds work [63], instead of serving as a proxy for a HTTP connection, the Mixes in Hordes serve as proxies for UDP connections. Path creation works analogous to Crowds, however, a multicast return is used instead of the reverse path. When an initiator sends a request, a multicast address is picked on which return responses should be broadcast. Hordes provides only sender anonymity, not receiver, because at some point the last Mix makes a connection to the end receiver. Unless the receiver also participates in an anonymous protocol, an attacker may perceive it as receiving data (and possibly responding).

One approach for anonymous communication is *Onion Routing* [67]. Such approach requires a set of onion routers, and anonymous connections through the network are multiplexed over longstanding socket connections among onion routers. One advantage of this approach is that, each onion router can only identify the previous and next hop router along a route, and data cannot be tracked en route. However, the sequence of

onion routers in a route is strictly defined at connection setup. When a participant desires to send a message, they a priori choose a set of onion routers, encrypt their message in the public keys of the Onion Routers and send the message, or onion on its way. Onion routing tries to make an initiator's request appear equally likely to come from any onion router. Therefore, sender anonymity is provided in onion routing.

Tor [68] is the next generation onion router which many improvements over the old onion routing design. For example, Tor separates the line between anonymity and "data cleansing". Data cleansing is the process of removing personal information (e.g. cookies, Active X Objects, etc), from message such as requests to web servers (HTTP). Many web servers will tag visitors with cookies to track user activity and discover how often visitors come back. Removal of this information is important to maintaining anonymity, Tor suggests the use of Privoxy [69], when using Tor to communicate anonymously. Tor also addresses the need for receiver anonymity or location-hidden services by introducing "rendezvous points"[2]. Location-hidden services allow the receiver to offer a TCP service, such as a web server, without revealing his IP address. This type of anonymity protects against distributed DoS attacks: attackers are forced to attack the onion routing network because they do not know the receiver's IP address.

2.3.4 Summary

Many solutions [61–68] have been proposed for protecting privacy and anonymity in the Internet environments, and some of them can be used in ad hoc networks as well. For example, both SDAR [23, 70–72] and ANODR [22] make use of the onion structure

[2]A rendezvous point is an onion router that contains instruction about how to contact a receiver anonymously.

proposed in [67]. However, because of several unique characteristics of ad hoc networks shown in Section 1.2, e.g. no central management, dynamic topology and membership, energy-constrained operation, and limited physical security, previous solutions workable in the Internet environments cannot be used directly in ad hoc networks. For example, in the onion routing scheme [67], an onion proxy undertakes the task of defining the route. Therefore, if the onion proxy is compromised, the anonymity property of routes is compromised at the same time. Besides that, the assumption of the existence of such an onion proxy is unsuitable for purely ad-hoc networks.

2.4 Secure Routing

Outside the ad hoc network community, secure routing in the Internet has received increased attention. The proposed solutions rely mainly on the existence of a line of defense, separating the fixed routing infrastructure from all other network entities. This is achieved by distributing a set of public keys/certificates, which signify the authority of the router to act within the limits of the employed protocol (e.g., advertise certain routes), and allow all routing data exchanges to be authenticated, non-repudiated and protected from tampering. However, such approaches cannot combat a single malicious router disseminating incorrect topological information. In addition, they are not applicable in the ad hoc network context because of impediments such as the absence of a fixed infrastructure and a central entity. More importantly, for an environment where node mobility and network topology rapidly change, such protocols also have high communication overhead because they send periodic routing messages even when

the network is not changing. The provision of comprehensive secure communication mandates that both *data forwarding* and *route discovery* be safeguarded.

Existing ad hoc routing protocols, such as DSR [13, 14] and *Ad Hoc On-demand Distance Vector* (**AODV**) [73, 74], are vulnerable to many kinds of attacks. It is fairly easy to inject fake routing messages or modify legitimate ones such that the operation of the network would be heavily disturbed (e.g., by creating loops or disconnecting the network). Therefore, many secure ad hoc routing protocols [16, 17, 45, 75–80] were proposed to make the routing in ad hoc networks resistant to various kinds of attacks. However, till recently, researchers begin to consider the anonymity issue [22–25], which is very important in ad hoc environments where the trust relationship between users is relatively weak.

2.4.1 Generic Secure Routing and Data Forwarding Protocols for Ad Hoc Networks

In this section, we review prior research in generic secure routing and data forwarding protocols for ad hoc networks. "Generic" in the context of this section means that, in these protocols, the authors consider only security requirements, and do not take additional requirements, e.g. anonymity, into account.

2.4.1.1 Promiscuously Auditing on Next Node

In [75], Marti et al. propose two mechanisms for detecting packet dropping. One is to detect misbehaving nodes and report such events, and the other is to maintain a set of metrics reflecting the past behavior of other nodes. The main idea behind is that

nodes work in the promiscuous mode, and overhear the transmissions of their successors. Not only do these nodes verify whether the packet was forwarded to the downstream node, but they also check the integrity of the forwarded packet. The assumptions for the above-mentioned work include a shared medium, bi-directional links, use of source routing (i.e., packets carry the entire route that becomes known to all intermediate nodes), and no colluding malicious nodes. It is also called as the *Watchdog* method.

Figure 2.9 illustrates an example to show how it works. A packet is transferred from S toward D through A, B, and C in sequence. After forwarding the packet to B, A can overhear B's transmission and verify that B has attempted to pass the correct packet to C. The solid line represents the intended direction of the packet sent by B to C, while the dashed line indicates that A is within transmission range of B and can overhear the packet transfer. Once a misbehaving node is detected, a report is generated and nodes update the rating of the reported misbehaving node. Because nodes may choose the best route comprised of relatively well-behaved nodes during data forwarding, the node which previously launched the packet dropping attack is bypassed.

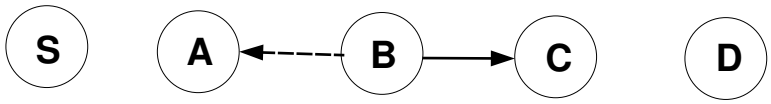


Figure 2.9: An Example of The Watchdog Method

The watchdog method exploits two features of ad hoc networks, namely, shared channel and source routing. Nevertheless, the plausibility of this solution is questionable for several reasons, and in fact the authors themselves provide a short list of scenarios of incorrect detection: (1) ambiguous collisions, (2) receiver collisions, (3) limited trans-

mission power, (4) false misbehavior, (5) collusion, and (6) partial dropping.

2.4.1.2 Secure Message Transmission

Secure Message Transmission (**SMT**) [76, 77] is a secure data forwarding protocol proposed by Papadimitratos and Haas. Given a topology view of the network, SMT determines a set of diverse paths between the source and the destination. Then, it introduces limited transmission redundancy across the paths, by dispersing a message into N pieces, so that successful reception of any M-out-of-N pieces allows the reconstruction of the original message at the destination. Upon reception of a number of pieces, the destination generates an acknowledgement informing the source of which pieces, and thus routes, were intact. In order to enhance the robustness of the feedback mechanism, the small-sized acknowledgments are maximally dispersed (i.e., successful reception of at least one piece is sufficient) and are protected by the protocol header as well.

SMT takes advantage of topological and transmission redundancies and utilizes feedback, exchanged only between the two communicating end-nodes, and thus remains effective even under highly adverse conditions. It does not impose processing overhead on intermediate nodes, while the end nodes make the routing decisions, based on the feedback provided by the destination and the underlying topology discovery and route maintenance protocols. Besides that, the fault-tolerance of SMT is enhanced by the adaptation of the ratio of required pieces to the total number of pieces. It can ensure 100% successful data transmission, even if 10 to 20 percent of the network nodes are malicious. The main weakness of SMT is that it is based on the correctness of the topological and routing information, which unfortunately has limited protection.

2.4.1.3 Secure-AODV

In [78], Zapata proposes the *Secure-AODV* (**S-AODV**) scheme to protect the AODV [73, 74] routing protocol. S-AODV assumes that each node has certified public keys of all network nodes, so that intermediate nodes can validate all in-transit routing packets. More specifically, the sender appends an RSA signature [52] and the last element of a hash chain [81] (i.e., the result of hashing a random number for n times). When the message is forwarded, intermediate nodes validate the signature and the hash value, generate the k-th[3] element of the hash chain and place it in the packet.

The main weaknesses of S-AODV includes: (1) the IP portion of the S-AODV packets can be easily compromised; (2) intermediate nodes can pretend that they are the neighbor of the destination, advertise arbitrarily high sequence numbers, and modify the route length; (3) the use of public-key cryptography imposes a high processing overhead on the intermediate nodes; (4) certificates are bound with fixed IP addresses. However, in ad hoc networks, newly-joined nodes may be assigned a IP address dynamically (e.g., DHCP [82]) or even randomly (e.g., Zero-Configuration [83]).

2.4.1.4 Secure Routing Protocol

Panagiotis Papadimitratos and Zygmunt Haas propose the *Secure Routing Protocol* (**SRP**) [17], which can be used with DSR [13, 14] or the *IntErzone Routing Protocol* (**IERP**) [84] in the *Zone Routing Protocol* (**ZRP**) [85, 86]. SRP requires a security association only between communicating nodes and uses this security association to authenticate route requests and route replies through the use of message authentication

[3] k is the number of traversed hops.

codes. The novelty of the scheme is that false route replies, as a result of malicious node behavior, are discarded partially by good nodes while in-transit towards the querying node, or deemed invalid upon reception.

SRP does not attempt to prevent unauthorized modification of fields. For example, a node can freely remove or corrupt the node list of a route request packet that it forwards. Besides that, to limit flooding, in SRP nodes record the rate at which each neighbor forwards route request packets, and gives priority to those route request packets sent through neighbors that less frequently forward the route request packets. Such techniques exacerbate the problem of greedy nodes. For example, a node that does not forward the route request packets gets a higher priority when it initiates route discovery.

2.4.1.5 Authenticated Routing for Ad-hoc Networks

The *Authenticated Routing for Ad-hoc Networks* (**ARAN**) secure routing protocol [16] proposed by Dahill et al. is conceived as an on-demand routing protocol that detects and protects against malicious actions carried out by third parties and peers in the ad hoc environment. ARAN introduces authentication, message integrity and non-repudiation as part of a minimal security policy for the ad hoc environment and consists of a preliminary certification process, a mandatory end-to-end authentication stage and an optional second stage that provides secure shortest paths.

The ARAN protocol protects against exploits using modification, fabrication and impersonation but the use of asymmetric cryptography makes it a very costly protocol to use in terms of CPU and energy usage. Furthermore, ARAN is not immune to the wormhole attack.

2.4.1.6 Secure Efficient Ad Hoc Distance Vector Routing Protocol

Hu, Perrig and Johnson present a proactive secure routing protocol, *Secure Efficient Ad-hoc Distance Vector Routing Protocol* (**SEAD**) [79], based on the *Destination-Sequenced Distance Vector* protocol (**DSDV**) [87]. In a proactive (or periodic) routing protocol nodes periodically exchange routing information with other nodes in attempt to have each node always know a current route to all destinations. Specifically, SEAD is inspired by the DSDV-SQ version of the DSDV protocol. The DSDV-SQ version of the DSDV protocol has been shown to outperform other DSDV versions in previous ad hoc networks simulations [88, 89].

SEAD deals with attackers that modify routing information broadcasted during the update phase of the DSDV-SQ protocol: in particular, routing can be disrupted if the attacker modifies the sequence number and the metric field of a routing table update message. Replay attacks are also taken into account. SEAD does not cope with wormhole attacks though the authors propose, as in the ARIADNE protocol [45], to use the TIK protocol to detect the threat.

2.4.1.7 Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks

In [45], Hu, Perrig and Johnson present an on-demand secure ad hoc routing protocol based on DSR that withstands node compromise and relies only on highly efficient symmetric cryptography. Ariadne guarantees that the target node of a route discovery process can authenticate the initiator, that the initiator can authenticate each intermediate node on the path to the destination present in the RREP message and that no

intermediate node can remove a previous node in the node list in the RREQ or RREP messages. As for the SRP protocol, Ariadne needs some mechanism to bootstrap authentic keys required by the protocol. In particular, each node needs a shared secret key (K_{SD}, is the shared key between a source S and a destination D) with each node it communicates with at a higher layer, an authentic TESLA [90, 91] key for each node in the network and an authentic "Route Discovery chain" element for each node for which this node will forward RREQ messages. Ariadne provides point-to-point authentication of a routing message using a message authentication code and a shared key between the two parties. However, for authentication of a broadcast packet such as RREQ, Ariadne uses the TESLA broadcast authentication protocol. Ariadne copes with attacks performed by malicious nodes that modify and fabricate routing information, with attacks using impersonation and, in an advanced version, with the wormhole attack. Selfish nodes are not taken into account.

2.4.1.8 Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks

Hu, Perrig and Johnson propose an approach to detect a wormhole based on *packet leashes* [80]. There are two types of packet leashes: geographical and temporal.

The main idea of packet leashes is that by authenticating either an extremely precise timestamp or location information combined with a loose timestamp, a receiver can determine if the packet has traversed an unrealistic distance for the specific network technology used. Temporal leashes rely on extremely precise time synchronization and timestamps in each packet. We can approximate a packet's travel time as the difference

between the receive time and the timestamp. To be more conservative, however, a node can choose to add the maximum time synchronization error, assuming that the sender's clock might be faster than the receiver's. Conversely, to allow all direct communication between legitimate nodes, a node can subtract the maximum time synchronization error, assuming that the sender's clock might be slower than the receiver's.

2.4.2 Anonymous Routing Protocols for Ad Hoc Networks

Among anonymous routing protocols proposed for ad hoc networks, only SDAR [23, 70–72] and ANODR [22] aim at pure ad hoc networks. In contrast, other anonymous routing or anonymous communication protocols are based on additional assumptions, e.g. AO2P [25] assumes the existence of a secure position service system.

2.4.2.1 SDAR: A Secure Distributed Anonymous Routing Protocol

In [23, 70–72], the authors propose a *Secure Distributed Anonymous Routing* (**SDAR**) protocol for ad hoc wireless networks, which is based on the onion routing protocol [67]. SDAR does not require the source node to gather and store information about the network topology. Instead, the source node initiates a path establishment process by broadcasting a path discovery message with certain trust requirements to all its neighboring nodes. The protocol is explained briefly as follows.

Firstly, the source node initiates a path establishment process by broadcasting a *path discovery* message with certain trust requirements to all of neighboring nodes. The format of the *path discovery* message is shown in Expression 2.4.

$$\left[\begin{array}{l} TYPE,\ TRUST\_REQ,\ TPK,\ E_{PK_R}(ID_R,\ K_S,\ PL_S),\ P_S, \\ E_{K_S}(ID_S,\ PK_S,\ TPK,\ TSK,\ SN_{Session\_ID_S},\ Sign_S(M_S)) \end{array}\right], \tag{2.4}$$

where

$TYPE$— the message type.

$TRUST\_REQ$— the trust requirement.

TPK/TSK— the temporary one-time public/private key.

PK_S/PK_R — the public key of the sender/receiver.

ID_S/ID_R — the identity of the sender/receiver.

K_S — a symmetric (session) key generated by the sender.

PL_S — the padding length set by the sender.

P_S — a padding implemented by the sender.

$SN_{Session\_ID_S}$ — a random number generated by the source for the current session.

M_S — $M_S = H(TYPE,\ TRUST\_REQ,\ TPK,\ TSK,\ ID_R,\ K_S,\ ID_S,\ PK_S,\ SN_{Session\_ID_S},\ PL_S,\ P_S)$.

$Sign_S(M_S)$ — M_S is signed with the private key of the source node S.

When a node i receives a path discovery message, it processes the message according to the following steps: (1) check if the message has already been received from other nodes; (2) check if the node is the sender's intended next hop; (3) check if the node is the destined receiver.

If the node is not the intended receiver, it adds a message which is encrypted with the TPK, and then forwards the new message to the neighbors whose trust level meets the source node's trust requirement. The format of the information appended is shown

in Expression 2.5. Besides that, node i needs to record $\langle Session\_ID_i,\ ID_{i-1},\ K_i\rangle$ to the internal mapping table.

$$E_{TPK}(ID_i,\ K_i,\ SN_{Session\_ID_i},\ Sign_{ID_i}(M_{ID_i})) \tag{2.5}$$

where

ID_i — the identity of node i.

K_i — a symmetric (session) key generated by node i.

$Session\_ID_i$ — a random number generated by node i source for the current session.

M_{ID_i} — $M_{ID_i} = H(M_{prev},\ ID_i,\ K_i,\ Session\_ID_i)$, and M_{prev} is the cumulative message that node i gets from its ancestor node $i-1$.

$Sign_{ID_i}(M_{ID_i})$ — M_{ID_i} is signed with the private key of node i.

If the node is the destined receiver, it first recovers the session keys for all the nodes along the path of the message. Afterwards, the destination generates a *path reverse* message, and then sends the message to the first node in the reverse path. Given that there are n nodes between the source and the destination along the path discovered, the format of the *path discovery* message is shown in Expression 2.6.

$$\left[\begin{array}{l} TYPE,\ E_{K_n}(E_{K_{n-1}}\ldots(E_{K_2}(E_{K_1}(E_{K_S}(\\ SN_{Session\_ID_1},\ K_1,\ SN_{Session\_ID_2},\ K_2,\ldots, \\ SN_{Session\_ID_{n-1}},\ K_{n-1},\ SN_{Session\_ID_n},\ K_n, SN_{Session\_ID_R},\ PL_R,\ P_R), \\ SN_{Session\_ID_S},\ SN_{Session\_ID_{S-1}},\ H(P),\ H_{K_S}(N_S)), \\ SN_{Session\_ID_1},\ SN_{Session\_ID_{S-1}},\ H(P),\ H_{K_1}(N_1)), \\ SN_{Session\_ID_2},\ SN_{Session\_ID_S},\ H(M_S),\ H_{K_2}(N_2)),\ldots), \\ SN_{Session\_ID_{n-1}},\ SN_{Session\_ID_{n-3}},\ H(M_{n-3}),\ H_{K_{n-1}}(N_{n-1})), \\ SN_{Session\_ID_n},\ SN_{Session\_ID_{n-2}},\ H(M_{n-2}),\ H_{K_n}(N_n) \end{array}\right], \quad (2.6)$$

where

$SN_{Session\_ID_{S-1}}$ — a random number that haves the same number of bits as any regular $SN_{Session\_ID_j}$ and is generated by the source node.

P — a padding that has the same length as any M_j.

M_i — $M_i = (E_{K_{i-1}}(M_{i-2}),\ SN_{Session\_ID_i},\ SN_{Session\_ID_{i-2}},\ H(M_{i-2}),\ H_{K_i}(N_i))$, where $H(M)$ and $H_{K_i}(M)$ are a one-way hash function and a keyed hash function using K_i as the key, respectively.

N_i — $N_i = (E_{K_i}(M_{i-1}),\ SN_{Session\_ID_i},\ SN_{Session\_ID_{i-2}},\ H_{K_i}(N_i))$.

Each intermediate node that receives the path reverse message uses the $SN_{Session\_ID_i}$ to retrieve the key for the session, removes one encryption layer, and forwards the message to the next node on the reverse path to the source node. The ID of the node from which the message was received is added to the successor node entry corresponding to the random number into the mapping table. When the source node receives the

message, it decrypts the message and passes the information about all the intermediate nodes (i.e., the route) to the higher application.

SDAR can protect the identities of the source and the destination from other nodes. In addition, it also ensures the route anonymity in the sense that adversaries outside the route have no knowledge about any node en route. However, there exist several weaknesses in SDAR. Firstly, the destination knows the identities of intermediate nodes en route so that a pair of adversaries can launch the protocol to collect the identities of other nodes in the network. Besides that, in SDAR, the authors add padding only to the original route request and route reply packets, while in the middle part of the route the route request and route reply packets are forwarded without being padded. As such, adversaries can easily deduce the distance from themselves to the source, even if they are not in the route discovered. Furthermore, the efficiency of this protocol is not good, and thus is vulnerable to DOS attacks, especially during the stage of flooding the RREQ packets.

2.4.2.2 ANODR: An ANonymous On Demand Routing Protocol

In [22], Kong et al. design the *ANonymous On Demand Routing* (**ANODR**) protocol. The design of ANODR is based on "broadcast with trapdoor information"[4], a novel network security concept which includes features of two existing network and security mechanisms, namely "broadcast" and "trapdoor information". Similar to Hordes [65,66], ANODR explores multicast/broadcast to improve recipient anonymity. However, ANODR is an on-demand protocol, and it extensively explores trapdoor information in broad-

[4]Given that $f : X \rightarrow Y$ is a trapdoor one-way function, trapdoor information is the secret that make the operation of finding preimages for $y \in Y$ feasible.

cast. These features are not discussed in Hordes' multicast mechanisms. In [22]. Kong et al. propose three different variants of ANODR: *ANODR-PO* (**Protected Onion**), *ANODR-BO* (**Boomerang Onion**), and *ANODR-TBO* (**Trapdoor Boomerang Onion**). Among them, ANODR-TBO is the most efficient one. Therefore, we only introduce this protocol here.

Anonymous route discovery in ANODR consists of two phase: route request denoted as RREQ and route response denoted as RREP. In [22], a communication source initiates the route discovery procedure by assembling an RREQ packet and locally broadcasting it. The RREQ packet is of the format shown in Expression 2.7.

$$< RREQ,\ seqnum,\ pk_{one},\ K_T(dest,\ K_c),\ K_c(dest),\ TBO > \tag{2.7}$$

where

$seqnum$— a globally unique sequence number.

pk_{one} — one-time public key from a public/private key pair ($pk_{one},\ sk_{one}$) generated by the forwarding node.

K_T — a secret key that the source shares with the destination.

$dest$ — the destination tag.

K_c — a commitment key.

TBO — a cryptographic onion that is critical for route pseudonym establishment.

The format of the RREP packet is shown in Expression 2.8.

$$< RREP,\ \{K_{seed}\}_{pk_{one}},\ K_{seed}(K'_c,\ TBO) > \tag{2.8}$$

where

K_{seed} — a locally unique random route pseudonym.

K'_c — the anonymous proof presented by the destination. Any forwarding node can verify the anonymous proof of trapdoor opening by checking $K_c(dest) \stackrel{?}{=} K'_c dest$.

Here ANODR employs "trapdoor commitment", a cryptographic concept explores the collision-resistant property of one-way functions. That is, given an output of one-way function $K_c(dest)$, it is computationally hard to find the input, or another input collision that can produce the same output. In ANODR, it makes use of the trapdoor commitment protocol proposed in TESLA [92]. In TESLA, the output of a one-way function is published as a commitment before the corresponding input is revealed. When both the input and output are available, any verifier can efficiently validate the commitment by applying the one-way function on them. In ANODR, $K_c(dest)$ embedded in RREQ packet is a public commitment made for the destination by the source. Later the destination node can present the input K_c as the proof of furnishing the commitment.

As to trapdoor boomerang onions, Figure 2.10 shows how they are compute, using an example of finding a path from A to E. When intermediate forwarding node, e.g. B, sees an RREQ packet, it embeds a random nonce N_B to the boomerang onion, encrypts the result with a random symmetric key K_B, then broadcasts the RREQ locally. The trapdoor information consists of N_B and K_B, and is only known to node B. The boomerang onion will be bounced back by the destination. After each local RREP broadcast, only the next hop (i.e., the previous hop in RREQ phase) can correctly open the trapdoor it made in the RREQ phase, hence the result is equivalent to a wireless

unicast. Then the node strips a layer of the boomerang onion and locally broadcasts the modified RREP packet.

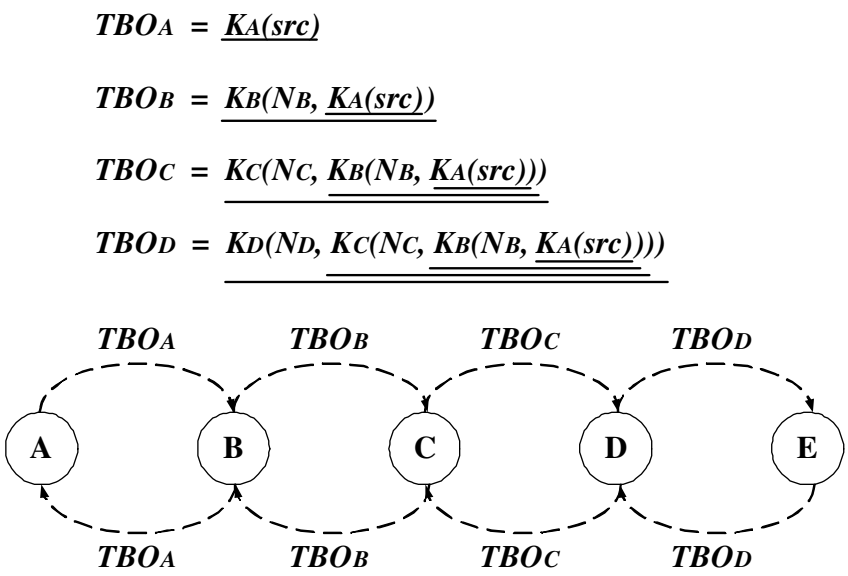


Figure 2.10: ANODR-TBO: anonymous route discovery using trapdoor boomerang onion (a single path showed from source A to destination E)

Compared to [23, 70–72], Kong et al. give a more comprehensive analysis on the anonymity and security properties achieved, and provide detailed simulation results on the efficiency of ANODR. In addition, ANODR is more efficient than SDAR at the data transmission stage.

