ABSTRACT

WANG, PAN. Securing Communication in Dynamic Network Environments. (Under the direction of Professor Douglas S. Reeves and Associate Professor Peng Ning).

In dynamic network environments, users may come from different domains, and the number of users and the network topology may change unpredictably over time. How to protect the users' communication in such dynamic environments, therefore, is extremely challenging. This dissertation has investigated multiple research problems related to securing users' communication in dynamic network environments, focusing on two kinds of dynamic networks, i.e., mobile ad hoc networks and overlay networks. It first introduces a secure address auto-configuration scheme for mobile ad hoc networks, since a precondition of network communication is that each user is configured with a unique network identifier (address). This proposed auto-configuration scheme binds each address with a public key, allows a user to self-authenticate itself, and thus greatly thwarts the address spoofing attacks, in the absence of centralized authentication services. Next, this thesis presents two storage-efficient stateless group key distribution schemes to protect the group communication of a dynamic set of users. These two key distribution schemes utilize one-way key chains with a logical tree. They allow an authorized user to get updated group keys even if the user goes off-line for a while, and significantly reduce the storage requirement at each user if compared with previous stateless key distribution schemes. Third, this thesis investigates the solution using cryptographic methods to enforce network access control in mobile ad hoc networks, whose dynamic natures make it difficult to directly apply traditional access control techniques such as firewalls. A functioning prototype demonstrates the proposed access control system is practical and effective. Finally, this dissertation introduces a k-anonymity communication protocol for overlay networks to protect the privacy of users' communication. Unlike the existing anonymous communication protocols that either cannot provide provable anonymity or suffer from transmission collision, the proposed protocol is transmission collision free and provides provable k-anonymity for both the sender and the recipient. The analysis shows the proposed anonymous communication protocol is secure even under a strong adversary model, in which the adversary controls a fraction of nodes, is able to eavesdrop all network traffic and maliciously modify/replay the transmitted messages. A proof-of-concept implementation demonstrates the proposed protocol is practical.

Securing Communication in Dynamic Network Environments

by

Pan Wang

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Engineering

Raleigh, North Carolina

2007

Approved By:

Dr. Gregory T. Byrd

Dr. Wenye Wang

Dr. Douglas S. Reeves
Chair of Advisory Committee

Dr. Peng Ning

I would like to dedicate this dissertation
to my mother, Zhilian Song and my father, Liuguo Wang,
with my deep appreciation for their love, encouragement and patience.

Biography

Pan Wang received his Bachelor's degree from the Department of Statistics at Renmin University of China in 1996. He received his Master's degree from the Department of Electrical and Computer Engineering at North Carolina State University in 2003, and then he continued with his Ph.D program at North Carolina State University. His research interests are in computer and information security, especially on distributed systems security, wireless security, and privacy protection in open systems. Currently, he is working on the anonymous communication problem.

Acknowledgements

Though only my name appears on the cover of this dissertation, many people have contributed to its production in many ways. I would like to express my sincere gratitude to all of them.

First of all, I would like to thank my advisor, Dr. Douglas S. Reeves. I have been fortunate enough to have an advisor who gave me the freedom to explore on my own, and the same time the guidance to recover when my steps faltered. His support helped me overcome many hard situations and made this dissertation possible. I would also like to thank my co-advisor, Dr. Peng Ning. He has been always there to listen and give advice. I am deeply grateful to him for posing challenging problems, insisting on both practical and theoretical significants for each research result, and teaching me how to do research. My thanks also go to the members of my committee, Dr. Gregory T. Byrd and Dr. Wenye Wang, for their valuable suggestions to my research.

I would also like to thank my colleagues and friends at North Carolina State University. They are: Kun Sun, Dingbang Xu, Donggang Liu, Young June Pyun, Fang Feng, Qinglin Jiang, Hua Li, Qinghua Zhang, Qing Zhang, Yi Zhang, Juan Du, Keith Irwin, and An Liu. Without their emotional and social support, this dissertation would not have been possible. A special thank goes to a nice couple, Dr. Ting Yu and Dr. Xiaosong Ma. I deeply appreciate their encouragements on my research, and thank them for helping me to improve my badminton skill. I really enjoyed the time that we played badminton games together.

Finally, and most importantly, I would like to thank my wife Weiwei Wu. Her support, encouragement, patience and unwavering love are undeniably the bedrock upon which the past ten years of my life have been built. I thank my parents, Zhilian Song and Liuguo Wang, for their faith in me. They have always been encouraging and supporting me to pursue my dreams. I would also like to thank my younger sister, Haiyan Wang, for taking care of our parents in China and thank my lovely niece, Yuqian Wang, for bringing some sunshine into my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Some electrical communication networks (or *networks* in short), e.g., peer-to-peer computer networks, allow entities to join and leave freely that may cause the change of their network topologies over time. To formalize the dynamic feature of such networks, we introduce the concept of *dynamic network environment* and define a network has dynamic network environment if the set of its participants and the network topology may change over time. A network with dynamic network environment is, therefore, also referred to as a *dynamic network*.

In dynamic network environments, the participants that are referred to as the *nodes* or the *users* may come from different organizations and have different interests. Their diverse interests often require the properties such as confidentiality, integrity and privacy for their communication. However, since the set of nodes and the network topology change dynamically and unpredictably over time, how to achieve these properties, i.e., to secure the users' communication, in such highly dynamic network environments is a challenging issue, if compared with that in static systems, in which the fixed network topology and the predetermined user memberships can be used to protect users' communication.

This dissertation has investigated several research problems related to securing users' communication in dynamic network environments. It focuses on two kinds of dynamic networks, i.e., mobile ad hoc networks and overlay networks.

Mobile ad hoc network is an autonomous collection of mobile users that communicate over wireless links. It promotes flexibility and mobility by not requiring fixed infrastructure such as cell sites or wireless access points. Users are free to move randomly and organize themselves arbitrarily. As a result, the user memberships and the network topology of a mobile ad hoc network may change dynamically.

Mobile ad hoc network has a wide range of applications. It is especially attractive for the applications that require rapidly-deployable communication between a dynamically changing set of participants, e.g., emergency/resue operations, disaster relief efforts, and military networks, due to the ease of deployment.

Overlay network is a computer network which is built on top of another network, e.g., Internet or telephone network. The overlay nodes are connected by virtual links (or "tunnels"), each of which corresponds to a physical path on the top of the underlying network infrastructure. The overlay has no control over packets routing in the underlying network between two overlay nodes, but it may control the sequence of overlay nodes a message traverses before reaching its destination. An overlay network does not require the deployment of new equipments in the underlaying network, as long as the participating nodes run the overlay protocol. Therefore, an overlay network can be easily constructed without the cooperation from Internet Service Providers (ISPs). It supports a dynamic population of users, i.e., each user can join and leave the overlay network freely, which leads to a dynamically changing network topology.

Overlay networks have a diverse set of applications, including distributed data storage (e.g., cooperative file system [36] and OceanStore [71]), distributed bandwidth sharing (e.g., Skype [7] and OverQoS [114]), and distributed file sharing (e.g., Gnutella [5] and KaZaa [6]). They have also been proposed as a way to improve Internet routing, such as quality-of-service and multicast, since the previous protocols (e.g., DiffServ [16]) require modification of all routers in the network.

1.1 Problem Statement

As stated earlier, users in a dynamic network may come from different domains. The dynamics of user memberships and network topology may make these users more worry about the security of their communication. There are three major security concerns: (i) confidentiality and integrity, (ii) access control, and (iii) privacy.

Confidentiality has been defined by the International Organization for Standardization (ISO) as "ensuring that information is accessible to those authorized to have access". It is one of the cornerstones of information security. *Integrity* means the assurance of authenticity and completeness of the received data. *Access control* means the restriction of participating a network, i.e., who has the right to join the network. *Privacy* generally means a user's secret and personal information that he/she does not want to be revealed publicly in the procedure of communication.

In electrical communication systems, as illustrated by figure 1.1, a user's input message that may be of the form of voice, a picture, or plain text in some particular language, such as English or Chinese, is converted into an electrical signal that is suitable for transmission. This electrical signal is transmitted through the physical channel or transmission medium. The destination receiver captures the electrical signal (basing on its network identifier and the employed transmission protocol). It then recovers the message contained in the received electrical signal as output signal.

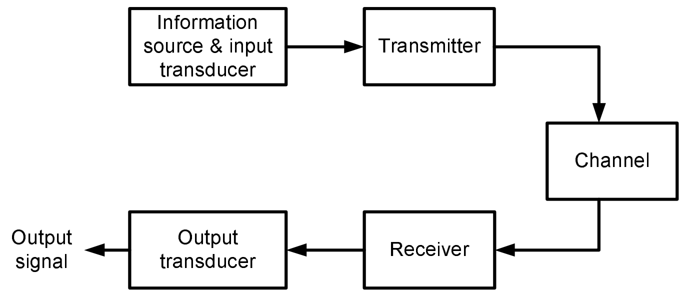


Figure 1.1: Functional block diagram of an electrical communication system.

Though electrical (network) communication provides convenience to users, it is vulnerable to a lot of attacks. This is because the communicating entities are not necessary the human beings, these entities may be apart from each other's visible distance, and the electrical signal may be captured/interrupted by other users' transmission devices.

An attacker may maliciously use other users' network identifiers, i.e., to spoof the latter, and then hijack the victims' traffic. Such an attack breaks the confidentiality of communication. It is a serious and realistic threat if there is no authentication mechanism allowing a node to prove the ownership of its network identifier (address). In some dynamic networks, e.g., mobile ad hoc networks, address auto-configuration is desirable since it is impossible to manually configure all nodes. Unfortunately, the existing auto-configuration protocols for mobile ad hoc networks lack an effective address authentication mechanism.

How to authenticate an address in the absence of centralized authentication services, e.g., to secure the address auto-configuration for mobile ad hoc networks, is a challenging task.

The attacker may modify the transmitted messages that threatens the integrity of application data. Even though there are mature and well-developed methods to protect the two-party communication (e.g., using the Diffie-Hellman key exchange protocol [40] to establish a secret key), how to efficiently secure the group oriented communication (i.e., one-to many or many-to-many communication pattern) in dynamic network environments is still relatively unexplored.

Without an access control mechanism, an illegitimate node may enter the network freely and then abuse the network resource, e.g., network bandwidth. Firewalls have been used to effectively enforce network access control, using network topology and service information, in conventional networks. However, they cannot be deployed directly in some dynamic network environments, e.g., mobile ad hoc networks, since there is no network topology information available. How to enforce access control in these dynamic network environments is a challenging issue.

An adversary may deduce some valuable information about a user via eavesdropping or tracing the latter's communication traffic. For instance, by tracing the online transactions, an attacker may be able to refer some private information about a customer (e.g., the IP address of the customer's computer, his/her account name, the email address, and even his/her social security number). A realistic proof of such a threat is the AOL case. That is, AOL disclosed 19 million Web search requests made by about 658,000 subscribers during a three-month period ending in May 2006. This released material includes 175 searches containing Social Security numbers, which can provide a stepping stone to identity theft [116]. Encryption can effectively protect the content of communication. However, it does not conceal the fact that two users are communicating. Therefore, it is desirable to design a communication protocol that can hide users' identities in their communication.

In this dissertation, we do not aim at designing a single comprehensive protocol that can cover all above security concerns. Since the original design of Internet protocol (IPv4) did not consider the requirement of communication security, it is almost impossible to develop such a protocol for current Internet. We focus on **finding solutions to address some of these security concerns in some existing dynamic networks**. These solutions would be helpful in developing a next generation Internet protocol that may cover all these security concerns.

1.2 Motivation

Some existing cryptographic methods may be used to address the above security concerns. For instance, two nodes can establish a shared secret key and use this secret key to protect the integrity of their transmitted messages. Furthermore, a dynamic system can be efficiently protected by using a secret (group) key known only to the participating members. However, these methods are still insufficient or limited in the dynamic network environments due to the latter' special features:

1. Dynamic Population of Participants. As stated above, the set of participants may change unpredictably over time in a dynamic network. It is very difficult to accomplish a task that requires the cooperation of all participants in such a dynamic network environment. For instance, the nodes may use a common secret key to protect their group communication. It is necessary to update this common secret key, i.e., to distribute a new group key to all remaining (authorized) nodes that are also referred to as the *legitimate* nodes, in order to remove compromised nodes from the network. However, some authorized nodes may not be able to get the updated group key, e.g., they were offline in the previous rounds of group key updating. As a result, these nodes are unfortunately isolated from group communication, even though they are still legitimate nodes. Statelessness, i.e., the capability that a legitimate node can get the updated group key does not rely on the success of its previous key update, is a desirable property for group key management in dynamic network environments. Fortunately, some existing group key distribution schemes do provide such a stateless property at a cost of high storage overhead at each node. Considering a node may have limited memory space, e.g., PDAs in a mobile ad hoc network, it is better to design a storage-efficient stateless group key distribution scheme.

2. Dynamic Network Topology. In some dynamic network environments, e.g., mobile ad hoc networks, the network topology may change rapidly and unpredictably over time, as the nodes join/leave the network or move freely. It is almost impossible to get the information about current network topology and distribute it to all nodes in time. As a result, no network topology information would be available in these dynamic network environments. Some existing techniques such as firewalls rely on the network topology and service information to prevent illegitimate users from intruding

the protected networks, i.e., to enforce the network access control. Therefore, it is hard to directly employ these techniques in the above dynamic network environments. A new approach that uses other techniques, e.g., cryptographic methods, to enforce network access control is desirable in dynamic network environments.

3. Lack of Trust Relationships. In some dynamic network environments, the nodes may come from different organizations. They may not be able to establish trust relationships between each other. As a result, a node cannot blindly rely on other nodes to protect its communication. Some existing anonymous communication protocols that are used to protect the privacy of users' communication do require the existence of trusted servers. They use the trusted servers to shuffle users' messages and thus hide the sender and/or the recipient of a transmitted message. These protocols, therefore, cannot be deployed in a dynamic network environment where no trusted server is available. A new anonymous communication protocol that does not rely on any trusted server is desirable.

1.3 Summary of Research Results

In this dissertation, I present the research results on securing communication in dynamic network environments, specifically focusing on mobile ad hoc networks and overlay networks. First, I introduce a secure address auto-configuration scheme for mobile ad hoc networks. A principle premise of network communication is that each node is configured with a unique network identifier (i.e., address). Since it is almost impossible to manually configure all nodes or to employ the Dynamic Host Configuration Protocol (DHCP) [46] (the highly dynamic topology may make the node running DHCP server permanently unreachable by all other nodes), secure address auto-configuration is thus desirable when deploying mobile ad hoc networks. The proposed address auto-configuration scheme binds each address with a public key. It allows a user to unilaterally authenticate itself. Such a self-authentication prevents a malicious user from declaring arbitrary address and greatly thwarts the address spoofing and some other attacks associated with address auto-configuration. The result of this research has been published in [123].

Next, I introduce two storage-efficient stateless group key distribution schemes. These two schemes are developed to efficiently distribute a common group key to a dynamic group, e.g., nodes in an overlay network, with less storage requirement at each user. The nodes thus can use the common group key to protect the secrecy and integrity of their multicast messages from illegitimate users. In the proposed two schemes, we assume each user is uniquely identified by an identifier (ID). We use a *Dual Directional Key Chains (DDKC)* structure, which employs two one-way key chains, to facilitate revocation of a set of users with consecutive IDs. By combining DDKCs with a logical key tree, we propose a stateless group key distribution scheme named *Key-Chain Tree (KCT)*. Given a group of total N users, the KCT scheme only requires $O(\log N)$ keys stored by each user, and it requires a key update message of size $O(R)$ measured by the length of encryption key[1] in order to revoke R users. As the KCT scheme requires $O(N)$ hash operations, we extend it to a *Layered Key-Chain Tree (LKCT)* scheme, which maintains the same storage overhead, slightly increases the communication overhead (still $O(R)$), but reduces the computation overhead in deriving a group key to $O(\sqrt{N})$ hash operations. The result of this research has been published in [120].

Third, I investigate the method using cryptographic techniques to enforce network access control in mobile ad hoc networks. The basic idea of the proposed method is to employ a network-wide access control (secret) key to authenticate all the packets transmitted in the network. Any packet that is not properly authenticated will be immediately dropped by the receiving node. As a result, a non-authorized node is prevented from injecting packets, i.e., accessing the network resources. A critical challenge is how to achieve a globally-synchronized network access control key once it is updated, considering the possibility of network partition and the absence of some legitimate nodes during the procedure of key update. To address this issue, we propose a distributed key synchronization approach that makes use of the stateless group key distribution schemes as presented above. This key synchronization method guarantees that whenever legitimate nodes communicate with each other, they will synchronize their access control keys and agree on the latest one needed to authenticate and verify data packets. We also present a packet retransmission mechanism to deal with received packets that cannot be verified immediately. This retransmission mechanism allows the involved nodes to synchronize their access control keys and establish or continue communication. We have implemented a functioning prototype of the proposed

[1]Some researchers chop a long key update message into a collection of short ones, while each short message only contains the new group key encrypted using a single secret key. They measure the communication overhead using the number of short messages. In such a scenario, our proposed schemes still have the complexity of $O(R)$.

network access control system, and tested it on some wireless communication devices. The measurements demonstrate the proposed system is practical and effective. The result of this research has been published in [121].

Finally, I propose a novel and scalable anonymous communication protocol for overlay networks. This proposed anonymous communication protocol does not rely on the existence of trusted servers. Most of all, unlike the existing anonymous communication protocols that either cannot provide provable anonymity or suffer from transmission collision, it is transmission collision free and provides provable k-anonymity for both the sender and the recipient, where k is a predetermined parameter that can be any number between 1 and $\mathcal{N}$ (the number of participants in the network).

In the proposed anonymous communication protocol, the nodes are organized into a set of logical rings and form an overlay network over the Internet. Within each ring, an anonymous transmission mechanism, which is the cornerstone of the proposed protocol, uses message batching and one-way key chain to make a node's message indistinguishable. It ensures (i) a node can send messages to any other node (in its ring) without disclosing identity to all nodes in the network, and (ii) a node is prevented from maliciously modifying or replaying any transmitted message in a ring. To anonymously communicate with a recipient that may be located in a different ring, a sender first conceals the ID of destination ring, in which the recipient resides, into its outgoing message. It then sends the message to a randomly chosen (agent) node in its local ring, following the above anonymous transmission mechanism. This agent node extracts the message and forwards it to the corresponding destination ring without knowing the sender's identity. The forwarded message is locally broadcasted in the destination ring, i.e., sent to all member nodes. The recipient thus receives the message without disclosing its identity. If each ring has at least k honest nodes, the proposed protocol, therefore, provides provable $k-$anonymity for both the sender and the recipient. Through theoretical analysis, we have shown the proposed protocol is secure even if a polynomial time adversary can eavesdrop all network traffic and control a fraction of nodes. We have also investigated the feasibility and performance of the proposed protocol by testing a proof-of-concept implementation on PlanetLab [4]. The result of this research will be published in [122].

1.4 Outline

The rest part of this dissertation is organized as follows. Chapter 2 describes some cryptographic tools and notation used through the dissertation. Chapter 3 presents a secure address auto-configuration scheme for mobile ad hoc networks. Chapter 4 introduces the two storage-efficient stateless group key distribution schemes to secure group communication in a dynamic set of users. Chapter 5 gives the research result on employing stateless group key distribution to enforce network access control for mobile ad hoc networks. Chapter 6 presents the proposed k-anonymous communication protocol for overlay networks. Finally, Chapter 7 concludes this dissertation and summarizes the accomplishment I have achieved in my Ph.D. study.

Chapter 2

Preliminaries: Cryptographic Tools and Notation

In this chapter, we first introduce some cryptographic tools/conceptes that will be employed by later proposed schemes, and then we show the notation frequently used throughout the dissertation.

2.1 Symmetric Key Encryption and Public Key Encryption

A cryptosystem uses mathematical techniques to transform (or *encrypt*) user's input, called *plaintext*, into an unreadable format, called *ciphertext*. Only those who possess a cryptographic *key* can decipher (or *decrypt*) the ciphertext into plaintext. The mathematical techniques used in the cryptosystem are referred to as the *encryption algorithms*, which are typically divided into two generic types: *symmetric key encryption* and *public key encryption*, based on whether the encryption key is made publicly available.

In symmetric key encryption, two users share a common secret key that is used for both encryption and decryption. The encryption functions must, by definition, be reversible since the two users need to be able to both encrypt and decrypt messages. Symmetric key encryption algorithms can be divided into stream ciphers and block ciphers. Stream ciphers, e.g., RC4 [102] and ISAAC [65], encrypt the bits of the message one at a time. Block ciphers, e.g., Blowfish [107] and AES [37], take a number of bits and encrypt them as a single unit.

Public key encryption is also known as asymmetric key encryption. In public key encryption, a user uses a pair of cryptographic keys, *public key* and *private key.* He publishes his public key which is used for encryption by any other user, and secretly keeps the private key which is used for decryption. It is computationally infeasible for an adversary to derive the private key from the public key. A message encrypted with a user's public key cannot be decrypted by anyone except the user processing the private key. The public key encryption thus ensures confidentiality. However, public key encryption is much slower than symmetric key encryption. The known public key encryption algorithms include El Gamal [47], RSA [97] and DSS [89].

In this dissertation, we frequently use symmetric key encryption and public key encryption together to protect the secrecy and the confidentiality of transmitted data messages, i.e., to ensure no user except the targeted recipient can access the encrypted content.

2.2 Semantic Security

The notion of semantic security was first put forward by Goldwasser and Micali in 1982 [54]. It is a widely-used definition for security in encryption algorithms. A symmetric key cryptosystem is considered semantically secure if no adversary, given an encryption of a message randomly chosen from a two-element message space determined by the adversary, can identify the message choice with probability significantly better than that of random guessing (1/2).

For a public key cryptosystem to be semantically secure, it must be infeasible for a computationally-bounded adversary to derive significant information about a message (i.e., the plaintext) when given only its ciphertext and the corresponding public encryption key. Semantically secure public key encryption algorithms include El Gamal [47] and Paillier [90]. RSA [97] can be made semantically secure (under stronger assumptions) through the use of random encryption padding schemes such as Optimal Asymmetric Encryption Padding [14].

In this dissertation, we require the semantic security for both the symmetric key encryption and the public key encryption, if without specific explanation. We will use the concept of semantic security to prove the proposed anonymous communication protocol in chapter 6 is provably secure.

2.3 One-way Function and One-way Key Chain

One-way function [81] is a function $\mathcal{H}()$ from a set X to a set Y, if $\mathcal{H}(x)$ is easy to compute for all $x \in X$, but for a random element $y \in Y$, it is computationally infeasible to find any $x \in X$ such that $\mathcal{H}(x) = y$. For instance, take $X = \{1, 2, 3, ..., 16\}$ and define $\mathcal{H}(x) = 3^x \; mod \; 17$. Given an input number between 1 and 16, it is relatively easy to find the image of it under $\mathcal{H}()$. However, given a random output number such as 7, without having a table of all inputs and corresponding outputs, it is hard to find x that $\mathcal{H}(x) = 7$.