Unfortunately, ANODR still has two main weaknesses. One is that the identity of the destination is known at least to the forwarding nodes en route, because they need this information to verify the validity of the RREP packet received. In fact, since the destination or the forwarding node cannot know in advance which neighbor is the next node along the route, it has to disclose the identity of the destination to all the neighbors. The other weakness is that, although ANODR can prevent adversaries from

knowing the exact location of the source or the destination, it is still possible for an internal adversary, i.e. an adversary en route, to deduce the distance, i.e. the number of hops, from itself to the source.

2.4.2.3 AO2P: Ad hoc On-demand Position-based Private Routing Protocol

Wu and Bhargava propose an anonymous routing protocol in [25], i.e. AO2P, in which the real identities of the source nodes, the destination nodes, and the forwarding nodes in the end-to-end connections are kept private. The format of the route request frame is shown in Figure 2.11.

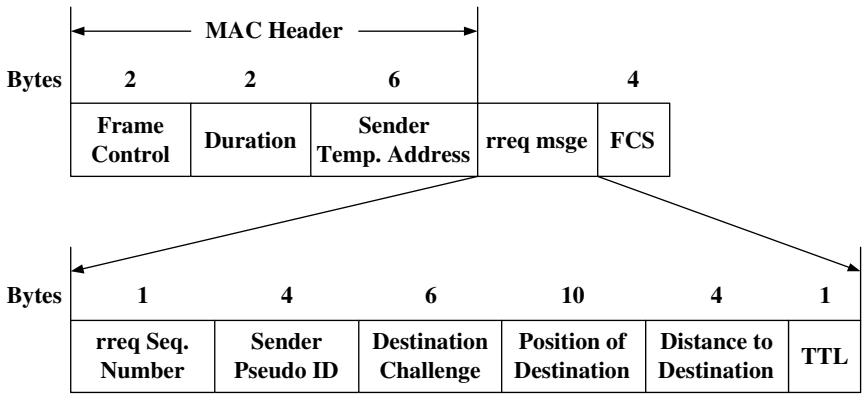


Figure 2.11: Frame of rreq message in AO2P routing protocol

As shown in Figure 2.11, AO2P does not provide location privacy, because the position of the destination is exposed in the network for route discovery. In addition, because the distance from the source to the destination is included in the route request, the location of the source in fact is also partially disclosed. AO2P assumes the existence of a secure position service system, which requires a number of fixed servers, and thus is not suitable for purely mobile ad hoc networks.

2.4.2.4 Other Anonymous Communication Protocols in Ad Hoc Networks

In [24], Zhang et al. propose an anonymous communication protocol, termed MASK. In this protocol, the authors assume that each node is assigned a large set PS_i of collision resistant pseudonyms by an off-line *Trusted Authority* (**TA**) beforehand. It cannot generate random pseudonyms by itself. Due to the limitation of storage on ad hoc network nodes, they cannot store a large number of pseudonyms, and thus these pseudonyms may be used out soon. Even worse, an adversary can launch a simple Denial of Service attack as follows. Adversary X sends out a request of mutual authentication using a fake (e.g. random generated) pseudonym to node Y. According to the scheme, Y will disclose one of pseudonyms assigned. Then the adversary moves around and sends such fake requests repeatedly, which either results in that the pre-assigned pseudonyms are used out in a short time, or let node Y refuse all the following requests. More importantly, in MASK, one RREQ message consumes one pseudonym of each node that could be reached by the source directly or indirectly. Besides that, MASK does not provide identity anonymity, because the real identifier of the destination node is disclosed to all the nodes en route during the route discovery process.

Čapkun et al. study the secure and privacy-preserving communications problem in hybrid ad hoc networks in [93]. Their scheme is based on the assumption that there exist network operators which have access to the locations and the identifiers of the registered mobile nodes.

2.4.3 Summary

One active area of securing mobile ad-hoc networks is secure routing. Many solutions, such as ARAN [16], S-AODV [78], SRP [17], Ariadne [45], SEAD [79], have been proposed for protecting popular routing protocols, such as AODV [73, 74], DSR [13, 14], and DSDV [87], from various passive and active attacks. However, due to some inherent limitations resulting from anonymity-related requirements, those solutions cannot be employed directly in anonymous routing protocols. For example, in [16], forwarding nodes need to verify route request and route reply packets with the source's and the destination's certificates. This conflicts with the goal of protecting the anonymity of the two communication parties in anonymous routing protocols.

Anonymous routing is a new and challenging field within secure routing. Current research work [22–25, 93] only provide limited anonymity properties, and are vulnerable to specific attacks. Therefore, it is highly desirable and challenging to design a new anonymous routing protocol that can satisfy all the anonymity and security-related properties, and at the same time minimize the communication and computation cost.

CHAPTER 3

Autonomous Key Management for Large Ad Hoc Networks

To overcome the challenges described in Section 1.5.1, i.e. scalability, flexibility, adaptivity, efficiency, security and robustness, we provide approaches at both the architecture and the algorithm level.

At the architecture level, we propose AKM which is based on a hierarchical structure and secret sharing to distribute cryptographic keys and provide certification services. In order to be employed in ad hoc networks, AKM is designed with several characteristics that are different from previous hierarchical key management schemes [94, 95]. Firstly, the hierarchical structure of AKM is a logical tree in which all the leaf nodes represent real wireless devices, while all the branch nodes only exist logically. In other words, AKM does not require the existence of real branch nodes (i.e. trusted key servers in [94, 95]), and thus is suitable for purely ad hoc environments. Secondly, flexibility and adaptivity can be obtained in AKM, since not only the structure of key management may change according to the increase/decrease of nodes, but also different parts of the structure have the freedom to set appropriate configurations to cope with various levels of risks. In addition, simulation results show that computation costs due to the dynamic property of ad hoc networks are very small under common threshold and region size settings. Thirdly, in AKM, secret keys of all branch nodes originate from one global

secret key either directly or indirectly. The secret key of each branch node is shared by its sub-nodes (either branch nodes or leaf nodes) using the Shamir secret sharing scheme [30]. Such secret sharing is performed recursively from top down to the lowest level. This characteristic allows us to issue certificates with different levels of assurance.

Once AKM is in operation, each real node holds a secret share which is used co-operatively with other nodes to maintain distributed key management services, such as assigning a secret share or a certificate to a newly-joined node.

At the algorithm level, we propose two algorithms, which are based on threshold cryptography and VSS [26, 27] and are independent from AKM. Both algorithms can resist active attacks. Given that there is no communication error, our first algorithm can assign a certificate within one round with help from a group of $2k-1$ nodes, in spite of active attacks. In contrast, the second algorithm involves help from only k nodes, although it may need more than one round to assign a certificate. Here, one round is defined to be the whole procedure that begins from a node requesting to be assigned a new certificate or renew its certificate to the combination and validation process of the certificate assigned or renewed. Simulation results show that, compared to the previous work [1–3, 19, 20], our second algorithm is not only much faster in a friendly environment, but it also works well in a hostile environment in which existing schemes work poorly. Furthermore, the process of generating partial certificates in our second algorithm is extremely fast. Such an advantage is critical in ad hoc networks where intrinsically the less help a node requests from its neighbors, the higher is the chance of obtaining the help. Consequently, using our second algorithm, a node can easily find enough neighboring nodes that provide the certification service.

3.1 Autonomous Key Management for Large Ad Hoc Networks

In this section, we first give an example to show the method of constructing the hierarchical structure in AKM, instead of presenting the details of the scheme, such as notation and definitions, directly.

3.1.1 An Example of Constructing the Hierarchical Structure in AKM

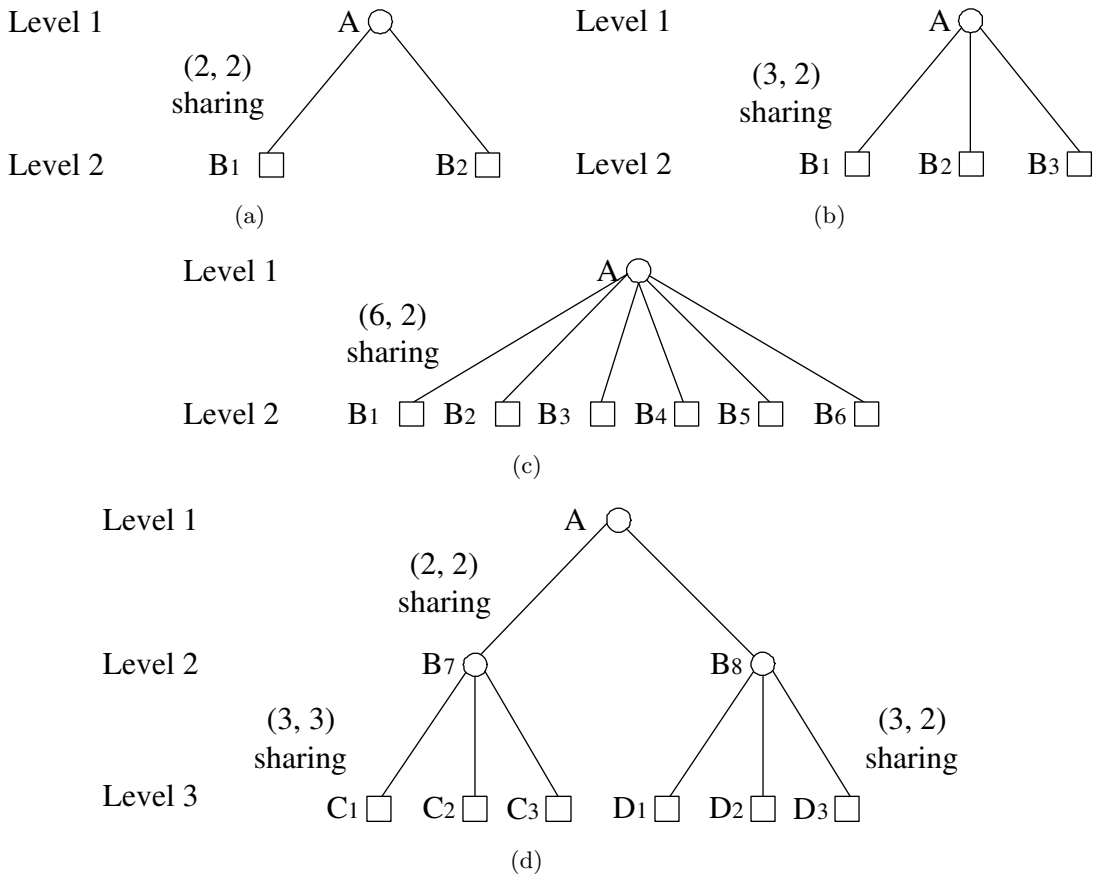


Figure 3.1: An example of constructing the hierarchical structure in AKM

Suppose that, at the beginning, there are two users who are physically nearby and

can communicate directly with each other. However, they do not trust each other. Therefore, they decide to generate a (2, 2) secret sharing system cooperatively[1]. A distributed VSS scheme (e.g., the one proposed in [96]) can be employed, since it is not suitable to assume the existence of a TTP in purely ad hoc networks. The structure of AKM at the initial stage is shown as Figure 3.1(a), where the squares and circles represent real nodes (i.e. physical devices) and logical nodes, respectively. Each user holds a share B_i $(i = 1, 2)$ of the secret A, where i is the identity of the user in the group, but none of them can deduce A by themselves. The secret can only be recovered under the cooperation of at least two users.

Afterwards, when a new user wants to join this group, she requires help from at least two members currently in the group. Once the new user is accepted, she is assigned with a new share of A corresponding to her identity (e.g. 3 in Figure 3.1(b)) in the group. And the configuration of the secret sharing system is changed from (2, 2) to (3, 2), as shown in Figure 3.1(b). This "Join" operation is explained in detail in Section 3.2.2.1.

Although the shares held by the current members are refreshed periodically to protect the shared secret, the system becomes more and more insecure when more and more users join the group. Suppose that, in our example, members in the group think that the system is insecure when the ratio of the threshold of the secret sharing system to the number of users in the group is less than 0.4. Thus, when the number of users reach six (as shown in Figure 3.1(c)), members in the group have to divide themselves into a few groups, such as two groups in the example. More specifically, B_1, B_2 and B_3 form a

[1]In some cases, if all the initial users think that the number of adversaries among them is less than a value k at that time, the threshold of the initial secret sharing system could be set to k, which is less than the number of initial users.

new group, members of which share a newly-generated secret B_7. Similarly, B_4, B_5 and B_6 form another group and share the secret B_8. The resulting structure of AKM after the division is shown in Figure 3.1(d). This operation increases the height of the tree structure, and is called the "Expansion" operation, which is to be explained in detail in Section 3.2.2.5. In AKM, B_7 and B_8 are generated in such a way that both of them are shares of the initial secret A. However, the secret sharing configurations within two newly-formed groups are independent from each other, and are determined by users that are initial members in the newly-created group. In this example, C_1, C_2 and C_3 (i.e. B_1, B_2 and B_3) totally distrust each other and employ a (3, 3) secret sharing scheme within group C_i. In contrast, D_1, D_2 and D_3 (i.e. B_4, B_5 and B_6) think that there are less than two adversaries among them between consecutive secret share updates. Thus, they choose a (3, 2) secret sharing scheme. Note that, as shown in Figure 3.1(d), the branch nodes holding B_7 and B_8 are logical nodes, and thus do not exist in reality. And secrets B_7 and B_8 are neither stored on any real node nor recovered explicitly at a later time.

Similarly, when more and more users want to join group C_i ($i = 1, 2, 3$) or group D_i ($i = 1, 2, 3$), they need help from at least k users in that group, where k is the threshold of the secret sharing configuration of that group (i.e. $k = 3$ for group C_i, and $k = 2$ for group D_i).

As shown in this example, the main challenge of designing AKM is to adapt the key management scheme to the dynamic environment like ad hoc networks securely (e.g. any secret held by the newly-generated virtual nodes should never be disclosed explicitly) and efficiently (e.g. the computation cost due to the membership variations should be

acceptable). In the following few sections, we first define the concepts used in AKM and present the assumptions employed in AKM in Section 3.1.2 and Section 4.2, respectively. In Section 3.1.4, we indicate the generic method of generating public/private key pairs and the secret shares of nodes in AKM, analyze the trade-off between the storage and the ability of assigning certificates with different levels of assurance, and show that how AKM can be scalable in terms of storage. In the same section, we also discuss about other characteristics provided by AKM, including self-organizing, adaptivity, and flexibility. Afterwards, we classify the membership-changing operations, and present the algorithms that achieve scalable and secure share updating in those operations in Section 3.2. In Section 3.4.3, we implement the two most costly membership-changing operations, and measure their costs under different settings.

3.1.2 Notation and Definitions

Here we explain the definitions of some concepts used in this paper, using a four-level ad hoc network system (shown in Figure 3.2) as the example:

- *Real Nodes*: the leaf nodes in the hierarchical structure of AKM. They have their own personal PKI key pairs, and they are corresponding to real devices in ad hoc networks. In Figure 3.2, node A to R are real nodes.
- *Virtual Nodes*: the branch nodes in the hierarchical structure of AKM. They are virtual and thus do not represent real devices in ad hoc networks. In Figure 3.2, the root node and node S to Z are virtual nodes.
- *Master Node*: the branch node that a node originates directly from. For example, node Z is the master node of node V to X.

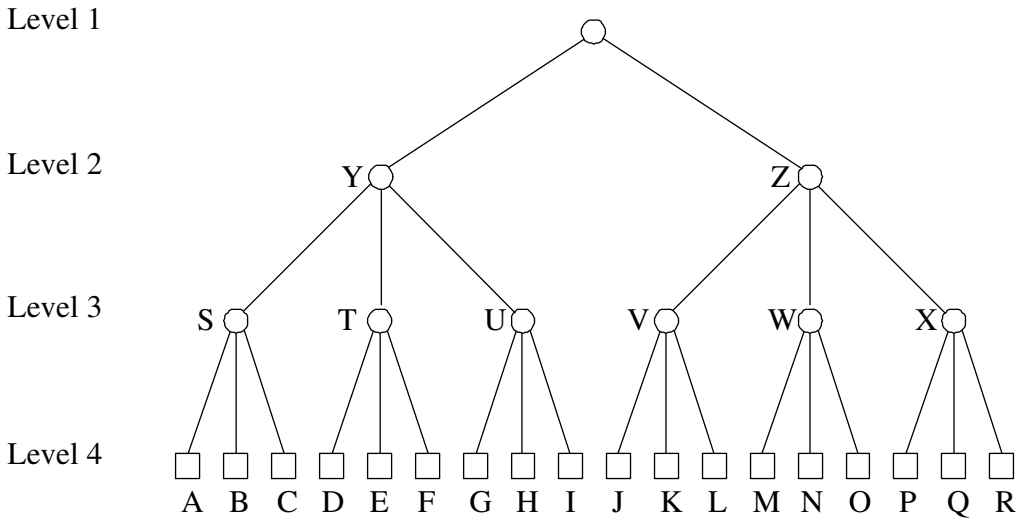


Figure 3.2: A four-level ad hoc network system

- *Region*: consists of all the real and virtual nodes originating directly from the same virtual node. Each virtual node corresponds to a region. For example, region T consists of real node D to F, and region Z consists of virtual node V to X.
- *Overall Region Size (ORS)*: the number of nodes which possess secret shares originated from the same secret, i.e. the secret key of the same region, between two consecutive secret share updates of this region. For example, we assume that Figure 3.2 shows the structure of the network just after a secret share update. At this time, the ORS of region S is 3. If node B leaves region S, the ORS of region S is still 3 until the next secret share update, even though in fact the number of the nodes currently in region S is 2. We propose this new parameter, in view of the fact that the leaving node may still keep a secret share of the secret key of the region.
- *Regional Trust Coefficient (RTC)*: the parameter to indicate how secure a region is. It is defined as the ratio of the threshold of a region to its ORS.

- *Global Trust Coefficient (GTC)*: the global lower limit on RTC.
- *Identifier (ID)*: the entity to represent a node.
- *Global Secret Key (GSK)*: the secret key of the whole ad hoc networks system.
- *Global Public Key (GPK)*: the public key of the whole ad hoc networks system.

We use the following notations (shown in Table 3.1):

Table 3.1: Notations in AKM

PK_i	the public key of a region "i"
SK_i	the secret share held by node "i". If node "i" is virtual, the secret share is also its secret key
ID_i	the identifier of a node "i"
pk_i	the public key of a real node "i"
sk_i	the secret key of a real node "i"
$SK_i(M)$	a message M is signed by SK_i

3.1.3 Assumptions

We make the following assumptions in this paper:

1. Each real node i has a personal PKI key pair [pk_i, sk_i]. The key pair can be generated by either a TTP or even the node itself.
2. Each real node joins and leaves the network randomly.
3. Given the public key of any virtual node, including the root node, it is computationally infeasible to obtain the corresponding secret key.
4. Between any two consecutive secret share updates, the number of adversaries that hold secret shares originated from the same secret key (i.e. the secret key of the same region) is less than k. The adversaries may still stay in the region, or have already left it.

3.1.4 Design of AKM

In AKM, all the regions' secret keys originate from GSK either directly or indirectly. The secret sharing process based on [30] is performed recursively from top down to the lowest level. Each node in AKM except the root virtual node, either real or virtual, holds a share of the secret of its master node and a PKI key pair. For a virtual node, the secret share that it holds is also its secret key, and its public key is generated from the secret key. In this scheme, we employ an asymmetric cryptographic scheme based on DLP [53] to generate the region's public key. For example, given that the secret key of region i is SK_i, its public key can be computed by $PK_i = g^{SK_i} \ (mod\ p)$. Unlike virtual nodes, each real node generates its own PKI key pair through some other method, e.g. by a TTP or even itself. However, it also needs to store a share of the secret key of its region (i.e. the secret key of its master node). For the sake of clarity, we list how real nodes and virtual nodes generate their keys in Table 3.2. Note that, to ensure the security of AKM, the secret key of any virtual node should never be recovered explicitly, even when we need it to assign a certificate. Instead, partial certificates are generated individually with those shares of the secret key, and then they are combined to create the certificate. Details of assigning or renewing certificates are presented in Section 3.3 at a later time.

Table 3.2: Key pairs and secret shares of nodes

	Real Node	Virtual Node
Secret Share	obtained during the secret sharing process	obtained during the secret sharing process
Secret Key	obtained before the initialization of AKM	the secret share is used as its secret key
Public key	obtained before the initialization of AKM	generated from its secret key after receiving its secret share

Besides the public key of itself, a real node needs to store public keys of other nodes in the hierarchical structure, since in purely ad hoc environments like ad hoc networks we cannot assume the existence of a trusted public server or directory storing public keys of other members. For a real node with sufficient storages, it will store two kinds of public keys: (1) public keys of itself and virtual nodes that are on its reverse path to the root. We denote the group of all these nodes as V; (2) public keys of nodes that are within the same region of any member in group V. For instance, in Figure 3.2, node A needs to store (1) PK_A, PK_S, PK_Y and GPK; (2) PK_B and PK_C (B and C are in the same region of A), PK_T and PK_U (T and U are in the same region of S), PK_Z (Z is in the same region of Y). Public keys of those higher-level virtual nodes are useful, when we want to verify certificates with different levels of assurance. The generation of certificates with different levels of assurance is to be presented in Section 3.3.1.3. However, there is a trade-off between the storage and this new characteristic provided by AKM. In reality, if a real node has limited storages, it can store a subset of these public keys. More specifically, it can discard public keys of those nodes that are d levels higher than it. For instance, in the previous example, when $d = 2$, public keys stored on node A consist of (1) PK_A, PK_S; (2) PK_B and PK_C (B and C are in the same region of A), PK_T and PK_U (T and U are in the same region of S). The minimum set of public keys that must be stored on a real node consists of (1) public keys of all the real nodes in the same region as itself; (2) the public key of its master node. These public keys are required to assign or renew common certificates (in contrast to certificates with different levels of assurance), which are signed by the secret key of the real node's master node, as presented in Section 3.3. Thus, A should at least store (1) PK_A, PK_B and

PK_C (A, B, C are within the same region); (2) PK_S (S is the master node of A). For simplicity, we assume that in Figure 3.2 the number of nodes in each region is n, and the secret sharing process within each region uses the same (n, k) configuration. Given that the height of the hierarchical structure is denoted as l, in such a system, there are n^{l-1} real nodes, and each of them needs to store $n+1$ (at minimum) to $(l-1)n+1$ (at maximum) public keys instead of storing public keys of all the n^{l-1} nodes.

AKM is self-organizing, since it does not rely on any TTP or TA at any stage. During the initialization, as shown in the example in Section 3.1.1, in contrast to [1–3] in which a TA is required to bootstrap the initial nodes, in AKM the role of the TA has been replaced by those initial nodes themselves, which jointly generate GSK using a distributed VSS scheme [96]. In addition, when the environment changes, e.g. some nodes join or leave, those nodes currently in a group/region would cooperate and adjust accordingly by themselves. In other words, our solution is adaptive to dynamic environments without a TTP. Details about node-based and region-based operations are shown in Section 3.2.2.

It should be absolutely clear that the proposed AKM scheme does not requires a preset hierarchical structure for all the nodes. If it did, this requirement would be really inappropriate for ad hoc networks where nodes are highly dynamic and thus we cannot determine their positions in advance. Instead, AKM only requires a small number of nodes, which are physically nearby and can communicate directly with each other, during the initialization. More importantly, the structure of the whole network may vary when more nodes join or leave. Therefore, the hierarchical structure of the whole network is adaptive. In the above example, the number of initial nodes needed is only k, and the

size of this network can be expanded to n^l at maximum at a later time. Actually, if there is no limit on the height of the hierarchical structure in AKM, AKM can contain nodes of arbitrary size.

Besides adaptivity, another main advantage of AKM is the flexibility of its structure. Each region[2] can determine its own size and the threshold of secret sharing, as long as it obeys the following rule: its RTC should be no less than GTC. The flexibility of the threshold parameter is very desirable, since in some cases different regions of an ad hoc network may face risks of different intensities. Thus, the threshold should not be set to be globally uniform. Moreover, such property is very useful in balancing the security and efficiency requirements.

3.2 Scalable Share Updates

In AKM, in order to improve the robustness of share updates, we distribute the functionality of the dealer in the Shamir secret sharing scheme [30] to many real nodes. The share updating process occurs under two conditions. One is at the time of the regular periodic renewal of secure shares. The other condition happens during the node-based and region-based operations.

To ensure the security of share updates, we use proactive secret sharing schemes [26, 32, 34, 37] to adapt the configuration of secret sharing to variations in a highly dynamic environment, such as ad hoc networks. However, because of their limited computation power, nodes in ad hoc networks cannot afford the costly computation

[2]In fact, as shown in the example in Section 3.1.1, it is not the virtual node representing the region but all the real nodes in this region that determine the secret sharing configuration during the initialization of the region, because the virtual node is logical and thus cannot take any operation by itself.

involved in the proactive scheme. To reduce the cost to a tolerable level, we keep the threshold of a region unchanged during its lifetime. It can greatly improve the efficiency of the proactive scheme, since the cost of changing the configuration of a secret sharing scheme from (n, k) to (n', k') [97,98] is very high, compared to the "Join" and "Leave" operations in AKM (described in Section 3.2.2.1 and Section3.2.2.2, respectively).

3.2.1 Regular Periodic Renewal of Secret Shares

In AKM, a proactive threshold cryptography scheme is used to enable nodes in a region to compute new shares from old ones in collaboration without disclosing the secret key of the region. It relies on the homomorphic property.

It is unnecessary to require all the nodes involved in the share refreshing process. Instead, the task can be done by only k nodes, since we assume that, between any consecutive secret share updates, the number of adversaries who hold secret shares originated from the same secret key is less than k. To detect those incorrect subshares, the VSS scheme [99] is employed.

Details are shown as follows. To renew the secret shares of all the n nodes in a region, firstly, k nodes are chosen from this region. Without lose of generality, we denote them as $(1, \cdots, k)$. Each of the k nodes, denoted as node i, randomly generates an (n, k) sharing of 0, denoted as $(SK_{i1}, SK_{i2}, \cdots, SK_{in})$, and then distributes the corresponding subshare SK_{ij} to node $j \in \{1, \ldots, n\}$. After receiving all the subshares generated by the k nodes, each node in the region, denoted as node j, can compute a new share from them and its old share ($SK'_j = SK_j + \sum_{i=1}^{k} SK_{ij}$). The new shares constitute a new (n, k) sharing of the service secret key. After refreshing, nodes remove the old shares and

use the new ones to generate partial signatures. Because the new shares are independent of the old ones, the adversary cannot combine old shares with new shares to recover the secret key of the service. Thus, the adversary is challenged to compromise k nodes in the same region between periodic refreshing.

3.2.2 Common Node-based and Region-based Operations

A comprehensive key management scheme must handle adjustments to secret shares subsequent to all membership-changing operations in the underlying communication system.

3.2.2.1 The "Join" operation

"Join" operations happen when one real node joins a region. It is, in fact, the process of changing a region's configuration from a threshold scheme of (n, k) to $(n+1, k)$. Given that node i joins a region, it first chooses a group of k nodes in this region denoted as group $G = \{1, \cdots, k\}$, and sends its request to them. Once node j receives the request, it checks node i's certificate and its CRL. If node j decides to serve the request, it calculates a partial share for node i as:

$$SK'_j = SK_j l_j(i) + \Delta_j \quad (mod\ q),$$

where Δ_j is the shuffling factor of node j, and $l_j(i) = \prod_{r=1, r\neq j}^{k} \frac{ID_i - ID_r}{ID_j - ID_r}\ (mod\ q)$. The shuffling factor is imported to prevent SK_j from being disclosed, because node i can easily recover SK_j from SK'_j if there is no shuffling factor. One method to generate the shuffling factor requires that each pair of nodes (j, r) in G exchanges a number S_{jr}, and

then Δ_j is computed as:

$$\Delta_j = \sum_{r=1, r\neq j}^{k} \sigma(j-r) \cdot S_{jr},$$

where $\sigma(x)$ is the sign function. Namely,

$$\sigma(x) = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ 0 & \text{otherwise} \end{cases}$$

Then node j returns the partial share to node i. After receiving k partial shares, node i can construct its secret share SK_i by adding them together:

$$\sum_{j=1}^{k} SK'_j = \sum_{j=1}^{k} SK_j l_j(ID_i) + \sum_{j=1}^{k} \Delta_j = SK_i \pmod{q}$$

and verify its secret share by:

$$g^{SK_i} = \prod_{j=1}^{k} (PK_j)^{l_j(ID_i)} \pmod{p}$$

3.2.2.2 The "Leave" operation

It happens when one node quits a region. Compared to the "Join" operation, this operation is easier. When nodes receive a "Leave" request from a node in the same region or detect that a node has left the region, they simply remove the certificate of that node from their key management records without recomputing secret shares. However, ORS does not decrease and thus RTC remains unchanged when a node "leaves" the region, because the node may still possess a secret share of the secret key of this region.

3.2.2.3 The "Merge" operation

"Merge" operations happen when the number of nodes within a region drops under its threshold. Given that the threshold remains unchanged, the RTC of a region drops very quickly if its ORS increases a lot. Therefore, we do not merge a region directly into a nearby region with the least size or the highest RTC. Instead, region i is divided into a few parts and each part is combined into one nearby region. Since the thresholds of the target regions are invariable, a "Merge" operation can be viewed as a series of "Join" operations.

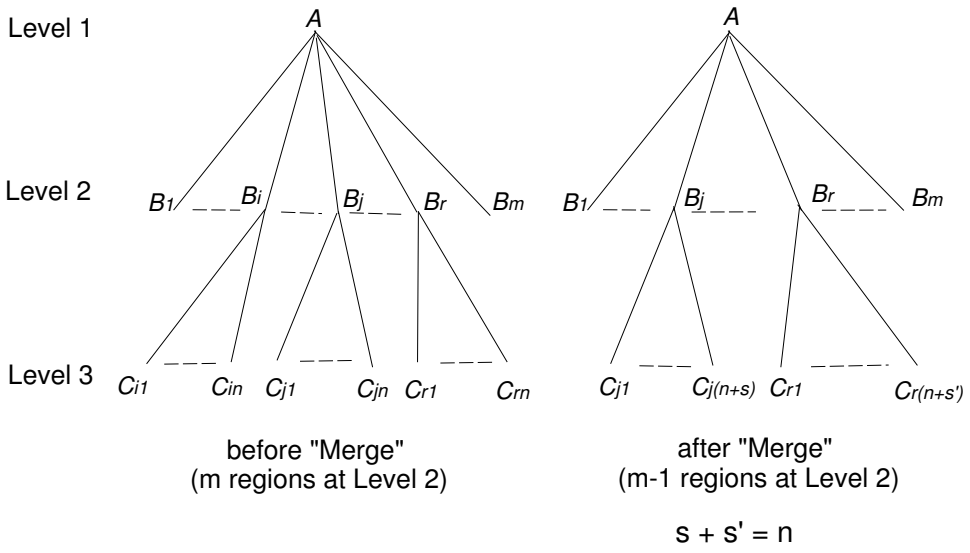


Figure 3.3: The "Merge" operation

3.2.2.4 The "Partition" operation

"Partition" operations happen when the RTC of a region drops under GTC or is lower than the security level expected. In AKM, the threshold of a region is fixed. Therefore, if the ORS of a region increases greatly, adversaries have a large chance to compromise

the threshold system. A straight-forward approach is to partition the region into two regions with almost the same size. In the hierarchical key tree, new regions lie at the same level as the original one. For example, as shown in Figure 3.4, region B_i with size $2n$ and the threshold k is partitioned into two regions B_i and B_{m+1}, each of which has n nodes and keeps its threshold as k. For the n nodes remaining in region B_i, they just need to renew their secret shares as discussed in Section 3.2.1.

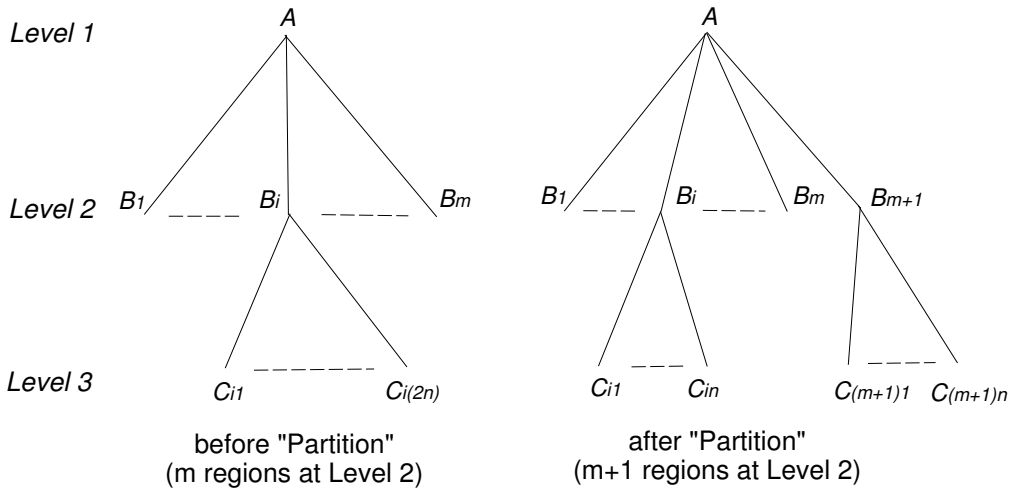


Figure 3.4: The "Partition" operation

In order to assign new secret shares to the n nodes in region B_{m+1}, firstly, region B_{m+1} chooses k regions at level 2, and then selects k nodes from each of the k regions. Without lost of generality, we denote the group of the k regions , the group of the k nodes from region B_j $(j = 1, \cdots, k)$, and the group of all these k^2 nodes as $G_B = \{B_1, \cdots, B_k\}$, $G_j = \{C_{j1}, \cdots, C_{jk}\}$, and $G = \{C_{11}, \cdots, C_{1k}, \cdots, C_{k1}, \cdots, C_{kk}\}$, respectively. Then, a "Partition" request signed by the secret key of region B_i is generated and multicasted to all the nodes in group G. The IDs of region B_i, B_{m+1}, and the

k regions are sent together with the request. By Lagrange interpolation, we know that:

$$SK_{B_i} = \sum_{j=1}^{k} SK_{B_j} l_{B_j}(ID_{B_i}) \quad (mod\ q) \tag{3.1}$$

where $l_{B_j}(ID_{B_i}) = \prod_{r=1, r\neq j}^{k} \frac{ID_{B_i} - ID_{B_r}}{ID_{B_j} - ID_{B_r}}\ (mod\ q)$. Since we know that:

$$SK_{B_j} = \sum_{h=1}^{k} SK_{C_{jh}} l_{C_{jh}}(0) \quad (mod\ q) \tag{3.2}$$

where $l_{C_{jh}}(0) = \prod_{r=1, r\neq j}^{k} \frac{ID_{C_{jr}}}{ID_{C_{jr}} - ID_{C_{jh}}}\ (mod\ q)$. Combine Eq. (3.1) and Eq. (3.2):

$$SK_{B_i} = \sum_{j=1}^{k} \sum_{h=1}^{k} SK_{C_{jh}} l_{C_{jh}}(0) l_{B_j}(ID_{B_i}) \quad (mod\ q) \tag{3.3}$$

Similarly, we get:

$$\begin{aligned} SK_{B_{m+1}} &= \sum_{j=1}^{k} \sum_{h=1}^{k} SK_{C_{jh}} l_{C_{jh}}(0) l_{B_j}(ID_{B_{m+1}}) \\ & \quad (mod\ q) \end{aligned} \tag{3.4}$$

where $l_{B_j}(ID_{B_{m+1}}) = \prod_{r=1, r\neq j}^{k} \frac{ID_{B_{m+1}} - ID_{B_r}}{ID_{B_j} - ID_{B_r}}\ (mod\ q)$

From Eq. (3.3) and Eq. (3.4), we get:

$$\begin{aligned} SK_{B_{m+1}} - SK_{B_i} &= \sum_{j=1}^{k} \sum_{h=1}^{k} SK_{C_{jh}} l_{C_{jh}}(0) R_j \\ & \quad (mod\ q) \end{aligned} \tag{3.5}$$

where $R_j = l_{B_j}(ID_{B_{m+1}}) - l_{B_j}(ID_{B_i})$.

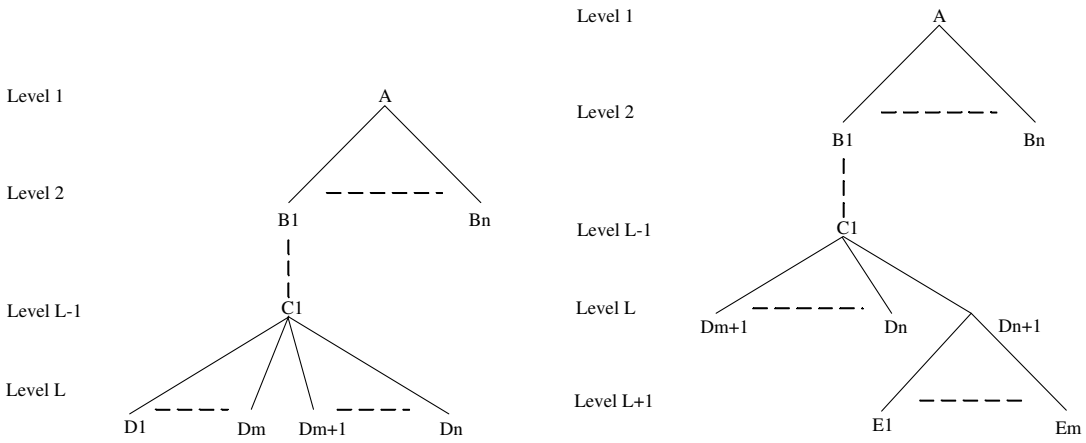
Consequently, to help nodes in region B_{m+1} generate the new shares of $SK_{B_{m+1}}$, each node C_{jh} in group G first computes $SK'_{C_{jh}} = SK_{C_{jh}} l_{C_{jh}}(0) R_j$. Then distributes the partial shares, i.e. the secret shares of $SK'_{C_{jh}}$, to nodes in region B_{m+1} using distributed VSS scheme proposed in [27]. According to Eq. (3.5), using the homomorphic property, each node in region B_{m+1} can compute its new share of $SK_{B_{m+1}}$ by adding k^2 partial shares from members in group G to its original share of SK_{B_i}.

3.2.2.5 The "Expansion" Operation

Similar to "Partition" operations, "Expansion" Operations happen when the RTC of a region drops under GTC or is lower than the security level expected. However, there is a special case in which we must take "Expansion" Operations. More specifically, when AKM has reached its capacity upper limit (namely, the RTCs of all the regions in AKM are equal to GTC), to ensure that the RTC of any region should not be less than GTC, we have to undertake a "Expansion" Operation, which increases the height of the hierarchical key tree.

For example, as shown in Figure 3.5(a), before the "Expansion" operation, the height of AKM is L. Now there is a new node that wants to join region C_1, but the RTCs of all the regions in AKM including region C_1 are equal to GTC. Therefore, the "Expansion" operation is executed. We assume that, original secret sharing of region C_1 is executed by following Eq. (3.6).

$$SK_{D_i} = a_0 + a_1 \cdot ID_{D_i} + \cdots + a_{k-1} \cdot ID_{D_i}^{k-1} \quad (mod\ q) \tag{3.6}$$



(a) Before "Expansion" (b) After "Expansion"

Figure 3.5: The "Expansion" operation

Firstly, chooses a group of m nodes from region C_1 which will be degraded to Level $L+1$, where $k \leq m \leq n-k+1$. Without loss of generality, let the group be $R = \{D_1, \cdots, D_k\}$. Following that, chooses a new identity denoted as $ID_{D_{n+1}}$ for the master node of all the nodes degraded. According to the Shamir secret sharing scheme [30], the secret share for node D_{n+1} can be calculated by the k nodes in group R. However, during the "Expansion" operation, this secret share would never be calculated out or recovered explicitly, since we do not assume the existence of a TA at this stage.

For simplicity, the same (n, k) threshold scheme is employed in the newly created region D_{n+1}. Without loss of generality, we assume that a new identity ID_{E_i} is assigned to the node whose old identity is ID_{D_i}, or the identity is chosen by the node itself. Then each node in group R calculates the following partial secret share denoted as SK'_{E_i} and distributes it to the node with the new identity ID_{E_i}.

$$SK'_{E_i} = SK_{D_j} \cdot l_{D_{n+1}} + \sum_{r=1}^{k-1} b_{jr} \cdot ID_{E_i}^r \quad (mod\ q), \tag{3.7}$$

where $l_{D_{n+1}} = \prod_{h=1,h\neq j}^{k} \frac{ID_{D_{n+1}} - ID_{D_h}}{ID_{D_j} - ID_{D_h}}$. Finally, each node recovers its new secret share in the new region by combining all the secret shares as Eq. (3.8).

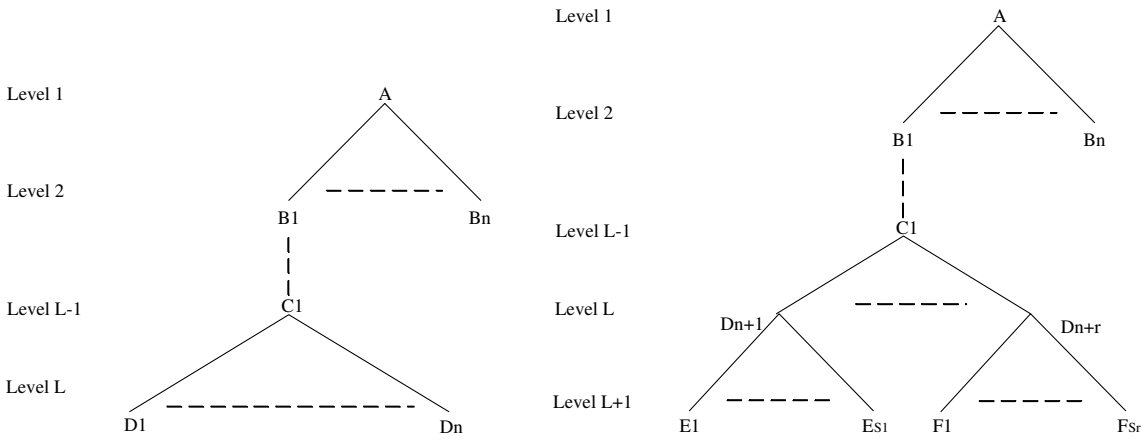
$$SK_{E_i} = b_0 + b_1 \cdot ID_{E_i} + \cdots + b_{k-1} \cdot ID_{E_i}^{k-1} \quad (mod\ q), \tag{3.8}$$

where $b_0 = SK_{D_{n+1}}$, $b_r = \sum_{j=1}^{k} b_{jr}$ $(r = 1, 2, \ldots, k-1)$. Therefore, all the coefficients of the secret sharing polynomial of the new region are cooperatively determined by the k nodes. At the end of the operation, the height of AKM increases to $L + 1$.

In Figure 3.5, there are n real nodes at level L before "Expansion". After performing the "Expansion" operation, m of them move to level $L + 1$, while others still remain at level L. From the security aspect, it means that those real nodes at level L have higher privileges than those moving to level $L + 1$, since each of them holds a share of SK_{C_1} directly. For nodes at level $L+1$, it requires k nodes of them to recover a share of SK_{C_1} cooperatively. It may not be desirable in some circumstances.

In Figure 3.6, the "Expansion" operation groups previous n real nodes under C_1 into r regions, and the size of each region is denoted as S_i $(i = 1, 2, \ldots, r)$. Therefore, we have $n = \sum_{i=1}^{i=r} S_i$. On one hand, the operation taken in Figure 3.6 can be viewed as a series of operations (i.e. r operations) described in Figure 3.5, which are undertaken separately for the r regions. On the other hand, the operation in Figure 3.5 can also be viewed as a special case of the operation described in Figure 3.6, in which $r - 1$ of r regions has only one member, and this member remains its old share instead of computing a new one.

For communication efficiency, when nodes cooperate to undertake region-based op-



(a) Before "Expansion" (b) After "Expansion"

Figure 3.6: Generic "Expansion" operation

erations, they may take their (relative) physical locations into consideration as well. In other words, they may prefer neighbors nearby to those nodes that are far away, when region-based operations, e.g. "Partition" and "Expansion" operations, are executed.

3.2.2.6 The "Contraction" Operation

"Contraction" Operations happen when AKM reaches its capacity lower limit. To ensure that the number of nodes in any region in AKM is not less than the threshold set before, we have to decrease the level of the structure. Similar to a "Merge" operation, a "Contraction" Operation can be viewed as a series of "Join" operations as well.

3.3 Certification Services Against Active Attacks

We described the architecture-level solution for key management in Section 3.1. In this section, we elaborate on our solution by presenting two new algorithms that are designed to protect certification services from active attacks.

3.3.1 Certificate Initialization and Renewal

There are two ways to issue a new certificate or renew a certificate. A node may be issued an initial certificate by an online or offline TA, after the authority verifies the principal's authenticity through external means (e.g., in-person ID). However, it is both costly for the TA to maintain certificates of all the nodes and is inconvenient for new nodes to request their certificates from the root TA. An alternative approach is to use any coalition of k networking nodes to issue an initial certificate via collaborative admission control for this new node. In AKM, we use the second approach, and extend it to support certification services with different levels of assurance. A certificate with the lowest level of assurance is assigned with the cooperation of k real nodes within the same region, while a certificate with higher-level assurance can be achieved with the coalition of more nodes from different regions.

In AKM, the certificate of node i, denoted as $CERT_i$, is a statement $cert_i$ that is signed by the secret key of its master node. The statement $cert_i$ consists of the association between node i and its public key, the ORS and the threshold of the master node, and the expiration time. The ORS and the threshold are included in $cert_i$, since receivers may be interested in these values and thus calculate the RTC of the region to which node i belongs. For example, they are useful in performing region-based operations.

In [1–3], Kong et al. employ the RSA scheme [52] to provide certification services. It works well under the flat structure. However, we find that it is not suitable for a hierarchical structure like AKM. In RSA [52], the certificate of node i is denoted as $CERT_i = (cert_i)^{SK} \ (mod\ N)$, where SK is the secret key of the master node of node i.

According to the requirements of RSA, SK should be coprime to $\phi(N)$, i.e. the Euler function of N. Namely,

$$gcd(SK,\ \phi(N)) = 1 \tag{3.9}$$

In AKM, all the secret keys of virtual nodes are generated during the recursive secret sharing process. In addition, in a purely ad hoc network, each node has the right to choose its ID instead of being predetermined by any entity. That is to say, during the secret sharing process, the secret shares are generated with arbitrary IDs based on the Shamir secret sharing scheme. Therefore, there is a high probability that they will not be coprime to $\phi(N)$, which $\equiv 0\ (mod\ 4)$.

To handle such problems, we designed a scheme based on the difficulty of DLP [53]. Both ElGamal [100] and the *Digital Signature Standard* (**DSS**) [101] require the computation of the inverse of the secrets, and such an operation is costly. To be more efficient, variants of the Schnorr signature scheme [102] and the signature scheme proposed by Park and Kurosawa [103] are used. Here we only show two algorithms based on the Schnorr signature scheme. In the former, node i selects a group of $2k-1$ nodes, and these nodes cooperate to assign node i a certificate. It requires only one round to assign a certificate. In the latter, node i selects a group of k nodes, and these nodes cooperate to assign node i a certificate. It may need more than one round to assign a certificate when there are malicious nodes launching active attacks.

The first algorithm was proposed by Stinson and Strobl [99], which is provably secure. The second algorithm is modified from [99] by us, which is also provably secure. It is very efficient but not fault-tolerant. These two algorithms, especially the second one,

are more efficient and secure than the scheme proposed by Kong et al. [1–3]. Firstly, both the algorithms proposed by Kong et al. cannot handle active attacks. They can only verify whether the combined certificate is valid, but fails to detect invalid partial certificates. Consequently, as long as there is one fake partial certificate among the k partial certificates chosen to generate the certificate, all the work done by honest nodes is useless. Even worse, adversaries can perform the attack without being caught. It makes the scheme proposed by Kong et al. inefficient for ad hoc networks. To prevent such attack, in our scheme nodes can verify each partial certificate to detect those malicious nodes. Secondly, our algorithms are based on DLP [53], which is faster than RSA [52] on which the algorithms proposed by Kong et al. are based. Thirdly, the time for generating a partial certificate in our algorithms is 7 to 150 times shorter than in the scheme proposed by Kong et al., when there is no communication error. Such advantage is critical in ad hoc networks where by nature the less help a node requests from its neighbors, the higher is the chance of obtaining the help. Furthermore, the algorithm proposed by Kong et al. requires a k-bounded offsetting to recover the real certificate [1–3], while we can generate the real certificate directly.

3.3.1.1 Assigning certificates based on $2k-1$ nodes

In this algorithm, a VSS scheme is employed to find out adversaries who launch active attacks. Since Pedersen's VSS scheme [27] requires a dealer, it is not suitable for ad hoc environments. Instead, we follow the distributed way proposed in Stinson's scheme [99] to achieve verifiable secure sharing. This algorithm requires cooperation from $2k-1$

nodes[3], and it ensures that the whole process of assigning a certificate can be finished within one round, in spite of active attacks launched by malicious nodes, since there are at most $k-1$ adversaries.