One-way key chain [75] is a chain of cryptographic keys generated by repeatedly applying a one-way function $\mathcal{H}()$ to a random number (key chain seed). For example, to construct a key chain of size L, the user randomly chooses a key chain seed K_L, and then computes $K_{L-1} = \mathcal{H}(K_L)$, $K_{L-2} = \mathcal{H}(K_{L-1})$, ..., until $K_0 = \mathcal{H}(K_1)$. K_0 generally is referred to as the *commitment* of the key chain. Due to the one-way property of the function $\mathcal{H}()$, given a disclosed k_i, it is computationally infeasible to compute any undisclosed k_j for $j > i$. However, a user can compute any K_j for $j < i$ efficiently, i.e., $K_j = \mathcal{H}^{(i-j)}(K_i)$.

In this dissertation, we use one-way chains to facilitate the revocation of users in the proposed stateless group key distribution schemes in chapter 4. The proposed anonymous communication protocol in chapter 6 also uses the one-way key chains to authenticate the origins and the orders of transmitted messages to prevent the replay/modification attacks. For convenience, the keys in a key chain are also referred to as the *key-chain keys* in this dissertation, in order to distinguish them from other encryption keys.

2.4 Birthday Paradox

In probability theory, the birthday paradox states that given a group of 23 (or more) randomly chosen people, the probability is more than 50% that at least two of them will have the same birthday. It can be generalized as follows: given n random integers drawn from a discrete uniform distribution with range $[1, d]$, what is the probability $p(n; d)$ that at least two numbers are the same? Mathematicians show:

$$p(n,d) = \begin{cases} 1 - \prod_{k=1}^{n-1}(1 - \frac{k}{d}) & n \leq d \\ 1 & n > d \end{cases} \qquad \Rightarrow \qquad n(p,d) \approx \sqrt{2d \ln \frac{1}{1-p}}$$

Birthday paradox is important to some cryptographic functions, e.g., hash function. It shows that if the output space is not large enough, a small number of collisions in a hash table, for the practical purposes, will be inevitable. An attacker thus may exploit it to launch some serious attacks, e.g., spoofing attacks.

In this dissertation, we use the birthday paradox to prove the security of the proposed address auto-configuration scheme in chapter 3, i.e., an attacker has to try a huge number of public keys (e.g., $> 2^{50}$) before being able to successfully spoof another node with a probability greater than 0.5. We also use the birthday paradox to illustrate that, in the proposed anonymous communication protocol for overlay networks in chapter 6, the probability that two nodes accidently use the same key chain to identify their messages can be ignored, given the key size is large enough (e.g., $>$ 160 bits).

2.5 Byzantine Problem and Approach

The Byzantine Problem grew out of attempts to control an unstable system while requiring certain degree of fault tolerance. That is, assuming that there are N generals. Some of them are good and some are faulty. They can communicate only via messages. The bad ones can forge messages, delay messages sent via them, send conflicting or contradictory messages, and masquerade as other generals. If a message from a "good" general is lost or damaged, then the good general is treated as a bad one. Such a Byzantine problem models real-world environments in which computers and networks may behave in unexpected ways due to hardware failure or malicious attacks. An algorithm solves the Byzantine problem if it gets all the good generals to agree within a bounded time.

Analysis and solutions of the Byzantine problem are given by Lamport *et al.* [76]. The gist of the theory on solutions to the problem is:

- If at least 1/3 of the generals are bad, then the good generals cannot reliably reach an agreement [91].

- If fewer than 1/3 of the generals are bad, then there are many algorithms, e.g., [44, 58, 91].

The proposed anonymous communication protocol in chapter 6 will use Byzantine approach to let the nodes in a ring (i) to reach an agreement on some public parameters, and (ii) to elect some agent nodes that will provide message forwarding service. It thus effectively prevents nodes from misbehaving.

2.6 Notation List

Table 2.1 shows some frequently used notation throughout this dissertation.

Table 2.1: Major Notation

PK_A	the public key of node A
SK_A	the corresponding private key to PK_A
$PK_A\{M\}$	the encryption of message M using public key PK_A
K_{A_i}	the i^{th} secret key generated by node A
$K_{A_i}\{M\}$	the encryption of message M using secret key K_{A_i}
$K_{A_i}^{-1}\{M\}$	the decryption of message M using secret key K_{A_i}
M_{A_i}	the i^{th} message sent by node A
MAC_{A_i}	the message authentication code computed using key K_{A_i}
$\mathcal{H}()$	a secure one-way function
$SIG_{PK}\{M\}$	the signature of message M using public key PK
$\|\|$	the concatenation of strings

Chapter 3

Securing Address Auto-Configuration - A Scheme for Mobile Ad Hoc Networks

A principle premise of successful network communication is that each node should be configured with a unique network identifier (address). In dynamic network environments, e.g., mobile ad hoc networks, manual configuration is not always possible and has some drawbacks, e.g., requires the human involvement. Automatic address configuration is therefore desirable in the deployment of dynamic networks.

The TCP/IP (Transmission Control Protocol/Internet Protocol) protocol suite is the foundation of modern Internet, and has been used as the primary platform for the development and implementation of networking applications. Hundreds and thousands TCP/IP applications have been developed over the last few decades. By configuring a mobile ad hoc network as an IP address-based network, we can easily transplant these applications into mobile ad hoc networks. Therefore, it is desirable to use an IP address as the logical identifer of a node and develop an automatic IP address assignment scheme for mobile ad hoc networks.

In traditional wired and fixed networks, address auto-configuration is usually performed by employing the DHCP protocol [46], where a fixed DHCP server assigns each node a unique IP address and authenticates the ownership of IP addresses. However, such a centralized method is not applicable in a peer-to-peer style mobile ad hoc network, since the highly dynamic topology may make the node running DHCP server permanently unreachable by all other nodes, even if the former is always online.

Recently, a number of address auto-configuration protocols for mobile ad hoc networks have been proposed, e.g., [20], [83], [87], [92], and [125]. These schemes focus on the issues related with address assignment and duplicate address detection. All of them lack an effective mechanism to authenticate the ownership of an assigned address. As a result, they are vulnerable to attacks such as IP address spoofing or impersonation. A malicious node may freely choose any configured node as a victim, spoof the IP address, and thus hijack the latter's traffic. Furthermore, a malicious node can even prevent an un-configured node from getting an IP address, e.g., by publishing some false address conflict messages or exhausting the available address space.

In this chapter, we introduce a secure IP address auto-configuration scheme for mobile ad hoc networks in the absence of centralized authentication services. The basic idea of the proposed scheme can be described as follows. A node first randomly selects a (long) public key (i.e., > 512 bits). It then uses the 32-bit (in IPv4) or 128-bit (in IPv6) hash value of this selected public key as its IP address. To prove the ownership of its IP address, the node needs to answer some cryptographic puzzles using its selected public key. A malicious node is therefore prevented from declaring arbitrary IP address, unless it can find a public key whose hash value equals the selected IP address. Our later analysis shows that, based on the birthday paradox, it is calculation-intensive to find such a key given a large address space, e.g., 128 bits. Further efforts are made to make the proposed address auto-configuration scheme more secure in the case of a small address space.

It is worth noting that the proposed address auto-configuration scheme is not limited to mobile ad hoc networks. It (or its variants) can be deployed in other dynamic network environments where address auto-configuration is required but no centralized server provides such auto-configuration and address authentication services.

The rest of this chapter is organized as follows. Section 3.1 summarizes the previous work on address auto-configuration for mobile ad hoc networks. Section 3.2 illustrates some major security threats to these previous schemes. Section 3.3 presents the basic scheme of the proposed secure address auto-configuration solution. Section 3.4 analyzes the security of the proposed scheme. Section 3.5 introduces an extended scheme suitable for a small address allocation space. Finally, section 3.6 summarizes our results.

3.1 Related Work

This section gives a brief review on some existing address auto-configuration schemes for mobile ad hoc networks.

Perkins et. al. [92] proposed a flood-oriented IP address auto-configuration scheme for mobile ad hoc networks. In their proposed scheme, a node first randomly picks an IP address in the specific range (169.254/16) and then broadcasts a request to determine whether this address has already been used. If no reply is received before a timer expires, the node assumes this address is not occupied and then takes for its own. Otherwise, the node reiterates its attempt with a new IP address. Clearly, duplicate addresses may happen in this scheme since it does not consider the scenarios of network partition and merge. The network partition means two subsets of nodes, i.e., subnetworks, cannot establish a communication path due to the failure(s) of communication link(s) and the merge means the rejoining of two partitioned subnetworks.

MANETConf is proposed by Prakash and Nesarigi [87]. It employs a distributed mutual exclusion algorithm to maintain a distributed pool of address. That is each configured node keeps a table of already assigned addresses in the network. When a newly arrived node, called "requester", wants to join the network, it searches for an already configured node, called "initiator", by broadcasting a request. The initiator replies, chooses an unassigned address, and seeks the permissions from all other configured nodes to assign this address. If no negative reply is received, the initiator then assigns the address to the requestor. Meanwhile, if a configured node does not reply at all, the initiator assumes this node has left the network and removes the corresponding address from the allocation table. MANETConf also uses a *universally unique identifier* (UUID) to identify each network partition. If two nodes find they are using different UUID, i.e., a partition merger is detected, they exchange their allocation tables and flood them in their partition. The node which finds its address in the allocation table of the other partition has to change its address.

Mohsin and Prakash [83] proposed a proactive scheme for dynamic allocation of

IP address in mobile ad hoc networks. Their proposed scheme splits the allocation table among all nodes and uses *buddy system* [126] for efficient table mergers. In this scheme, each configured node has a block of addresses (referred to as the allocation table), and it is the only node managing that block of addresses. When one of them (we refer to it as the *initiator*) receives a request from a newly arrived node, it chooses a unique address from its block and assigns this address to the requester. The initiator also splits its block into two and assigns one half to the requester. The initiator and the requester then become each other's *buddy node*. Since a node's crash or abrupt leaving may cause address leak, i.e., part of address space is not available for future allocation, a synchronization procedure is further deployed for the purpose of detecting address leaks. For a released or leaked block, it is possible to merge it with that of its buddy node.

Furthermore, Vaidya [117] proposed a weak duplicate address detection scheme for auto-configuration in mobile ad hoc networks. His scheme assumes each node is pre-assigned a unique key, which can be the media access control address, a large random number, or some other information. The tuple <IP address, Key> thus can be used to uniquely identify a node, even if two nodes happen to have chosen the same IP address. Boleng [20] presented another address auto-configuration scheme with variable-length addresses. Weniger [125] proposed a passive duplicate address detection mechanism based on classic link state routing. None of the above schemes considered the security issues in address auto-configuration. They are vulnerable to some attacks that will be discussed in section 3.2.

To the best of our knowledge, [29] is the first paper attempting to design a secure address auto-configuration scheme for mobile ad hoc networks. It employs the buddy system proposed in [83] to allocate the address, and uses inter-certification to authenticate an allocated IP address. In [29], an un-configured node first randomly chooses a public key. It then obtains one specific address as well as a block of IP addresses like that in [83]. After that, the node asks its initiator and other configured nodes to generate certificates that bind the node's IP address with its chosen public key. Unfortunately, this inter-certification mechanism still cannot effectively prevent address spoofing attack since the other nodes may not be able to verify the public key of the initiator. The attacker also can enforce any configured node to change its address by initiating a false merger. The details of these attacks will be discussed in the next section.

3.2 Security Threats in Previous Schemes

In this section, we will discuss three major security threats associated with address auto-configuration in mobile ad hoc networks. Other threats, e.g., attacks against the routing protocol, are beyond the scope of our discussion.

3.2.1 Address Spoofing Attack

Clearly, due to the lack of an authentication mechanism, previous auto-configuration schemes are vulnerable to address spoofing attacks, in which an attacker may maliciously configure itself with another node's IP address to either impersonate the victim or hijack the latter's traffic. Figure 3.1 shows an example of the address spoofing attack. Let node A with IP address a communicate with node D via nodes B and C. An attacker X that wants to hijack the traffic from node A configures itself with IP address a. When node C receives a neighbor discovery message, e.g., hello message in AODV protocol [1], from the attacker X, it may mistakenly believe that node A becomes its direct neighbor. As a result, node C updates its routing table and redirects the traffic to attacker X.

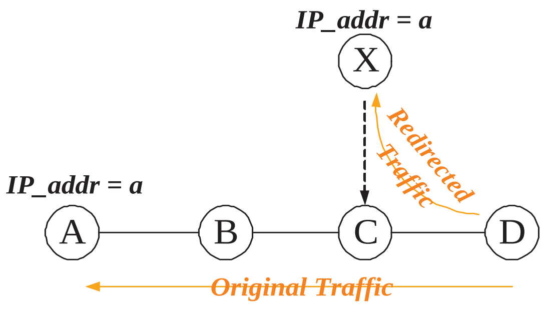


Figure 3.1: An example of address spoofing attack and traffic hijack.

The inter-certification mechanism in [29] is used to address such spoofing attacks. However, since a node may not be able to verify the initiator's public key, an attacker still can launch the spoofing attack. For instance, if node C in figure 3.1 does not know the public key of node A's initiator, node X randomly chooses two public keys, uses one of them to generate a certificate for itself, and declares this certificate was obtained from its initiator. Node C will be fooled as it cannot verify the public key of node A's initiator.

3.2.2 False Address Conflict Attack

In [87] and [92], the assignment of a new address requires an approval of all configured nodes. An attacker can take advantage of this to prevent a newly arrived node from getting an IP address, e.g., by sending negative replies. Since the victim node cannot verify the authenticity of such negative replies, it has to give up the chosen address and try another one. If the victim node continuously receives negative replies, it is prevented from entering the network.

In [29] and [83], an attacker may broadcast some false address conflict messages, i.e., initiating a false merger and publishing a false allocation table which marks the victim node's address as one used by another partition. Since the victim node cannot verify the authenticity of an address conflict, it has to give up its current address and seek for a new one according to the previous auto-configuration schemes. As a result, the victim node's normal communication is interrupted, and a lot of network bandwidth is consumed by the traffic introduced by the unnecessary changing of address.

3.2.3 Address Exhaustion Attack

IP address is one of the most important resources in mobile ad hoc networks. An attacker could maliciously claim as many IP addresses as possible. If all valid IP addresses are occupied/exhausted by the attacker, a newly arrived node will not be able to get an IP address and thus is prevented from entering the network. Figure 3.2 shows an example of such address exhaustion attacks. The attacker X declares multiple identities, introduces a number of phantom nodes $(G_1, ..., G_v)$ into the network, and thus exhausts the available address space. Due to the lack of a centralized authority, it is very difficult to detect these phantom nodes and all previous auto-configuration schemes cannot prevent such an address exhaustion attack. (It is worth noting that if a large address space is deployed, applying and maintaining a large number of IP addresses requires a lot of resources. An attacker may use false address conflict attacks, in stead of the address exhaustion attacks, to prevent a newly arrived node from getting an IP address.)

The buddy system [29, 83] mitigates such an address exhaustion attack by employing a distributed address allocation table. However, an attacker still could exhaust the available address space at a light cost. For instance, suppose the configured node C has an address block with a size of 2^r in figure 3.2. An attacker X contacts with node C for

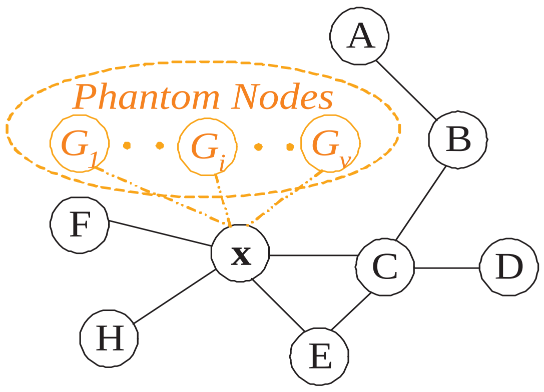


Figure 3.2: An example of address exhaustion attack.

address assignment using a faked ID G_1. As a result, node C loses half of its address block after assigning an address to G_1. After repeating this procedure r times with different faked IDs, the attacker X deprives all address blocks from node C. Similarly, the attack X can deprive other nodes' address blocks and eventually exhaust all available address space in a mobile ad hoc network.

3.3 Proposed Scheme

In this section, we present a secure address auto-configuration scheme for mobile ad hoc networks. This proposed scheme binds a node's IP address with a public key via a one-way hash operation, in the absence of centralized authentication services. The owner of an address thus can use the bound public key to unilaterally authenticate itself. Such an unilateral authentication greatly thwarts the attacks discussed in the previous subsection.

We assume that the targeted mobile ad hoc network is a completely private network. All address space is therefore available for address auto-configuration. In other words, all 32 bits (in IPv4) or 128 bits (in IPv6) can be used for addressing the nodes in the network. Next, we assume the nodes in the network keep a loose time synchronization. Each node embeds a time-stamp into its outgoing message. As a result, any replayed message, i.e., a replay attack, will be detected by the receiver.

In the propose scheme, to get an IP address automatically, an un-configured node, suppose node A, first randomly generates a public/private key pair (PK_A / SK_A) and a secret key (K_A), and immediately saves these keys to a secure local storage. Next, node A

calculates a 32-bit (in IPv4) or a 128-bit (in IPv6) hash value of PK_A, i.e. $a = \mathcal{H}(PK_A)$. Node A then temporarily uses the resulting value, a, as its IP address, initiates a timer, and broadcasts a *Duplicate Address Probe* (DAP) message illustrated as follows, where t_A is a time-stamp used to prevent potential replay attacks.

$$DAP = \{a, 0, PK_A, t_A, K_A\{t_A\}, SIG_{PK_A}\{0||t_A||K_A\{t_A\}\}\}$$

If a configured node, suppose node B, finds the IP address in a received DAP message is the same as its own, it will verify the authenticity of this DAP message. Node B first tests whether $a = \mathcal{H}(PK_A)$ holds. If it does, node B then verifies the signature using the received PK_A. If the signature proves correct, node B further verifies whether $PK_B = PK_A$ holds and whether $K_B^{-1}\{K_A\{t_A\}\} = t_A$ holds to prevent the replay attacks. If at least one of these two equalities does not hold, node B then broadcasts an *Address Conflict Notice* (ACN) message shown as follows, to inform the corresponding node A of an address conflict, where t_B is a time stamp generated by node B. Otherwise, node B simply discards the received DAP message, since its IP address does not conflict with a.

$$ACN = \{a, 1, PK_B, t_B, K_B\{t_B\}, SIG_{PK_B}\{1||t_B||K_B\{t_B\}\}\}$$

If no ACN message is received before the timer expires, node A assumes that the IP address a has not been used and begins using this address. If node A receives an ACN message from another node, e.g., node B, it first verifies this ACN message before trying another IP address. Node A checks the embedded time stamp t_B to see whether the received ACN is expired. If not, node A further checks whether $a = \mathcal{H}\{PK_B\}$ holds and whether the signature is correct based on the embedded PK_B. If all the verification results are positive, node A is sure that IP address a is in use by another node. Node A then has to repeat the auto-configuration procedure with another pair of public/private keys. Otherwise, node A simply discards the received ACN message.

It is possible that two configured nodes happen to have the same IP address after a merger of partitions. In this case, when the address conflict is detected, e.g., by employing the passive detection method proposed in [125], one of these two nodes (e.g., node A) broadcasts an ACN message. Once receiving the ACN message, the other node (e.g., node

B) first verifies the authenticity of this ACN as was discussed earlier. If the verification result is positive, node B then checks whether it is the source of this ACN message to prevent replay attacks. That is node B checks whether the embedded public key PK_A is same as its own, decrypts the cypher-text $K_A\{t_A\}$ with its secret key K_B, and checks whether the resulting value matches with the embedded time stamp t_A. If all results are positive, node B treats this ACN message as a replayed one and simply discards it. Otherwise, node B gives up its current IP address and starts a procedure of address auto-configuration.

Two nodes that have conflicted address may receive an ACN message from each other. To prevent them from simultaneously changing their IP addresses, only the node, whose received time-stamp in the DAP message is smaller than the one it sent out, should change its IP address.

3.4 Analysis

The proposed scheme provides a kind of unilateral authentication for the ownership of an IP address. It forces an attacker to find a public key whose hash value equals the victim's IP address before being able to launch an attack (e.g., spoofing attack). To find such a key requires a number of computations or the storage of key/address pairs. In this section, we analyze the likelihood of an address collision and the likely computation/storage cost if an attacker wants to launch address spoofing attacks or address exhaustion attacks, given each node randomly chooses its public/private key pair.

$$Pr_1 = 1 - \frac{\mathcal{C}!}{\mathcal{C}^n(\mathcal{C}-n)!} \tag{3.1}$$

If we assume the outputs of the one-way hash function $\mathcal{H}$ are evenly distributed in the address space $\mathcal{C}$, the address collision problem (i.e., two different public keys having the same hash value) is analogous to the birthday paradox presented in chapter 2. The probability of such address collision is expressed by equation 3.1, where n is the number of nodes who randomly choose a pair of public/private keys. Clearly, the probability of address collision increases as the number of nodes in the network increases, given a fixed address space $\mathcal{C}$. Figure 3.3 plots the probability of address collision given a 32-bit address space

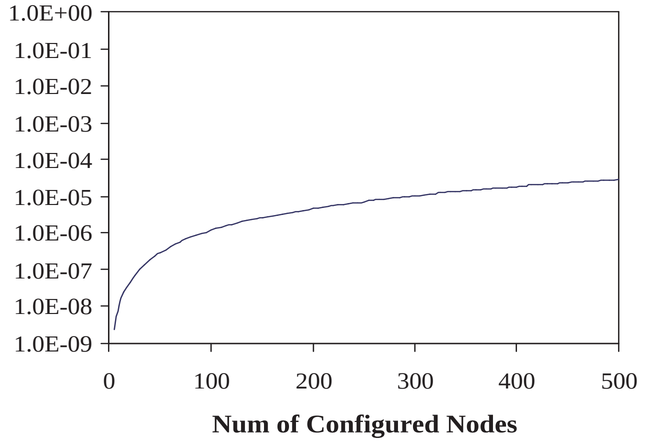


Figure 3.3: Probability of address collision given a number of configured nodes (for IPv4).

(i.e., IPv4) and variant number of configured nodes, and it proves the above conclusion.