Let p and q be two large primes such that q divides $p-1$, and let G_q be the unique multiplicative subgroup of $\mathbb{Z}_p$ with order q. Let m be the statement claiming that a new node i's public key is PK_i, let $h(\cdot)$ be a one-way hash function: $\{0,\ 1\}^* \rightarrow \mathbb{Z}_q$.

Firstly, node i chooses a group of $2k-1$ nodes from its neighbors. Without loss of generality, let the group be $G = \{ID_1, \cdots, ID_{2k-1}\}$. Then node i broadcasts the request m together with the IDs of the $2k-1$ nodes among the group G. To achieve VSS, we need to generate a random shared secret denoted as r within group G. Details is shown as follows.

Once a node $j \in G$ receives the request and decides to serve the request, it chooses $r_j, r'_j \in \mathbb{Z}_q$ at random, and verifiably shares them among G acting as the dealer according to Pedersen's VSS scheme. Let the sharing polynomials be $f_j(u) = \sum_{t=0}^{k-1} a_{jt}u^t$, $f'_j(u) = \sum_{t=0}^{k-1} a'_{jt}u^t$, where $a_{j0} = r_j$, $a'_{j0} = r'_j$. Let the public commitments be $C_{jt} = g^{a_{jt}} \cdot h^{a'_{jt}} \ (mod\ p)$ for $t \in \{0, \cdots, k-1\}$, where g and h are two generators of G_q and no one knows $\log_g h$. Let H_0 be the set of nodes which are not detected to be cheating. Then the shared secret r is defined as $r = \sum_{j \in H_0} r_j$, and node j sets its share of the secret as $e_j = \sum_{t \in H_0} f_t(ID_j) \ (mod\ q)$, and the value $e'_j = \sum_{t \in H_0} f'_t(ID_j) \ (mod\ q)$.

Next, each node j $(\in H_0)$ broadcasts $A_{jk} = g^{a_{jt}}$ for $t \in \{0, \cdots, k-1\}$, and each

[3]On consideration of the dynamic property of ad hoc networks, a relative higher number of nodes may need to be involved in this algorithm.

node s in H_0 can verify the values broadcasted by other nodes in H_0 by checking if

$$g^{f_j(ID_s)} = \prod_{t=0}^{k-1} (A_{jt})^{ID_s^t} \quad (mod\ p) \tag{3.10}$$

If the check fails for an index j, node s complains against node j by broadcasting the values $f_j(ID_s)$, $f'_j(ID_s)$ that satisfy

$$g^{f_j(ID_s)} \cdot h^{f'_j(ID_s)} = \prod_{t=0}^{k-1} (C_{jt})^{{ID_s}^t} \quad (mod\ p) \tag{3.11}$$

but do not satisfy Equation (3.10). For node i which received at least one valid complaint, other nodes run the reconstruction phase of Pedersen's VSS scheme to compute r_j, $f_j(\cdot)$, A_{jt} for $t = 0, \cdots, k-1$. Therefore, all the players in H_0 set $X_j = g^{r_j}$.

After performing these steps, the following equations hold:

$$X = \prod_{j \in H_0} X_j = g^r \quad (mod\ p), \quad f(u) = r + a_1 u + \cdots + a_{k-1} u^{k-1}$$

where $a_t = \sum_{j \in H_0} a_{jt}$, for $t \in 1, \cdots, k-1$, and $f(j) = e_j\ (mod\ q)$ for $\ j \in H_0$.

$$C_t = g^{a_t} \quad (t = 0, \cdots, k-1)$$

In [96], the above scheme is proved to be robust under the assumption that $k \leq \frac{n}{2}$. In AKM, this assumption is satisfied, since the threshold of a region is normally set to be much less than the number of nodes in this region.

Then for each node $j \in H_0$, reveals its partial certificate:

$$\gamma_j = e_j + h(m||X) \cdot SK_j \quad (mod\ q),$$

and γ_j can be verified by:

$$g^{\gamma_j} = X \cdot \prod_{t=1}^{k-1} C_t^{j^t} \cdot PK_j^{h(m||X)}$$

for all $j \in H_0$.

Let (SK, PK) denote the key pair of the region to which the $2k-1$ nodes belong. Let H_1 denote the set of nodes not detected to be cheating in the above step. After verifying the partial certificates, node i selects an arbitrary subset $H_2 \subseteq H_1$ with $|H_2| = k$. Without loss of generality, we denote $H_2 = \{1, \cdots, k\}$, and compute:

$$\sigma = \sum_{j \in H_2} \gamma_j l'_j(0) \qquad (mod\ q)$$

where $l'_j(0) = \prod_{r=1, r\neq j}^{k} \frac{ID_r}{ID_r - ID_j}\ (mod\ q)$. Then node i's signature for m (i.e. $CERT_i$) is the pair $(X,\ \sigma)$. Since $\sigma = e + h(m||X)SK\ (mod\ q)$, other nodes can verify the certificate by:

$$g^{\sigma} = X \cdot PK^{h(m||X)} \quad (mod\ p)$$

3.3.1.2 Assigning certificates based on k nodes

Although the algorithm presented Section 3.3.1.1 can complete the certification service within one round, both the computation and communication overheads for achieving

VSS are heavy for nodes in ad hoc networks. The complexities of the computation and communication cost of VSS are $\mathbf{O}(k)$ and $\mathbf{O}(k^2)$, respectively. It is mainly due to the costly commitment process. To solve this problem, in this section we present another algorithm which requires lightweight computation. This algorithm needs cooperation from only k nodes, but it may take more than one round to assign a certificate, when malicious nodes are launching active attacks. Although this algorithm is not fault-tolerant, we can distinguish honest and malicious nodes, and those honest nodes are selected directly as members of group G of the next round. The complexities of both the computation and communication cost of one round of our second algorithm are $\mathbf{O}(1)$. Simulation results show that, when there is no communication error, more than 96% of certification renewals can be finished within two rounds, even if malicious nodes launch active attacks. It is much higher than that of the scheme proposed by Kong et al. [1–3], which declines very fast when the threshold increases.

Let p and q be two large primes such that q divides $p-1$, let G_q be the unique multiplicative subgroup of $\mathbb{Z}_p$ with order q, and let g be a generator of G_q. Let m be the statement claiming that a new node i's public key is PK_i, let $h(\cdot)$ be a one-way hash function: $\{0,\ 1\}^* \rightarrow \mathbb{Z}_q$.

Details of the algorithm for generating a threshold Schnorr signature are shown as follows. Firstly, node i chooses a group of k nodes from its neighbors. Without loss of generality, let the group be $G = \{ID_1, \cdots, ID_k\}$. Then node i broadcasts the request m together with the IDs of the k nodes in group G.

Once a node $j \in G$ receives the request and decides to serve the request, it first chooses a random integer $e_j \in \mathbb{Z}_q$ and broadcasts $x_j = g^{e_j} \ (mod\ p)$ and $PK_j =$

$g^{SK_j} \ (mod\ p)$ within the group G. Then node j calculates its partial certificate γ_j that is specific to group G

$$\gamma_j = e_j + h(m||X) \cdot SK_j \cdot l_j(0) \quad (mod\ q)\ ,$$

where $l_j(0) = \prod_{r=1,r\neq j}^{k} \frac{ID_r}{ID_r - ID_j} \ (mod\ q)$ and $X = \prod_{j=1}^{k} x_j \ (mod\ p)$, and returns it to node i. Node i can verify the partial certificate as follow:

$$g^{\gamma_j} = x_j \cdot PK_j^{h(m||X)l_j(0)} \quad (mod\ p) \tag{3.12}$$

If the k partial certificates are valid, node i calculates $\sigma = \sum_{j=1}^{k} \gamma_j$, and its signature for m (i.e. $CERT_i$) is the pair (X, σ). Other nodes and node i can verify the certificate by:

$$g^{\sigma} = X \cdot PK^{h(m||X)} \quad (mod\ p) \tag{3.13}$$

where the public key of the region to which k nodes belong is denoted as PK. In practice, node i can first verify (X, σ) using Equation (3.13). If it is valid, the task is completed. Otherwise, node i then verifies the partial certificates using Equation (3.12).

3.3.1.3 Assigning certificates with higher level assurance

In AKM, all the secret shares originate from the same global secret key. Making use of this property, our scheme can provide the ability to assign certificates with different levels of assurance at relatively small costs. Section 3.3.1.1 and Section 3.3.1.2 present how to assign a certificate with the lowest level assurance. Both of the two algorithms

can be extended to assign certificates with higher-level assurance, but here we just discuss the latter with an example of assigning a certificate with 2-level assurance.

As shown in Figure 3.7, the secret key of a virtual node A (i.e. SK_A) is distributed to n virtual nodes $\{B_1, B_2, \cdots, B_n\}$ using an (n, k) threshold scheme, and again SK_{B_j} ($j = 1, 2, \cdots, n$) is distributed to n_j real nodes $\{C_{j1}, C_{j2}, \cdots, C_{jn_j}\}$ with an (n_j, k_j) threshold scheme. For simplicity, we assume that $n_j = n$ and $k_j = k$, for all $j = 1, 2, \cdots, n$.

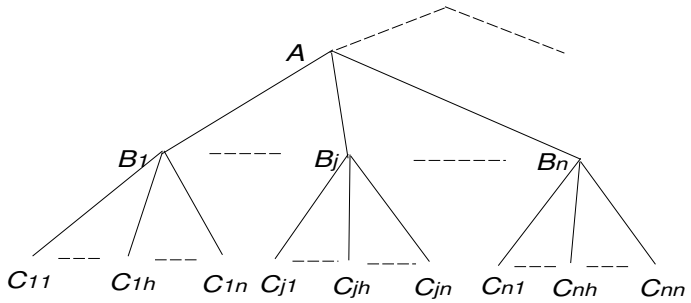


Figure 3.7: An example of assigning a certificate with 2-level assurance

Node i which wants to get a certificate with 2-level assurance first needs to choose k regions, and then choose k real nodes from each of these k regions.

Without losing of generality, let the group of k^2 nodes be $G = \{C_{11}, \cdots, C_{jh}, \cdots, C_{kk}\}$, which belong to k regions $\{B_1, B_2, \cdots, B_k\}$. Then node i broadcasts the request m together with the node IDs of the k^2 real nodes and k regions among the group G.

Once the request is received, real node C_{jh} which decides to serve the request from node i first chooses a random integer $e_{jh} \in \mathbb{Z}_q$ and broadcasts $x_{jh} = g^{e_{jh}}$ and $PK_{jh} = g^{SK_{jh}}$ within the group G. Then node C_{jh} calculates its additive share γ_{jh} that is specific to group G

$$\gamma_{jh} = e_{jh} + h(m||X)SK_{jh}l_{jh}(0)\lambda_j(0) \quad (mod\ q)$$

where $X = \prod_{j=1}^{k} \prod_{h=1}^{k} x_{jh} \ (mod\ p)$, $l_{jh}(0) = \prod_{r=1, r\neq h}^{k} \frac{C_{jr}}{C_{jr}-C_{jh}} \ (mod\ q)$, and $\lambda_j(0) = \prod_{r=1, r\neq j}^{k} \frac{B_r}{B_r - B_j} \ (mod\ q)$, and returns it to node i. Node i can verify the partial certificate as follow:

$$g^{\gamma_{jh}} = x_{jh} \cdot (PK_{jh})^{h(m||X) l_{jh}(0) \lambda_j(0)} \quad (mod\ p) \tag{3.14}$$

If all the k^2 partial certificates are valid, node i calculates $\sigma = \sum_{j=1}^{k} \sum_{h=1}^{k} \gamma_{jh}$, and its signature for m (i.e. $CERT_i$) is $(X,\ \sigma)$. Other nodes and node i can verify the certificate by:

$$g^{\sigma} = X \cdot (PK_A)^{h(m||X)} \quad (mod\ p). \tag{3.15}$$

In this scheme, to obtain a certificate with b level assurance, node i needs the cooperation of at most k^b honest nodes. Such requirement is reasonable and this scheme is efficient, since in AKM k^b is much smaller than the size of the whole ad hoc network, i.e. n^b. Again, in practice, node i can first verify $(X,\ \sigma)$ using Equation (3.15). If it is valid, the task is completed. Otherwise, node i then verifies the partial certificates using Equation (3.14).

3.3.2 Certificate Revocation

In AKM, the CRL is based on the accusations from other nodes. In a multi-hop wireless network like ad hoc network, the reliability of an accusation is based on the security of all the nodes that pass and broadcast the accusation. As such, the further an accusation comes from, the higher the probability that this accusation is compromised or malicious is. As a result, in ad hoc networks, messages from far away are not as trustworthy as

those generated by neighbors (nodes or regions). In addition, large accusation range results in rapidly increased communication and storage overheads. Therefore, the range of the accusations is limited to the same region. If node i receives an accusation on node j, it marks node j as "suspect" when there are less than k accusations towards it. Otherwise, node j is marked as "compromised" and is added into node i's CRL. In addition, node i launches a request to the k nodes which accused node j to sign a certificate revocation message. For example, in Figure 3.2, a (3, 2) threshold system is employed within region T, and node E and F accuse on node D. Thus, they can cooperate to generate and send out a message signed by SK_T to revoke node D's certificate.

3.4 Simulations on Security and Efficiency

In highly dynamic environment like ad hoc networks, small region size may result in rapid variances of the structure of the key tree. On the other hand, if the size is too large, we may have problems in intra-region routing. Current on-demand routing protocols, such as AODV [73, 74] and OLSR [104], handle well when the size of the ad hoc network is around 100 to 250 nodes. Thus, it is suitable to set the region size within this range. Let p_n be the probability of a node being compromised, p_r be the probability of a region being compromised, and n be the size of the region. Table 3.3 shows the settings on the threshold of a region under different conditions. From the table, we find that, when p_n is not less than 0.01, to ensure p_r lower than 10^{-4}, the threshold of a region should be set to at least 7 and 11 for a region with 100 and 250 nodes, respectively.

Table 3.3: Settings on the threshold of a region

p_n	n = 100		n=250	
	$p_r < 10^{-4}$	$p_r < 10^{-5}$	$p_r < 10^{-4}$	$p_r < 10^{-5}$
0.01	7	8	11	13
0.05	16	17	28	30
0.1	24	26	45	48

3.4.1 Hierarchical Structure vs. Flat Structure

To show advantages of the hierarchical structure over the flat structure in key management based on threshold cryptography, we compare the probabilities of GSK being compromised in AKM and [1–3]. In our implementation, a 3-level structure is employed, and the threshold of any region in AKM is set to be 10. As shown in Figure 3.8, under the flat structure, the security of an ad hoc network is very weak when the size of the whole network increases. However, under the hierarchical structure, the probability of GSK being compromised is very small and quite stable in spite of the great size of the ad hoc network.

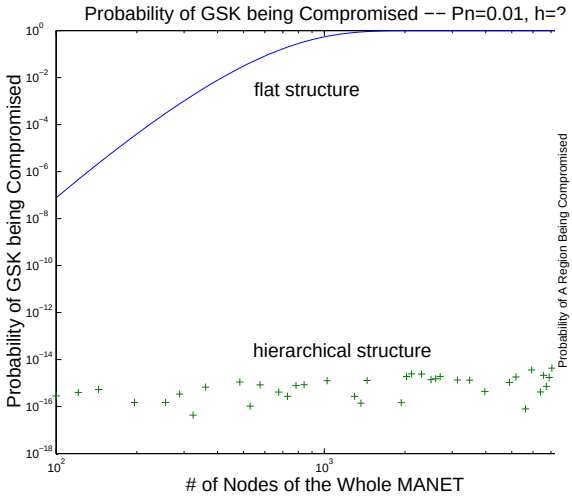


Figure 3.8: Flat structure vs. hierarchical structure

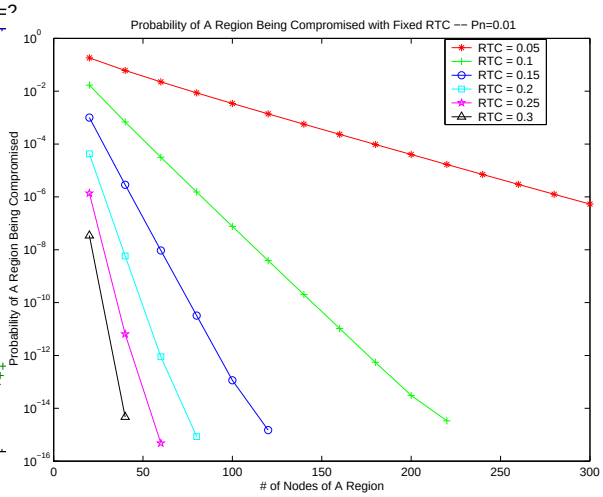


Figure 3.9: P_r with fixed RTC

3.4.2 Regions with the Same RTC

As shown in Figure 3.9, the higher the RTC is, the smaller p_r is. Therefore, it is suitable to use RTC as an approximate index of the security condition of a region. In addition, the higher the RTC is, the faster p_r decreases. Due to these two properties, a higher RTC is good for the sake of security. However, on the other hand, it results in a higher threshold which requires more computation power.

In addition, we find that, when both the RTC and the region size are small, this region is easy to be compromised. For example, in Figure 3.9, given that the p_n is 0.01 and the RTC of a region is 0.05, p_r may be higher than 0.01 when the size of the region is less than 75. Consequently, we need to be aware of this during the generation of a region. In other words, when the number of initial nodes in a region is small, the threshold should be set to a relative high value.

3.4.3 Computational Costs of Region-based Operations

Table 3.4: Computation cost of "Partition" and "Expansion" operations

ORS	Threshold	"Partition" Operation (msec)			"Expansion" Operation (msec)		
		GPSS	GNSS	Total	GPSS	GNSS	Total
100	5	8.87	0.03	8.90	1.59	0.01	1.60
100	10	18.59	0.11	18.70	4.52	0.02	4.54
100	15	30.96	0.28	31.24	7.91	0.02	7.93
100	20	46.24	0.60	46.84	11.69	0.02	11.71
100	25	62.00	0.80	62.80	16.42	0.03	16.45
250	10	47.63	0.11	47.74	11.00	0.03	11.03
250	20	124.20	0.68	124.88	27.67	0.03	27.70
250	30	205.27	1.29	206.56	54.71	0.07	54.78
250	40	302.27	2.24	304.51	90.16	0.07	90.23
250	50	422.73	3.77	426.50	135.78	0.07	135.85

Since both "Merge" and "Contraction" operations can be viewed as a series of "Join"

operations, our simulation focuses on "Partition" and "Expansion" operations. We run the simulation on a Pentium III 800 laptop.

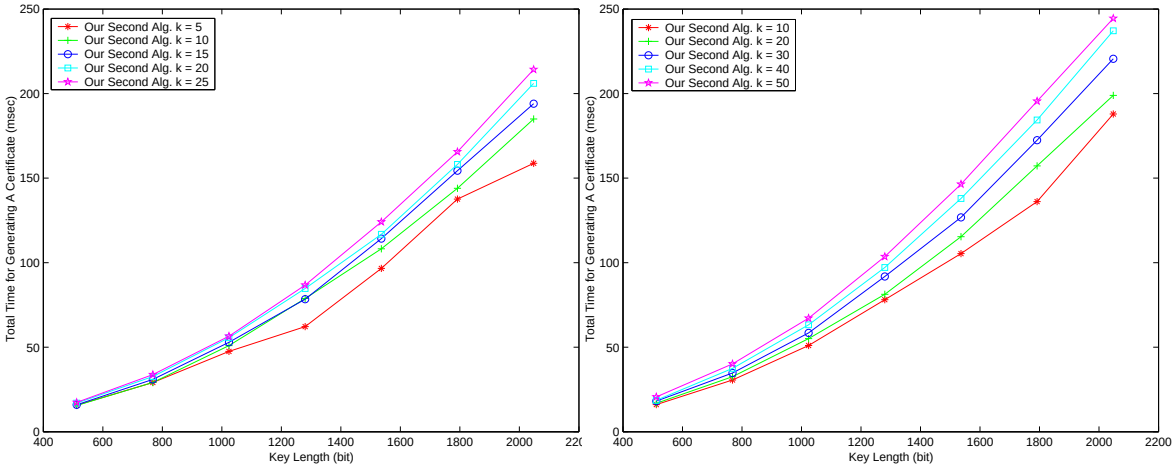
Table 3.4 shows the computation cost of "Partition" and "Expansion" operations under different ORS and threshold. In the table, GPSS stands for time for generating partial secret shares for all the nodes in the newly generated region, while GNSS stands for time for generating the new secret share. Simulation results show that the computation cost of both "Partition" and "Expansion" operations is quite small under common threshold and region size settings. For example, when the ORS of a region is 100 and its threshold is 15, it only takes 31 milliseconds to complete the "Partition" operation. As to the "Expansion" operation, the whole cost is less than 8 milliseconds.

3.4.4 Computational Costs of Certification Services

We measure and compare the performance of the algorithm proposed by Kong et al. [1–3] and our second algorithm on a Pentium III 800 laptop. Both of algorithms are implemented in Java. We run the experiments under different settings, e.g. different key lengths, thresholds, and region sizes. For each setting, we run the two algorithms for 20 times respectively and then calculate the average values.

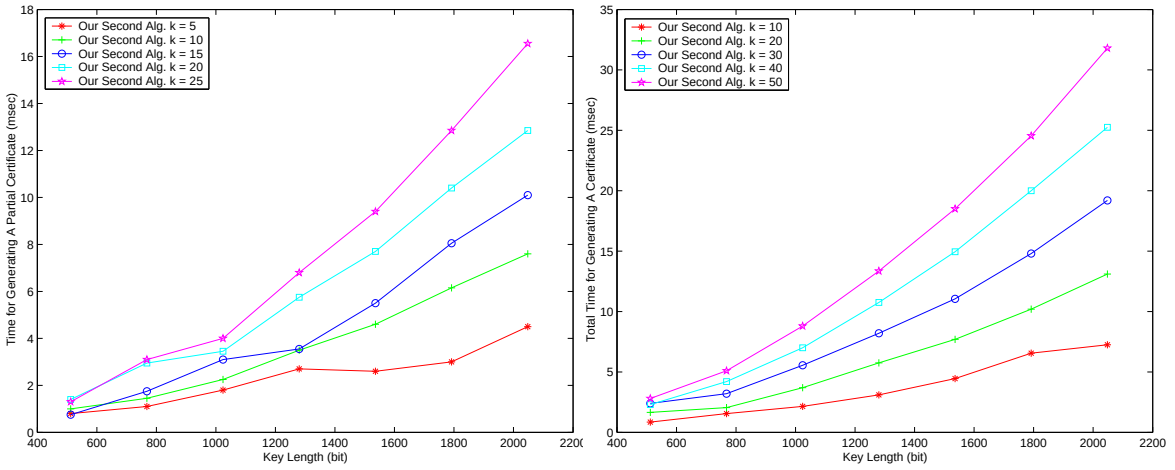
Here, the time for generating or renewing a certificate is calculated as the sum of all the processes taken by nodes, including the requesting node and those neighbors that provide certification services, as described in the algorithms. The main objective of this simulation is to show that compared to the algorithm proposed by Kong et al. [1–3] our second algorithm has much looser requirements on the computational power of nodes. Moreover, both Kong et al.'s algorithm and our second algorithm require help from

k nodes, and have similar communication costs. Therefore, we do not consider the communication overhead, when comparing the two algorithms.



(a) n = 100 (b) n = 250

Figure 3.10: Total time for generating or renewing a certificate in our second algorithm



(a) n = 100 (b) n = 250

Figure 3.11: Time for generating a partial certificate in our second algorithm

As shown in Figure 3.10, the total time for generating or renewing a certificate in our second algorithm varies from 20 to 250 milliseconds in the experiments, depending on the setting on the key length, threshold, and region size. In particular, when the

key length is 1024 bits, it takes our second algorithm around 40 to 70 milliseconds to generate or renew a certificate. We also find that, in our second algorithm, the process that a neighbor generates a partial certificate for the new node is very fast. As shown in Figure 3.11, such process takes less than 32 milliseconds under all the settings tested in the experiments. In particular, when the key length is 1024 bits, it takes less than 9 milliseconds to generate a partial certificate.

To compare with previous work [1–3], in Figure 3.12 and Figure 3.13, we show the ratio of the total time for generating or renewing a certificate in the algorithm proposed by Kong et al. [1–3] (denoted as T_{C-KONG}) to that of our second algorithm (denoted as T_{C-OUR}) and the ratio of the time for generating a partial certificate in the algorithm proposed by Kong et al. (denoted as $T_{PC-KONG}$) to that of our second algorithm (denoted as T_{PC-OUR}), respectively. As shown in Figure 3.12, our second algorithm is more efficient, when we consider the total time for assigning a new certificate. For instance, when the key length is 1024 bits, our second algorithm is around six to eight times faster than the algorithm proposed by Kong et al. As to the process of generating a partial certificate, the efficiency is greatly improved in our second algorithm. For example, when the key length is 1024 bits, our second algorithm is around 20 to 80 times faster. Consequently, using our second algorithm, a node can easily find enough neighbor nodes to provide the certification service, since very little effort is involved.

From empirical results, we notice that the performance of our algorithm is tightly related to the key length and the threshold. The larger the key length is, the more time we need to complete the certification service. However, compared to the scheme proposed by Kong et al. [1–3], our second scheme is less sensitive to this parameter, and

thus is more efficient. Similarly, the larger the threshold is, the more time we need to complete the certification service.

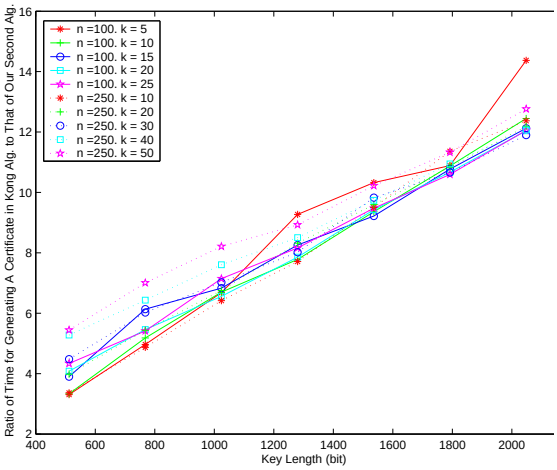


Figure 3.12: Ratio of T_{C-KONG} to T_{C-OUR}

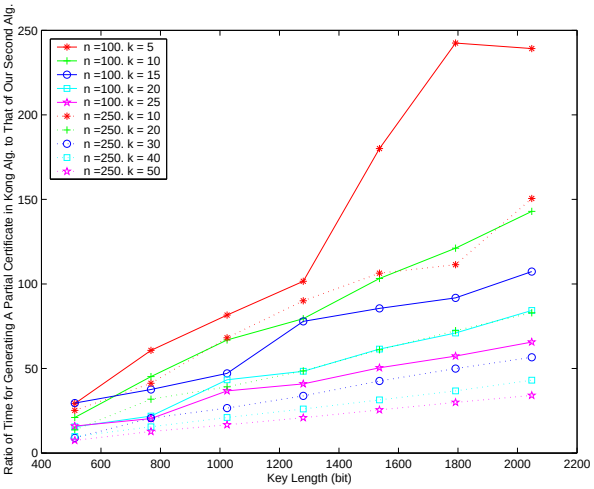


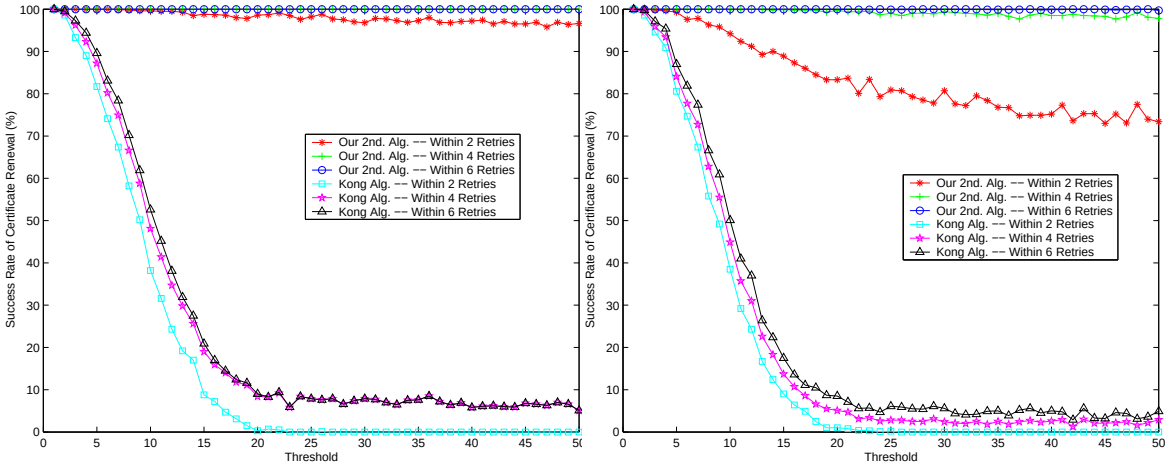
Figure 3.13: Ratio of $T_{PC-KONG}$ to T_{PC-OUR}

3.4.5 Certification Services Under Active Attacks

To compare the efficiency of certification services under active attacks, we use the success rate of certification renewals within certain rounds and the average rounds of retries before successfully assigning or renewing a certificate as the evaluation metrics. Here, we denote the success rate of certification renewals within r rounds and the wireless channel error rate as SR_r and e, respectively.

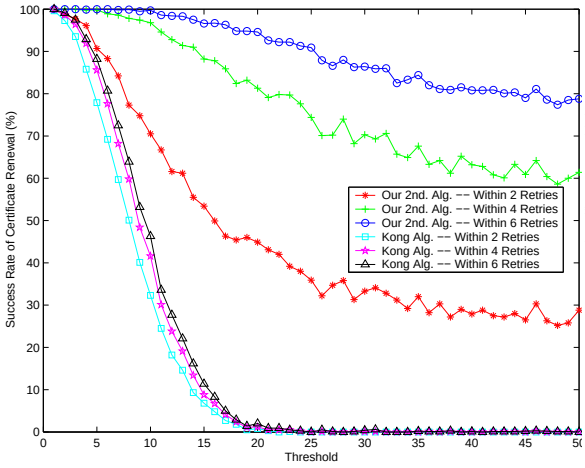
We run the simulation in a 600m X 600m network with 100 or 250 nodes, and the speed of nodes ranges from 1 m/s to 20 m/s. The random way-point model is applied to emulate node mobility pattern. In the simulation, we consider the scenarios of different wireless channel error rates, from no error (0%) to high error rate (10%). For each scenario, if the certification renewal fails due to either active attacks or communication

errors, the node which requests the certification service would replay the request up to a maximum number of retries. Typically, we set the maximum number of retries to 100 in the simulation.

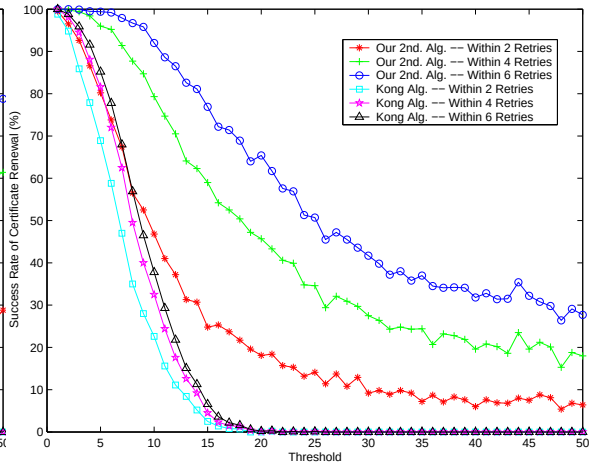


(a) error rate = 0%

(b) error rate = 1%



(c) error rate = 5%

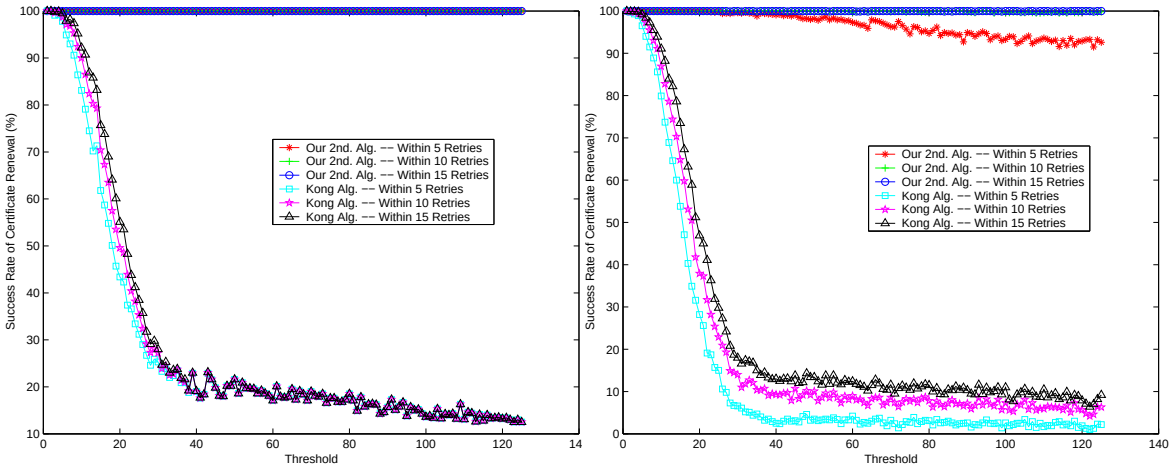


(d) error rate = 10%

Figure 3.14: Success rate of certification renewals – n = 100

As shown in Figure 3.14 and Figure 3.15[4], in all the simulation models, the success rate of the algorithm proposed by Kong et al. [1–3] declines very quickly when the

[4]Note that, due to limited space in Figure 3.14, Figure 3.15, and Figure 3.16, we denote the algorithm proposed by Kong et al. [1–3] as a short term "Kong Alg." in these three figures. However, it does not mean that the algorithm is designed by Kong only.



(a) error rate = 0%

(b) error rate = 1%

(c) error rate = 5%

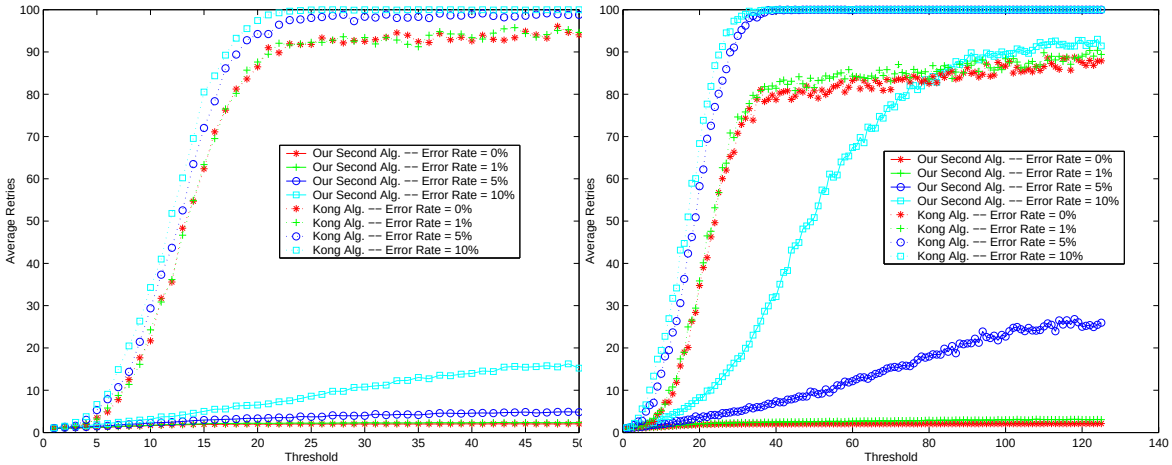
(d) error rate = 10%

Figure 3.15: Success rate of certification renewals – n = 250

threshold increases. For example, even when there is no communication error, for a region with size 100 and p_n is 0.01, if the threshold of this region is set to be 5, SR_2 is 81.7%. However, if the threshold increases to 10, it drops to 38.2%. In contrast, in our second algorithm, SR_2 decreases only 0.2%, i.e. from 100% to 99.8%, while the threshold increases from 5 to 10.

As shown in Figure 3.14 and Figure 3.15, the success rate of our second algorithm is always higher than that of the algorithm proposed by Kong et al. [1–3]. However,

we also notice that in our second algorithm, the higher the wireless channel error rate is, the faster the success rate of certification renewals declines. More specifically, our second algorithm works well under low wireless channel error rates, e.g. $e = 1\%$. When e increases (e.g. under heavy communications), SR_r is more sensitive to the variations on the threshold. For instance, as shown in Figure 3.14, when $e = 1\%$ and $n = 100$, SR_2 decrease by 5.4%, while the threshold increases 5 from 10. However, when $e = 10\%$ and $n = 100$, for the same variation of the threshold, SR_2 decreases by 33.4% instead. In such cases, to improve the success rate, we need to either choose a small threshold or increase the number of retries. The former is more effective. In addition, in AKM, to ensure the security, the RTC of one region should not be less than GTC. Therefore, we need to limit the size of a region, when the wireless channel error rate is high.



(a) n = 100 (b) n = 250

Figure 3.16: Average retries of certification renewals

Figure 3.16 shows the average retries of certification renewals in the algorithm proposed by Kong et al. [1–3] and our second algorithm under different wireless channel error rates. In all the cases, compared to our second algorithm, the algorithm proposed

by Kong et al. takes more rounds to complete the certification renewal. Similar to the experimental results on the success rate of certification renewals, in our second algorithm, the higher the wireless channel error rate is, the faster the average retries of certification renewals raises when the threshold of the region increases.

3.4.6 Summary

The empirical results on our architecture level solution (i.e. AKM) show that, on one hand, the hierarchical structure of AKM makes it more scalable compared to the flat structure schemes [1–3]. On the other hand, the computation cost of the two most expensive region-based operations, i.e. "Partition" and "Expansion", are quite small under common threshold and region size settings. In other words, simulation results indicate that AKM is more scalable and efficient compared to previous work.

As to our algorithm level solution, especially the second algorithm for certification services, simulation results show that our second algorithm is more efficient under both hostile and good environments.

CHAPTER 4

EASE: The Efficient Anonymity and Security-Enabled Routing Protocol

Anonymity is an indispensable part of the comprehensive solution of ad hoc network security, and under certain privacy-vital environments it is a fundamental requirement. For example, in a battle field, we not only want to ensure that adversaries cannot disclose the content of our communications (i.e., confidentiality) or disable the communications (i.e., availability and integrity), but also expect that the identities and location information of parties in communications are anonymous to adversaries. Otherwise, adversaries may deduce important information about the location or mobility model of communication parties, which can be used to locate the target of their physical attacks at a later time.

4.1 Design Goals

We intend to design a routing protocol which can protect the privacy of nodes and routes, and at the same time ensure other properties, such as security and efficiency. As a first step, we identify the expected goals or properties that we want to achieve in EASE as follows:

4.1.1 Ensure Privacy

4.1.1.1 Identity Anonymity

Identity anonymity consists of the following requirements: (a) No one knows the real identities of the source and the destination, except themselves; (b) The source and the destination have no information about the real identities of intermediate nodes en route.

4.1.1.2 Location Privacy

Location privacy consists of the following requirements: (a) No one knows the exact location of the source or the destination, except themselves; (b) Other nodes, including both those nodes outside the route discovered and the intermediate nodes en route, have no information about their distance, i.e. the number of hops, from either the source or the destination. This requirement is optional, but it is desirable in keeping both identity and location anonymity of the source or the destination, especially when the distance is just one hop.

For a protocol satisfying (a), we say that such a protocol provides *weak location privacy*; for a protocol satisfying both (a) and (b), we say that such a protocol provides *strong location privacy.*

4.1.1.3 Route Anonymity

Route anonymity consists of the following requirements: (a) Adversaries, either en route or outside the route, cannot trace a packet flow back to its source or destination; (b) For adversaries not in the route, they have no information on any part of the route; (c) It is difficult for adversaries to infer the transmission pattern and motion pattern of the

source or the destination;

4.1.2 Ensure Security

The protocol should ensure that the discovered route could function properly (namely, the protocol can find the route correctly and efficiently) under different attacks.

4.1.3 Ensure Efficiency

The protocol should be efficient in the sense of both computation and communication costs. More specifically, we expect that less cryptographic operations (especially asymmetric encryption/decryption operations) are involved during the route discovery process, and both the route discovery and the route maintenance processes do not generate a large number of messages.

4.2 Assumptions

In this paper, we assume that (1) there is a shared secret between the source and the destination; (2) wireless links are symmetric. Namely, if node A is in transmission range of some node B, then B is in transmission range of A as well; (3) Each node can change the source address of its outgoing *Medium Access Control* (**MAC**) frames, so that adversaries cannot trace the node based on its unique MAC address; (4) adversaries have unbounded eavesdropping capability but bounded computing and node intrusion capabilities.

4.3 Design of EASE

The design of EASE is mainly based on two ideas. The first one is to make use of the shared secret between the two communication parties and generate the variable part of the route request/reply in a way similar to the encryption of data with one-time key pad. Thus, adversaries without the secret cannot deduce meaningful information by sniffing. Moreover, we employ XOR and rotation operations to ensure that the length of the variable part is unchanged, and thus frustrate adversaries' attempt on finding patterns of variations on the route discovery packets, a common type of attacks against previous anonymous routing protocols for ad hoc networks. The other idea is to make use of the shared secrets between consecutive pairs of nodes en route, which are setup during the route discovery process, to provide a local repair mechanism. Since most of communication costs of anonymous routing protocols result from the flooding route request process, the local repairing mechanism is expected to improve the efficiency of anonymous routing protocols.

The EASE protocol consists of the following parts: *Route Request, Route Reply, Data Transmission, and Route Maintenance*. The notions of different types of packets involved in the route discovery process are shown in Table 4.1.

RREQ	Route Request Packet	RREP	Route Reply Packet
RERR	Route Error Packet	RRPR	Route Repair Packet
RUPD	Route Update Packet		

Table 4.1: Notions of types of packets involved in route discovery

As showed in Figure 4.1, we denote the source node, nodes en route, and the destination node as S, X_i $(i = 1, 2, \ldots, n)$, and D, respectively. n denotes the number of

nodes between the source and the destination.

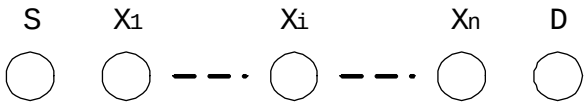


Figure 4.1: The route from source S to destination D

4.3.1 Route Request

During the route request process, each node en route denoted as X_i $(i = 1, 2, \ldots, n)$ receives a route request with the following format:

$$[RREQ,\ seq,\ K_T(dest,\ K_s,\ U_{orig}),\ K_s(seq,\ END),\ PK_{i-1},\ U_{i-1}],$$

where

seq — the sequence number of each session.

K_T — the secret shared between the source and destination.

dest — the identity of the destination D.

K_s — a session key of current session.

END — a sign showing that the destination has received the route request.

PK_{i-1} — the public key of the one-time key pair generated by the previous node X_{i-1}. PK_0 is the one-time public key chosen by the source S.

U_{orig} — a random number chosen by the source S.

U_{i-1} — a number generated by X_{i-1} $(i = 1,\ 2, \cdots,\ n)$. For $i = 1$, U_0 is generated by the source S.

For U_i $(i = 1, 2, \cdots, n)$ in RREQ, X_i computes it according to Equation (4.1):

$$U_i = f(U_{i-1},\ S_i) = (U_{i-1} \oplus S_i) \gg p_x, \tag{4.1}$$

where S_i is a random number chosen by X_i with size p_x. When $i = 0$, U_0 is calculated by the source S according to Equation 4.2:

$$U_0 = f(U_{orig},\ S_0) = (U_{orig} \oplus S_0) \gg p_x, \tag{4.2}$$

U_{orig} and S_0 are random numbers chosen by the source S with size p_s and p_x, respectively. Note that, in Equation (4.1) and Equation (4.2), $\oplus$ means the operation that S_i, the length of which is p_x, XORs with the least p_x bits of U_{i-1}. Thus, the computation denoted by Equation (4.1) and Equation (4.2) includes two steps. The output of the first step is a number with size p_s. The least p_x bits of the output is the result that S_i XORs with the least p_x bits of U_{i-1} or U_{orig}, while the higher bits are the same as the corresponding bits of U_{i-1} or U_{orig}. The next step is to right shift the result of the first step for p_x bits.

Let H_{max} denote the maximum number of hops that S wish the route to be. Then, we have:

$$p_s = (H_{max} + 1) \cdot p_x \tag{4.3}$$

For instance, given that the length of the random number chosen by X_i, i.e. S_i, is 16 bits, the source wants to discover a route between the destination and itself, and expects the length of the route is no more than 10 hops (i.e. $H_{max} = 10$). According

to Equation (4.3), we know that $p_x = 176$, and thus generate a random number U_{orig} with 176 bits during the generation of the route request packet. The setting of H_{max} is tightly related to the network density and mobility. Intuitively, the higher H_{max} is, the more routes will be found during the route discovery process. However, due to the dynamic property of ad hoc networks, the routes might be broken after some time period. Therefore, it may be inefficient to store a large number of routes, since most of them are unusable at a later time. Obviously, there is a trade-off between the route availability and the storage while choosing H_{max}.

Once receiving the RREQ packet, each forwarding node denoted as X_i first checks whether *seq* has been recorded in its RREQ buffer table. If yes, it simply discards the packet without decrypting the third element of the RREQ packet. Otherwise, X_i tries to decrypt $K_T(dest, K_s, U_{orig})$.

If fails, X_i first adds a new record into the RREQ buffer table. The format of a record in the RREQ buffer table of X_i is shown as follows:

$$[seq,\ PK_{i-1},\ K_s(seq, END)]$$

Then X_i generates U_i as shown in Equation (4.1), and replaces PK_{i-1} and U_{i-1} with its one-time public key (i.e. PK_i) and U_i, respectively. Finally, X_i broadcasts the modified packet locally.

If successful, it means that X_i is the destination node of this packet, since only the destination can successfully decrypt the packet. Afterwards, D compares U_{orig}, which is recovered from the third element of the RREQ packet, to U_n (i.e. the sixth element of

the RREQ packet). The destination discards those RREQ packets whose U_n's have been modified by more than H_{max} nodes. In other words, the destination ignores those routes, the lengths of which are longer than the hop limit. For those routes with fewer hops, the destination can figure out the exact distance from the source through the comparison. Thereafter, depending on whether EASE provides the multiple-path functionality[1], D may send out a RREP packet for each route with less than H_{max} hops or only for the first route with less than H_{max} hops, and at the same time adds a new record into its local route table. To counterattack sniffing, unlike AODV [73, 74], the destination needs to forward RREQs received like intermediate nodes.

At the end of the route request process, each node en route has the one-time public key of the previous node, and the destination has knowledge about the number of routes found between S and D and the length of each route.

4.3.2 Route Reply

During the route reply process, each node en route denoted as X_i $(i = 1, 2, \ldots, n)$ receives a route reply with the following format:

$$\begin{bmatrix} RREP,\ \{T_{i+1}\}_{PK_i}, \\ T_{i+1}(seq,\ K'_s,\ TPK,\ V_{i+1},\ K_T(H_{route},\ V_{orig},\ TSK)) \end{bmatrix}$$

where

T_{i+1} — a random number chosen by X_{i+1}, which is used as the shared secret between X_i and X_{i+1} after the routing discovery process.

[1]The empirical results and analysis on employing the multiple-path mechanism is given in Section 4.5.2.

K'_s — the proof that the destination has recovered the secret, i.e. the session key K_s, from the third element of the RREQ packet.

TPK, TSK— the key pair generated for the current session.

V_{i+1} — a number generated by X_{i+1}. For $i = n$, V_{n+1} is generated by the destination D.

H_{route} — the number of hops that this RREP packet is supposed to be forwarded before reaching the source.

V_{orig} — a random number chosen by the destination D.

For V_i $(i = 1, 2, \cdots, n)$ in RREP, X_i computes it according to Equation (4.4):

$$V_i = g(V_{i+1},\ T_i) = (V_{i+1} \oplus T_i) \gg q_x \tag{4.4}$$

where T_{i+1} is a random number chosen by X_{i+1} with size q_x. When $i = n + 1$, V_{n+1} is calculated by the destination D according to Equation (4.5):

$$V_{n+1} = g(V_{orig},\ T_{n+1}) = (V_{orig} \oplus T_{n+1}) \gg q_x \tag{4.5}$$

V_{orig} and T_{n+1} are random numbers chosen by the destination D with size q_d and q_x, respectively.

For the sake of anonymity, q_d cannot be equal to $H_{route} \cdot q_x$. Otherwise, adversaries can easily obtain the information about the route length by sniffing the RREP packets. Therefore, q_d should be set a few q_x bits longer than $H_{route} \cdot q_x$. Namely, we have

$$q_d > H_{route} \cdot q_x$$

For example, if we assume that the H_{route} of a given RREP packet is 7 and q_x is set to be 128 bits, we may set q_d to be 1280 bits. Alternatively, we can set q_d according to Equation (4.6), in spite of what the exact length of the route is, although it might be communicationally inefficient when the route length is much shorter than H_{max}.

$$q_d = (H_{max} + 1) \cdot q_x \tag{4.6}$$

Once receiving the RREP packet, each forwarding node denoted as X_i first tries to decrypt $\{T_{i+1}\}_{PK_i}$, and recovers the last element of the RREP packet. Since the last element is encrypted by T_{i+1}, only X_i can decrypt it. Then X_i extracts *seq* from the recovered information, and checks whether *seq* has been recorded in its RREQ buffer table. If no, it simply discards the packet without any furtherer checking. Otherwise, X_i extracts K'_s from the recovered information. Thereafter, X_i also needs to make sure that the RREP packet is from the destination. It can be verified by Equation (4.7), because only the destination D can recover K_s from the RREQ packet. If Equation (4.7) is not satisfied, X_i simply discards this RREP packet.

$$K'_s(seq, END) \stackrel{?}{=} K_s(seq, END), \tag{4.7}$$

After successfully verifying the validity of the RREP packet, X_i chooses a random number T_i with size q_x, and computes V_i from V_{i+1} and T_i according to Equation (4.4). Following that, X_i builds a new record in its route table. Then computes $\{T_i\}_{PK_{i-1}}$ and $T_i(seq, K'_s, TPK, V_i,\ K_T(H_{route}, V_{orig}, TSK))$, which are used to replace the last two elements of the RREP packet received. Finally, X_i broadcasts the modified RREP

packet locally.