Instead of focusing on a particular victim, an attacker may want to target at a set of configured nodes, i.e., to see if it can spoof one of them. Equation 3.2 shows the probability of getting a public key whose hash value equals one of the n given IP addresses after having tested m different inputs. Based on this equation, we can derive the minimum number (i.e., a threshold) of public keys tested before an attacker can successfully spoof one of the n configured nodes with a probability greater than p. Equation 3.3 shows how to calculate such a threshold. Clearly, this threshold also depends on the size of address space and the number of configured nodes. Figure 3.4 plots the minimum number of public keys that an attacker has to test before being able to get a key to spoof one of a given number of configured nodes in IPv4, with a probability greater than 0.5. It shows that as the number of configured nodes increases, the threshold of minimum number of tested public keys decreases.

$$Pr_2 = 1 - (\frac{\mathcal{C} - n}{\mathcal{C}})^m \tag{3.2}$$

$$Pr_2 > p \quad \Rightarrow \quad m > \frac{\ln(1-p)}{\ln(\frac{\mathcal{C}-n}{\mathcal{C}})} \tag{3.3}$$

An attacker might do the calculation offline, buffer the resulting key/IP address pairs, and use them to break the proposed scheme, instead of computing them online. Equation 3.4 shows the probability of at least one IP address collision, given the attacker

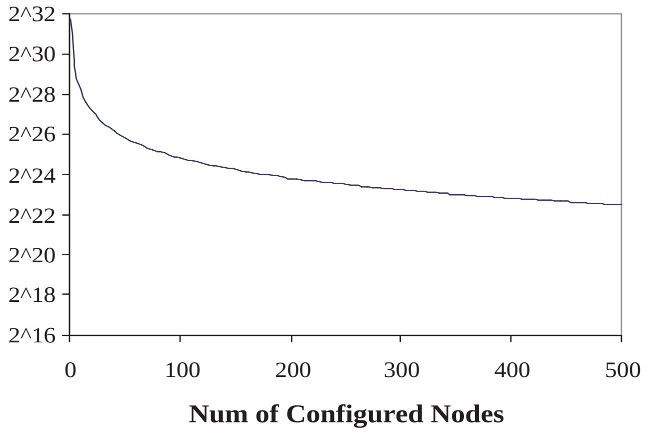


Figure 3.4: Minimum number of public keys tested to successfully spoof a node with a probability *p>0.5*, given n configured nodes (for IPv4).

buffers w key/IP address pairs and there are m configured nodes. We can see that the more key/IP address pairs buffered, the higher this probability is. This probability also increases as the number of configured nodes grows, given a fixed number of key/IP address pairs buffered by the attacker. Figure 3.5 plots the collision probability in the case of 2^{20} key/IP address pairs are buffered, given various number of configured nodes. It shows that this collision probability is quite low, even if an attacker spends a lot of storage resources (132MB memory in this example, assuming the public key is 1024 bits long) on buffering the key/IP address pairs.

$$Pr_3 = 1 - \frac{(\mathcal{C} - w)!}{\mathcal{C}^m(\mathcal{C} - w - m)!} \tag{3.4}$$

3.5 Extension for IPV4

The security performance of the proposed scheme (or the cost for an attacker to break this scheme) relies on the size of address space, $\mathcal{C}$. In IPv4, an attacker may relatively easily find a public key whose hash value equals a given 32-bit IP address. Figure 3.4 shows that after trying 2^{31} public keys, an attacker can get one whose hash value equals a given IP address with a probability of 0.5. Considering the high speed of modern mobile computing devices, such an amount of computation is not a big challenge.

Inspired by [117], which uses some extensional information combined with the IP

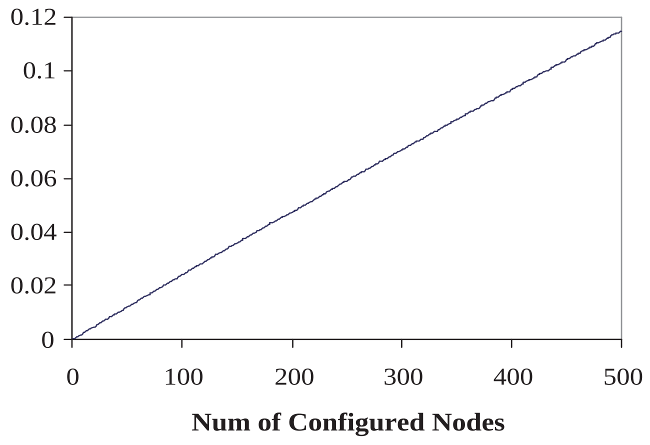


Figure 3.5: Probability of at least one address conflict with some configured nodes, given an attacker buffers 2^{20} public key/address pairs (for IPv4).

address to uniquely identify a node and to prevent the packets from being routed to a wrong destination in the case of duplicate IP addresses, we propose an extension of the proposed scheme for the scenario of a small address allocation space, i.e., IPv4.

A straight-forward solution is that a node first randomly chooses its public/private key pair (PK/SK) and a secret key K, and auto-configures itself as that in the basic scheme. It next directly uses the public key PK as the *key* in [117]. In the procedure of route discovery or neighbor discovery, the source node, i.e., the node that initiates the route discovery, embeds its public key PK into the outgoing packets, e.g., the $RREQ$ messages or the $HELLO$ messages. After receiving these packets, the intermediate nodes verify whether the source node's IP address equals the hash value of the embedded public key PK. If it does, they record the source node's IP address and the corresponding public key and then forward the packets. Otherwise, they drop the packets. As a result, a node now is uniquely identified by a tuple, <IP address, public key>. In the case that two nodes happen to have the same IP address but different public key, the intermediate nodes still can tell from them based on their public keys, and the packets still can be routed to the correct destination.

However, directly using a node's public key as the extensional information is not a good solution. This is because each intermediate node has to buffer the embedded public key in a received routing request. An attacker may exploit such a buffering to launch a resource consumption attack, e.g., by sending a large number of route requests. To thwart such a resource consumption attack, we suggest that a node calculates a 32-bit *padding* value, where *padding* = $\mathcal{H}$(IP address || PK), and uses this padding as the extensional information. A node is thus uniquely identified by a tuple <IP address, padding>,

Since a node is now identified by a virtual 64-bit address or a larger one if the public key is used as the extensional information, the likelihood of address collision is much closer to zero according to the previous analysis (far less than 10^{-13} for 500 nodes with a 32-bit padding). As a result, the computation or storage cost for launching an address spoofing attack or an address exhaustion attack greatly increases. For instance, if node X in figure 3.1 wants to successfully spoof node A and redirect the traffic with a probability of 0.5, it has to try over 2^{63} public keys. Otherwise, node C can tell node X from node A even if they have the same IP address. Such computation cost is large enough even for a PC with Pentium 4 2.1GHz processor (for the 1024-bit public key, it requires over 1000 days computation based on the data from [38]). The extended scheme thus enhances the security of address auto-configuration in the scenario of a small address allocation space, e.g. IPv4, at a cost of a little increased communication and storage overhead.

3.6 Summary

This chapter has discussed several security threats in implementing address auto-configuration for mobile ad hoc networks, and proposed a scheme to thwart these threats. In the proposed scheme, a node first randomly selects a public key and then uses the hash value of this public key as its IP address. The owner of an IP address thus can use the corresponding public key to authenticate itself. Such an unilateral authentication effectively thwarts the attacks against address auto-configuration in mobile ad hoc networks, as it significantly increases the computational and/or storage cost to launch these attacks.

Chapter 4

Securing Group Communication - Two Stateless Group Key Distribution Schemes

In a communication system, a member may want to send messages to all other members (i.e., one-to-many model) or a set of members may want to talk with another set of members (i.e., many-to-many model). Such communication patterns are referred to as group communication. Group communication has a diverse range of applications: video conference, distance learning, distributed databases, data replication, multi-party games and distributed simulation, network broadcast services and many others. Regardless of the application environment, security services are necessary in order to protect the privacy and integrity of group communication.

A straightforward method of securing group communication is to use a secret (group) key known only to the participating users. However, in dynamic network environments, this group key has to be updated, whenever a new member asks for joining the network/group, if authentication succeeds and access control checks the validity of the request. A new group key needs to be securely distributed to all the members of the new group (the group key is thus also referred to as the *session* key, where a session is defined by the event of key updating). Such a group key update is necessary because otherwise the new user may be able to decrypt the packets transmitted before he joined the group (for that he would have to record the encrypted packets and decrypt them after joining). Similarly, the group key also has to be changed whenever a member leaves the group . Otherwise the leaving member could still have access to the group exchanged information. A group key management mechanism is therefore required in order to secure the group communication, and a secure group key management scheme (or the secure group communication) must satisfy:

1. Only authenticated members are allowed to join
2. Integrity and confidentiality of group-messages is guaranteed
3. Current members cannot obtain keys used in the past (backward confidentiality)
4. Past members cannot obtain keys used in the future (forward confidentiality)

Group key management schemes can be classified into two categories:

Group Key Agreement All members participate and contribute some information to generate the group key. Group key agreement schemes are based on Diffie-Hellman key agreement algorithm [42]. In Diffie-Hellman key agreement algorithm, two parties, named Alice and Bob, choose and agree on a large prime number p and a prime root module p, i.e., a prime number g. Alice chooses a large random number a as her private key and Bob similarly chooses a large number b. Alice then computes $A = g^a \pmod{p}$, which she sends to Bob. Bob computes $B = g^b \pmod{p}$, which he sends to Alice. Now both Alice and Bob can compute their shared secret key $K = B^a \pmod{p} = A^b \pmod{p} = g^{ab} \pmod{p}$.

Group Key Distribution A single entity, i.e., a group key manager, is responsible for managing the group keys and the users' accessibility to the group communication. This group key manager shares one (or a set of) personal secret key(s) with each group member. To revoke compromised users from the group or to update the group key, the group key manager uses a set of personal secret keys only known to the valid members to encrypt the new group key individually. The latter can decrypt one of the encrypted values to obtain the new group key. Group key distribution, therefore, is also referred to as *group key revocation*. These two notions are exchangeable in this dissertation. Group key distribution schemes can be classified into either *stateful* ones or *stateless* ones, based on the interdependency of key update messages.

Statelessness means the key update messages are independent from each other. A legitimate user can get the updated group key, as long as the user has the corresponding key update message, even if he/she missed all previous rounds of key update.

Definition 4.1 *Given a group key distribution scheme $\mathcal{A}$, for any two consecutive key update messages (i.e., the old key update message M_1 and a new one M_2) transmitted by the group key manager, if any authorized user can decrypt the new group key K from M_2 without having M_1, the key update messages are said to be* independent *from each other and $\mathcal{A}$ is a stateless key distribution scheme.*

On the contrary, in a stateful group key distribution scheme (e.g., LKH [119, 127]), a legitimate user's state (fail/success) in current round of group key update will affect its ability to decrypt future group keys. That is, a legitimate user who misses current round of key update may not be able to participate in future group communication, even if it is not revoked by the group key manager. The user has to contact the group key manager to regain the access to group communication. This may impose heavy communication burden on the latter.

Considering the dynamics of user memberships and the unstableness of communication channels in dynamic network environments, it is better to design a stateless group key distribution scheme.

Stateless group key distribution schemes provide the stateless flexibility by having users store a number of auxiliary keys [73], and the number of auxiliary keys increases as the group size grows. Several recent schemes have managed to reduce the storage requirements for users by taking advantage of techniques such as pseudo random number generators [59, 86]. Though these advances make stateless group key revocation schemes practical in most typical applications, there are still some applications in which it is either necessary or desirable to further reduce the storage requirements. For example, when a tamper-resistant smart card is used (e.g., in a satellite TV system) to decrypt the group key, the more tamper-resistant memory required to store the keying materials, the higher each smart card will cost. Given a typical smart card with 1K bytes tamper-resistant memory and 128-bit keys, the SD scheme [86] can only support a group with about 1000 users.

This chapter presents two storage-efficient stateless group key distribution schemes. These two schemes are based on a *Dual Directional Key Chains (DDKC)* structure, which assumes each user is uniquely identified by an ID and employs two one-way key chains to facilitate revocation of a set of users with consecutive IDs. By combining DDKCs with a logical key tree, a storage-efficient stateless group key distribution scheme named *Key-Chain Tree (KCT)* is proposed. Given a group of total N users, the KCT scheme only requires $O(\log N)$ auxiliary keys stored by each user, and requires $O(R)$ keys in a key update message in order to revoke R users. However, the KCT scheme requires $O(N)$ number of hash operations to decrypt a new group key. To reduce the computation overhead, an extension named the *Layered Key-Chain Tree (LKCT)* scheme is introduced. LKCT maintains the same storage overhead, slightly increases the communication overhead (still $O(R)$ keys), but reduces the computation overhead in deriving a group key to $O(\sqrt{N})$ number of hash operations. Compared with the previous stateless group key distribution schemes such as SD [86] and LSD [59], these two proposed schemes significantly reduce the storage requirements with slightly more computation and communication overheads. For instance, considering a group with 2^{20} users and 128-bit keys, each user needs to store 211 keys (3,376 bytes) in the SD scheme, 90 keys (1,140 bytes) in the LSD scheme, but only 40 keys (640 bytes) in the proposed schemes. The proposed schemes thereby provide another trade-off between communication, computation, and storage.

The rest of this chapter is organized as follows. Section 4.1 discusses the related work. Section 4.2 presents the DDKC structure and the two proposed group key distribution schemes. Section 4.3 analyzes the performance of the proposed schemes and compares them with some existing schemes. Finally, section 4.4 summarizes the results.

4.1 Related Work

Securing group communication, which is also referred to as the *broadcast encryption* problem, has received attentions from both the network and cryptography communities. A number of approaches [22, 30, 48, 59, 60, 74, 82, 86, 110, 105, 119, 127] have been proposed. The surveys on group key management are available in [10, 45, 72]. This section gives a brief outline of the existing group key management schemes.

Fiat and Naor [48] first formally studied the broadcast encryption problem. With $O(tn^2\log(t))$ user stored keys and $O(t^2n\log^2(t))$ messages, their proposed schemes allow a group key manager to revoke any number of users where at most t of them collude. Blundo *et al.* [17, 19] and Stinson *et al.* [112] studied broadcast encryption in the unconditionally secure model. They gave the lower and the upper bounds on the communication cost and a user's storage overhead. Luby and Staddon [79] showed the tradeoff between the storage overhead and the communication cost.

Group Diffie-Hellman key exchange [111] is a group key agreement protocol. It is a nature extension of Diffie-Hellman key agreement algorithm to support n-party group operation. In this protocol, the group members agree on a pair of primes (p and q) and starts calculating the intermediate values and the final group key in a distributed manner. The first member calculates the first value (q^{x_1}) and passes it to the next member. Each subsequent member receives the set of intermediary values and raises them using its own secret number generating a new set [98]. For instance, the fourth member receives the set $\{q^{x_2x_3}, q^{x_1x_3}, q^{x_1x_2}, q^{x_1x_2x_3}\}$, it then generates $\{q^{x_2x_3x_4}, q^{x_1x_3x_4}, q^{x_1x_2x_4}, q^{x_1x_2x_4}, x^{x_1x_2x_3x_4}\}$. The last member ($n$) calculates the group key K ($K = q^{x_1 \cdots x_n} \bmod p$). It raises all intermediate values to its secret value and multicasts the whole set. Each group member extracts its respective intermediate value and calculates K. The setup time is, therefore, linear (in term of n) since all member must contribute to generating the group key.

Wallner *et al.* [119] and Wong *et al.* [127] independently discovered the Logical Key Hierarchy (LKH) scheme (also called the Key Graph scheme). In this scheme, the group key manager maintains a hierarchical key tree as illustrated by figure 4.1. The nodes of the tree hold the secret keys which are referred to as the *key-encryption-keys*, as these keys are used to encrypt the group key. The leaves of the tree correspond to the group members. Each member receives a copy of key-encryption-keys corresponding to each node in the path from its leaf to the root. The key held by the root of the tree is the group key. Figure 4.2 illustrates the necessary encryptions when a node joins in the group. The similar encryptions are required when a member leaves (or to revoke a member from the group). LKH is an efficient stateful group key distribution method. It requires each user to store $\log(n)$ keys and the group key manager to broadcast $2\log(n)$ messages (or a message with $2\log(n)$ keys) for a key update operation, where n is the number of legitimate users.

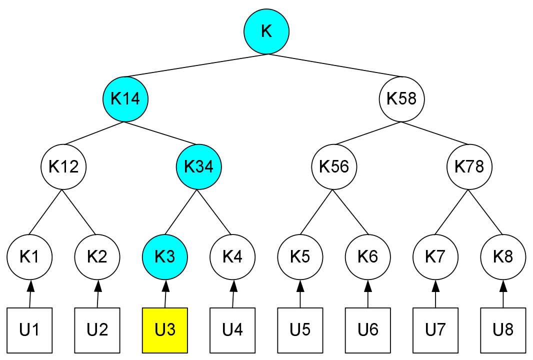


Figure 4.1: An Illustration of the Key Tree in LKH

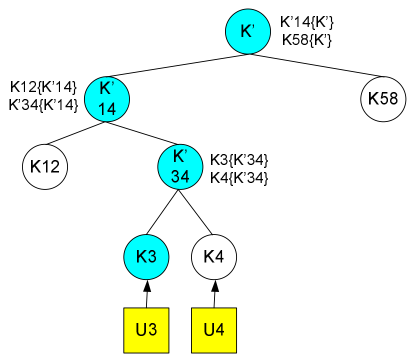


Figure 4.2: Necessary encryption when a member (u_3) joins in the LKH scheme

Naor *et al.* [86] first proposed two stateless group key distribution schemes, termed Complete Subset (CS) method and Subset Difference (SD) method. These two schemes organize users with a tree structure, i.e., users are assigned to the leaves of a full binary tree. In CS scheme, each internal node in the tree corresponds to a secret key, called *subset cover key*, only known to the corresponding *subset*, i.e, the leaves (users) in the subtree rooted at this internal node. In the SD scheme, a subset is redefined as follows. Two nodes in the key tree, i and j (j is the descendant of i) define a subset $\mathcal{S}_{i,j}$, i.e., $\mathcal{S}_{i,j} = \{\text{leaf } u \mid u \text{ is the descendant of } i \text{ but not of } j\}$. Each subset is assigned a secret key, which is called as a subset cover key and can be derived only by the members of the subset. In both schemes, the group key manager uses a collection of subset cover keys, where the remaining legitimate nodes know at least one of them, to encrypt the new group key individually. Figure 4.3 illustrates the subset cover mechanism in SD scheme. To revoke users u_1 and u_4 in figure 4.3 (a), group key manager divides the remaining users into subsets $\mathcal{S}_{3,7} = \{u_2\}$ and $\mathcal{S}_{2,12} = \{u_5, u_7, u_8\}$, and uses the corresponding subset cover keys to encrypt the new group key individually. Figure 4.3 also illustrates the stateless property. If user u_5 did not receive the key update message in figure 4.3(a), he can still decrypt the next key update message in figure 4.3(b) when user u_6 is further revoked from the group. Given the maximum total number of users N and each user has $\log(N)$ keys, the CS scheme can revoke any R users with $O(R\log(N/R))$ messages. The SD scheme reduces the message number to $O(R)$ while it increases the user storage overhead to $O(\log^2(N))$ and requires $O(\log(N))$ cryptographic operations. Compared with CS method, the proposed schemes in this chapter have a less communication overhead but a slightly higher storage overhead. Compared with

SD method, the proposed schemes significantly reduce the storage overhead and maintain the same communication overhead.

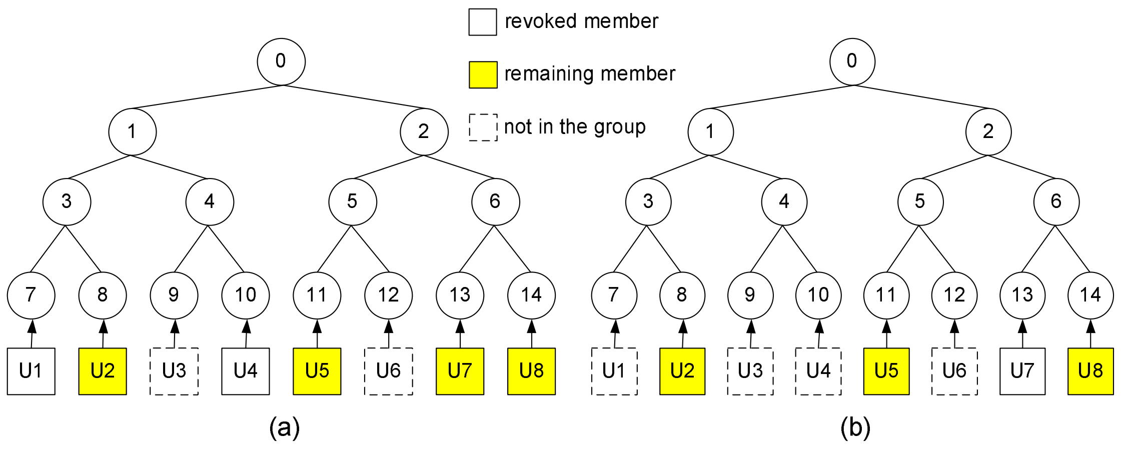


Figure 4.3: The Illustration of Subset Difference Method

Halevy and Shamir [59] proposed a variant scheme of SD, i.e., Layered Subset Difference (LSD). LSD reduces the storage overhead from $O(\log^2(N))$ to $O(\log^{1+\epsilon}(N))$ with increased communication overhead. Their experiments show that the average communication overhead is close to $2R$, which is 1.6 times larger than that in SD. The proposed schemes in this chapter have the similar communication overhead but a less storage requirement, if compared with LSD method.

Chen and Dondeti [33] evaluated the performance of stateful and stateless group key distribution schemes, i.e., they analytically compared the storage cost and the communication cost of LKH scheme and SD method in immediate and batch key update scenarios. Their conclusion is that LKH performs better in immediate key update and small batch key update, whereas SD performs better in the case of larger batch key update.

In addition to the property of statelessness, some recent works address the *self-healing* property that a group member could recover the missed session keys from the latest key update message on its own. Staddon *et al.* [110] first proposed a self-healing group key distribution approach, which is based on two dimensional t-degree polynomials. Their scheme is improved by Liu and Ning [77]. Blundo *et al.* [18] further presented a new mechanism for implementing the self-healing approach.

Canetti *et al.* [26] discussed the tradeoff between communication overhead and the storage overhead of rekeying operations in a multicast group. Two useful conclusions are:

(i) let $b(n)+1$ be the maximal number of the keys held by any user in the multicast group, the rekeying communication costs, $c(n)$, satisfy $c(n) \geq n^{1/b(n)} - 1$, and (ii) for structure preserving protocols, the rekeying communication costs satisfy $c(n) \geq bn^{1/b} - b$, where $b+1$ denotes the maximal number of keys held by any user in the multicast group.

4.2 The Proposed Schemes

Let $\mathcal{N}$ be the set of all potential users, where $|\mathcal{N}| = N$, and let $\mathcal{R}$ be the set of revoked users, where $\mathcal{R} \subset \mathcal{N}$ and $|\mathcal{R}| = R$. The goal of group key distribution is to have the group key manager transmit a key update message efficiently over a broadcast channel shared by all users so that any user $u \in \mathcal{N} \setminus \mathcal{R}$ can decrypt this message, but any users in $\mathcal{R}$ cannot decrypt this message properly, even if they collude with each other in an arbitrary manner.

This section introduces two group key distribution schemes that are both stateless and storage-efficient. These two schemes allow a legitimate user to derive the new group key from the received key update message even if he/she has missed several rounds of key updates. We start from a Dual Directional Key Chains (DDKC) scheme, which itself is not a complete group key distribution scheme, but the foundation of the proposed schemes, and postpone the details of the proposed schemes to the later subsections.

4.2.1 Dual Directional Key Chains

A DDKC is composed of two one-way key chains with equal length, a *forward key chain* (K^F) and a *backward key chain* (K^B). The keys in the forward key chain are referred to as the *forward keys*, and $K^F_{i,j}$ is used to denote the forward key assigned to u_j (user j) in subset $\mathcal{N}_i$. Similarly, the keys in the backward key chain are called the *backward keys*, and $K^B_{i,j}$ is used to denote the backward key assigned to u_j in subset $\mathcal{N}_i$.