Upon receiving the RREP packet, the source S can extract the H_{route} and the shared secrets along the route from the packet (i.e. $T_1,\ T_2, \ldots,\ T_{H_{route}}$), and then record them into the local route table. In our scheme, instead of only recording the shared secret with the first forwarding node, the source needs to record all the shared secrets from itself to the destination. In Section 4.3.4, we present how this additional effort benefits the route maintenance process. Since these secrets are generated randomly and are used only for this specific route from S to D, the source cannot deduce the identities of those forwarding nodes with the knowledge.

At the end of the route discovery process, each forwarding node has established shared secrets with the previous and next nodes, and masters the public key for the current session. As to the source and the destination, they have additional knowledge on the number of routes found and their lengths together with the private key for the current session. Moreover, only the source knows the shared secrets established along the route.

4.3.3 Data Transmission

To realize anonymous data transmission, we need to make sure that adversaries are not able to read or deduce information about the source and destination from data packets, and such information is only open to entities holding corresponding secrets. It is definitely not a good idea to encrypt the whole data packet using the shared secrets, although this solution is workable in theory; otherwise, each node has to try to decrypt the whole content of every packet received before deciding whether to accept it or not.

Consequently, this method results in a huge amount of computational cost.

In EASE, we provide a solution by making use of the shared secrets between any two consecutive nodes (i.e. T_i). Our idea is to construct some small-size information which is sent together with the data packet so that a forwarding node only needs to verify a small size information instead of the whole packet. It is similar to the construction of route pseudonym in [22], but is more simple and efficient. The small size information denoted as TAG is constructed as follows.

Given that, node X_i and node X_{i+1} share a secret denoted as T_{i+1}. Let $H_K()$ be a keyed fast one-way function, which uses K as the key. The format of TAG on the packet from X_i to X_{i+1}, denoted as TAG_i, is calculated as $H_{T_{i+1}}(N)$, where N is a non-decreasing number.

The data transmission process is similar to the route discovery process. Any forwarding node broadcasts the data packet to its neighbors, and then neighbors verify the validity of the TAG. If the packet passes the verification, the forwarding node recalculates and replaces the TAG. In addition, before broadcasting the packet to its neighbors, the content of data packets should be shuffled by an efficient encryption so that the adversaries cannot match payload contents to trace data forwarding. If the packet fails to pass the verification, it is discarded. Such a process is repeated until the packet reaches the destination.

One major challenge in anonymous data transmission is to ensure the route anonymity when we assume that time synchronization is not available. And such an assumption is reasonable for pure ad hoc networks where there is no central trusted party. In EASE, the non-decreasing numbers held by two consecutive nodes are synchronized, because

they are initialized to the same number when generating the new records in the local route table during the route discovery process, and are increased per packet received or sent in this route.

4.3.4 Route Maintenance

4.3.4.1 Motive of Providing The Local Repair Functionality

In ad hoc networks, we wish that the discovered anonymous route is robust and efficient against failures due to the following reasons: (1) node mobility; (2) join/leave operations of nodes; (3) nodes en route are hacked, and refuse to provide the data-forwarding function. A straightforward solution is to re-launch the route discovery process. However, in most anonymous routing protocols, route requests are broadcasted to flood the whole network, and thus it is costly to launch a new route discovery process. As such, this method is inefficient under ad hoc networks in which the network is highly dynamic and with poor physical protection, and we have to launch new route discovery process frequently. Alternatively, we can try to store all the routes found during the route discovery process. It is only workable under certain circumstances and has a few limitations. The effectiveness and limitations of this method are to be discussed in detail in Section 4.5.2.

In [105], authors show that local repair has significant performance advantages in for large networks with increasingly longer routes (e.g., 10 or more hops). Consequently, it is desirable to have a method to repair the current route instead of launching a new routing request. Unfortunately, due to the anonymity limitations, previous local repair mechanisms, for traditional networks or even ad hoc networks with no anonymity requirements, cannot be employed directly in the anonymous routing protocols for ad hoc

networks. In our scheme, we make use of the shared secrets along the route found, which are obtained during the previous route discovery process, to find a usable route with less computation and communication overheads. At the same time, our scheme ensures that the local repair process does not impair the anonymity property of the route. The idea of designing the locally-repairing mechanism is based on two observations: (1) The two communication parties, namely the source and destination, can have some extra privileges over nodes en route, as long as such privileges do not compromise the anonymity of other nodes; (2) Knowing the secret shares along the route does not help the source deduce the identities of the forwarding nodes en route, because shared secrets used by any part of the route are totally randomly chosen and are used only for this specific route from S to D. Similarly, for nodes en route, the knowledge of the shared secrets along a middle part of the route does not compromise the anonymity of nodes en route, especially when the length of this middle part is very short, e.g. two hops.

4.3.4.2 The Local Repair Mechanism

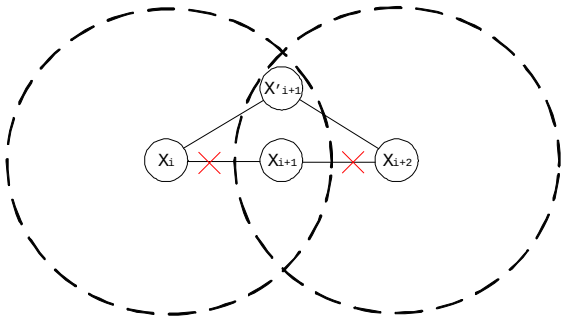


Figure 4.2: The repair of damaged route

We assume that, nodes can detect route failures when re-transmission count exceeds a predefined number. For example, as shown in Figure 4.2, X_i detects that the route to

X_{i+1} is not available any more. Upon detection, it looks up the corresponding entry in its forwarding table, finds the current TAG information that it shares with the previous node and the secret shared with the next node, and then broadcasts a route error packet with the following format:

$$[RERR,\ TAG_j,\ T_j(\{T_{i+1},\ Time\}_{TPK})],$$

where

TAG_j — the TAG that the current node shares with the previous node en route.

T_j — the shared secret that the current node shares with the previous node en route.

$Time$ — a time stamp for the RERR packet.

Intermediate nodes receiving such a RERR packet forward it to the previous nodes after replacing the TAG and re-calculating the last element of the RERR packet. The process is repeated until the RERR packet reaches the source. After extracting T_{i+1} from the packet, S compares it with the record in its local route table, and finds out the exact node, X_i here, reporting the route error. To discover a new route to the destination, S sends out a RRPR packet along the previous route. The data portion of the packet is of the following format:

$$[RRPR,\ TAG_{j+1},\ T_{i+1}(T_{i+2}, Time', H_2(T_{i+2}, Time'))]$$

where

TAG_{j+1} — the TAG that the current node shares with the next node en route.

$H_2()$ — a collision-resistant cryptographic hash function.

$Time'$ — a time stamp for the RRPR packet.

Each node en route simply forwards the packet, if it did not send a RERR message for the broken route before. When the packet reaches X_i, X_i decrypts the packet and extracts T_{i+2} which becomes a shared secret between X_i and X_{i+2}. Following that, X_i launches a route request from itself to X_{i+2}. It is similar to the previous route discovery process between S and D but with fewer hops. One major difference is that the identity of the destination in the RREQ packet is replaced with a broadcasting address. To verify whether it is the destination of the packet, a receiver needs to first recover K_s from the third element and then use it to decrypt the fourth element. If the receiver can recover *seq* successfully, it is the destination of this RREQ packet. In other words, compared to the global route discovery process, a receiver of a local repair RREQ packet has to execute one more decryption. The other major difference is that in the local repair process, we set a very short time period denoted as $Time_{LR}$ as the lifetime of the RREQ packet. The setting of $Time_{LR}$ is related to the random jitter before sending out a packet. A receiver simply ignores the local repair RREQ packet, if it is timeout.

Here, as shown in Figure 4.2, we assume that a new route via X'_{i+1} is found. Upon the completion of the route discovery between X_i and X_{i+2}, X_i sends out a RUPD packet with the following format:

$$[RUPD,\ TAG_j,\ T_j(\{T'_{i+1},\ T'_{i+2},\ Time''\}_{TPK})],$$

where

TAG_j — the TAG that the current node shares with the previous node en route.

T_j — the shared secret that the current node shares with the previous node en route.

T'_{i+1} — the shared secret that X_i shares with X'_{i+1}.

T'_{i+2} — the shared secret that X'_{i+1} shares with X_{i+2}.

$Time''$ — a time stamp for the RUPD packet.

Finally, once receiving the RUPD packet, S updates the corresponding record in its route table with T'_{i+1} and T'_{i+2}. In Figure 4.2, there is one node between X_i and X_{i+2} on the new route. For the case that there are no intermediate node or more than two hops on the repaired portion of new route, it is handled in the same way. However, in practice, we only consider routes with less than three hops, because the longer the route is, the higher is the probability that the route breaks within a certain time frame.

4.4 Analysis on Anonymity and Security

Firstly, we need to make clear that the *Security* term discussed in this section does not include issues about security of the content of data packets being transmitted. It is easy to see that security of the content of data packets is orthogonal to anonymity and security of the route protocol.

4.4.1 Passive Attacks & Active Attacks

Attacks against anonymous and secure routing in ad hoc networks can be classified into two types:

- *Passive Attacks* typically involve unauthorized "listening" to the routing packets or silently refusing to execute the function requested. The former type of attacks might be an attempt to gain routing information from which the attacker could extrapolate data about the positions of each node in relation to the others. Such an attack is usually impossible to detect, since the attacker does not disrupt the operation of a routing protocol but only attempts to discover valuable information by listening to the routed traffic.
- *Active Attacks* are meant to degrade or prevent message flow between the nodes. They can cause a degradation or complete halt in communications between nodes. Normally, such an attack involves actions performed by adversaries, for instance, the replication, modification, and deletion of exchanged data.

Typically, adversaries may launch both passive and active attacks at the same time, and the information obtained from the former can be used to enhance the effectiveness of the latter. For example, adversaries may sniff broadcast data and record specific signs that are used to identify the route, and then launch *Denial of Service* (**DOS**) or *Distributed Denial of Service* (**DDOS**) attacks by sending or broadcasting fake data using recorded signs.

For the anonymity and security analysis in this section, we consider attacks from both internal nodes (i.e. in the route) and external nodes (i.e. out of the route).

4.4.2 Anonymity Analysis

Here, we want to check whether EASE has achieved anonymity-related goals defined in Section 4.1, namely identity anonymity, location privacy, and route anonymity. In

the context of anonymity analysis, we assume that all the nodes including nodes on the discovered route are potential adversaries and are interested in the privacy information about the two communication parties and discovered routes.

4.4.2.1 Identity Anonymity

In [22, 24], the identity of the destination is disclosed to all the nodes en route. In contrast, in [23,70–72] the identities of forwarding nodes are disclosed to the destination.

In EASE, there is no node identity involved except the destination's identity, namely *dest*, in the RREQ packet. Fortunately, *dest* is encrypted by the shared secret between the source and the destination, and thus it is known only to the two communication parties. Therefore, EASE can ensure identity anonymity in ad hoc networks.

4.4.2.2 Location Privacy

The idea of current attacks on location privacy is to overhear the route request and route reply packets and then deduce the distance from the source or the destination by checking the length of those packets.

In [23, 70–72], the authors add padding only to the original route request and route reply packets, while in the middle part of the route the route request and route reply packets are forwarded without being padded. In addition, each forwarding node appends fixed-length information, including the id of the node and a session key (shared encryption key generated by the node), etc., to the route request packet. Therefore, every node receiving the route request packet can deduce the distance between the source and itself. In [22], authors propose to add random padding to the packets, after calculating

the onion[2] at each step. This method works well, when adversaries are not in the route. However, in order to calculate and replace the onion in the route request and route reply packets, internal nodes (nodes en route) need to have full knowledge about the actual size of the onion received. Consequently, their work is still vulnerable to internal nodes. In [25], the location of the destination is open to all the nodes in the network.

In EASE, we undertake two different methods to counterattack traffic analysis launched by both external and internal nodes. On one hand, a random padding is added at each forwarder's decision so that external nodes cannot match payload length to get useful information about the location of the source or the destination. On the other hand, more importantly, the lengths of the meaningful contents of the route request and the route reply do not increase along the route so that internal nodes cannot deduce how far they are from the source or the destination.

4.4.2.3 Route Anonymity

Current attacks on route anonymity are based on *Traffic Analysis* [106]. The theory behind all these attacks is to detect common information among sniffed packets, and assume that any two packets are transferred along the same route, if they have information in common. The "common information" could be either identical content (e.g., the same sequence number) in sniffed packets, or identical time consumed by handling sniffed packets, or certain pattern of variations (e.g., the increase of the length of the packets).

In EASE, hop-by-hop payload shuffle is employed to prevent adversaries from match-

[2]Onion is a cryptographic data structure which was first proposed in [67].

ing the content of packets. The second case is also referred as *Time Analysis*. In timing analysis, temporal dependency between transmissions can be used by advesaries to trace a victim message's forwarding path. One usual method to thwart timing analysis is to use mixing technique [107–109]. More specifically, we can use a buffer to store and reshuffle the sequence of received data packets, and at the same time inject dummy packets into the buffer if necessary. As to the third case, we ensure that the length of packets does not change during the transmission, since the increase of the packet length could be one signal for route tracing.

In [23, 70–72], when the route reply is sent back to the source, each forwarding node removes one encryption layer, i.e. fixed-length information, from the route reply that it receives. Obviously, such a variation could be one signal for adversaries to tracing the route. In [25], nodes generate a pseudo ID and a temporary MAC address instead of using their real identities and MAC addresses. Unfortunately, such IDs and temporary MAC addresses are unchanged during the data transmission. Therefore, adversaries can easily identify the whole route by sniffing and comparing pseudo IDs and temporary MAC addresses in data packets.

Table 4.2 shows the comparison of the anonymity-related properties achieved in known anonymous routing protocols for ad hoc networks. In the table, $SDAR$, $ANODR$, $MASK$, and $AO2P$ stand for the anonymous routing protocols proposed in [22–25], respectively. As shown in Table 4.2, $EASE$ and $MASK$ are the only protocols providing complete strong location privacy. However, $MASK$ in fact is a routing protocol in which all the nodes involved in the route, i.e. both the source/destination pair and intermediate nodes en route, are from the same party, and thus can construct shared secrets

	SDAR	ANODR	MASK	AO2P	EASE
Identity Anonymity of The Source and The Destination	√	X	X	√	√
Identity Anonymity of Forwarding Nodes en Route	X	√	√	√	√
Weak Location Privacy	√	√	√	X	√
Strong Location Privacy (external nodes)	X	√	√	X	√
Strong Location Privacy (internal nodes)	X	X	√	X	√
Route Anonymity	X	√	√	X	√

Table 4.2: Comparison of the anonymity property of routing protocols

among themselves before launching the route discovery process. Therefore, to be more accurate, $MASK$ is not a generic anonymous routing protocol for ad hoc networks.

4.4.3 Security Analysis

4.4.3.1 Passive Attacks

The simplest attack on the route protocol is that adversaries or selfish nodes silently refuse to perform functions requested in the protocol. In normal routing protocols, the watchdog model [75] can be employed to detect such actions. However, in anonymous routing, the route reply is modified hop-by-hop and is supposed to be undistinguishable from other route replies. Therefore, by nature, we cannot figure out which route a given sniffed route reply belongs to, since it is a trade-off between anonymity and security. The only usable solution is to discover and maintain multiple routes at the stage of route discovery.

4.4.3.2 DoS Attacks

According to the target of the attack, DoS attacks in the context of anonymous routing can be classified into two types: *Multiple-to-One* attacks and *One-to-Multiple* attacks.

In the former attacks, multiple adversaries (or one adversary with strong power) may cooperate to exhaust the resource of a given target. The most critical step of such attacks is to identify the target, either its identity or its exact location. In [22–24], the identities of either the destination or nodes en route are disclosed, while in [25] the exact location of the destination is known to all the nodes receiving the route request. Our protocol is immune to this type of attacks, since both identity anonymity and location privacy are ensured in EASE.

As to the latter attacks, one adversary can send fake route request packets which exhaust the computation resources of all consequent nodes along the route, since those nodes would perform the cryptographic computation as requested in the protocol. In EASE, such attacks are prevented by (a) little computation, i.e., a XOR operation and a shift operation, is involved in modifying the RREQ packet before rebroadcasting; (b) employ the hop-by-hop authentication on the RREP packet. In [23, 70–72], the computation involved in handling the route request consists of two parts: signing the route request received with the receiver's private key and encrypting all the information added by this node with a temporary public key. Compared to [23, 70–72], the computational cost of this process in [22] is much lower. However, it still requires a symmetric key encryption to calculate the onion.

4.4.3.3 Attacks on Route Maintenance

One possible attack is that adversaries send fake route error packets to fool the source to choose another route or even re-launch the route discovery process. It makes no sense when adversaries en route launch such an attack. Therefore, in the context of attacks on route maintenance, we only consider adversaries which are not in the route.

In [23,70–72], there is no shared secret between consecutive nodes en route, and thus a node detecting route failures has difficulties in informing the source such failures. In [25], although there exist shared secrets, i.e. pseudo ID and temporary MAC address pairs, between nodes en route, such information is unchanged during the data transmission. Therefore, adversaries can generate fake route error messages using the information collected from previously sniffed data packets.

In EASE, no adversary out of the route can construct fake route error packets, because it does not hold any secret with any node en route, which is necessary to generate the TAG in the route error packet.

4.4.3.4 Wormhole Attacks

In *Wormhole Attacks* [12], an attacker records packets received at one location in the network, tunnels them to another location, and retransmits them into the network. Hu, Perrig, and Johnson propose an approach to detect wormhole attacks based on packet leashes [12]. The key intuition is that by authenticating either an extremely precise timestamp (i.e., *temporal leashes*) or location information combined with a loose timestamp (i.e., *geographical leashes*), a receiver can determine if the packet has traversed a distance that is unrealistic for the specific network technology used. Both of the

	SDAR	ANODR	MASK	AO2P	EASE
Passive Attacks	√	√	√	√	√
Multiple-to-One DoS Attacks	X	X	X	X	√
One-to-Multiple DoS Attacks	X	X	√	√	√
Attacks on Route Maintenance	X	√	√	X	√
Wormhole Attacks	√	√	√	√	√

Table 4.3: Comparison of the security property of routing protocols

solutions can be easily integrated into EASE without any conflict.

Table 4.3 shows the comparison of the security-related properties achieved in known anonymous routing protocols for ad hoc networks.

4.5 Simulation Results & Efficiency Analysis

In this section, we analyze the efficiency of EASE and compare it with other generic anonymous routing protocols [22, 23, 70–72] from two aspects: computation costs and communication costs.

Here, we do not consider MASK [24] and AO2P [25], due to the following reasons. In [24], the authors assume that a TA assigns each node in a group with a sufficiently large set of collision-resistant pseudonyms and a corresponding secret point set. And such information is used to generate shared session key and link identifier pairs, which are similar to the TAG in EASE, among nodes in the same group before launching the route discovery process. Apparently, MASK aims at finding anonymous routes among nodes in the same party instead of all the nodes in the network, and the low computational cost of handling route requests is based on costly before-hand computation of generating shared session key and link identifier pairs. In the AO2P protocol [25], the authors assume that there exist a small set of trusted nodes which can act as secure position

	Asymmetric Encryption or Decryption	Symmetric Encryption or Decryption	Others
SDAR	2	0	none
ANODR	0	1	none
EASE	0	0	one XOR and one shift

Table 4.4: Cryptographic operations for handling a route request

	Asymmetric Encryption or Decryption	Symmetric Encryption or Decryption	Others
SDAR	0	1	none
ANODR	2	2	none
EASE	2	2	none

Table 4.5: Cryptographic operations for handling a route reply

servers, and they only concern about the identity anonymity but ignore the disclosure of the location and route information. Consequently, in the section, we only compare EASE with SDAR [23, 70–72] and ANODR [22].

4.5.1 Computational Costs

In Table 4.4 and Table 4.5, we show the cryptographic operations that intermediate nodes carry out to modify a route request or route reply packet before forwarding it. Standard cryptographic algorithms can be employed for asymmetric and symmetric encryption/decryptions in both EASE and other anonymous routing protocols. For example, we can use RSA (1024-bit key) and AES (128-bit key & block) for asymmetric and symmetric encryption/decryption, respectively. Table 4.6 shows the performance of different cryptosystems on an iPAQ3670 pocket PC with Intel StrongARM 206MHz CPU [22]. For asymmetric key cryptosystems, the table shows processing latency per encryption/decryption operation. For symmetric key cryptosystems (the five AES final

Cryptosystem		decryption	encryption
ECAES (160-bit key)		42ms	160ms
RSA (1024-bit key)		900ms	30ms
El Gamal (1024-bit key)		80ms	100ms
AES/Rijndael (128-bit key & block)		29.2Mbps	29.1Mbps
RC6 (128-bit key & block)		53.8Mbps	49.2Mbps
Mars (128-bit key & block)		36.8Mbps	36.8Mbps
Serpent (128-bit key & block)		15.2Mbps	17.2Mbps
TwoFish (128-bit key & block)		30.9Mbps	30.8Mbps

Table 4.6: Processing overhead of various cryptosystems

candidates), the table shows encryption/ decryption bit-rate.

As shown in Table 4.4, in the terms of handling route requests, in EASE forwarding nodes perform only one XOR operation and one shift operation, instead of carrying out symmetric or asymmetric encryptions in SDAR [23, 70–72] and ANODR [22]. As to the computation cost of handling route replies, as shown in Table 4.5, SDAR requires only one symmetric decryption, while EASE and ANODR have the same amount of computation, i.e. two symmetric encryption/decryption and two asymmetric encryption/decryption.

However, in all the generic anonymous routing protocols for ad hoc networks, including SDAR, ANODR, and EASE, minimization of the computation costs involved in handling a route request are more important than reducing the costs of handling a route reply, because the route request is forwarded to flood the whole network. In contrast, the route reply is only forwarded reversely along the route found. Simulation results in Section 4.5.2 show that, the number of route requests received is much larger than the number of route replies received during the route discovery process. For example, when the simulation model is "single path without local repair" and the number of nodes in the network is 50, the number of route requests received is around 9 to 11 times

as the number of route replies received. According to Table 4.4 and Table 4.5, SDAR consumes two asymmetric encryption/decryption while handling a RREQ packet, while EASE and ANODR consumes two asymmetric encryption/decryption while handling a RREP packet. In addition, in [23, 70–72], the asymmetric cryptographic operations taken by a forwarding node are executed upon a large amount of data, including the whole RREQ message received, the identity of this node, a session key and a random number generated by this node, the length of which increases along the route. Furthermore, the computation cost of a symmetric encryption/decryption is a few orders faster than that of an asymmetric encryption/decryption.

We also implemented the cryptographic operations performed in EASE and ANODR for handling route requests. The program was written in Java, and was executed on a Pentium IV 2.2G desktop. In the implementation for EASE, we set the length of random number being XORed, i.e. p_x in Section 4.3.1, to be 16 bits, and the maximum number of hops, i.e. H_{max}, was set to be 10 hops. In the implementation for ANODR, we used the same setting that authors mention in [22], i.e. set the length of the onion as 400 bits for a distance of 10 hops. Simulation results show that, EASE is around 10 to 12 times faster than ANODR. Therefore, EASE is the most efficient one among the three generic anonymous routing protocols, while SDAR is the most costly one.

4.5.2 Communication Costs

The routing protocols and related simulation models are implemented using *Java in Simulation Time / Scalable Wireless Ad hoc Network Simulator* (*JiST/SWANS*) [110, 111], which is a Java-based simulation engine developed by Cornell University. JiST

is efficient, out-performing existing highly optimized simulation runtimes both in space and time.

We ran the simulation in a 3000m X 3000m network, and the node transmission radius was set to be 625m. The Random Way-point model [13] was applied to emulate node mobility pattern. According to the model, a node travels to a random chosen location in a certain speed and stays for a while before going to another random location. In our simulation, for each test, a pair of the source and the destination is chosen randomly from nodes in the network. The source begins to send the first data message 10 seconds after the start of the simulation. Afterwards, it sends one data message per second, until all the 100 data messages have been sent out. The simulation is terminated after running for 160 seconds. For each setting of the simulations, we ran the simulation program for 40 times and then calculate the average values.

4.5.2.1 Simulation Models

There are many methods that can improve the performance of traditional routing protocols. Some of the most frequently-used methods include repairing the broken part of the route locally, storing multi-paths during the route discovery process, and forwarding the route requests with a certain probability. For the former two methods, we design the following communication simulation models for analyzing the communication costs:

SP-NLR only store the first route found, and do not employ any local repair mechanism.

MP-NLR store all the routes found during the route discovery process, but do not employ any local repair mechanism. After the route discovery process, the

source chooses the shortest path to transmit data. If the path is broken, the source will choose the shortest one among other routes found to continue the data transmission.

SP-LR only store the first route found, and employ the local repair mechanism provided in EASE. If the route is broken, the source will launch the local repair mechanism. If fails to repair the route locally, it will begin a new route discovery process.

MP-LR store all the routes found during the route discovery process, and employ the local repair mechanism provided in EASE as well. After the route discovery, the source chooses the shortest path to transmit data. If the path is broken, it will first launch the local repair mechanism. If fails to repair the route locally, the source will the shortest one among other routes found. Such process is repeatedly, until all the routes found in the previous route discovery process are broken and unable to be repaired locally. Then, the source has to begin a new route discovery process.

As to the latter method, i.e. forwarding the route requests with a certain probability, we will discuss it separately in Section 4.5.2.5

In the original protocols proposed, both SDAR [23, 70–72] and ANODR [22] only support $SP - NLR$, but they can be extended to support $MP - NLR$ as well. In contrast, EASE can support all the four simulation models.

4.5.2.2 Metrics for Evaluating the Simulation

Metrics we used for evaluating the anonymous routing protocols include: (1) the numbers of route discovery packets (including RREQ, RREP, RERR, and RRPR, RUPD, if any) sent and received, which are denoted as $RDISs$ and $RDISr$, respectively; (2) the numbers of all the packet (including all the route discovery packets and data transmission packets, i.e. data packets denoted as DATA and acknowledge packets denoted as ACK) sent and received, which are denoted as $PACKs$ and $PACKr$, respectively; (3) the numbers of route reply packets sent and received, which are denoted as $RREPs$ and $RREPr$, respectively. The reason that we pay more attention to route reply packets is based on the observation that, in both EASE and AONDR, the computational cost of handling a RREP packet is much larger than that of handling a RREQ packet or a packet of any other type, due to the costly asymmetric encryption/decryption; (4) the success rate of transmitting data messages. It is denoted as $SucRate$.

4.5.2.3 Different Node Densities

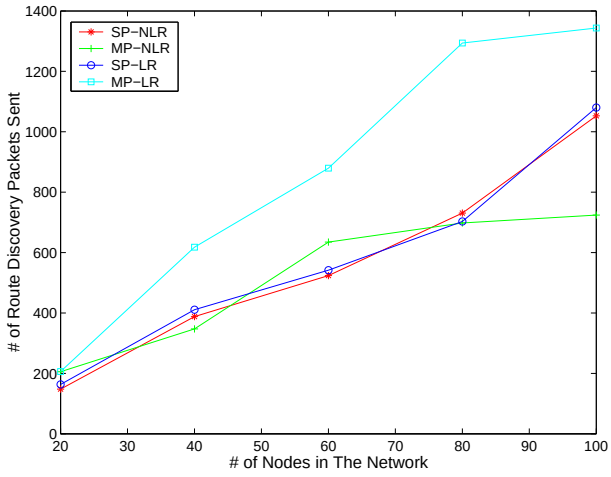


Figure 4.3: # of route discovery packets sent under different node densities

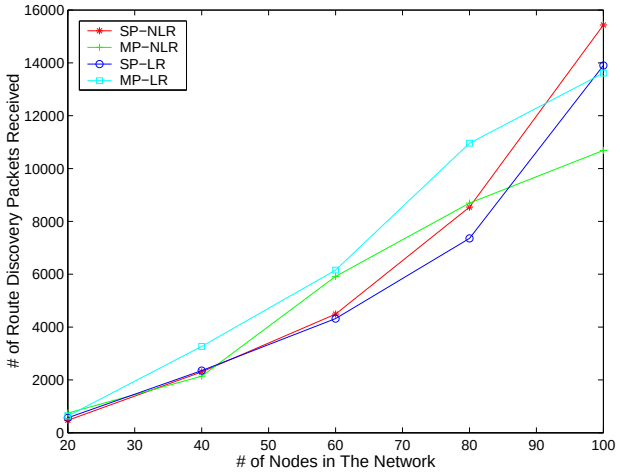


Figure 4.4: # of route discovery packets received under different node densities

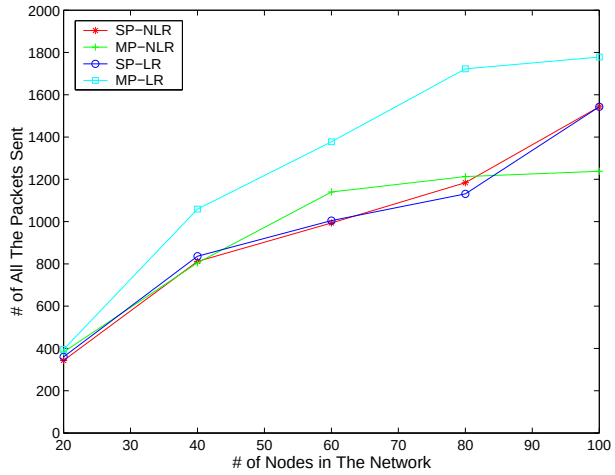


Figure 4.5: # of all the packets sent under different node densities

In this simulation, we test all the four models, i.e. SP-NLR, MP-NLR, SP-LR, and MP-LR, under different node densities. Let n denote the number of nodes within the simulation field or network. Five different node densities, i.e. $n = 20,\ 40,\ 60,\ 80,\ 100$, are simulated to analyze the efficiencies of the four models under low ($n \leq 40$), medium ($40 < n < 80$), and high ($n \geq 80$) density, respectively. In the simulation, mobility speed is chosen randomly from 0 to 20 m/sec, and the pause time is set to 0 second. The

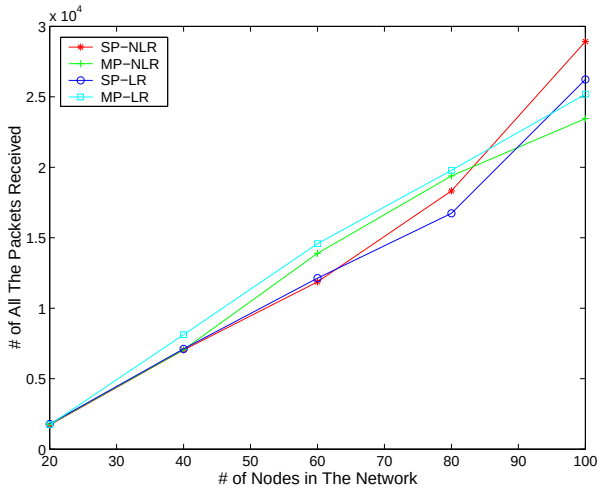


Figure 4.6: # of all the packets received under different node densities

simulation results under different node densities are shown in Figure 4.3 to Figure 4.9.

SP-LR vs. SP-NLR As shown in Figure 4.3 to Figure 4.9, SP-LR and SP-NLR are similar in terms of RDISs, PACKs, and RREPs/RREPr, but SP-LR has smaller RDISr and PACKr when the node density is high. This advantage results in less computation overhead involved in handling the route discovery and data transmission packets in the SP-NLR model when the node density is high. Besides that, we notice that in Figure 4.9 SP-LR has the highest success rate of transmitting DATA messages (i.e. SucRate), and the SucRate of SP-LR is around 1% to 8% higher than that of SP-NLR.

SP-LR vs. MP-NLR As shown in Figure 4.3 to Figure 4.6, compared to SP-LR, MP-NLR generates and receives more route discovery and data transmission packets when the node density is low to medium. However, we notice that MP-NLR is less sensitive to the increase of node density, and thus it is the simulation model with the least RDISs/RDISr and PACKs/PACKr when the node density is high. It seems that MP-NLR is a good choice when the network has a high density. Unfortunately, according

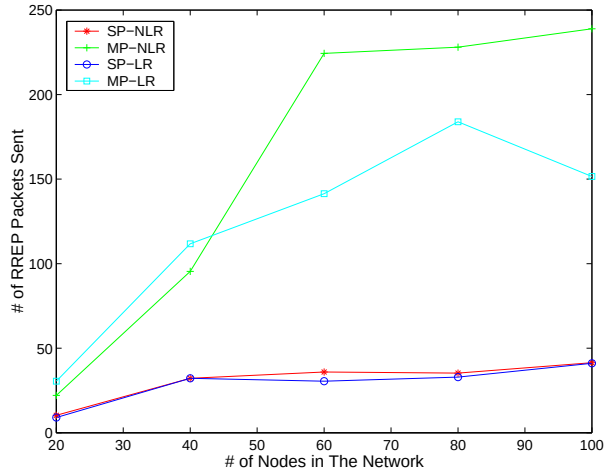


Figure 4.7: # of RREP packets sent under different node densities

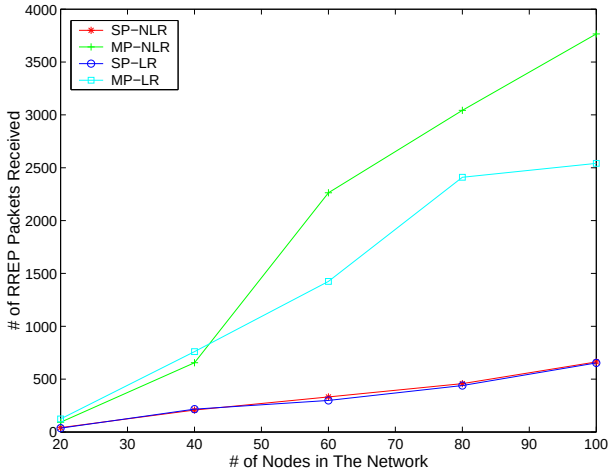


Figure 4.8: # of RREP packets received under different node densities

to Figure 4.7 and Figure 4.8, the RREPs/RREPr in MP-NLR is around 3 to 7 times as those in SP-LR when the node density is medium or high. Considering the costly operations (i.e. asymmetric encryption/decryption) involved in handling RREP packets, MP-NLR is less efficient than SP-LR. Moreover, the SucRate of MP-NLR is around 6% to 13% less than that of SP-LR.

SP-LR vs. MP-LR Compared to SP-LR, MP-LR has relative higher RDISs/RDISr and PACKs/PACKr, especially when the node density is medium. In addition, the

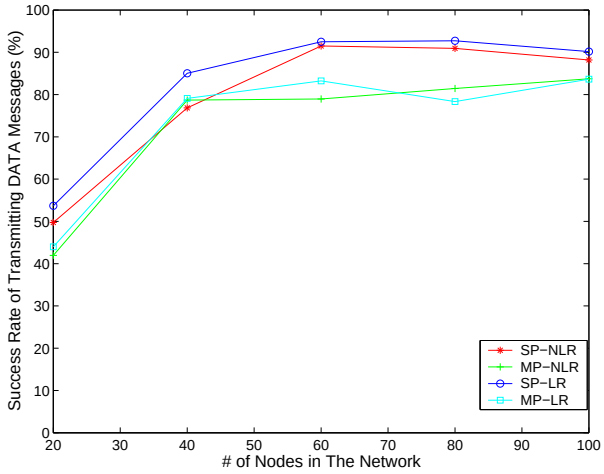


Figure 4.9: Success rate of transmitting DATA messages under different node densities

RREPs/RREPr of MP-LR are around 3 to 6 times as those of SP-LR, and the SucRate of MP-LR is around 6% to 14% higher than that of SP-LR.

4.5.2.4 Different Mobility Settings

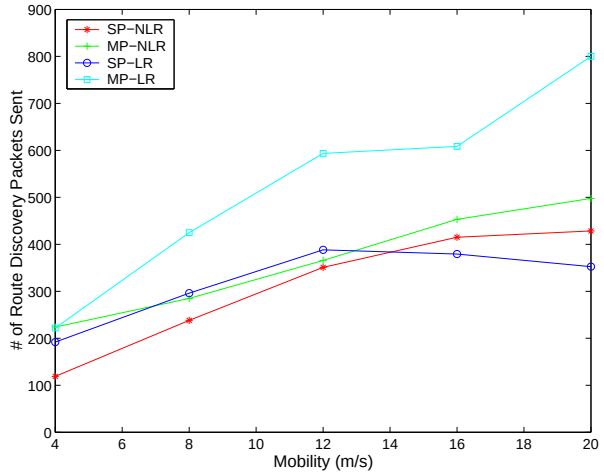


Figure 4.10: # of route discovery packets sent under different mobility settings

Similar to the simulation under different node densities, we test all the four models, i.e. SP-NLR, MP-NLR, SP-LR, and MP-LR, under different mobility settings. Suppose

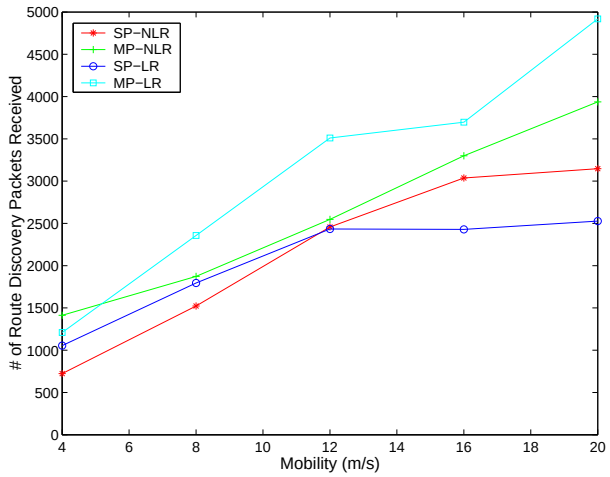


Figure 4.11: # of route discovery packets received under different mobility settings

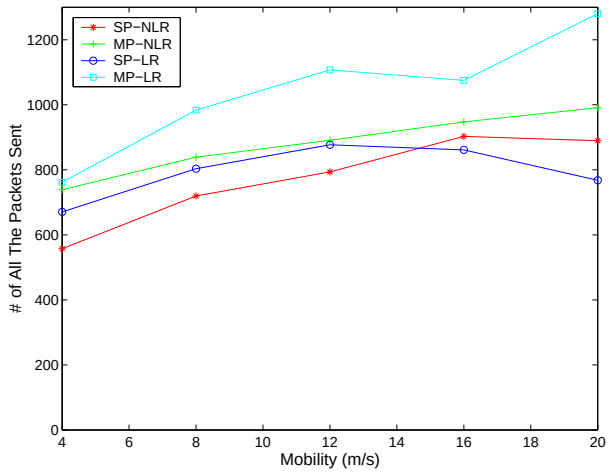


Figure 4.12: # of all the packets sent under different mobility settings

that, the speed of any node in the network is chosen randomly from 0 to v_{max}. Five different mobility settings, i.e. $v_{max} = 4,\ 8,\ 12,\ 16,\ 20\ m/s$, are simulated to analyze the efficiencies of the four models under low ($v_{max} \leq 8\ m/s$), medium ($8\ m/s < v_{max} < 16\ m/s$), and high ($v_{max} \geq 16\ m/s$) mobility, respectively. In the simulation, the pause time is set to 0 second. The simulation results under different mobility settings are shown in Figure 4.10 to Figure 4.16.

SP-LR vs. SP-NLR According to Figure 4.10 to Figure 4.15, SP-NLR generates

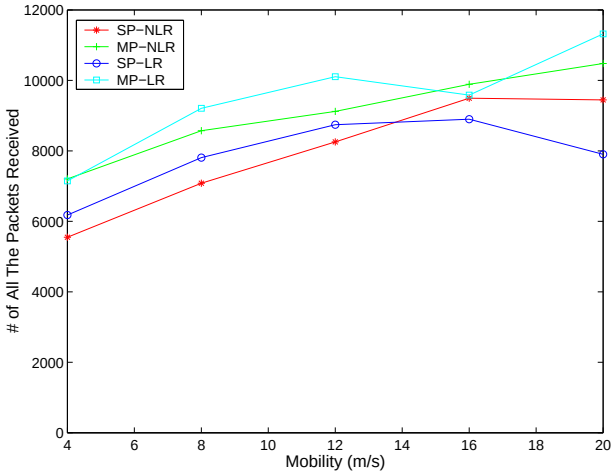


Figure 4.13: # of all the packets received under different mobility settings

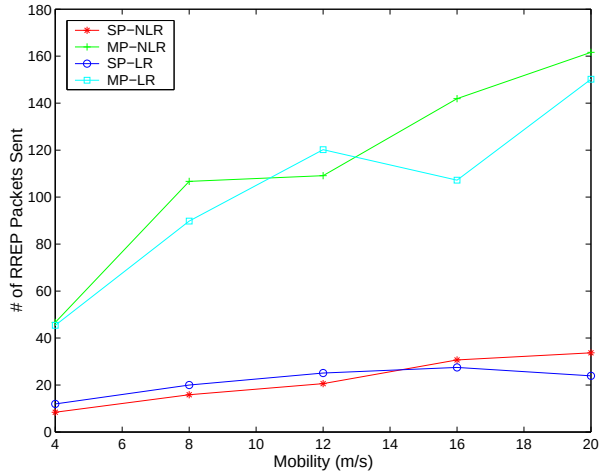


Figure 4.14: # of RREP packets sent under different mobility settings

and receives less route discovery and data transmission packets when the mobility is low. However, when the node mobility increases, especially when $v_{max} \geq 16\ m/s$, SP-LR is more efficient in the sense that it has less RDISs/RDISr, RREPs/RREPr, and PACKs/PACKr. Similarly, SP-LR has higher SucRates (i.e. around 2.5% to 5.5% higher than that of SP-NLR), when the node mobility is medium to high.

SP-LR vs. MP-NLR As shown in Figure 4.10 to Figure 4.13, compared to MP-NLR, SP-LR is less sensitive to the increase of mobility. More specifically, MP-NLR

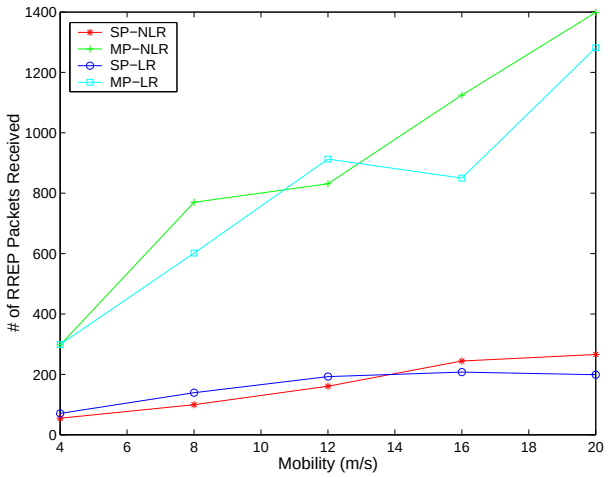


Figure 4.15: # of RREP packets received under different mobility settings

generates and receives similar or slightly higher numbers of route discovery and data transmission packets when the mobility is low to medium, but it tends to be inefficient when the mobility is medium to high. As to the RREPs/RREPr, in Figure 4.14 and Figure 4.15, we find that SP-LR is much more efficient when the mobility is medium or high. More specifically, MP-NLR generates and receives around 4 to 7 times RREP packets as SP-LR does. Besides that, the SucRate of SP-LR is around 4% to 13% higher than that of MP-NLR, when the mobility is medium or high.

SP-LR vs. MP-LR As shown in Figure 4.10 to Figure 4.15, SP-LR is more efficient than MP-LR, and such an advantage is more obvious when the node mobility increases. For example, when the mobility is medium to high, i.e. $v \geq 8\ m/s$, MP-LR generates and receives around 4 to 6 times RREP packets as SP-LR does. Besides that, the SucRate of SP-LR is around 6% to 14% higher than that of MP-LR, when the mobility is medium or high.

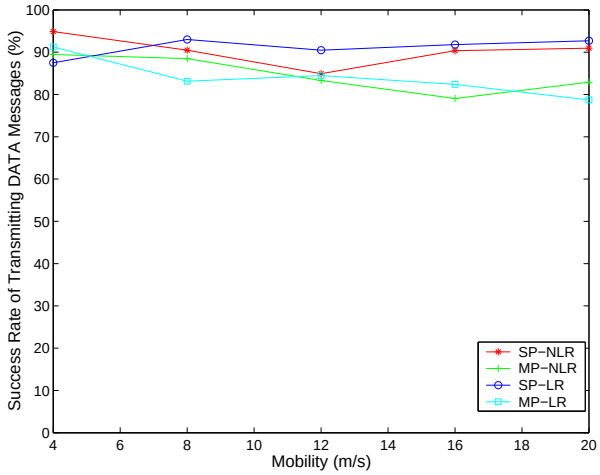


Figure 4.16: Success rate of transmitting DATA messages under different mobility settings

4.5.2.5 Forwarding Route Requests with A Certain Probability

In this simulation, we try to simulate the effectiveness of the method that forwards route requests with a certain probability denoted as p in improving the efficiency of anonymous routing protocols. We simulate the SP-LR model with six forwarding probabilities ranging from 50% to 100%. The empirical results are shown in Table 4.7. This table shows that, when the forwarding probability is 90%, we get the best SucRate, i.e. 90.90%. Besides that, compared to the case that nodes always forward the route requests received, i.e. $p = 100\%$, the network generates around 10% less RREP packets and around 5% to 9% less RDIS and PACK packets, when $p = 90\%$. However, when we try to decrease the forwarding probability further, as shown in Table 4.7, either the network generates more RREP, RDIS, and PACk packets, e.g. $p = 80\%$, or the SucRate drops around 5% to 9%, e.g. $p = 50\%, 60\%, 70\%$.

	RREPs/RREPr	RDISs/RDISr	PACKs/PACKr	SucRate
50%	38.15/302.4	484.2/3703.7	951.6/10371.6	81.30%
60%	34.15/288.15	391.3/3026.35	873.45/9871	85.55%
70%	37.45/305.6	463.95/3580.1	930.7/10094.3	81.85%
80%	38.55/333.3	457.05/3492.2	943.6/10626.95	89.05%
90%	35.6/280.2	442.45/3350.85	901.9/9455.8	90.90%
100%	38.4/312.75	463.55/3462.85	947.75/10347.7	89.90%

Table 4.7: Forwarding route requests with different probabilities

4.5.2.6 Communication Overhead over AODV

EASE has higher communication overhead compared to AODV, which does not have full anonymity and security supports. It is mainly due to the usage of one time public key, i.e. PK_i, and One-time Key Pad like random secrets (denoted as OKP secrets), i.e. U_{orig}, U_i, V_{orig}, V_i, although several fields in AODV, e.g. the IP address of next path node [74], are unnecessary in EASE and thus removed. The overhead is related to a few factors, including H_{max}, p_x/q_x, and q_d. Note that, the purposes of employing the OKP secrets in route request and route response are different. In the former, it is to let the destination deduce the length of the route found, and thus p_x can be set to a small number, e.g. 16, so long as the probability that the destination makes an incorrect judgement due to the collisions is very tiny. Instead, in route response, the shared secrets between consecutive nodes en route are used in anonymous data transmission at a later time, and thus q_x needs to be set to a larger number, e.g. 128. Given that H_{max} is set to 10 and q_d is chosen according to Equation (4.6), the overheads of EASE on route request and route response are around three and eight times, respectively, as that of AODV.

4.5.2.7 Summary

According the empirical results and analysis from Section 4.5.2.3 to Section 4.5.2.5, overall, SP-LR is the best choice, when taking all the metrics into consideration. It works well when the network becomes denser. Furthermore, compared to other models, SP-LR is more efficient when the mobility increases. Consequently, as the only anonymous routing protocol supporting the local repair mechanism, compared to SDAR [23, 70–72] and ANODR [22], *EASE* is more efficient in highly dynamic environments like ad hoc networks.

CHAPTER 5

Conclusion

In this chapter, we summarize our work in two major areas of securing ad hoc networks, i.e. key management and anonymous secure routing. The contributions achieved in these two areas are listed. Furthermore, we point out possible future work related to or based on our work presented in this dissertation.

5.1 Contributions of Our Work

The main contributions of our work can be divided into three parts:

5.1.1 Key Management Model for Large Ad Hoc networks

Handle Key Management Problem in Large Ad Hoc Networks Current research work [1–3,19,20] can handle only limited number of nodes. When the number of nodes increases, all of them become either inefficient or insecure. In this dissertation, we have proposed the AKM scheme which can handle ad hoc networks with a large number of nodes.

Self-organizing Regions In a dynamic environment, such as ad hoc networks, regions at different places or even the same region at different times may face different intensities of risks. In AKM, not only the structure of key management may change with the increase/decrease in the number of nodes,

but also different parts of the structure have the freedom to set appropriate configurations to cope with various levels of risks.

Risk Isolation Unlike in the flat structure, the hierarchical structure helps isolate risks from different regions. In addition, as shown in Section 3.4.1, GSK is well protected in AKM.

RTC – Security Index We proposed a new concept RTC, and the simulation results in Section 3.4.2 show that it is suitable to use RTC as the index of security situation of a region, if the two global rules are followed.