A DDKC may be used to facilitate the revocation of a set of users that have consecutive IDs. Specifically, the group manager constructs a DDKC and assign a key in each key chain to each user. For example, figure 4.4 shows a DDKC with 8 keys in each key chain and 8 users. Each user is assigned the key bellow it in both key chains, e.g., assign K^F_5 and K^B_4 to u_5. In general, each user u_i gets K^F_i in the forward key chain and K^B_{N+1-i} in the backward key chain. Obviously, u_i can compute all the keys K^F_j for $j > i$ in the

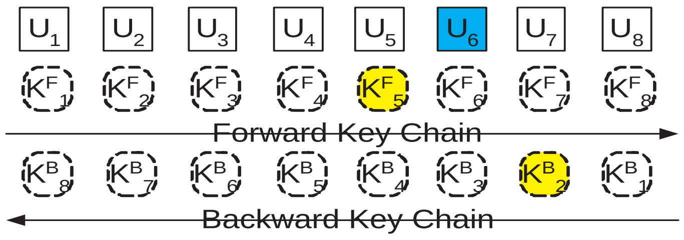


Figure 4.4: An example of dual directional key chains

forward key chain and K_j^B for $j > N+1-i$ in the backward key chain. This property can be used to facilitate revoking a user. For example, if u_6 in figure 4.4 denotes a revoked user, the non-revoked users can be divided into two subsets $\mathcal{T}_1 = \{u_1, ..., u_5\}$ and $\mathcal{T}_2 = \{u_7, u_8\}$. K_5^F is only known by the users in subset $\mathcal{T}_1$ and K_2^B is only known by users in subset $\mathcal{T}_2$. In general, if an encryption key K is only known by the users in a subset T_i, it is said that subset T_i is *covered* by K and K is the *subset cover key* for T_i. Therefore, K_5^F *covers* subset $\mathcal{T}_1$ and K_2^B *covers* subset $\mathcal{T}_2$. The group key manager may use these two subset cover keys, K_5^F and K_2^B, to encrypt the new group key separately and broadcast the cipher-texts to the group. All users except for u_6 will be able to derive at least one of K_5^F and K_2^B and then decrypt the new group key.

This observation leads to the following DDKC revocation scheme, which is only intended to revoke one or a set of users with consecutive IDs. The schemes that can revoke arbitrary sets of users will be discussed in later subsections. Clearly, this DDKC scheme is a subset-cover algorithm [86], i.e., the valid group members are partitioned into a collection of (disjoint) subsets and the new group key is encrypted with the subset cover keys one-by-one.

System Setup Construct a DDKC as follows:

1: For a given potential group size N and a set of users $\mathcal{T}=\{u_1, ..., u_n\}$, $n < N$, the group key manager randomly chooses two initial key seeds, *the forward key seed* S^F and *the backward key seed* S^B.

2: The group key manager repeatedly applies a one-way hash function $\mathcal{H}$ to these initial key seeds, respectively, and gets two *one-way key chains* of equal length N, *the forward key chain* $K_i^F = \mathcal{H}[K_{i-1}^F] = \mathcal{H}^{i-1}[S^F]$ and *the backward key chain* $K_i^B = \mathcal{H}[K_{i-1}^B] = \mathcal{H}^{i-1}[S^B]$.

3: The group key manager assigns the secret keys, K_i^F and K_{N+1-i}^B to user u_i via a secure channel.

User Revocation In order to revoke a set of consecutive users $\{u_i, u_{i+1}, ..., u_j\} \subset \mathcal{T}$, the group key manager first locates two encryption keys, K^F_{i-1} and K^B_{N-j}, if they exist. The group key manager then uses these two encryption keys to encrypt the new group key separately and broadcasts both cipher-texts in the key update message. Thus, only the remaining (non-revoked) users can get the new group key by calculating one of the encryption keys (either K^F_{i-1} or K^B_{N-j}) and then decrypting the new group key.

Analysis

Recall the definition of a secure revocation scheme in the work of Naor *et al.* [86]. Consider an adversary $\mathcal{B}$ that gets to (i) select adaptively a set $\mathcal{R}$ of receivers and obtain I_u (i.e., the secret information that u receives) for all $u \in \mathcal{R}$. By adaptively means that $\mathcal{B}$ may select messages M_1, M_2 ... and revocation set $\mathcal{R}_1$, $\mathcal{R}_2$, ... (the revocation sets need not correspond to the actual corrupted users) and see the encryption of M_i when the revoked set is $\mathcal{R}_i$. Also $\mathcal{B}$ can create a ciphertext and see how any (non-corrupted) user decrypts it. It then asks to corrupt a receiver u and obtains I_u. This step is repeated $|\mathcal{R}|$ times (for any $u \in \mathcal{R}$). And (ii) choose a message M as the challenge plaintext and a set R of revoked users that must include all the ones it corrupted (but may contain more). $\mathcal{B}$ then receives an encrypted message M' with a revoked set $\mathcal{R}$. It has to guess whether $M' = M$ or $M' = R_M$ where R_M is a random message of similar length. A revocation scheme is secure if, for any (probabilistic polynomial time) adversary $\mathcal{B}$ as above, the probability that $\mathcal{B}$ distinguishes between the two cases is negligible.

Naor *et al.* proved that any subset-cover algorithm is secure if it satisfies the key-indistinguishability property, i.e., the encryption keys which are used to encrypt the new group key are indistinguishable for the non-legitimate users. (please refer to Theorem 10 in [86] for details). Since the DDKC scheme is a subset-cover method, to prove it is secure, we only need to show that DDKC scheme satisfies the key indistinguishability.

Key Indistinguishability was defined in [86]. Let $\mathcal{A}$ be a Subset-Cover revocation algorithm that defines a collection of subsets $S_1, ..., S_w$. Consider a feasible adversary $\mathcal{B}$ that (i) selects i, $1 \le i \le w$, and (ii) receives the I_us for all $u \in \mathcal{N} \setminus S_i$. $\mathcal{A}$ satisfies the key-indistinguishability property *if* the probability that $\mathcal{B}$ distinguishes L_i (the real case) from a random key R_{L_i} of similar length (the random case) is negligible, i.e. $|Pr[\mathcal{B}\ outputs\ 'real'\ |L_i] - Pr[\mathcal{B}\ outputs\ 'real'\ |R_{L_i}]| \le \varepsilon$, where ε denotes a threshold that is negligible to zero.

Theorem 4.1 *The encryption (subset cover) keys produced by the DDKC scheme are key-indistinguishable for the non-legitimate users, if $\mathcal{H}$ is a perfect cryptographic hash function.*

Proof The random oracle methodology [25, 27] shows that given a random oracle and a query x, if the oracle has not been given the query x before it generates a random response which has uniform probability of being chosen from anywhere in the output domain. If the one-way hash function $\mathcal{H}$ is a perfect cryptographic hash function, without knowing the initial key seed S^F or S^B, it is computationally infeasible for an attacker to figure out the encryption keys, K^F_{i-1} and K^B_{N-i}. Even given the personal secrets, K^F_i and K^B_{N-i+1}, it is also computationally infeasible for a revoked user $u_i \in \mathcal{R}$ to calculate the encryption keys K^F_{i-1} and K^B_{N-i}. Thus, the encryption keys in the DDKC scheme are indistinguishable from random keys for any user $u_i \notin \mathcal{N} \setminus \mathcal{R}$. $\square$

The properties related to the performance of the DDKC scheme is summarized as follows:

Theorem 4.2 *In the DDKC scheme, (i) the user storage overhead is 2 keys; (ii) the length of the broadcast key update message is at most 2 keys; and (iii) the average computation overhead is less than $(N - R - 1)/2$.*

Proof (i) Clearly, the storage overhead for each user is the initially assigned two personal keys. (ii) According to the scheme, the group key manager needs at most two encryption keys to exclude a single user or a continuous block of users. The new group key is encrypted using these two encryption keys separately. The length of a key update message is therefor at most 2 keys (measures by the length of the key). (iii) A user's computation cost depends on both its and the revoked users' positions. Suppose the new group key is encrypted using K^F_{i-1} and K^B_{N-j} separately. A legitimate user u_p $(p < i)$ needs $i - p - 1$ hash operations to get K^F_{i-1}. Similarly, u_q $(q > j)$ needs $q - j - 1$ hash operations to get K^B_{N-j}. The average computation overhead $\theta = \frac{\sum_{p=1}^{i-1}(i-p-1)+\sum_{q=j+1}^{N}(q-j-1)}{N-j+i-1}$ where $R = j - i + 1$. When $i = 1$ or $j = N$, θ reaches its maximum value $\frac{N-R-1}{2}$. $\square$

4.2.2 Key-Chain Tree Revocation Scheme

Generally, the group key manager needs to revoke multiple users which might not be adjacent to each other. The proposed DDKC method unfortunately cannot handle such situations. Inspired by the logical key trees in the SD scheme [86], we propose to use a tree structure along with (multiple) DDKCs to address this problem. The resulting scheme is named the *key-chain tree (KCT)* scheme as illustrated by 4.5. This KCT scheme allows the group key manager to revoke any number of users regardless of their positions, and works very well even when the excluded users collude with each other in an arbitrary way.

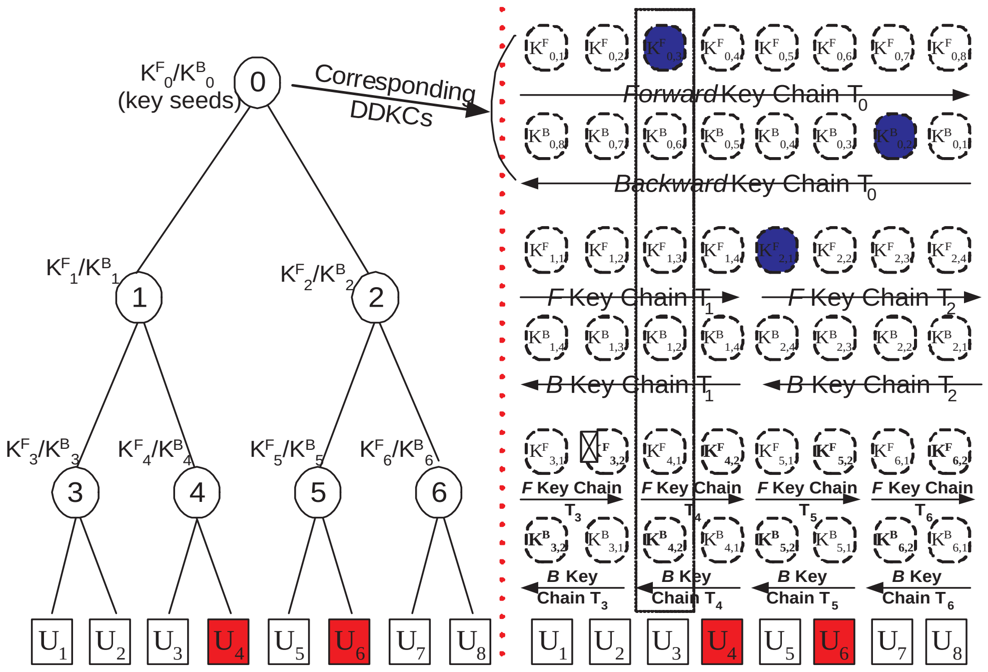


Figure 4.5: An example key-chain tree

System Setup Given the maximum group size N, the KCT scheme maps the users to the leaves of a binary tree. (For convenience, we assume $N = 2^d$, where d is an integer, and we do not distinguish the node and the ID associated with the node in this chapter.) A subgroup G_i is defined as the collection of users in the subtree rooted at an internal node i. Each subgroup G_i is associated with a DDKC. Therefore, there are totaly $N - 1$ DDKCs corresponding to the $N - 1$ subgroups. Figure 4.5 shows an example of a key-chain tree. Moreover, each user u_i is associated with $\log(N)$ DDKCs along the path from the root to leaf i. The corresponding backward/forward keys are assigned to u_i as its personal secrets. For instance, in figure 4.5, the personal secrets of u_3 are $\{K^F_{0,3}, K^B_{0,6}, K^F_{1,3}, K^B_{1,2}, K^F_{4,1}, K^B_{4,2}\}$. $K^F_{i,j}$ denotes the key at position j in the forward key chain associated with subgroup G_i. $K^B_{i,j}$ is defined similarly.

User Revocation Clearly, the R revoked users partition the remaining set of users into at most $R+1$ contiguous blocks. each block of non-revoked users is referred to as a *contiguous subset (of users)*. Formally, there exists a contiguous subset $S_{m\_n} = \{u_i \mid m < i < n,\ m = 0 \ or\ u_m \in \mathcal{R},\ n = N+1 \ or\ u_n \in \mathcal{R},\ u_i \notin \mathcal{R}\ for\ all\ i\}$. Similar to the DDKC scheme, the group key manager needs to find a set of encryption keys to cover every $u_i \in \mathcal{N} \setminus \mathcal{R}$. For instance, the set of cover keys in figure 4.5 is $\{K^F_{0,3}, K^B_{0,2}, K^F_{2,5}\}$.

The *cover key discovery* algorithm is given as follows:

```
1: Sort the revoked users by their IDs. Assume the resulting IDs are $r_1, r_2, ..., r_R$.
2: Find $K^F_{0,r_1-1}$ and $K^B_{0,r_R+1}$, if they exist.
3: Let $i = 1$.
4: In the logical key tree, find the least common ancestor $V$ of users $u_{r_i}$ and $u_{r_{i+1}}$. Let
   $V_l$ and $V_r$ be the two children of $V$ such that $u_{r_i}$ is a descendant of $V_l$ and $u_{r_{i+1}}$ is a
   descendant of $V_r$. Let $p_l$ denote the relative position of user $u_{r_i+1}$ in subtree $V_l$ and
   $p_r$ denote the relative position of user $u_{r_{i+1}-1}$ in the subtree $V_r$. Find the cover keys,
   $K^B_{V_l,p_l}$ and $K^F_{V_r,p_r}$, if they exist[1].
5: Repeat step 4 with $i = i + 1$, until $i = R - 1$.
```

Analysis

Before proving KCT is secure, we first show a contiguous subset in KCT satisfies the key-indistinguishability property.

Lemma 4.1 *The encryption keys for a contiguous subset in KCT is key-indistinguishable for users not in this subset.*

Proof Given a contiguous subset S_{m_n}, if $m = 0$, the subset S_{0_n} is covered by the encryption key $K^F_{0,n-1}$, since this encryption key is only known by user u_j, $0 < j < n$. Thus, $K^F_{0,n-1}$ is key-indistinguishable to any user $u_i \notin S_{0\_n}$. Similarly, if $n = N+1$, $K^B_{0,N-m}$ is key-indistinguishable to any user $u_i \notin S_{m\_N+1}$.

[1]Non-existing means either u_{r_i} is the rightmost node in subtree V_l or $u_{r_{i+1}}$ is the leftmost node in subtree V_r. For example, u_4 is the rightmost node of subtree 1 in figure 4.5.

If $m \neq 0$ and $n \neq N+1$, find the least common ancestor V of u_m and u_n. Let V_l and V_r be the left child and the right child of V, respectively. Therefore, u_m is the descendant of V_l and u_n is the descendant of V_r. Consider all possible scenarios: (1) $V_l = u_m$ and $V_r = u_n$ i.e.,$n = m + 1$. This means S_{m_n} is empty. Thus, no encryption key is needed. (2) $V_l \neq u_m$ and $V_r \neq u_n$. Let p_{m+1} denote the relative position of leaf u_{m+1} in subtree V_l and p_{n-1} denote the relative position of leaf u_{n-1} in subtree V_r. If $K^B_{V_l,p_{m+1}}$ exists, it is only known by a legitimate user u_i where $\{u_i \mid u_i \in G_{V_l}, m < i < n\}$. Similarly, if $K^F_{V_r,p_{n-1}}$ exists, it is only known by a legitimate user $u_j \in \{u_j \mid u_i \in G_{V_r}, m < i < n\}$. Clearly, $\{u_i \mid u_i \in G_{V_l}, m < i < n\} \cap \{u_j \mid u_i \in G_{V_r}, m < i < n\} = S_{m\_n}$. Therefore, $K^B_{V_l,p_{m+1}}$ and $K^F_{V_r,p_{n-1}}$ are key-indistinguishable to any user $u_i \notin S_{m\_n}$. □

Since the contiguous subsets in KCT are disjoint with each other, it is obvious that they satisfy the key-indistinguishability. Therefore, based on Theorem 12 in [86], the KCT scheme does provide a secure encryption of the messages even if the revoked users collude with each other.

Theorem 4.3 summarizes the properties related to the performance of the KCT scheme.

Theorem 4.3 *The KCT scheme requires (i) message length of at most* $2R$ *keys, (ii)* $2\log(N)$ *keys stored at each receiver, and (iii) a single decryption operation and at most* $N-1$ *one-way hash function operations to decrypt a key update message.*

Proof (i) For a contiguous subset S_{m_n}, where $m \neq 0$ and $n \neq N+1$, it requires at most 2 encryption keys to cover this subset, while there are at most $R-1$ such subsets. For subset S_{m_n}, where $m = 0$ or $n = N+1$, a single encryption key is enough. Therefore, the length of the key update message is no greater than $1 + 2(R-1) + 1 = 2R$ keys. (ii) Each user is associated with $\log(N)$ DDKCs along the path from the leaf to the root, and keeps 2 keys for each DDKC. Thus, the total number of keys that a user needs to store is $2\log(N)$. (iii) It is clear that the computation overhead of a legitimate user varies. It not only depends on the set of the revoked users and their positions in the logical tree, but also on the position of this legitimate user. The upper bound is decided by the size of the contiguous subset of which this non-revoked user is a member. Since the maximum contiguous subset size is N in the case of no revoked user at all, the computation overhead in the KCT scheme is bounded to $N-1$ one-way function operations, and one decryption operation. □

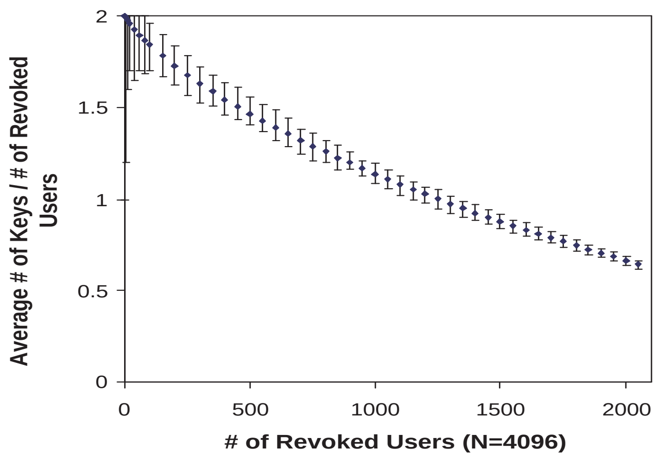


Figure 4.6: Averaged communication cost for KCT scheme based on simulation

Theorem 4.3 shows the worst case of communication overhead in KCT scheme. The real cost depends on $\mathcal{N}$, $\mathcal{R}$ and the positions of the revoked users. Simulation experiments were performed to further examine the average cost for randomly revoked users. They shows that the average cost is lower than the upper bound $2R$. Figure 4.6 shows the communication overhead obtained in the simulations, in which R users are randomly selected to be revoked from a group of 4,096 users. For each data point, the average, the minimum, and the maximum values measured (basing on 10,000 times of experiments) for the indicated number of revoked users are shown.

4.2.3 Layered Key-Chain Tree Scheme

The main disadvantage of the KCT scheme is its high computation overhead, which involves $N - 1$ one-way hash function operations in the worst case. In this subsection, an extension, i.e., the *layered key-chain tree (LKCT)* scheme is introduced to reduce such computation overhead.

The basic idea of the LKCT scheme is to divide a large group into a collection of smaller subgroups as illustrated in Figure 4.7. The upper-layer KCT is used to cover the subgroups, and the lower-layer KCTs are used to cover the real group members.

System Setup Given the maximum group size N, the group key manager first constructs a KCT (upper-layer KCT) of size $\sqrt{N}$. Each leaf node in the upper-layer KCT corresponds to a subgroup of $\sqrt{N}$ users. The group key manager then constructs another $\sqrt{N}$ lower-layer KCTs, each of which has $\sqrt{N}$ leaves. Each leaf of a lower-layer KCT

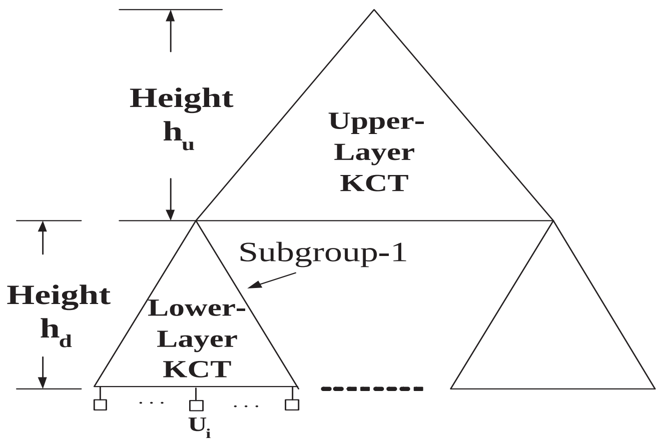


Figure 4.7: An illustration of a LKCT

corresponds to a user. A subgroup is defined as the users in a lower-layer KCT.

When a new user joins the group, the group key manager puts it in a subgroup that has less than $\sqrt{N}$ users, and maps it to an unused leaf node in the corresponding lower-lay KCT. The group key manager then assigns the keys in both KCTs to the new user following the KCT scheme.

User Revocation To revoke a set of users, the group key manager uses the KCT scheme to revoke the subgroup which contains some revoked users from the upper-layer KCT, and then for each lower-layer KCT including revoked users, the group key manager revokes these users with the lower-layer KCTs corresponding to their subgroups

For instance, in order to revoke u_i in figure 4.7, the group key manager first finds a set of encryption keys to cover all the subgroups except for subgroup 1 in the upper-layer KCT. This step requires determining at most 2 encryption keys. After that, the group key manager needs to find another set of encryption keys to cover the remaining users in subgroup 1. It is clear that this revocation also requires at most 2 keys. Finally, the group key manager uses the resulting keys to encrypt the new group key separately and broadcasts all the cipher-texts in the key update message. All the users except u_i can decrypt the new group key from the key update message.

Analysis

LKCT is an extension to KCT. It can easily prove that the subsets in LKCT also satisfy the key-indistinguishability property as those in KCT. (The details are omitted in order to avoid repetition.) Therefore, the key update messages in LKCT are inaccessible to the revoked users even they collude with each other.

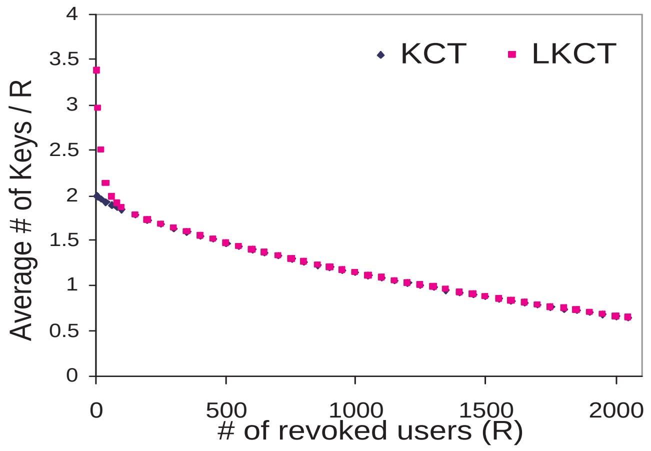


Figure 4.8: Average communication cost (N=4096) based on simulation

The performance properties of LKCT scheme is summarized as follows.