Small Computation Costs Involved Simulation results show that computation costs due to the variations, including node-based and region-based operations, are very small under common threshold and region size settings.

5.1.2 certification services against Active Attacks

Robust against Active Attacks Existing research work [1–3, 19, 20] can only resist passive attacks, such as dropping the certificate request, and are vulnerable under active attacks, such as the returning of a fake reply to the node requesting certification services. We propose two algorithms, both of which can defend active attacks on certification services.

Efficient Simulation results in Section 3.4.5 show that, even if adversaries do not launch active attacks, our second algorithm is still faster than [1–3]. For example, when the key length is 1024 bits, our second algorithm is around six to eight times faster.

Easy to Find Enough Neighbors That Provide certification services

The process of generating partial certificates in our second algorithm is extremely fast. For example, when the key length is 1024 bits, our algorithm is around 20 to 80 times faster. Such advantage is critical in ad hoc networks where intrinsically the less help a node requests from its neighbors, the higher chance it gets the help. Consequently, using our second algorithm, a node can easily find enough neighbor nodes which provide the certification service.

5.1.3 Anonymous Secure Routing

Re-define Anonymity and Security Requirements on Ad Hoc Network Routing Protocols In view of the fact that, in previous work [22–25, 70–72], the definition of anonymity is incomplete. We re-define the anonymity and security properties that are supposed to be achieved in anonymous secure routing protocols.

Provide "Identity Anonymity" Compared to previous work [22–24, 70–72], ASR provides a new anonymity property, "Identity Anonymity". Namely, no one involved in the routing protocol knows the identities of any other node en route, except that the source and destination know the identity of each other.

Provide "Strong Location Privacy" Another additional anonymity property achieved by ASR is "Strong Location Privacy". It means that no one knows the exact or even relative (i.e. the number of hops, from either the source or the destination) location of the source or the destination.

Anonymity and Security Analysis on ASR and Other Protocols We

also give a detailed analysis on the anonymity and security properties achieved in ASR and the previous work [22–25, 70–72].

Provide A Local Repair Mechanism without Compromising Security and Anonymity We proposed a local repair mechanism so that, instead of re-launching the costly flooding-based route discovery process, EASE can repair the broken part locally without compromising security and anonymity properties defined.

Simulate and Analyze Methods That Might be Able to Enhance Efficiency We simulated various types of methods that might be able to enhance the efficiency of anonymous routing protocols, e.g. storing multiple paths during the route discovery process and supporting the local repair mechanism, under different network and mobility models. The analysis on the simulation results shows that EASE is more efficient compared to previous solutions in highly dynamic environments like ad hoc networks.

5.2 Future Work

5.2.1 Key Management

As to the key management issue in ad hoc networks, in this dissertation, we managed to extend threshold schemes from the flat structure to the hierarchical structure. Such a change brings many advantages over those schemes based on the flat structure. At the same time, it also opens up a few new issues.

In AKM, RTC is proposed as a rough index of security situation of a region. As

shown in Section 3.4.2, it is not very accurate, since we notice that the security condition of a region may vary, even if its RTC remains unchanged. In addition, it is feasible within certain range – for a region with size less than 250 and RTC less than 0.1 or with size less than 120 and RTC less than 0.15. Therefore, it is still an open question to find a parameter that is capable of representing the security condition of a region more accurately and can be widely applied to different environments.

Currently, research work on different aspects of ad hoc network security, such as key management and security routing, were conducted separately. As a result of it, some solutions which are suitable for one aspect are not compatible with requirements of another aspect. Therefore, it is really desirable and challenging to build a more general structure which can be the uniform basis on which we can design schemes satisfying the requirements of different components. Consequently, one possible research direction is to incorporate the semi-dynamic hierarchical structure proposed in AKM into other areas of ad hoc network security or even other research fields of ad hoc network, such as network structure.

Another possible research direction is the security of group-based actions. All current research work is limited to the issue of a single node. In reality, it is common that nodes may join or leave as a group. Although group-based actions can be handled as a series of actions of single node, it takes more time to complete the authentication process. In addition, sometimes it is desirable to keep the relationship between group members even when they join a new environment. In AKM, the network is region-based. If treat each region as a group and nodes in the region as members of the group, we can make use of the key management and certification services provided by AKM to achieve the security

of group-based actions.

5.2.2 Privacy in Sensor Networks

Compared to ad hoc networks, on one hand, sensor networks allow the existence of certain more powerful nodes to collect and analyze information from normal sensors. Therefore, it is possible to setup the Trust Authority (TA) in the network. It seemingly makes the job of securing sensor networks more easier. On the other hand, however, due to the extremely limited computation and storage capability of normal sensors, the usage of Public Key Cryptography (PKC) is highly restricted. As such, EASE cannot be used directly in sensor networks, because the cost of hop-by-hop public key encryptions and verifications during the route discovery process is unaffordable. It would be a great challenge for researchers to find efficient solutions that can provide privacy in sensor networks.

Bibliography

[1] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for mobile ad-hoc networks," in *IEEE 9th International Conference on Network Protocols (ICNP'01)*, 2001.

[2] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang, "Self-securing ad hoc wireless networks," in *Seventh IEEE Symposium on Computers and Communications (ISCC'02)*, 2002.

[3] ——, "URSA: Ubiquitous and robust access control for mobile ad hoc networks," *IEEE/ACM Transactions on Networking*, 2004.

[4] S. Sesay, Z. Yang, and J. He, "A survey on mobile ad hoc wireless network," *Information Technology Journal*, vol. 3, no. 2, pp. 168–175, 2004.

[5] F. Peña-Mora, R. Aldunate, and M. Nussbaum, "Availability analysis of an ad-hoc DSMS for disaster relief environments," in *Proceedings of the 22nd International Conference of the Chilean Computer Science Society (SCCC 2002)*, Nov. 2002, pp. 59–71.

[6] S. P. McGrath, E. B. Entin, R. S. Gray, and L. Shay, "The ActComm project: Mobile agents and ad hoc routing meeting military requirements for information superiority," in *IEEE Military Communications Conference (MILCOM 2001)*, vol. 1, Oct. 2001, pp. 28–31.

[7] M. Choudhary, P. Sharma, and D. Sanghi, "Secure multicast model for ad-hoc military networks," in *Proceedings of the 12th IEEE International Conference on Networks (ICON 2004)*, vol. 2, Nov. 2004, pp. 16–19.

[8] J. A. Davis, A. H. Fagg, and B. N. Levine, "Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks," in *Proceedings of the Fifth International Symposium on Wearable Computers*, Oct. 2001, pp. 141–148.

[9] J. P. G. Sterbenz, R. Krishnan, R. R. Hain, A. W. Jackson, D. Levin, R. Ramanathan, and J. Zao, "Survivable mobile wireless networks: Issues, challenges, and research directions," in *Proceedings of the ACM Workshop on Wireless Security (WiSe'02)*, 2002, pp. 31–40.

[10] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Proceedings of the 7th International Workshop on Security Protocols, Lecture Notes in Computer Science 1796*, 1999, pp. 172–194.

[11] N. Asokan and P. Ginzboorg, "Key agreement in ad hoc networks," *Computer Communications*, vol. 23, pp. 1627–1637, 2000.

[12] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless ad hoc networks," in *Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, 2003.

[13] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing*, vol. 353, 1996.

[14] D. B. Johnson, D. A. Maltz, and J. Broch, "DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks," *Ad Hoc Networking*, 2001.

[15] R. Anderson and M. Kuhn, "Tamper resistance - a cautionary note," in *Proceeding of 2nd USENIX Workshop on Electronic Commerce*, 1996.

[16] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer., "A secure routing protocol for ad hoc networks," in *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP)*, 2002.

[17] P. Papadimitratos and Z. J. Haas, "Secure routing for mobile ad hoc networks," in *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, January 2002.

[18] L. Zhou and Z. J. Haas, "Securing ad hoc networks," *IEEE Network Magazine, Special Issue on Network Security*, vol. 13, no. 6, 1999.

[19] B. Lehane, L. Doyle, and D. O'Mahony, "Shared RSA key generation in a mobile ad hoc network," in *Proceedings of IEEE Military Communications Conference (MILCOM 2003)*, 2003.

[20] A. Khalili, J. Katz, and W. Arbaugh, "Toward secure key distribution in truly ad-hoc networks," in *IEEE Workshop on Security and Assurance in Ad Hoc Networks, in Conjunction with the 2003 International Symposium on Applications and the Internet*, 2003.

[21] S. Čapkuny, L. Buttyán, and J.-P. Hubaux, "Self-organized public-key management for mobile ad hoc networks," EPFL/IC, Tech. Rep. 2002/34, May 2002.

[22] J. Kong and X. Hong, "ANODR: ANonymous on demand routing with untraceable routes for mobile ad-hoc networks," in *Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'03)*, 2003, pp. 291–302.

[23] K. El-Khatib, L. Korba, R. Song, and G. Yee, "Secure dynamic distributed routing algorithm for ad hoc wireless networks," in *International Conference on Parallel Processing Workshops (ICPPW'03)*, 2003.

[24] Y. Zhang, W. Liu, and W. Lou, "Anonymous communications in mobile ad hoc networks," in *IEEE INFOCOM 2005*, Mar. 2005.

[25] X. Wu and B. Bhargava, "AO2P: Ad hoc on-demand position-based private routing protocol," *IEEE Transactions on Mobile Computing*, to appear.

[26] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proceedings of the 28th IEEE Symposium on the Foundations of Computer Science*, 1987, pp. 427–437.

[27] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology – Crypto'91, Lecture Notes in Computer Science 576*, 1991, pp. 129–140.

[28] Y. Desmedt, "Society and group oriented cryptography," in *Advances in Cryptology - Crypto '87 Proceedings, Lecture Notes in Computer Science 0293*. Springer-Verlag, 1988, pp. 120–127.

[29] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *Proceeding of Advances in Cryptology - CRYPTO '89, Lecture Notes in Computer Science 0435*, 1990, pp. 307–315.

[30] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, November 1979.

[31] M. Abdalla, S. Miner, and C. Namprempre, "Forward security in threshold signature schemes," in *Topics in Cryptology - CT-RSA 2001, LNCS 2020*, 2001, pp. 441–456.

[32] Y. Frankel and Y. G. Desmedt, "Parallel reliable threshold multi-signature," Dept. of EECS, University of Wisconsin-Milwaukee, Tech. Rep. TR-92-04-02, 1992.

[33] S. Jarecki., "Proactive secret sharing and public key cryptosystems," Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Sept. 1995.

[34] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing or: How to cope with perpetual leakage," in *Advances in Cryptology – Crypto '95, Lecture Notes in Computer Science 963*, 1995, pp. 457–469.

[35] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive public-key and signature schemes," in *Proceedings of the Fourth Annual Conference on Computer Communications Security*, 1997, pp. 100–110.

[36] Y. Frankel, P. Gemmel, P. MacKenzie, and M. Yung, "Optimal resilience proactive public-key cryptosystems," in *Proceedings of the 38th Symposium on Foundations of Computer Science*, 1997.

[37] Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung, "Proactive RSA," in *Advances in Cryptology – Crypto '97, Lecture Notes in Computer Science 1294*, 1997, pp. 440–454.

[38] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract)," in *Proceedings of the 26th IEEE Annual Symposim on Foundations of Computer Science*, Oct. 1985, pp. 383–395.

[39] J. C. Benaloh, "Secret sharing homomorphisms: Keeping shares of a secret secret," in *Proceedings of the 6th Annual International Conference: Advances in Cryptology (CRYPTO 1986), Lecture Notes in Computer Science 263*, 1986, pp. 213–222.

[40] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust threshold DSS signatures," in *Advances in Cryptography – Proceedings of Eurocrypt '96, Lecture Notes in Computer Science 1070*, 1996, pp. 354–371.

[41] R. Gennaro and S. Micali, "Verifiable secret sharing as secure computation," in *Proceedings of Advances in Cryptology – EUROCRYPT'95, Lecture Notes in Computer Science 921*, 1995, pp. 50–63.

[42] O. Goldreich, S. Micali, and A. Wigderson, "How to prove all NP statements in zero-knowledge and a methodology of cryptograhpic protocol design," in *Proceedings of Advances in Cryptology—CRYPTO '86, Lecture Notes in Computer Science 263*, 1986, pp. 171–185.

[43] T. Rabin, "Robust sharing of secrets when the dealer is honest or cheating," *Journal of the ACM*, vol. 41, no. 6, pp. 1089–1109, Nov. 1994.

[44] T. Rabin and M. Ben-Or, "Verifiable secret sharing and multiparty protocols with honest majority," in *Proceedings of the 21st Symposium on the Theory of Computing*, May 1989, pp. 73–85.

[45] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom 2002)*, 2002, pp. 12–23.

[46] S. Yi and R. Kravets, "MOCA: Mobile certificate authority for wireless ad hoc networks," in *2nd Annual PKI Research Workshop Program (PKI 03)*, 2003.

[47] M. Narasimha, G. Tsudik, and J. H. Yi, "On the utility of distributed cryptography in P2P and MANETs: The case of membership control," in *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03)*, Nov. 2003, pp. 336–345.

[48] N. Saxena, G. Tsudik, and J. H. Yi, "Admission control in peer-to-peer: Design and performance evaluation," in *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2003, pp. 104–114.

[49] J.-P. Hubaux, L. Buttyn, and S. Čapkun, "The quest for security in mobile ad hoc networks," in *Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2001.

[50] S. Čapkun, L. Buttyán, and J.-P. Hubaux, "Self-organized public-key management for mobile ad-hoc networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, January-March 2003.

[51] Y. Desmedt, "Threshold cryptography," *European Transactions on Telecommunications*, vol. 5, no. 4, pp. 449–457, 1994.

[52] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[53] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, Oct. 1996, ch. 3, p. 103.

[54] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proceedings of Advances in Cryptology (CRYPTO 2001), Lecture Notes in Computer Science 2139*, 2001, pp. 213–229.

[55] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proceedings of Advances in Cryptology (ASIACRYPT 2001), Lecture Notes in Computer Science 2248*, 2001, pp. 512–532.

[56] J. C. Cha and J. H. Cheon, "An identity-based signature from gap diffie-hellman groups," in *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography: Public Key Cryptography, Lecture Notes In Computer Science 2567*, 2003, pp. 18–30.

[57] S. Čapkun and J.-P. Hubaux, "BISS: Building secure routing out of an incomplete set of security association," in *The Second ACM Workshop on Wireless Security (WiSe)*, 2003, pp. 21–30.

[58] S. Čapkun, J.-P. Hubaux, and L. Buttyan, "Mobility helps security in ad hoc networks," in *The 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003)*, 2003, pp. 46–56.

[59] S. Garfinkel, *PGP: Pretty Good Privacy.* OReilly & Associates, 1995.

[60] P. Zimmermann, *The Official PGP Users Guide.* MIT Press, 1995.

[61] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[62] Anonymizer.Com, "Online privacy services," http://www.anonymizer.com.

[63] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security (TISSEC)*, vol. 1, no. 1, pp. 66–92, 1998.

[64] O. Berthold, H. Federrath, and S. Kospell, "Web MIXes: A system for anonymous and unobservable internet access," in *Proceedings of Workshop on Design Issues in Anonymity and Unobservability (DIAU'00), Lecture Notes in Computer Science 2009*, 2000, pp. 115–129.

[65] C. Shields, "Secure hierarchical multicast routing and multicast internet anonymity," Ph.D. dissertation, Computer Engineering, University of California, Santa Cruz, 1999.

[66] C. Shields and B. N. Levine, "A protocol for anonymous communication over the internet," in *ACM Conference on Computer and Communications Security (CCS 2000)*, 2000, pp. 33–42.

[67] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," *IEEE Journal on Selected Areas in Communications, Special Issue on Copyright and Privacy Protection*, vol. 16, no. 4, pp. 482–494, 1998.

[68] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, 2004.

[69] "Privoxy," http://www.privoxy.org/ retrieved on June 23, 2005.

[70] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba, "SDAR: A secure distributed anonymous routing protocol for wireless and mobile ad hoc networks," in *The Fourth International IEEE Workshop on Wireless Local Networks (WLN 2004)*, Oct. 2004, pp. 618–624.

[71] ——, "A novel solution for achieving anonymity in wireless ad hoc routing protocol," in *Proceedings of Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor and Ubiquitous Networks (ACM PE-WASUN'2004)*, Nov. 2004.

[72] ——, "An efficient secure distributed anonymous routing protocol for mobile and wireless ad hoc networks," *Computer Communications Journal*, vol. 28, no. 10, pp. 1193–1203, June 2005.

[73] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *WMCSA'99*, 1999.

[74] C. Perkins, E. Royer, and S. Das, "Ad hoc on-demand distance vector routing," June 2001, draft-ietf-manet-aodv-08.txt, IETF MANET Working Group.

[75] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Mobile Computing and Networking*, 2000.

[76] P. Papadimitratos and Z. J. Haas, "Secure data transmission in mobile ad hoc networks," in *Proceedings of the 2003 ACM Workshop on Wireless Security*, 2003, pp. 41 – 50.

[77] P. Papadimitratos and Z. Haas, "Secure message transmission in mobile ad hoc networks," *Elsevier Ad Hoc Networks Journal*, vol. 1, no. 1, 2003.

[78] M. G. Zapata, "Secure ad hoc on-demand distance vector (SAODV) routing," Aug. 2002, internet draft, draft-guerrero-manet-saodv-00.txt.

[79] Y.-C. Hu, D. B. Johnson, and A. Perrig, "SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks," in *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002)*, June 2002, pp. 3–13.

[80] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless ad hoc networks," Department of Computer Science, Rice University, Tech. Rep. TR01-384, Sept. 2002.

[81] L. Lamport, "Password authentication with insecure communication," *Comm. of ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[82] R. Droms, "Dynamic host configuration protocol," Mar. 1997, iETF RFC 2131.

[83] M. Hattig, "Zero-conf IP host requirements," Aug. 2001, draft-ietf-zeroconf-reqts-09.txt, IETF MANET Working Group.

[84] Z. J. Haas and M. R. Pearlman, "The performance of query control schemes for the zone routing protocol," in *Proceedings of ACM SIGCOMM'98*, Aug. 1998.

[85] Z. J. Haas, "A new routing protocol for the reconfigurable wireless networks," in *Proceedings of the IEEE ICUPC*, 1997.

[86] Z. J. Haas, M. R. Pearlman, and P. Samar, "The zone routing protocol (ZRP) for ad hoc networks," July 2002, internet draft, draft-ietf-manet-zone-zrp-02.txt.

[87] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector (DSDV) routing for mobile computers," in *Proceedings of ACM SIGCOMM'94*, Aug. 1994.

[88] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *MOBICOM*, 1998, pp. 85–97.

[89] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based performance analysis of routing protocols for mobile ad hoc networks," in *Proceedings of MOBICOM*, 1999.

[90] A. Perrig, R. Canetti, J. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," in *IEEE Symposium on Security and Privacy*, 2000.

[91] A. Perrig, R. Canetti, D. Song, and J. Tygar, "Efficient and secure source authentication for multicast," in *Proceedings of NDSS 2001*, 2001.

[92] A. Perrig, R. Canetti, D. Tygar, and D. Song, "The TESLA broadcast authentication protocol," *RSA CryptoBytes*, vol. 5, no. 2, pp. 2–13, 2002.

[93] S. Čapkun, J.-P. Hubaux, and M. Jakobsson, "Secure and privacy-preserving communication in hybrid ad hoc networks," EPFL-IC, Tech. Rep. IC/2004/10, 2004.

[94] P. McDaniel and S. Jamin, "A scalable key distribution hierarchy," Electrical Engineering and Computer Science, University of Michigan, Tech. Rep. CSE-TR-366-98, 1998.

[95] J. Kong, H. Luo, K. Xu, D. L. Gu, M. Gerla, and S. Lu, "Adaptive security for multi-layer ad-hoc networks," *John Wiley InterScience Press journal, Special Issue of Wireless Communications and Mobile Computing*, 2002.

[96] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystem," in *Eurocrypt '99*, 1999, pp. 295–310.

[97] Y. Desmedt and S. Jajodia, "Redistributing secret shares to new access structures and its applications," Department of Information and Software Systems Engineering, George Mason University, Tech. Rep. ISSE-TR-97-01, July 1997.

[98] T. M. Wong, "Decentralized recovery for survivable storage systems," Ph.D. dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, May 2004.

[99] D. R. Stinson and R. Strobl, "Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates," in *The 6th Australasian Conference on Information Security and Privacy (ACISP 2001), Lecture Notes in Computer Science 2119*, 2001, pp. 417–434.

[100] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inform. Theory*, vol. 31, pp. 469–472, 1985.

[101] N. I. for Standards and Technology, "Digital signature standard (DSS)," 1998, fIPS 186-1.

[102] C. P. Schnorr, "Efficient signature generation for smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 239–252, 1991.

[103] C. Park and K. Kurosawa, "New ElGamal type threshold digital signature scheme," *IEICE TRANS Fundamentals Special Section on Cryptography and Information Security*, vol. E79-A, no. 1, pp. 86–93, 1996.

[104] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, L. Viennot, and I. Rocquencourt, "Optimized link state routing protocol," Sept. 2001, internet draft, draft-ietf-manet-olsr-06.txt.

[105] S.-J. Lee, E. M. Belding-Royer, and C. E. Perkins, "Scalability study of the ad hoc on-demand distance vector routing protocol," *International Journal of Network Management*, vol. 13, no. 2, pp. 97–114, 2003.

[106] J.-F. Raymond, "Traffic analysis: Protocols, attacks, design issues, and open problems," in *DIAU00, Lecture Notes in Computer Science 2009*, 2000, pp. 10–29.

[107] A. Pfitzmann, B. Pfitzmann, and M. Waidner, "ISDN-MIXes: Untraceable communication with very small bandwidth overhead," in *Proc. GI/ITG-Conference "Kommunikation in Verteilten Systemen" (Communication in Distributed Systems)*, 1991, pp. 451–463.

[108] D. Kesdogan, J. Egner, and R. Bschkes, "Stop-and-go-MIXes providing probabilistic anonymity in an open system," in *Second International Workshop on Information Hiding, Lecture Notes in Computer Science 1525*, 1998, pp. 83–98.

[109] O. Berthold, H. Federrath, and M. Kohntopp, "Project anonymity and unobservability in the internet," in *Computers Freedom and Privacy Conference 2000 (CFP 2000), Workshop on Freedom and Privacy by Design*, 2000.

[110] R. Barr, "An efficient, unifying approach to simulation using virtual machines," Ph.D. dissertation, Cornell University, May 2004.

[111] R. Barr, Z. J. Haas, and R. V. Renesse, "JiST: Embedding simulation time into a virtual machine," in *EuroSim Congress on Modelling and Simulation*, Sept. 2004.

APPENDIX A

Thesis-related Publications

A.1 Book Chapters

1. Ren Kui, Bo Zhu, Zhiguo Wan, and Wenjing Lou, invited chapter in the book "*Progress in Computer Network Research*", Nova Science Publishers, in preparation.

A.2 Journal Publications

2. Bo Zhu, Feng Bao, Robert H. Deng, Mohan S. Kankanhalli, and Guilin Wang, "*Efficient and Robust Key Management for Large Mobile Ad-hoc Networks*", Journal of Computer Networks, Volume 48, Issue 4, July 2005, pp. 657-682.

A.3 Conference Publications

3. Bo Zhu, Guilin Wang, Zhiguo Wan, Mohan S. Kankanhalli, Feng Bao, and Robert H. Deng, "*Providing Robust Certification Services Against Active Attacks in Ad Hoc Networks*", The 24th IEEE International Performance Computing and Communications Conference (IPCCC 2005), April 7-9, 2005, Phoenix, Arizona, USA.
4. Zhiguo Wan, Bo Zhu, Robert H. Deng, Feng Bao, and Akkihebbal L. Ananda,

"*DOS-Resistant Access Control Protocol with Identity Confidentiality for Wireless Networks*", The Sixth IEEE Wireless Communications and Networking Conference (WCNC 2005), March 13-17, 2005, New Orleans, LA, USA.

5. Bo Zhu, Zhiguo Wan, Mohan S. Kankanhalli, Feng Bao, Robert H. Deng, "*Anonymous Secure Routing in Mobile Ad-Hoc Networks*", The 29th Annual IEEE Conference on Local Computer Networks (LCN 2004), Tampa, Florida, U.S.A., November 16-18, 2004.
6. Bo Zhu, Tieyan Li, Huafei Zhu, Mohan S. Kankanhalli, Feng Bao, Robert H. Deng, "*Trust Establishment in Large Scale Grid Settings*", The Third International Conference on Grid and Cooperative Computing (GCC 2004), LNCS 3251, pp. 317-324, October 21-24, 2004 (Acceptance Rate 23%=99/426).
7. Guilin Wang, Xiaoxi Han and Bo Zhu, "*On the Security of Two Threshold Signature Schemes with Traceable Signers*", The First International Conference on Applied Cryptography and Network Security (ACNS 2003), LNCS 2846, pp. 111-122, 2003 (Acceptance Rate 16%=30/191).