Theorem 4.4 *The LKCT method requires (i) message length of at most* $4r$ *keys, (ii)* $2\log N$ *keys stored at each receiver, and (iii) at most* $\sqrt{N}-1$ *one-way function operations and a single decryption to decrypt a key update message.*

Proof (i) Assume the revoked R users are distributed in w low-layer KCTs, denoted as $g_1, ..., g_w$. Further assume in the low-layer KCT g_i, there are x_i revoked users, where $i = 1, ..., w$ and $x_1 + x_2 + ... + x_w = R$. According to Theorem 4.3, it needs at most $2w$ keys to revoke the w lower-layer KCTs. Moreover, for each low-layer KCT g_i, where $i = 1, ..., w$, it needs at most $2x_i$ keys to revoke the x_i users in g_i. It totally needs $2w + 2x_1 + ... + 2x_w = 2w + 2R \leq 4R$ keys to generate the key update message. (ii) Each user needs to store keys in the upper-layer KCT and the lower-layer KCT in which it resides. The total number of keys each user needs to keep is $2\log\sqrt{N} + 2\log\sqrt{N} = 2\log N$. (iii) To compute the group key, a legitimate user needs to use either the upper-layer KCT if no user in its subgroup is revoked, or the lower-layer KCT otherwise. Thus, according to Theorem 4.3, it needs at most $\sqrt{N}-1$ hash operations to compute the encryption key, and one decryption operation to get the updated group key. □

By using the layers, the LKCT scheme significantly reduces the computation overhead, from at most $N-1$ to at most $\sqrt{N}-1$, at the expense of a slightly increased communication overhead. Figure 4.8 compares the communication overhead between LKCT and KCT for revoking R users from a group of 4,096 users. It shows that the communication overhead in LKCT scheme is still low for the average case.

4.3 Performance Comparison and Discussion

This section compares the performance of the proposed schemes with some existing stateless group key revocation approaches. Table 4.1 shows the storage, the communication, and the computation complexities of these schemes. It shows that the SC scheme and the proposed schemes have a lower storage overhead than SD and LSD. However, the SC scheme has a much higher communication overhead than the others. The SC scheme has the lowest computation overhead, while the proposed schemes have the highest computation overhead. The proposed schemes trade off the storage and communication overhead with the computation overhead.

Table 4.1: Performance Comparison on Major Features

	Storage Overhead	Communication Overhead	Computation Overhead
SC [86]	$O(\log(N))$	$O(r\log(N/R))$	$O(\log(\log(N)))$
SD [86]	$O(\log^2(N))$	$O(R)$ worst case ($2R$)	$O(\log(N))$
LSD [59]	$O(\log^{1+\epsilon}(N))$	$O(R)$ worst case ($4R$)	$O(\log(N))$
KCT	$O(\log(N))$	$O(R)$ worst case ($2R$)	$O(N)$
LKCT	$O(\log(N))$	$O(R)$ worst case ($4R$)	$O(\sqrt{N})$

Table 4.1 only compares the worst case performance of the methods. The real performance (especially the communication cost) of a stateless group key revocation scheme not only depends on the number of revoked users R, but also on the revoked users' positions in the logical tree. Simulation were performed to compare the average communication cost of these schemes. Given a chosen number of revoked users, R, the simulation was repeated 10,000 times using different random generator seed. The average number of keys in a key update message of each scheme is show by figure 4.9.

As shown in figure 4.9, (i) the SC scheme has the highest average communication overhead (which is much higher than the others), (ii) the SD scheme has the lowest average

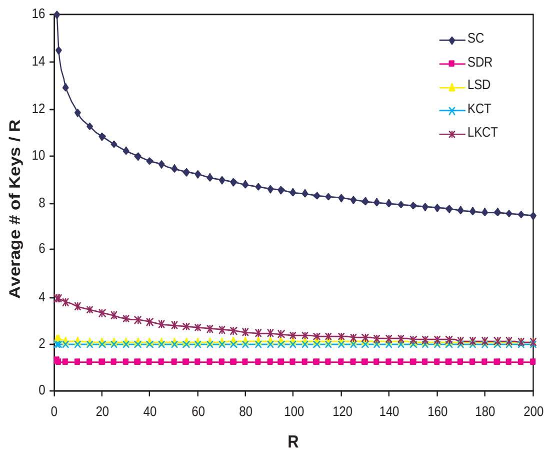


Figure 4.9: Communication overhead ($N=2^{16}$)

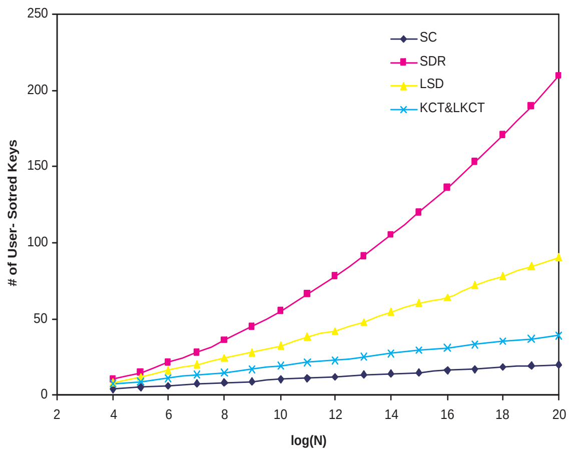


Figure 4.10: User storage overhead vs. group size

communication overhead, (iii) the KCT scheme has a slightly higher average communication overhead than SD, but lower than LSD, and (iv) if R is large enough, the LKCT scheme has a similar average communication overhead to LSD.

Figure 4.10 shows the storage overhead of each scheme. Clearly, the SD scheme has the highest storage overhead. The proposed schemes have a lower storage overhead than SD and LSD, but higher than SC.

The above analysis shows that the SC scheme is a better solution for scenarios in which a user has a high communication bandwidth but very limited memory and computation power. The LKCT and KCT schemes are suitable for scenarios in which a user's memory and communication bandwidth are critically limited. If a user has enough memory to store all the keys, both SD and LSD are more suitable than the other schemes, and SD is better than LSD on communication overhead.

It is worth noting that the proposed schemes perform better when the revoked users are clustered rather than randomly distributed. Since group members may be ordered based on specific organizational requirements, e.g., geographic location, the scenario of excluding a cluster of users from a group may be common in real life.

The higher computation overhead of the proposed schemes is not a major obstacle in real implementation. [38] shows the speed of computing a MD5 hash function on a Pentium 4 2.1Ghz processor is 204.55 Megabytes/second. It thus takes about $40\mu s$ for 1024 hash operations with 64 bit key lengths, which is the maximum computation overhead for the LKCT scheme with 1 million users. The real processing time may be a little bit longer, since the cost of hashing many short messages is different from that of hashing large quantities of data.

Jakobsson [62] as well as Coppersmith and Jakobsson [35] proposed schemes to improve the performance of the one-way key chain, which requires only $O(log(N))$ storage (i.e., keys) and $O(log(N))$ computation (i.e., hash operations) to access an element. By adopting their schemes, the proposed KCT and LKCT can achieve the same theoretic complexity as the SD [86] scheme on the performances. That is, the computation overhead is further reduced to $O(log(N))$ hash operations, while the storage overhead is increased to $O(log^2(N))$ keys.

4.4 Summary

Due to the dynamics of memberships and the instability of communication channels, a stateless group key distribution scheme is desired in order to protect group communication in dynamic network environments. In this chapter, two storage-efficient stateless group key revocation schemes are introduced. They reduce the user storage requirement by trading-off with the communication and computation costs. Given a group of N users, the proposed schemes only require each user to store $O(\log N)$ symmetric keys, and allow a group key manager to revoke R users by broadcasting a key update message at the size of $O(R)$ measured by the length of symmetric key, at the cost of $O(N)$ and $O(\sqrt{N})$ hash operations respectively. They are especially suitable for scenarios in which the memory and the bandwidth are critically limited.

Chapter 5

Access Control - A System for Mobile Ad Hoc Networks

Network access control is a challenging issue in dynamic network environments. Without access control, an attackers can easily gain access to the network and launch various attacks. For instance, an attacker may inject a large number of “bogus” or spurious packets into the network, simply to consume network bandwidth.

Firewalls have been used to enforce network access control, using network topology and service information. They are particularly valuable for guarding against resource consumption or Denial of Service (DoS) attacks. However, the dynamic population of users and the changing network topology make it difficult to directly apply techniques such as firewalls in some dynamic network environments, e.g., mobile ad hoc networks.

In this chapter, we propose to enforce network access control in mobile ad hoc networks using cryptographic methods. In the proposed approach, packets are authenticated by means of a network-wide access control (secret session) key, which is only known by the approved nodes (i.e., legitimate nodes). Any packets that are not authenticated correctly will be immediately dropped by the receiving node. As a result, the packets from illegitimate nodes are prevented from propagating from the network.

A critical challenge is how to manage such access control keys. Due to the need to remove compromised nodes, this access control key may have to be updated frequently. The dynamic network topology and frequent communication failures make it difficult to achieve a globally-synchronized key after each key update. Nodes wishing to communicate may therefore hold different session keys, which must somehow be synchronized. To the best of our knowledge, no existing key distribution schemes can ensure such key synchronization, *even if* the key manager is assumed to be always online.

To address this issue, we introduce a distributed key synchronization method making use of *stateless group key distribution* schemes (e.g., [59, 86, 110]). As described in the previous chapter, stateless group key distribution schemes have a nice property that a legitimate node can get the updated group key from a received key update message, even if the node has missed several rounds of key update. The proposed key synchronization method guarantees that whenever legitimate nodes communicate with each other, they will synchronize their access control keys and agree on the latest one needed to authenticate and verify data packets. We also design a packet retransmission mechanism to deal with the received packets that cannot be verified immediately. This can happen if a new access control key has not been fully distributed throughout the network. This proposed packet retransmission mechanism allows the involved nodes to synchronize their access control keys and establish or continue communication

The rest of this chapter is organized as follows. Section 5.1 introduces the related work on access control for mobile ad hoc networks. Section 5.2 presents the network and attack models. Section 5.3 discusses the proposed system and the critical techniques for synchronizing session keys. Section 5.4 analyzes the communication overhead and the security of the proposed system. Section 5.5 shows a proof-of-concept implementation and the test results on some wireless devices. Section 5.6 summarizes this chapter.

5.1 Related Work

Basagni, et. al [12] suggested using a symmetric key to secure the communication in a mobile ad hoc network. Their underlying idea is similar to the proposed approach. However, they suggested using hierarchical group key management, and did not give a solution on how to synchronize the session keys and how to handle the unverified packets in the case of key un-synchronization.

Recently, Zhu, et. al., [129] designed a lightweight hop-by-hop authentication protocol (LHAP) for mobile ad hoc networks. LHAP uses the TESLA broadcast authentication [93] technique and the one-way key chain. It requires clock synchronization between nodes, and cannot completely prevent the propagation of forged packets. For instance, an attacker can use TESLA keys eavesdropped from a legitimate node A to authenticate its forged packets. It then fools the nodes that set up a trust relationship with A, but lose contact for a while, to accept these forged packets.

D. Kraft and G. Schafer [70] proposed a distributed access control scheme for consumer operated mobile ad hoc networks. Their scheme relies on a web-of-trust approach (like PGP [130]) for authentication. Such schemes are not designed for secure group communication and therefore are not suited for the purpose of network access control.

Ioannidis, et. al., [61] presented a distributed firewall system. In their proposed system, a centrally defined security policy is propagated to each network endpoint, which executes this security policy to filter out unwanted packets. IPsec is used to authenticate users, protect traffic, and securely distribute credentials. This approach is likely to be impractical in a mobile ad hoc network, since IPSec is a point-to-point protocol, and managing a security association between every pair of nodes will be difficult.

R. Canetti, et. al., [24] proposed a host architecture for secure Internet multicast, which is somewhat similar to the proposed method. However, their architecture requires each member to set up an IPsec security association with the key manager, and does not solve the problem of key un-synchronization. Thus, it is impractical to be used as a network access control system for mobile ad hoc networks.

5.2 Underlying Models

In this section, we present the network model and the attack model used in the proposed access control system.

5.2.1 Network Model

We assume the wireless links in the network are bidirectional, i.e., if node A is in the transmission range of node B, node B must also be in the transmission range of A.

We notice that without a logically centralized authority, a faulty entity can behave arbitrarily and thus defeat an access control system. Therefore, we assume all legitimate nodes come from one domain and their accesses to mobile ad hoc network are controlled by a *key manager* that is not required to be always online. We argue that this assumption is reasonable, if we want to enforce access control in mobile ad hoc networks.

All nodes register at the key manager offline before joining the network. Each of them is pre-configured with a unique identifier (ID), ID certificate, current network access control key, and a set of personal secret keys used for stateless key distribution purposes. We also assume neighboring nodes know each other's ID by verifying the ID certificate.

Whenever a new network access control key needs to be distributed, e.g., after a compromised node is detected, the key manager broadcasts a key update message to the network, or gives the key update message to a newly joined node. Note that the methods for detecting compromised nodes are not in the scope of this report.

5.2.2 Attack Model

The proposed network access control system relies on cryptographic techniques to control the nodes' access to a mobile ad hoc network. An attacker with unbounded computing capability is capable of compromising any practical security protocol. We thus assume that the attackers have bounded computing capability. They cannot break the adopted stateless group key distribution scheme.

An attacker may wish to inject packets into the network with the goal of depleting the resources of nodes relaying the packets. Even though neighboring nodes will not forward bogus packets injected by an attacker, they will have to spend some resources on verifying these packets. Such a local resource consumption attack is possible and cannot be prevented by the proposed scheme.

5.3 Proposed Access Control Scheme

As we mentioned, a mobile ad hoc network is vulnerable to resource consumption attacks, if it cannot prevent the bogus packets from being propagated. Unfortunately, the dynamic natures of mobile ad hoc networks prevent them from deploying any conventional firewalls [34], which depend on network topology to filter out the bogus packets.

One potential solution is to deploy the technique of digital signature [51, 88], i.e., a node only accepts the packets digitally signed by other legitimate nodes. However, this approach itself might be a target to the attackers, since the asymmetric cryptographic operation is very expensive. For example, by sending the packets with faked signatures, the attacker can easily make a victim node to waste its computation power on verifying these bogus packets.

In this section, we present a symmetric key based network access control system for mobile ad hoc networks. That is, a key manager (not required to be always online) controls the nodes' accesses to the mobile ad hoc network by selectively distributing a common network access control key. Only authorized nodes which have not been revoked by the key manager can get such a common key.

Each node uses this network access control key to authenticate every outgoing packet, i.e., to insert a *Packet Authentication Header* (PAH). Figure 5.1 illustrates a possible way of adding the PAH, and table 5.1 shows the functionality of each field in the PAH.

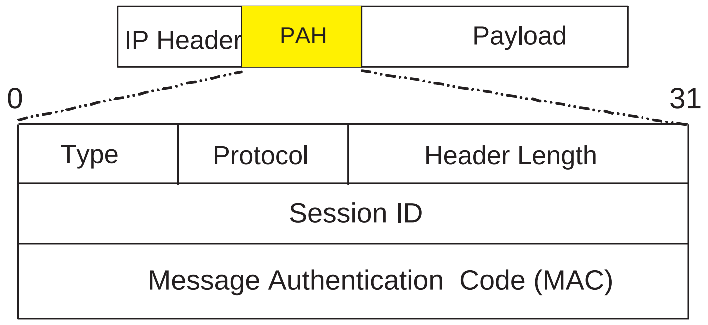


Figure 5.1: An illustration of packet authentication header.

The nodes also use the network access key to verify the PAH of a received packet, either intended for them or transiting through them. They immediately drop any packet not properly authenticated. As a result, a non-authorized node that is either unknown to or revoked by the key manager is prevented from injecting packets into the network.

Table 5.1: Fields in the Packet Authentication Header

Type	The type of a packet, either a control packet or a data packet
Protocol	The protocol value in original IP header, since the latter is replaced with an unsigned value (e.g., 199). So that a packet can be restored once it is accepted at the destination
Header Length	Length of the packet authentication head
Session ID	The session ID of network access control key used in the packet
MAC	Message authentication code

Clearly, the network access control key may have to be updated due to the revocation of malicous/compromised nodes. Since each key update (rekeying) event defines a key session, the access control keys are also referred to as session keys. In this chapter, we use network access control key and session key interchangeably, and we assume a legitimate node only uses the most recent session key it has received.

To make the proposed symmetric key based network access control system practical, two critical issues need to be solved.

1. The proposed system requires all legitimate nodes agree on the same access control key at the time they communicate. However, no existing group key distribution scheme can guarantee that all nodes receive the latest session key after a key update. Maintaining a synchronized session key therefore becomes a non-trivial task, given the dynamic nature of mobile ad hoc networks.

2. Two legitimate nodes cannot communicate in the proposed system, if a more recent key update message only reaches one of them, unless they can agree on one specific session key. Thus, another challenging problem is how a node handles the received packets authenticated with different session keys, to ensure secure establishment/continuation of communication with corresponding senders.

In the following subsections, we present solutions to these two issues. We first introduce a key synchronization method, which exploits the stateless feature of stateless group key distribution schemes. This method assists in the distribution of latest network access control (session) key to all legitimate nodes. Next, we introduce a packet retransmission mechanism when two communicating nodes have inconsistent views of the network access control keys. This mechanism allows the involved nodes to synchronize their keys, and ensures the data is delivered and correctly authenticated.

5.3.1 Synchronization of Network Access Control Keys

Problem Definition

In the proposed network access control system, ideally, all legitimate nodes should use the same (latest) session key at any time. However, due to communication failures, the offline of some nodes, and changes in the network topology, a key update message may fail to propagate across the entire network. As a result, two legitimate nodes may simultaneously hold different session keys. We refer to such a scenario as *key un-synchronization*.

Definition 5.1 *Let U denote a set of legitimate nodes, and $K_U^{(t)}$ denote the set of session keys used by the nodes in U at time t. U is* key-synchronized *at time t, if $|K_U^{(t)}| = 1$. Otherwise, we say U is* key-unsynchronized*.*

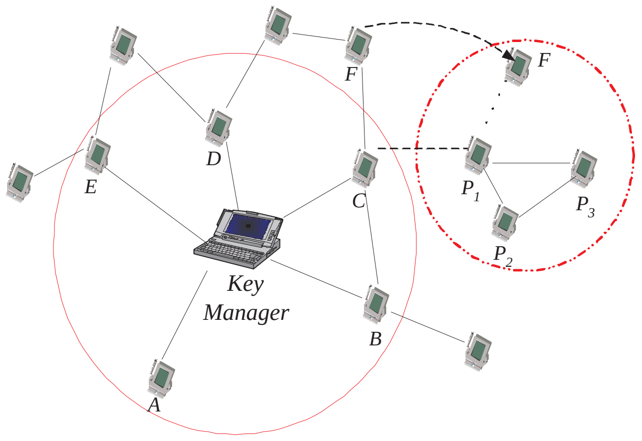


Figure 5.2: An example of key un-synchronization

Key un-synchronization is a fatal threat to the proposed network access control system. It prevents key-unsynchronized nodes from establishing normal communication. For instance, due to the link failure between node C and node P_1 in Figure 5.2, nodes P_1, P_2 and P_3 missed the latest key update message and used an old session key K_o. Suppose node F, which holds the latest session key, roams into the communication range of nodes P_1, P_2 and P_3. Even though these four nodes are still legitimate nodes, nodes P_1 , P_2 and P_3 cannot communicate with node F, because their packets are authenticated/encrypted with different session keys. From the view of the shared session key, we can say that node F is *logically partitioned* from nodes P_1, P_2 and P_3, although they are physically connected.

Several possible methods could be employed to remedy such key un-synchronization based on assistance from the key manager. For instance, nodes that become aware they have not received the newest session key could directly contact the key manager to get the key, or the key manager could periodically rebroadcast the key update messages. However, such methods are overly-dependent on the key manager. The key manager could be unreachable by a large fraction of nodes due to temporary partitioning of the network, *even if* it is always online. In such a case, nodes that can physically reach each other by some path will not be able to synchronize their session keys.

In the following, we present a distributed key synchronization method. The detailed algorithm of the proposed method will be presented in the next subsection, due to the interaction between key synchronization and data packet transmission.

Proposed Method

The proposed key synchronization method depends upon the advances in stateless group key distribution [59, 86, 110]. It helps distribute the latest session key to all legitimate nodes, and it guarantees that whenever two legitimate nodes communicate with each other, they will synchronize their session keys and agree on the most recent one to use.

In the proposed key synchronization method, each node buffers (without modification) the key update message it most recently receives. It directly transmit this buffered message to other nodes that may be using an older session key. The corresponding nodes can then extract the new session key from the received message if they were not revoked by the key manager. This is because that the stateless key distribution schemes allow a legitimate user to get the updated group key as long as the user has the corresponding key update message, *even if* the user has been off-line for a while or missed several previous rounds of key update. In this way, two key-unsynchronized nodes that want to communicate can synchronize their session keys without relying upon the key manager, if both of them are legitimate nodes.

We notice that an attacker may send "bogus" key update messages. This could lead to a resource consumption attack, if the verification of a key update message involves expensive operations (e.g., asymmetric cryptographic operations). To mitigate this attack, all session keys in the proposed system are generated from a one-way key chain. The key manager sequentially uses these session keys for network access control, and pre-distributes K_0 or the current session key K_i to a legitimate node as the commitment of the key chain. As discussed in chapter 2, a node thus can verify the authenticity of a new session key (or a new key update message) with a limited number of hash operations that are much cheaper than an asymmetric cryptographic operation.

In order to accelerate the convergence of the network access keys in a mobile ad hoc network, we suggest that each node periodically broadcasts a *beacon message*. This beacon message can be either the node's buffered key update message, or a special message, e.g., the (authenticated) hello message in some routing protocols. From a received beacon message, a local receiver can detect whether it is key-synchronized with the sender. If not, the receiver can initiate a procedure of key synchronization.

5.3.2 Packet Retransmission in Case of Key Un-synchronization

As was mentioned earlier, in the case of key un-synchronization, a legitimate node may receive some packets authenticated with a different session key rather than the one it holds. For convenience, we refer to such received packets as *unverified packets*, since they cannot be immediately verified by the receiver. Simply accepting or forwarding an unverified packet is not acceptable. Otherwise, an attacker can easily defeat the proposed network access control system by using a spurious PAH.

There are two important design factors: (a) how a receiver synchronizes the session key with the corresponding sender, and (b) whether a receiver buffers an unverified packet. A legitimate node has four major options to avoid key un-synchronization or to handle an unverified packet. These options are:

1. Synchronizes the keys with all nodes along the communication path first, to avoid the key un-synchronization in future data transmission.

2. Buffers the unverified packets first, then synchronizes the session key with the sender, and finally verifies these buffered packets.

3. Simply drops the unverified packets in case of key un-synchronization.

4. Drops the unverified packets and asks the sender to retransmit the packets, while a key synchronization is triggered during the retransmission.

The first three options each have serious drawbacks. The first option cannot exclude the possibility that a transmission cannot complete because of frequent key updates. The second option exposes legitimate nodes to memory consumption attacks. That is, the victim nodes are enforced to buffer the bogus packets authenticated with a "newer" session key, while waiting to receive the necessary session key update message. The third option prevents two legitimate nodes from establishing or continuing communication when they are (or become) key-unsynchronized.

In the proposed network access control system, we propose to use the fourth option. It assists the communicating nodes to reestablish communication in the case they become key-unsynchronized, constrains the propagation of any unverified packet to a single hop, and avoids the memory consumption attack since the receiver is not required to buffer any unverified packets.

Proposed Approach

In this subsection, we present the details of the proposed packet retransmission and key synchronization algorithm. We start with the pseudo-code for unicast data packets, shown in figure 5.3.

In the packet receiving part, if the sender and the receiver are key-synchronized, i.e., the session ID in the received packet is equal to the receiver's, the latter verifies the PAH immediately. Otherwise, the receiver needs to synchronize the session key with the sender and requests a packet retransmission.

The receiver sends a retransmission request to the sender, if the sender has a more recent session key. After receiving this retransmission request, the sender retransmits the data packet with its buffered key update message. The receiver then extracts the new session key from the attached key update message, and verifies the retransmitted packet. Conversely, the receiver sends a retransmission request to the sender and attaches its own buffered key update message. The sender extracts the new session key from the attached key update message, authenticates the data packet with this key, and retransmits the data packet. The receiver can then verify the retransmitted packet. Thus, any unverified unicast packet is prevented from propagating beyond a single hop.

The proposed algorithm uses an ACK mechanism to ensure a data packet is received correctly. Thus, it requires the sender to retain a packet until it is ACKed. If the received packet is verified, the receiver replies with an authenticated ACK. The sender is then free to purge the acknowledged packet from its sending buffer.

The proposed algorithm may allow some packets from a revoked node to be successfully delivered and verified, after the node has been revoked, in the following way. Assume there are 3 nodes S, X, and D, all synchronized with the same session key, and

```
Locally broadcast a beacon message every t time unites
When sending a packet P
    insert a PAH to P
When receiving a packet P from sender X
   CASE 1: P is a data packet
   if P's session ID > node's current one
      drop P, send a retransmission request to X
      along with a key synchronization request
   else if P's session ID < node's current one
      if X is in the list of revoked nodes
         do nothing
      else  send a retransmission request to X
         along with the buffered key update message
   else  verify the PAH in P
      if PAH is correct
         accept P and send ACK to X
      else  drop P
   CASE 2: P is a key update message
   if P is authenticated for a new session key
      buffer P, extract new key, forward P,
      and calculate the list of revoked nodes
   else drop P
   CASE 3: P is a retransmission request
   if a new key update message is attached
      extract the new session key and verify PAH in P
      if verification proves correct
         send ACK to X
      else drop p
   else retransmit the data packet
      along with the buffered key update message
   CASE 4: P is an ACK
      remove the acked packet from the buffer
```

Figure 5.3: Pseudo-code for key synchronization and packet retransmission

suppose S sends a packet to D through X. X will verify the packet and forward it to D, but in the meantime D may receive a new session key update message (which revokes S from the group). When X forwards the packet to D, the packet will fail to be verified. D will request retransmission and send the new session key update message to X. X will update its session key, reauthenticate the packet, and retransmit it to D, which will verify the packet and accept it.

A solution to this problem would require an additional check. In essence, each receiving node must check if the source of a packet has been revoked according to the session key held by the receiver. If so, the packet is dropped and communication fails. Otherwise, the receiver synchronizes the session key with the sender and forwards the packets to the next node in the path. A configuration parameter can be set to implement this more stringent requirement. As a result, no packet will be verified and accepted by a node that has received a key update message revoking the sender of that packet.

It is worth noting that the proposed algorithm must be modified slightly in the case of a *broadcast* data packet. In this case, we assume ACKs are not used, in order to avoid the ACK implosion problem (i.e., network bandwidth exhaustion from transmitting too many ACKs triggered by one broadcast packet). Accordingly, a data packet that is broadcast is not stored by forwarding nodes. A receiver that requests retransmission therefore has to send its retransmission request to the source of the broadcast packet. The source can then perform the normal processing described above, and rebroadcast the packet.

5.4 Analysis

5.4.1 Correctness

In the proposed system, a node can tell whether it is key-synchronized with the sender by examining the PAH in the received packet. If the sender and the receiver are key-unsynchronized, the receiver will employ the proposed key synchronization algorithm to synchronize their session keys. A more recently buffered key update message is therefore exchanged between these two nodes. Meanwhile, any improperly authenticated key update message will be detected and filtered out since a node can easily verify the authenticity of a key update message based on its current session key. The corresponding node which holds the old key will extract the new session key from the received key update message, and as a result, these two nodes converge on the newer of their two keys. For a communication route, all nodes enroute will synchronize on the most recent key they have, once the destination node finally verifies the received packet.

Theorem 5.1 *Given a communication path formed by the legitimate nodes and the time interval of key updates is much larger than the packet propagation delay, the proposed key synchronization scheme ensures that these legitimate nodes can communicated with each other and all of them agree on the latest session key they have after the destination node finally verifies the received packet.*

Proof Suppose node S tries to communicate with node D via n intermediate nodes, $U_1 ... U_n$. These $n + 2$ legitimate nodes use v session keys, while $v \leq n + 2$ and K_v is the latest one they have. It is clear that if there were new key update messages continuously arriving in the procedure of key synchronization, these nodes had to continuously update their session keys and might not be able to agree on the latest one. To prevent such scenarios from happening, we assume that the time interval between the key updates is much larger than the propagation delay between S and D. Therefore, these nodes can finish the procedure of key synchronization before a new session key is published by the key manager. This assumption is reasonable. Since the key manager only updates the session key when the set of legitimate nodes changes or based on certain time period, e.g., per day or per week, the time interval between two consecutive key updates generally is larger than several minutes. (It is hard to image a mobile ad hoc network that there is a node joining or leaving every minute). While, the propagation delay between S and D generally is less than several seconds, if n is not extremely large.

Now, suppose the session key K_i is used by S, where $i \leq v$, and S already has a route from itself to D. S first sends the data packet to the down-streamed node U_1. There are three possible scenarios, when the packet arrives at U_1. (i) U_1 also uses K_i. It then verifies the received packet and forwards the packet to U_2. (ii) U_1 uses an old session key K_j, where $j < i \leq v$. U_1 drops this received packet and asks the corresponding sender, which is the source node S in this case, to retransmit the data packet, while a key synchronization request is attached. After receiving the retransmission request, S encloses its buffered key update message to the data packet and retransmits it. If the attached key update message proves authentic, U_1 can decrypt K_i from it and update the context of its previously buffered key update message. U_1 then uses the new session key to verify the retransmitted data packet and forwards it to U_2. (iii) U_1 uses a more recent session key K_j, where $i < j \leq v$. U_1 now is key unsynchronized with not only the sender but also all up-streamed nodes. U_1 sends a retransmission request to the source node, while its buffered key update message is attached. The propagation of this retransmission request will trigger all up-stream nodes to update their session keys. After receiving the retransmission request and decrypting the new session key K_j, S retransmits the data packet which is authenticated with K_j now. U_1 then verifies the retransmitted packet and forwards it to U_2. For simplicity, we call the above procedure as one round of key synchronization. Now, U_2 verifies the received packet, launches anther round of key synchronization if there is a key un-synchronization, and then forwards the packet to $U_3 \ldots$

Lately, as the data packet arrives at node U_y $(1 < y \leq n)$, which holds the latest session key K_v, another round of key synchronization is triggered. U_y and all up-stream nodes then agree on K_v. Sequentially, all the down-stream nodes will also update their session keys to K_v as the data packet arrives, since S uses K_v now. D may also ask for a retransmission, if it uses K_q, where $q \neq v$. However, after D finally verifies the received packet, all these $n + 2$ nodes agree on K_v. Clearly, achieving such key convergence requires at most $n + 1$ rounds of key synchronization. □

5.4.2 Security

The proposed system employs a symmetric key to prevent non-authenticated packets from propagating beyond a single hop, and thus enforces the network access control. However, an attacker could launch some attacks against the proposed system.

A malicious node may cause unnecessary communication by broadcasting a forged beacon message, transmitting a key synchronization request, or sending data packets with spurious PAH. The legitimate nodes will respond by either sending a key synchronization request or transmitting their buffered key update messages, wasting both network bandwidth and battery power. It is not possible to completely prevent such attacks in the proposed system. However, a malicious node can only trigger the legitimate nodes to send a key synchronization request, which generally is much smaller than a buffered key update message, by declaring a newer session key has been used. So the effects of such resource consumption attacks are localized. We speculate that identifying such attacks and the source of the attacks is easier as a result.

An inside attacker, i.e., a compromised but undetected and therefore not revoked node, may refuse to forward information. This may cause a logical (key) partition of the network. The proposed system cannot prevent such a logical partition caused by an inside attacker. However, it can be avoided if the network is highly-connected. Otherwise, the issue of nodes which refuse to forward packets is known as a “selfish node” problem for mobile ad hoc networks, for which several solutions have been proposed (e.g. [80]).

The proposed system cannot guarantee a global key synchronization in the case of network partitioning. Therefore, a newly revoked node may enter a subnetwork that still uses an old session key and cause damage. This is a limitation of the proposed scheme, and no method based on centralized revocation can overcome such a limitation. However, we argue that the mobility of nodes accelerates the convergence of the session key of the whole network. A revoked node will eventually be excluded in the proposed system, if a node using the latest session key joins that subnetwork.

Since the total number of sessions is predetermined by the length of the one-way key chain, an attacker may attempt to deplete the sessions (and thus the lifetime of the network) by frequently joining and leaving, requiring the key manager to frequently update the session key. To mitigate this attack, the key manager can generate a long key chain. Coppersmith and Jakobsson [35] already proposed a scheme to improve the performance of the one-way key chain, which requires only $O(log(N))$ storage and $O(log(N))$ computation to access an element, where N is the total number of keys. Furthermore, since a node needs to register at the (centralized) key manager offline to gain network access, repeated registration can be easily identified.

5.4.3 Overhead

The computation overhead comes from the operations of packet authentication and verification. A source node needs to calculate a PAH for every outgoing packet, and a receiver needs to verify this PAH before accepting or forwarding the packet. Since symmetric cryptography is used, calculating and verifying a PAH will be very fast and cheap in the proposed system. Such routine operations will not cause a heavy burden.

The communication overhead arises from several sources. First, each packet contains a PAH header, which introduces a message overhead. Second, the periodically broadcasted beacon message presents another source of communication overhead. Since the length of a beacon message is quite small (on the order of an IP header plus a PAH header), such a cost is tolerable. Finally, a node may have to retransmit the data packet, when it is key-unsynchronized with the receiver. In the worst case, a data packet may need to be retransmitted l times before it is successfully delivered to the destination, where l is the length of the path. However, since key un-synchronization generally does not happen frequently, the communication overhead caused by the packet retransmission is tolerable. In the next section, we present experimental results that investigate how frequently key-synchronization is required, and the percentage of retransmitted packets (extra communication delay).

5.4.4 Performance

We evaluated the performance of the proposed method by means of simulation. We created our own simulator, which implemented the transmission of data between nodes. For evaluation purposes, we assumed shortest path routing information was maintained at all times. This level of detail was sufficient to evaluate the effectiveness and performance tradeoffs of the proposed method.

We generated a mobile ad hoc network with n nodes in a 1km x 1km area. Nodes were initially placed in a random way. Each node's movement was simulated as a random walk [23], with a maximum speed of 20m/s. The communication range for each node was uniformly set to 200m. Each data point measured is the average of 2000 simulations, using different seeds for random number generation. The average percentage of nodes with the latest session key was then measured after each round of key update for 8 consecutive rounds, supposing only the session key needs to be updated, and no revoked nodes.

The average percentage of reachable nodes from a randomly-selected source node, referred to as *reachability* in this paper, affects the performance of the proposed system. Figure 5.4 shows that for a fixed geographic area and transmission range, as the total number of nodes increases, the reachability increases. In the following experiments, we simulated a network with either 40 nodes, for which the reachability is 70%, or a network with 80 nodes, for which the reachability is 98%.

Next, we compared the proposed key synchronization scheme with the stateful and

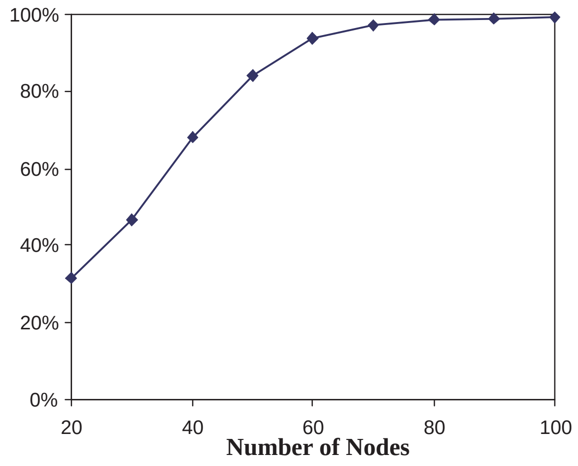


Figure 5.4: Ave-percentage of reachable nodes from a randomly chosen node

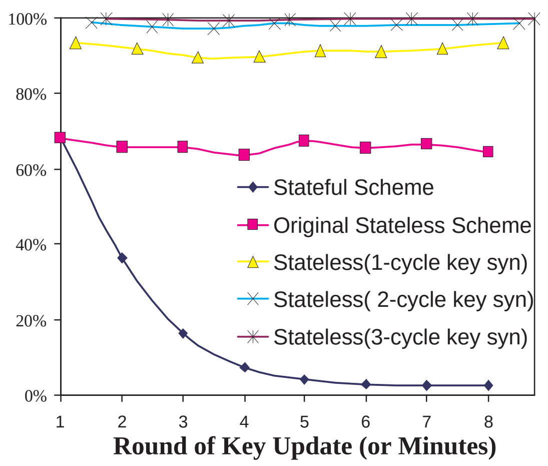


Figure 5.5: Ave-percentage of nodes received the latest key (P_{lost}=0, 40 nodes)

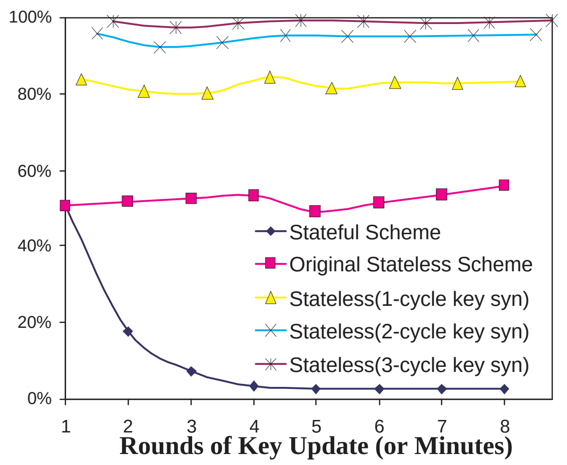


Figure 5.6: Ave-percentage of nodes received the latest key (P_{lost}=0.25, 40 nodes)

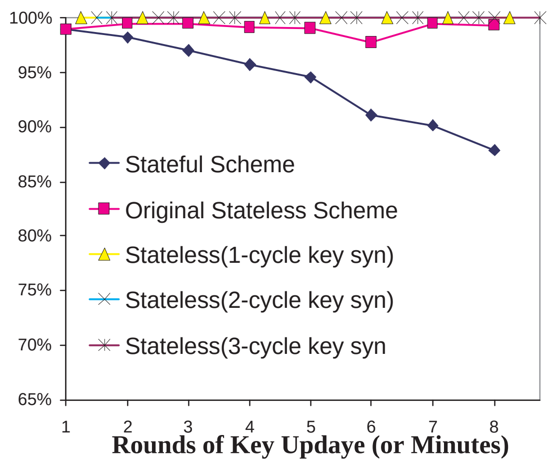


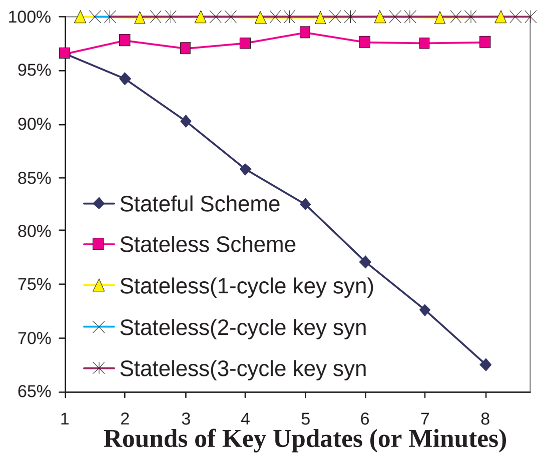
Figure 5.7: Ave-percentage of nodes received the latest key (P_{lost}=0, 80 nodes)

the original stateless key distribution schemes on the average percentage of nodes that have the latest session key after key updates. We assume the key manager updates the session key once each minute. The nodes in the proposed scheme broadcast a beacon message and synchronize session keys with neighbor nodes (if necessary) once every 15 seconds, i.e., there are 3 *cycles of key synchronization* between two consecutive key updates. Each cycle of key synchronization is assumed to complete before the next one begins. Data packets were not exchanged, so that only the effectiveness of key distribution was measured. The comparison results are shown in figures 5.5 - 5.8.

Figure 5.5 shows that if without a key synchronization scheme, the stateful group key distribution scheme has a very bad performance. After the key manager updates the session key 6 times, almost no legitimate node can get the latest session key, even though it is still connected with the key manager. Such bad performance is due to the accumulation of the nodes which missed the previous key updates. For the stateless group key distribution scheme, the average percentage of nodes which get the latest session key is close to the average percentage of the reachable nodes. This is because that the key update is only based on the correspondent key update message and independent from a node's previous statuses of key updates. However, there are still a lot of nodes (about 30%) cannot get the latest session key from the broadcast key update message. On the contrast, the proposed key synchronization scheme greatly improves the performance of the original stateless group key distribution. After one cycle of key synchronization ($15s$ later), the average percentage of nodes which get the latest session key increases from about 70% to 95%. After 3 cycles of key synchronization, almost every node (99%) receives the latest session key.

Figure 5.6 shows that even considering the factor of packet loss, the proposed key synchronization method still achieves a very good performance, i.e., about 98% nodes get the latest session key after 3 cycles of key synchronization. Figure 5.7 and Figure 5.8 compare the performances in the scenario of more nodes (80 nodes). They also confirm that the proposed key synchronization method can improve the performance of the original stateless group key distribution scheme and assist the nodes to agree on the latest session key. (In this case, almost 100% nodes get the latest session key).

Figure 5.9 shows the packet retransmission ratio (due to key un-synchronization) for various packet sending rates, given a specific source and destination pair that communicate over a 100 minute period. It shows that (i) a denser network has a lower retransmission rate due to the lower probability of key un-synchronization, and (ii) the packet retransmis-



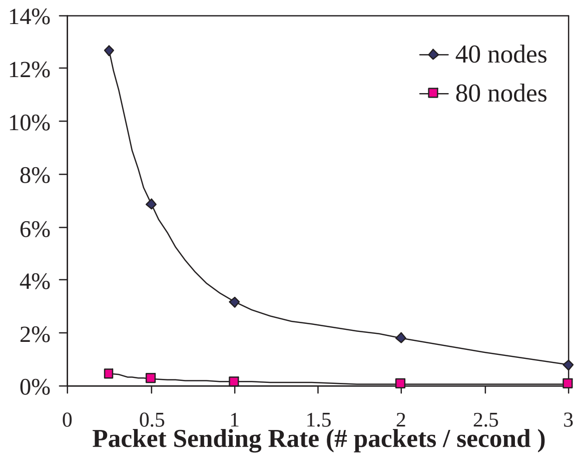


Figure 5.8: Ave-percentage of nodes received the latest key (P_{lost}=0.25, 80 nodes)

Figure 5.9: Averaged packet retransmission rate due to key un-synchronization

sion ratio decreases as the sending rate increases, since more data packets will be successfully transmitted after each key synchronization.

5.5 Implementation Test

We implemented and tested a prototype of the proposed symmetric key based network access control system at IP layer. Our implementation consists of two modules, the group rekeying module and the packet authentication module.

The group rekeying module runs in the user space. It employs the stateless group key distribution scheme proposed in [77]. This module extracts the new session key and passes it to the packet authentication module via a kernel message. It also buffers the latest key update message for future key synchronization.

The packet authentication module runs in kernel space and is based on the Netfilter [2] architecture. It uses the HMAC-MD5 authentication algorithm, and will send a signal to the group rekeying module once receiving an unverified packet. Figure 5.10 illustrates how this module is implemented using Netfilter.

We tested our implementation on a small scale test bed consisting of a DELL Pentium IV laptop running Red Hat Linux 9.0 (kernel version 2.4.20), and two COMPAQ iPAQ 3970 PDAs running Familiar v0.7.2. (kernel version 2.4.19-rmk-pxa1-hh30). All of

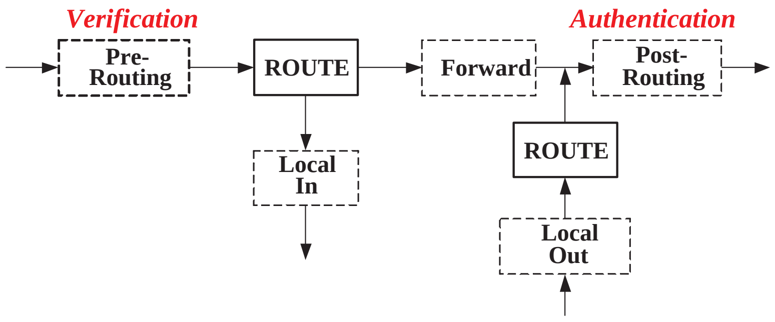


Figure 5.10: Structure of Implementation on Netfilter

these were equipped with Lucent Orinoco wireless cards. The laptop acted as the key manager and the PDAs acted as regular nodes.

We first tested the functionality of packet authentication and verification. We booted up our access control system on the laptop and a PDA A. They communicated with each other correctly, e.g., ping each other. We then used another PDA B, which did not run the access control system, to ping A. There was no ping reply from A. The log file in A showed the ping requests were dropped since they failed in packet verification.

Next, we tested the stateless feature by manually letting A miss several rounds of key updates. A was out of key synchronization with the key manager and could not communicate with it. However, after receiving the new key update message, A returned to the network immediately.

We also tested the key synchronization method. We let B also run our network access control system. However, we manually let B use an old session key, i.e., A and B were key unsynchronized. When we used B to ping A, we found A sent its buffered key update message to B. The latter updated its session key to the latest one. Clearly, it proves the proposed key synchronization method work well.

Finally, we tested the revocation mechanism by revoking A from the network. The log file in A showed it could not decrypt the news session key from the key update message any more. We used A to ping B, there was no ping reply back. This also proves A could not decrypt the new session key after the revocation. A was isolated from the network. In summary, all tests performed as expected and demonstrated that access control worked properly.

5.6 Summary

In this chapter, we have introduced a network access control system for mobile ad hoc networks based on stateless group key distribution. This system utilizes a network-wide symmetric (access control) key to filter out the packets originated from unauthorized nodes. It synchronizes the access control keys when necessary for successful delivery, with proper access control. A preliminary implementation demonstrates the proposed system is practical. We address that such an access control system can also be deployed in other dynamic network environments where the population of users and the network topology change dynamically over time.

Chapter 6

Privacy Protection - A k-Anonymous Communication Protocol

The privacy of communication has become a critical issue on the Internet. Encryption schemes protect the contents of communication, but do not conceal the fact that two users are communicating. In many situations, users may wish to make their communication anonymous. For instance, a customer placing an online order may not want his/her transactions to be traced. As another example, if the item ordered by this person can be delivered electronically (e.g., an electronic book or a digital movie), he/she may not want his/her destination address (e.g., the email account, the IP address of his/her computer) to be identified.

In the past two decades, a number of anonymous communication protocols (e.g., [8, 13, 15, 31, 39, 56, 69, 84, 100, 101]) have been proposed. Most of them originate from Chaum's two seminal approaches: mixnet [31] and DC-net [32]. The mixnet family protocols (e.g., [39, 57, 85, 100, 101]) use a set of "mix" servers that shuffle the received packets to make the communication path (including the sender and the recipient) ambiguous. As these protocols do not enforce a strict transmission pattern on each user, they rely on the statistical properties of background traffic that is also referred to as the *cover traffic* to achieve the desired anonymity, and thus cannot provide provable anonymity. As a result, these protocols are not suitable in dynamic network environments where no trusted server is available. The DC-net family protocols (e.g., [8, 32, 53, 56]) utilize secure multi-party computation techniques. They provide provable anonymity without relying on trusted third parties. However, these protocols suffer from the transmission collision problem which does not have a practical solution [56].

In this chapter, we introduce a simple and scalable anonymous communication protocol. The proposed protocol provides provable $k-$anonymity for both the sender and the recipient without transmission collision. That is, the sender and the recipient are indistinguishable from the other $k-1$ participants, where k is a predetermined parameter that can be any number between 1 and $\mathcal{N}$ (the number of participants in the network).

In the proposed protocol, the participants, which are referred to as the nodes, are organized into a set of logical rings and form an overlay network over the Internet. Within each ring, an anonymous transmission mechanism, which is the cornerstone of the proposed protocol, uses message batching and one-way key chains to make a node's message indistinguishable. It ensures (i) a node can send messages to any other node (in its ring) without disclosing identity to all nodes in the network, and (ii) a node is prevented from maliciously modifying or replaying any transmitted message in a ring.

A sender utilizes the above anonymous transmission mechanism to anonymously communicate with a recipient that may be located in a different ring. That is, the sender conceals the ID of the destination ring in which the recipient resides into its outgoing message. It sends the message to a randomly chosen agent node in its local ring, following the above anonymous transmission mechanism. This agent node extracts the message and forwards it to the corresponding destination ring without knowing the sender's identity. The forwarded message is locally broadcast in the destination ring, i.e., sent to all member nodes. The recipient thus receives the message without disclosing its identity. If each ring has at least k honest nodes, the proposed protocol provides provable $k-$anonymity for both the sender and the recipient.

The analysis shows the proposed protocol is secure under a strong adversary model, in which the adversary controls a fraction of nodes, is able to eavesdrop all network traffic and maliciously modify/replay the transmitted messages. We have completed a proof-of-concept implementation of the basic communication module and tested it on PlanetLab [4]. The test results demonstrate the proposed protocol is practical.

The rest of this chapter is organized as follows. Section 6.1 describes the nomenclature and attacks which have appeared in anonymous communication. Section 6.2 provides an overview of the related work. Section 6.3 introduces the system and threat models and some symbols used by the proposed protocol. Section 6.4 presents the proposed $k-$anonymous communication protocol in detail. Section 6.5 provides the anonymity, security and performance analysis. Finally, section 6.6 summarizes the research results.

6.1 Background

In this section, we first describe the nomenclature of anonymity defined by previous work then introduce some attacks against anonymous communication.

6.1.1 Nomenclatures

The concept of *anonymity* in information management has been discussed in early research work [94, 96, 101, 118] where anonymity is defined as the state of being not identifiable within a set of possible subjects who might take an action. We recall the definition introduced by Andreas Pfitzmann [94] where anonymity is defined in terms of unlinkability or unobservability.

Unlinkability Unlinkability of two or more entities means that, within an anonymous (communication) system, these entities are no more and no less related than they are related concerning a-priori knowledge. That is the probability of those entities being related stays the same before (a-priori knowledge) and after the run within the system (a-posteriori knowledge of the attacker).

Unobservability Unobservability is the state of the entities of interest being indistinguishable from other entities where the entities of interest can be those which are sending and receiving messages. It means that, e.g., sending messages is not discernible from random noise.

In this dissertation, we use the term “anonymity” as a synonym for “anonymity in terms of unlinkability”. Three types of communication properties: *sender anonymity*, *recipient anonymity* and *relationship anonymity* have been defined.

- Sender anonymity means that a particular message is not linkable to any sender and no message is linkable to a particular sender. That is for a transmitted (user) message m, the probability that an adversary links m to a specific user i (as the sender of m) is equal to the reciprocal of the size of the anonymity set in which the user resides:

$$\forall i \in \mathcal{N}, \quad \forall m \in \bigcup_{j \in \mathcal{N}} M_j \quad : \qquad Pr_{(m \in M_i)} = \frac{1}{|\mathcal{N}|}$$

  where $\mathcal{N}$ denotes the anonymity set (i.e., all nodes in the network) and M_j denotes the set of messages sent by user j. If an anonymous communication system can guarantee the achievement of this probability, we say it provides provable $|\mathcal{N}|$ degree anonymity to the sender.

- Recipient anonymity similarly means a particular message cannot be linked to any recipient and that to a particular recipient, no message is linkable, i.e.,

$$\forall i \in \mathcal{N}, \quad \forall m \in \bigcup_{j \in \mathcal{N}} M_j \quad : \qquad Pr_{(m \in \overline{M}_i)} = \frac{1}{|\mathcal{N}|}$$

  where $\overline{M}_j$ denotes the set of messages for which user j is the targeted recipient. Similarly, we say an anonymous communication system provides probable $|\mathcal{N}|$ degree anonymity to the recipient if it can guarantee the achievement of the above probability.

- Sender-recipient relationship anonymity (or relationship anonymity in short) is a weak property. It means that the sender and the recipient cannot be identified as communicating with each other, though it may be clear they are participating in some communications.

The above anonymities are also referred to as the *full anonymities*, since they guarantee that an adversary cannot infer anything about the sender, recipient or communication relationship of a transmitted message, from the network traffic.

k-Anonymity is a weaker guarantee of anonymity, when compared with full anonymity. Ahn, Bortz and Hopper [8] defined it as the property that an adversary was able to learn something about the sender or the recipient of a particular message, but could not narrow down its search to a set of less than k participants. In other words, k-anonymity guarantees that the adversary is not able to guess the sender or the recipient of a particular message with probability non-negligibly greater than $1/k$. Ahn, Bortz and Hopper further defined the *sender (recipient) k-anonymity* as the properties that the sender (recipient) of a transmitted message is indistinguishable from at least $k-1$ other honest nodes.

6.1.2 Attacks against Anonymous Communication

As stated earlier, by eavesdropping and analyzing the network traffic, an adversary may learn users' communication patterns such as who communicate with whom, when, how long, etc. In the following, we give a brief overview on some attacks against anonymous communication and how the proposed protocol defends against these attacks.

Replay Attacks In replay attacks, an attacker first obtains a copy of a valid message that was sent in an anonymous communication system. The attacker then sends many copies of it into the system at some later time, which will, in turn, send out many identical messages. By detecting which address receives this increase in packet traffic following the replay attack, the attacker can easily identify the intended recipient. Such replay attacks are a most serious and realistic threat if the participants in an anonymous communication system cannot distinguish and prevent the propagation of replayed packets.

In the proposed protocol a node uses a one-way key chain to authenticate the origin and the order of its packets at an intermediate (forwarding) node without disclosing the node's identity. It thus effectively prevents the replay attacks.

Contextual Attacks Normally, two entities that are communicating with each other don't "talk" at the same time. That is, when one party is sending, the other usually is silent. Therefore, via observing the communication patterns (when users send and receive), an adversary may deduce the communication relationships between the users. Furthermore, an adversary may infer the communication relationships by counting the number or frequency of packets sent and received by a user. (This might requires precise timing information). This is also referred to as a packet counting attack or frequency analysis attack. As marked by [99], contextual attacks are particularly effective for real-time interactive communication.

In the proposed protocol, the message batching makes the sender indistinguishable from other nodes in the ring, and the multicast conceals the recipient. Even though the communication pattern is publicly known (i.e., an adversary knows the rings in which the sender and the recipient reside respectively) the adversary cannot locate them with a probability significantly better than $1/k$ if each ring has at least k honest nodes. The proposed protocol thus prevents the contextual attacks at a cost of increased communication latency.

Timing Attacks An adversary may use the temporary dependency between transmissions to correlate the sniffed packets and thus trace a communication route.That is, given the set of messages coming in the network and the set of messages out of the network, an attacker could use the route timing information to correlate the messages in the two sets. This is referred to as the *timing analysis attack* or the *latency attack*.

For example, suppose there are two communication routes, one having a latency of 1 seconds and the other 3 seconds. Assume there are two messages coming in the network at 0:00 and 0:01, and two messages leave at 0:01 and 0:02. If the adversary knows the latency information of each route, he can easily correlate the incoming messages and the outgoing messages without requiring any expensive computations.

The proposed protocol uses message batching to prevent an adversary from using the temporary information of message transmission. That is, a node will not forward the received messages until it receives exactly one message from each node in the ring.

Message Tagging Attacks In message tagging attacks, two collusive attackers try to trace a communication route by tagging (e.g., slightly modify) the transmitted packets. That is one attacker tags messages in such a way that its accomplice can spot them. If the other attacker does receive the tagged messages, these two attackers then know they are in the same route and thus obtain some information about the route.

A slight variant of such message tagging attacks is that an adversary purposely delay the forwarding of packets. The adversary can thus obtain some information about the communication relationships, so long as the delays can be presumably detected.

In the proposed protocol, each node uses a unique one-way key chain to authenticate its packets at an intermediate node. Any modification of a transmitted packet will be immediately identified by the intermediate forwarding nodes. Furthermore, the message batching makes the forwarding delay introduced by the attacker indistinguishable.

Flow Marking Attacks Fu et al.,[50] discussed the degradation of anonymity in a flow-based wireless mix network under flow marking attacks. In a flow marking attack, an adversary embeds a periodic pattern of marks into traffic flows. By tracking these marks, the adversary can discover the communication relationships between users. To effectively and efficiently detect the pattern of marks, the adversary can combine frequency analysis and convert the abstract pattern of marks in the time domain to easily detect invariant frequency components.

The proposed protocol defeats the flow marking attacks by organizing the nodes with a set of logical rings and enforcing each node to follow a strict transmission pattern. A node has to use the message batch to send its messages along the local ring. As the number of messages in each message batch is predetermined (i.e., equals the number of nodes in the local ring) a node is prevented from introducing any marked traffic flow into the local ring. A node is also prevented from inserting packets or traffic flows into other rings due to the organization of rings.

Flushing Attacks Flushing attack was first mentioned in [31]. It is also known as spam attack or $n-1$ attack. In some anonymous communication systems, an intermediate node generally waiting till it has n messages before "flushing" (i.e., forwarding the messages). An adversary, therefore, can send $n-1$ messages and easily associate messages leaving the intermediate node with those having entered, and thus obtain the information about users' communication relationships.

In the proposed protocol, a transmitted message batch contains exactly one message from each node in the ring, and a node can detect whether a received message batch contains one message from each member in the ring. This prevents an adversary from inserting bogus packets into a message batch and thus defeats the flushing attacks.

6.2 Related Work

Most anonymous communication protocols originate from David Chaum's two seminal approaches: mixnet [31] and DC-net [32]. In mixnet, a sender encrypts its outgoing message and the ID of recipient using the public key of a trusted server which is called "mix". The mix decrypts the message and then forwards it to the recipient. To make the incoming and outgoing messages uncorrelatable, the mix needs to shuffle the messages and routes, e.g., to delay and reorder the messages. To deal with the possibility of compromising the single mix, mixnet has been extended in [57, 69, 95, 100] where a set of mixes are used. Recently, Möller [84] presented a provable secure public-key encryption algorithm for mixnet. This algorithm is adopted by Mixminion [39].

DC-nets [32, 118] is an anonymous broadcast protocol employing the secure multiparty sum computation method. It provides provable security in the absence of trusted parties. But, it is vulnerable to the transmission collision. That is, two or more players may transmit in the same message slot and thus no message will be delivered successfully, even when all players are honest. There is no practical solution to solve such a transmission collision [56]. Furthermore, this protocol has a poor scalability. It requires $\Omega(n^3)$ protocol messages for each transmitted anonymous message in a network of n users. Pfitzmann and Waidner [96] therefore suggested to implement superposed sending on a physical ring network. It is important to note that the ring is designed to reduce the communication overhead and prevent attackers from eavesdrop all lines, instead of using for anonymously message delivery directly. Recently, Ahn et al., [8] extended DC-net to achieve $k-$anonymity for message transmission. Golle and Juels [56] presented new DC-net constructions to detect cheating with high probability. Compared with DC-net family protocols, the protocol avoids the problem of transmission collision, and it is more efficient, i.e., $O(n^2)$ protocol messages per anonymous message, where n is the number of nodes in the ring.

Crowds [101] is based on the idea of hiding one's actions within the actions of many others, and is designed for anonymous web browsing, i.e., hiding the sender from the recipient. In Crowds, upon receiving a request, a member records from whom it received the request, and then either submits the request directly to the target server, or forwards it to another randomly chosen member. The web server, therefore, cannot determine the identity of the requestor from the received packet. However, Crowds does not hide the recipient and the packet content from the enroute nodes. Compared with Crowds, the proposed protocol provides better anonymity protection. It not only hide the sender but also conceals the recipient from all nodes in the network.

Hordes [108] is a multicast-based anonymous communication protocol derived from Crowds. It is intended to require less processing by members, and to prevent members from being able to read the replies. The sender sends a request to the destination server using randomized routing analogous to that in Crowds. The sender also includes with this request a multicast address, for a group of which it is a member. Upon receiving a request, the destination server multicasts its response to the mutlicast address provided. As a result, Hordes also aims for sender anonymity. Compared with Hordes, the proposed protocol provides better anonymous protection at a cost of increased communication overhead. Its ring-based message transmission mechanism ensures all nodes in the network cannot identify the sender and the recipient of a transmitted message.

Beimel and Dolev [13] designed a family of anonymous message delivery protocols that simulate the operation of public transportation system in digital world. That is the sender and receiver are modeled as bus stations. The pieces of information that senders send to recipients are modeled as passengers. The "buses", i.e., messages, traverse on the network, and each piece of information is allocated a seat within a bus. The basic method of BUS scheme suffers from the huge size of transmitted message, i.e., $O(N^2)$, even when the number of nodes in the network (i.e., N) is not very large. Furthermore, the seat discloses the sender's identity to the recipient. The authors suggested using random seating and cluster to reduce the number of seats and hide sender from recipient. However, this will make the recipient not able to send a reply back and pieces of information may be lost due to seating collision. The proposed protocol is somewhat similar to BUS system. That is, both of them use a logical ring structure. However, the protocol can hide the sender from the recipient without the problem of transmission collision.

Golle et al., [55] designed a universal re-encryption algorithm and thus developed a universal mixnet. This universal mixnet holds no keying information. It conceals the communication from everyone but the sender and recipient. However, it does not provide sender anonymity with respect to the recipient. A recipient can trace the a message intended for her/him throughout the mixing process. Compared with universal mixnet, the proposed protocol provides better anonymous protection. It can hide the sender from the recipient which cannot be achieved in a universal mixnet.

Tarzan [49] is a peer-to-peer anonymous IP network overlay. It achieves anonymity by a combination of multiple layers of encryption, and multi-hop routing, like mix-nets. Each node in the overlay chooses a set of mimics. The sender chooses a communication path that adversaries cannot easily influence. All nodes on the path exchange cover traffic with their mimics. This cover traffic is intended to prevent a global observer doing traffic analysis from identifying the sender and the recipient. Unfortunately, Tarzan cannot provide provable anonymity which is a nice advantage of the proposed protocol.

Sherwood et al.[106] proposed an anonymous communication protocol that is called p^5 for peer-to-peer networks. p^5 provides sender and recipient anonymity by transmitting encrypted packets at a constant rate to all participants in the network. It scales by partitioning the network into anonymizing broadcast groups. However, the broadcast groups are implemented as peer-to-peer unicast trees instead of multicast trees. The reliability and efficiency of communication, therefore, is sacrificed. The proposed protocol outperforms p^5 for provable anonymity.

TOR [43] is the second generation Onion Routing system. In TOR, each onion router maintains a TLS [41] connection to every other onion router, and it also maintains a long-term identity key and a short-term onion key. The identity key is used to sign TLS certificates, router descriptors and directories. The onion key is used to set up communication circuit and negotiate ephemeral keys. Therefore, TOR provides perfect forward secrecy, integrity checking and hidden-location service that are not addressed in the original onion routing scheme. However, TOR still cannot provide provable anonymity.

[95] is designed to provide anonymous communication service for PTT subscribers. [64] introduces a Mixes scheme for realtime application based on narrow-band ISDN (Integrated Services Digital Network). [124] presents a scheme for protecting the anonymity of message recipients in an untrusted network, relying on the use of multicast and incomparable public key cryptosystem. [109] uses formal methods to analyze probabilistic anonymity systems like Crowds and confirms that anonymity degrades with a larger crowd. [52] also analyzed the anonymity using formal methods.

It is worth noting that since the proposed protocol uses a logical ring structure, it is related to previous work on Token ring network. Token-Ring local area network (LAN) technology was developed by IBM in the early 1980s and standardized as IEEE 802.5. Stations on a Token-Ring LAN are logically organized in a ring topology with data being transmitted sequentially from one ring station to the next with a control token circulating around the ring controlling access. This technology is aimed at media access control, although it has the potential for natural anonymity in communication.

6.3 Models and Symbols

We assume the participating nodes voluntarily cooperate with each other to provide an anonymizing service. All nodes are potential originators of anonymous communication. Each node has a unique ID. For simplicity, we assume a node's ID is its IP address, and we will not distinguish between the node's ID and its IP address in the rest of the paper. Each node has a public/private key pair authenticated by a trusted authority. Two nodes can authenticate each other and establish a secure and reliable end-to-end connection by employing, for example, IPsec [66, 67, 68]. For simplicity, we assume the nodes' public keys have a unform size, e.g., 1024 bits.

We assume there is an effective broadcast/multicast mechanism in overlay networks. This broadcast/multicast mechanism could be achieved by using unicast. That is, the sender unicasts a copy of message to each member in the multicast group. Some existing multicast schemes for overlay networks may use a tree structure to improve the efficiency. It is worth noting that multicast in overlay networks is different from multicast in IP protocol. The latter uses a class D IP address to denote a multicast group and requires the cooperations from the routers. Please refer to [63] for details about multicast methods in overlay networks.

We assume the adversary can eavesdrop all network traffic. The adversary may also control a fraction of nodes in the network, and it may modify/replay/drop a transmitted message. The objective of the adversary may be to identify the sender/recipient of a given message, or to trace the end-to-end communication relationship.

Finally, we assume the cryptographic algorithms used by the proposed protocol, e.g., symmetric key encryption, public key encryption, one-way functions, and one-way key chains are secure.

Table 6.1 shows some special symbols used by the proposed protocol.

6.4 The Proposed Protocol

In the proposed protocol, the participating nodes are organized with a collection of logical ring structures. Within each ring, a transmission mechanism using message batch ensures a node can anonymously send messages to any other member node without disclosing its identity. To anonymously communicate with a recipient that may reside in another ring,

Table 6.1: Some Symbols Used in the Proposed Anonymous Communication Protocol

S_{ID}	session ID
B_{ID}	message batch ID
PK_A	the public key of node A
$PK_A\{M\}$	encryption of message M using public key PK_A
K_{A_i}	the i^{th} secret key generated by node A
$K_{A_i}\{M\}$	encryption of message M using secret key K_{A_i}
$K_{A_i}^{-1}\{M\}$	decryption of message M using secret key K_{A_i}
M_{A_i}	the i^{th} message sent by node A
MAC_{A_i}	message authentication code computed using secret key K_{A_i}
R_{AB}	one-way key chain generated by node A and assigned to node B
R_{AB_i}	the i^{th} key-chain key in key chain R_{AB}
$f()$	a secure one-way function whose output is 1024-bit long
KF_i	key chain update flag with value i, where $i = 0$ means the flag is not set
CK	commitment of key chain update. Contains a node ID and a commitment of new key chain if KF is set. Otherwise, it is a random value
RF	a receiver flag
N_i	a random number

a sender follows the transmission mechanism and sends its messages to a randomly selected agent node in the local ring. These messages contain an ID of the destination ring in which the recipient resides. If the destination ring is the local ring, i.e., the sender and the recipient are in the same ring, the agent node broadcasts the messages locally, i.e., to all member nodes in the ring. Otherwise, it forwards the messages to a node in the destination ring. The latter broadcasts the received messages locally. The recipient thus receives the messages without disclosing its identity. If each ring has at least k honest nodes, as we will show through our analysis, the proposed protocol will provide provable k-anonymity protection to both the sender and the recipient.

In the following subsections, we will present the proposed protocol in detail. We first introduce its node entry and network topology control mechanism, which ensures the rings are of approximately equal size and prevents the colluding attackers from taking over a local ring. Next, we describe the transmission mechanism that allows a node to anonymously send messages to other member node in the local ring. This transmission mechanism is the cornerstone of the proposed protocol. It conceals the sender of a transmitted message from other nodes in the local ring, and it uses one-way key chains to prevent the reply attacks and the packet modification attacks. After that, we show the solution that allows two arbitrary nodes to communicate anonymously. This solution is based on the presented transmission mechanism and multicast. Finally, we discuss the update of key chains used by the nodes to anonymously identify the origin and the order of their messages in the proposed protocol.

6.4.1 Node Entry and Network Topology Control

The proposed protocol utilizes a hierarchical structure to organize the network, as illustrated in figure 6.1. The participating nodes are organized into small (logical) rings. Each ring is uniquely identified by a ring ID, and has at least k nodes, where k is a predetermined parameter representing the least degree of anonymity provided by the ring. The nodes in a ring fall into two categories, *regular node* and *super node*. A regular node is a node that has no direct connection to the nodes in other rings, while a super node is one that provides the message forwarding service across rings. Regular nodes rely on the local super node(s) to forward packets across rings. A super nodes also has the responsibility for local ring management, such as accepting new nodes. Each ring may have multiple super nodes in order to avoid a single point of failure. In the following, we will present the mechanisms on nodes joining and leaving, super nodes selection, rings splitting and merging.

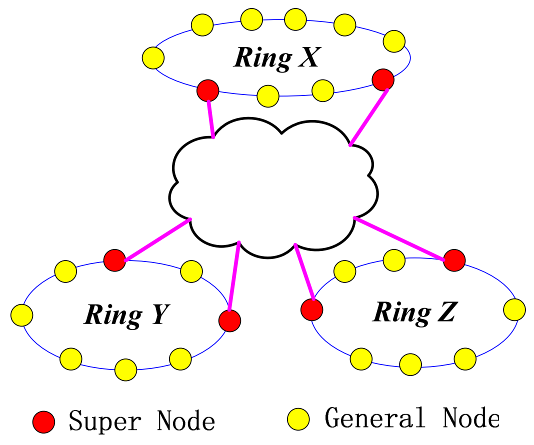


Figure 6.1: An illustration of network topology of the proposed protocol.

User Joining and Leaving

To join the overlay network, a new node first contacts any existing node (e.g., by broadcasting a request to find such an existing node) or a centralized *directory server* (if there is such a server). For simplicity, we assume there is a directory server. This server redirects the new node to a corresponding super node based on the *admission rule*. That is the lower v bits of the hash value of a member node's ID should equal those of the ring ID, where v is the number of bits used for indicating the ring ID. Such a grouping mechanism is used in many peer-to-peer systems, e.g., [53, 104, 113]. It ensures the rings are of approximately equal size in the proposed protocol, and it also prevents attackers from crowding out all but one honest node in a targeted ring.

The new node then sends its ID and public key to the corresponding super node. This super node adds the new node into the local ring if the latter satisfies the admission rule, while the latter's position in the ring is unimportant for purposes of anonymity and may be determined purely randomly. The super node next broadcasts an update message informing all members (including the new node) about the new ring structure, the nodes' public keys, and some parameters used in the ring, e.g., current session ID, uniform message size and encryption algorithm, where a session is defined by the change of local ring structure, e.g., node joining or leaving. This broadcast message is digitally signed by the super node to prevent it from being maliciously modified.

In order to prevent the super node from maliciously transmitting different ring structure information to different node in the ring, a Byzantine agreement method as discussed in [103, 128] is adopted. That is once receiving a new update message from the super node, a node digitally resigns this message and then forwards (i.e., rebroadcasts) it to every other node in the ring. As a result, each node in the ring will ideally receive $n-1$ copies of (digitally signed) update message, if we ignore the possibility of packet loss. If considering the packet loss, each node then has to receive $2t+1$ update messages, where t is the tolerant number of malicious nodes in the ring. By checking whether the received update messages are consistent with each other, a node can identify the malicious nodes (including the super node) who try to publish faked update message, since each message is digitally signed by both the super node and the forwarding node.

Once proving the received update message correct and finding the new node satisfies the admission rule, each node in the local ring then updates its stored information about ring structure, nodes' public keys and other parameters. As a result, the new node is granted admission into the overlay network.

A node may leave or suddenly fail during the anonymous data transmission. Therefore, the ring has to be repaired once a node leaving or failure is reported or detected. In the proposed protocol, a node is responsible for detecting whether its previous and next hop nodes (in the ring) are still online, e.g., no ACK message received from the next hop node for the transmitted TCP packets implies a potential node offline. Once a node finds its previous or next hop node, saying node P_i, is offline, it broadcasts a node failure notification to the rest nodes in the ring. The latter double check whether node P_i is really offline, e.g., by sending a ping message. If the checking result is positive, i.e., node P_i is really offline, a node broadcasts a signed confirm message in the ring. By using the Byzantine approach, all remaining nodes will agree on whether node P_i is really offline. The corresponding super node then can update the ring structure as that of node joining.

Super Node Selection

The selection mechanism of super nodes has been discussed in earlier literature [28, 78, 113, 128]. In the proposed protocol, we use the method presented in [28] which is also adopted by the Steward project [9]. That is, the nodes take the position of super node in the order of their IDs. For instance, the two nodes with the lowest IDs will be automatically elected as the super nodes first and all nodes set their local view number to 1. When these two super nodes are suspected faulty or a timeout is triggered, the nodes increase their local view number and broadcast *new-presentative* messages, which contains their view numbers. Upon receiving a set of $2t+1$ *new-presentative* messages proposing the same view number, the two nodes with the second lowest ID will be elected as the super nodes, and so on. Please refer to [9, 28] for details.

Ring Splitting and Merging

The size of each ring and the total number of rings can grow and shrink over time as the nodes join and leave. Once a ring has 2ϑ or more nodes, it will split into two rings, where each new ring has at least ϑ nodes and ($\vartheta \geq k$). It is worth noting that the nodes can pre-calculate the size of each new ring after a potential split, based on current ring's structure (i.e., node list) and the admission rule. The split only happens when each new ring will have at least ϑ nodes. We will discuss the value of ϑ in later section. The IDs of the two new rings and their members can be decided as follows:

1. Increase the number of bits denoting the IDs of new rings by one (i.e., $\upsilon = \upsilon + 1$);
2. The ID of one new ring ID is the original ring ID following a bit 0, and the other ring ID is the original ring ID following a bit 1. For example, the (old) Ring 4 will split into two rings, a (new) Ring 4 and a Ring 12.
3. The nodes calculate the list of member nodes of each new ring based on their stored information of the original ring and the admission rule as described earlier. They then join the new ring and vote for the new super node(s).

The elected super nodes establish connections with other super nodes in the network and inform the latter about the ring split, i.e., the IDs of the new rings and the new super nodes. That is the elected super nodes either send an update message endorsed by the ring members to the directory server if it exists or broadcast such an update message in the overlay network. In the later case, the receiving super nodes may have to use Byzantine agreement method to reach a consistent view about the network topology and the list of super nodes. Please refer to [103, 128] for the details of Byzantine agreement methods in peer-to-peer environments. For convenience, we call two rings sister rings, if they share the same $\upsilon - 1$ bits from the most significant bits in their ring IDs.

A ring needs to merge with its sister ring once the number of member nodes drops below ϑ. Correspondingly, the number of bits denoting the ID of the merged ring reduces one, i.e., setting the most significant bit to zero, the nodes vote for new super nodes, and the new super nodes inform all other super nodes about the mergence of rings.

It is worth noting that in the procedure of ring split and merge, (i) the members will not be able to anonymously transmit data to other nodes, as the old ring structure is unavailable now, and (ii) the old super nodes need to forward the packets received from other rings into the local ring. So that a node can still receive data without disclosing its identity but temporarily cannot send data out.

6.4.2 Anonymous Transmission in a Local Ring

The anonymous transmission mechanism in a local ring is the cornerstone of the proposed protocol. It effectively hides the sender of an anonymous message from the other nodes including the super node forwarding this message. In this subsection, we present the details of this transmission mechanism: the transmission initialization and the message transmission along the ring and the detection of adversaries that do not follow this transmission mechanism. We defer the description of the solution that allows two arbitrary nodes to anonymously exchange messages to the next subsection.

Transmission Initialization

A ring needs a two-stage initialization once a new session is triggered, where a session is defined by the event of ring structure changing, e.g., node joining and leaving. The purpose of such an initialization is two-fold. First, the nodes need to set up a message batch that contains exactly one message from each node in the ring without disclosing the sender's identity of each message. A node's message is thus mixed with the other nodes' messages in the message batch. Second, each node secretly assigns one unique one-way key chain to each of the other nodes in the ring without disclosing its identity. These key chains are used to identify the order of received messages and thus prevent the message replay attacks. They also allows a node to verify whether the received message batch contains exactly one message from each node in the ring. The later analysis will show these key chains will not affect the anonymity provided by the proposed protocol.

After the two-stage initialization, a node can send data to any other node in the ring anonymously by replacing its old message in the received message batch. Figures 6.2 and 6.3 illustrate the procedure of transmission initialization.

Stage One: Message Batch Construction To start the initialization, a specific super node that is referred to as the *starting node* (e.g., the super node A in figure 6.2) sends an initialization packet (P_{A1}) to its next hop in the clockwise direction of the ring (referred to as the *ring direction*). This initialization packet is signed by node A. It consists of a n-slot key vector and an encrypted dummy message, where n is the number of nodes in the ring. Each slot conceals a secret key, a session ID and a message authentication code. For instance, the first slot is $PK_0\{K_1, S_{ID}, MAC_1\}$ and the i^{th} slot is $K_{i-1}\{...K_1\{PK_i\{K_i, S_{ID}, MAC_i\}\}...\}$. The dummy message is used to for two purposes: (i) to construct the message batch, and

(ii) to anonymously assign each member node a unique one-way key chain used in later anonymous communication. It has a standard size and a strict format, i.e., a n-slot key vector plus the anonymizing payload, where the elements in a key vector slot of the dummy message are different from those in the key vector slot of the initialization packet. Such standard message size and message format are two public parameters of the ring and are applied to all transmitted messages in the ring. We will discuss the message format and how the key chains are distributed later.

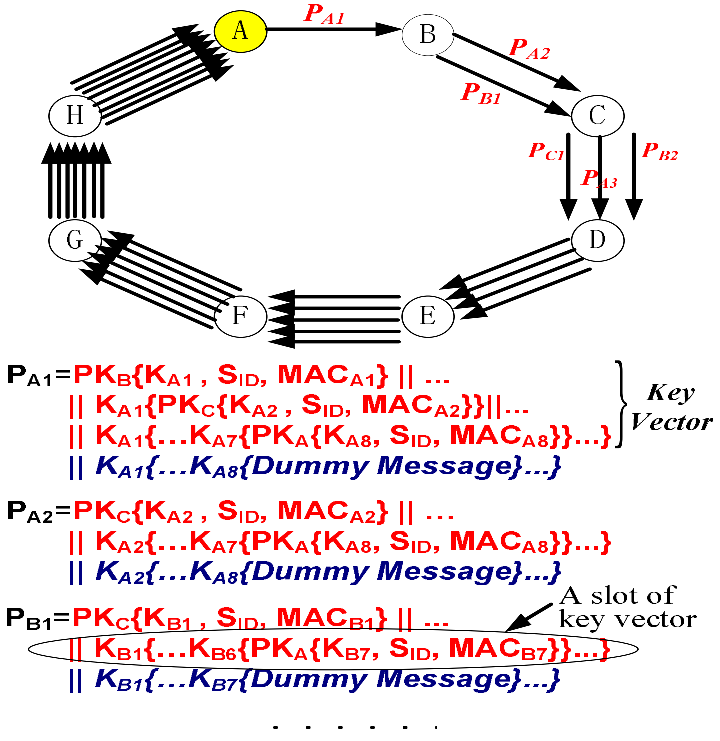

A
B
C
D
E
F
G
H
P_{A1}
P_{A2}
P_{B1}
P_{C1}
P_{A3}
P_{B2}
$P_{A1}=PK_B\{K_{A1}, S_{ID}, MAC_{A1}\} \| \ldots$
$\| K_{A1}\{PK_C\{K_{A2}, S_{ID}, MAC_{A2}\}\}\|\ldots$
$\| K_{A1}\{\ldots K_{A7}\{PK_A\{K_{A8}, S_{ID}, MAC_{A8}\}\}\ldots\}$
Key Vector
$\| K_{A1}\{\ldots K_{A8}\{Dummy\ Message\}\ldots\}$
$P_{A2}=PK_C\{K_{A2}, S_{ID}, MAC_{A2}\} \| \ldots$
$\| K_{A2}\{\ldots K_{A7}\{PK_A\{K_{A8}, S_{ID}, MAC_{A8}\}\}\ldots\}$
$\| K_{A2}\{\ldots K_{A8}\{Dummy\ Message\}\ldots\}$
A slot of key vector
$P_{B1}=PK_C\{K_{B1}, S_{ID}, MAC_{B1}\} \| \ldots$
$\| K_{B1}\{\ldots K_{B6}\{PK_A\{K_{B7}, S_{ID}, MAC_{B7}\}\}\ldots\}$
$\| K_{B1}\{\ldots K_{B7}\{Dummy\ Message\}\ldots\}$
. . . . . .

Figure 6.2: An illustration of the construction of a message batch. The sender precalculates the MACs in reverse order, e.g., MAC_{A8} first, then MAC_{A7}, and so on. The MAC in a key vector slot is calculated based on the secret key concealed in this slot and the rest of the packet assuming this slot becomes the first slot. For example MAC_{A2} is computed based on K_{A2} and $K_{A2}\{PK_D\{K_{A3}, S_{ID}, MAC_A3\}\}..||K_{A2}\{..\{DummyMessage\}..\}$. The digital signature is not shown in the figure.

Once receiving the initialization packet from the starting node, the immediate downstream node (e.g., node B in figure 6.2) uses its private key to decrypt the first key vector slot. It thus gets the secret key, session ID and message authentication code concealed in this slot. The node then checks the packet signature, decrypted session ID, and message authentication code to ensure the correctness and integrity of the packet, while the message authentication code is calculated based on the concealed secret key and the rest of the packet. If the checking result proves correct, the node removes the signature and the first key vector slot, and it uses the secret key to decrypt the remaining slots and the dummy message. After that, the node sends the changed initialization packet and a new one generated by itself to the next hop in a random order. These two initialization packets are digitally signed by the node, and they have the same number of key vector slots (i.e., $n-1$) and packet size.

The other member nodes in the ring follow the same steps, i.e., after receiving d initialization packets from the previous hop, decrypt each of them, remove the first key vector slot, and then forward $d+1$ initialization packets to the next hop at a random order, where d is the number of hops between a node and the starting node. As a result, the number of transmitted initialization packets increases by one at each hop, while the size of the packet decreases gradually. Finally, the starting node receives n dummy messages that have uniform size. These n dummy messages form the message batch.

Detection of Packet Modification: A node may get a negative result when it checks the MAC of a received initialization packet. The attacker could be the previous hop node who modifies the packet or the packet's original sender who intentionally conceals a wrong MAC. To detect the attacker, the receiver, e.g., node B in figure 6.2, immediately reports an error about the received packet, i.e., broadcasting the received initialization packet and the decrypted K_{A1}, S_{ID} and MAC_{A1}. The other nodes can verify the error report because (i) P_{A1} is signed by node A and thus is non-repudiated, and (ii) the encryption of (broadcast) K_{A1}, S_{ID} and MAC_{A1} using node B's public key should equal the first key vector slot of P_{A1}. Node B's previous hop (node A) then needs to prove it forwarded the packet correctly, i.e., to show a received packet whose forwarding result should be P_{A1} and the decryption of the first key vector of this packet. If node A cannot show the corresponding packet, it is identified as the attacker. Otherwise, the detection continues until one upstream node cannot prove it forwarded the packet correctly. This node is identified as the attacker. The nodes revoke the attacker from the ring and restart the transmission initialization.

Lemma 6.1 *A polynomial time adversary cannot link an honest node with a specific (received) initialization packet or vice versa if more than one of the received initialization packets originated from honest nodes.*

Proof Suppose an honest node A receives l initialization packets while only one of them (M_0) originated from an honest node (suppose node B). Node A decrypts M_0 with a secret key K concealed in the first key vector slot of M_0. The resulting new packet is denoted as M_1. Node A sends M_1 and its own initialization packet M_2 (at a random order) to the next hop. Since K is encrypted using node A's public key, it is computationally infeasible for a polynomial time adversary $\mathcal{A}$ to calculate K from M_0. Without knowing K, $\mathcal{A}$ cannot link M_0 with M_1 with a probability significantly better than 1/2 (i.e., random guessing), recalling the semantic security of symmetric key encryption. In other words, $\mathcal{A}$ cannot link node A with M_2 or link node B with M_1 with a probability significantly better than 1/2, even $\mathcal{A}$ can link node B with M_0. Similarly, if there are r received initialization packets originated from different honest nodes, $\mathcal{A}$ cannot successfully identify the sender of each (honest) initialization packet with a probability significantly greater than $1/r$. □

Stage Two: Message Batch Checking and Key Chain Distribution As an intermediate node may maliciously replace the received initialization packets in stage one, the nodes need to verify whether the resulting message batch contains exactly one dummy message from each node in the ring. Therefore, the nodes need to assign a unique key chain to each member node secretly in stage two. These key chains are used to identify the origin and the order of the nodes' messages without disclosing the latter's identities. They not only prevent the potential replay attacks in later anonymous communication but also allow a node to verify whether a received message batch contains exactly one message from each member node in the ring. Figure 6.3 illustrates the procedure of message batch checking and key chain distribution by showing the propagation of node E's dummy message without losing universality.

The starting node (e.g., node A in figure 6.3) forwards the message batch (formed at stage one) to the next hop, while each message is digitally signed by the starting node. As stated earlier, a dummy message and the later transmitted anonymous data message consist of a n-slot key vector and the fixed-lengthed anonymizing payload. Each key vector slot has a standard size decided by the key length of the employed public key encryption algorithm. As illustrated in figure 6.3, each slot contains a receiver ID, a session ID (S_{ID}), a recipient flag (RF), a key chain update flag (KF), a new key-chain key (R_n) in the assigned key chain, a key chain update commitment (CK), a one-time secret key (K_i), a random number (N_i), and a message authentication code (MAC_i).

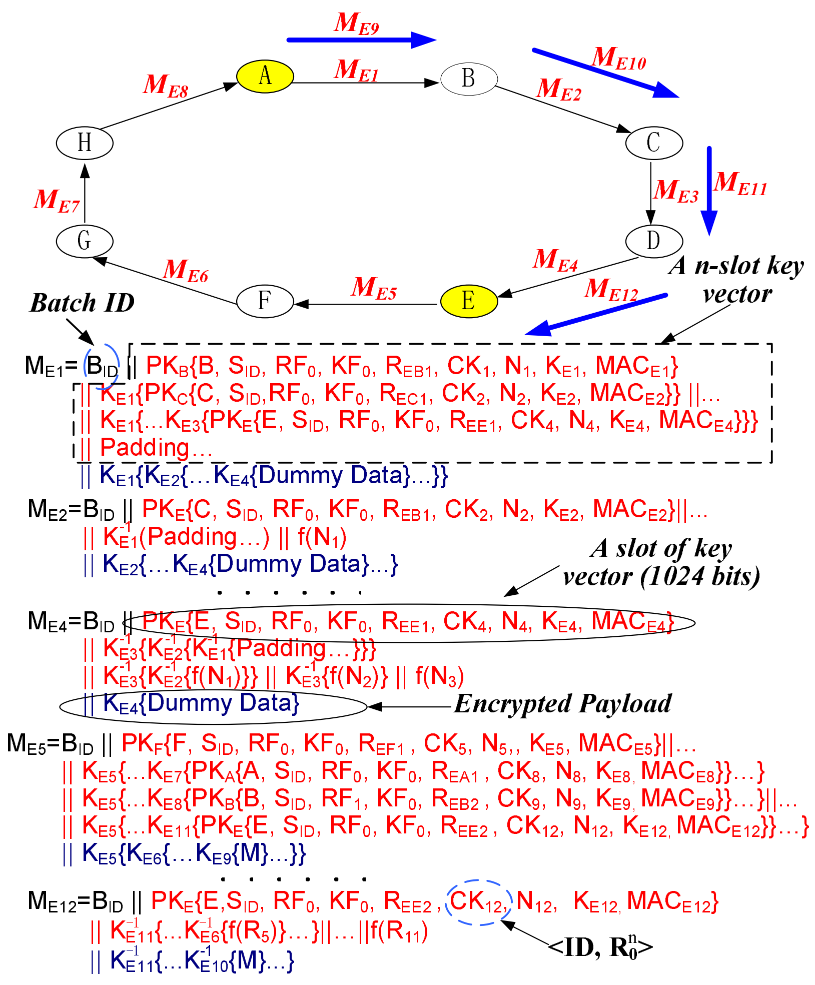

M_{E9}
M_{E10}
M_{E1}
M_{E2}
M_{E8}
A
B
C
H
M_{E3}
M_{E11}
M_{E7}
G
D
M_{E6}
M_{E4}
A n-slot key vector
M_{E5}
M_{E12}
F
E
Batch ID
$M_{E1}= B_{ID} \,||\, PK_B\{B, S_{ID}, RF_0, KF_0, R_{EB1}, CK_1, N_1, K_{E1}, MAC_{E1}\}$
$||\, K_{E1}\{PK_C\{C, S_{ID}, RF_0, KF_0, R_{EC1}, CK_2, N_2, K_{E2}, MAC_{E2}\}\} \,||\ldots$
$||\, K_{E1}\{\ldots K_{E3}\{PK_E\{E, S_{ID}, RF_0, KF_0, R_{EE1}, CK_4, N_4, K_{E4}, MAC_{E4}\}\}\}$
$||$ Padding…
$||\, K_{E1}\{K_{E2}\{\ldots K_{E4}\{\text{Dummy Data}\}\ldots\}\}$
$M_{E2}=B_{ID} \,||\, PK_E\{C, S_{ID}, RF_0, KF_0, R_{EB1}, CK_2, N_2, K_{E2}, MAC_{E2}\}||\ldots$
$||\, K_{E1}^{-1}(\text{Padding}\ldots) \,||\, f(N_1)$
$||\, K_{E2}\{\ldots K_{E4}\{\text{Dummy Data}\}\ldots\}$
A slot of key vector (1024 bits)
$M_{E4}=B_{ID} \,||\, PK_E\{E, S_{ID}, RF_0, KF_0, R_{EE1}, CK_4, N_4, K_{E4}, MAC_{E4}\}$
$||\, K_{E3}^{-1}\{K_{E2}^{-1}\{K_{E1}^{-1}\{\text{Padding}\ldots\}\}\}$
$||\, K_{E3}^{-1}\{K_{E2}^{-1}\{f(N_1)\}\} \,||\, K_{E3}^{-1}\{f(N_2)\} \,||\, f(N_3)$
$||\, K_{E4}\{\text{Dummy Data}\}$
Encrypted Payload
$M_{E5}=B_{ID} \,||\, PK_F\{F, S_{ID}, RF_0, KF_0, R_{EF1}, CK_5, N_5, K_{E5}, MAC_{E5}\}||\ldots$
$||\, K_{E5}\{\ldots K_{E7}\{PK_A\{A, S_{ID}, RF_0, KF_0, R_{EA1}, CK_8, N_8, K_{E8}, MAC_{E8}\}\}\ldots\}$
$||\, K_{E5}\{\ldots K_{E8}\{PK_B\{B, S_{ID}, RF_1, KF_0, R_{EB2}, CK_9, N_9, K_{E9}, MAC_{E9}\}\}\ldots\}||\ldots$
$||\, K_{E5}\{\ldots K_{E11}\{PK_E\{E, S_{ID}, RF_0, KF_0, R_{EE2}, CK_{12}, N_{12}, K_{E12}, MAC_{E12}\}\}\ldots\}$
$||\, K_{E5}\{K_{E6}\{\ldots K_{E9}\{M\}\ldots\}\}$
$M_{E12}=B_{ID} \,||\, PK_E\{E, S_{ID}, RF_0, KF_0, R_{EE2}, CK_{12}, N_{12}, K_{E12}, MAC_{E12}\}$
$||\, K_{E11}^{-1}\{\ldots K_{E6}^{-1}\{f(R_5)\}\ldots\}||\ldots||f(R_{11})$
$||\, K_{E11}^{-1}\{\ldots K_{E10}^{-1}\{M\}\ldots\}$
$<ID, R_0^n>$

Figure 6.3: An illustration of message batch checking and key chain distribution. We assume M is encrypted data. The digital signature of each message is not shown in the figure. Clearly, the sender precalculates the MACs in a reversed order, e.g., in message M_{E5}, node E computes MAC_{E12} first, then MAC_{E11}, and so on.

When a node receives a message in the message batch, e.g., node B receives M_{E1}, it decrypts the first key vector slot and checks the integrity of M_{E1} based on the value of each field in the slot. If the checking result proves negative, node B starts the detection of modification as described in stage one. Otherwise, node B continues processing M_{E1}. From the receiver flag RF, node B knows whether it is the targeted receiver. As the transmission initialization is not finished yet, the RF flag should not be set, i.e., node B is not the receiver. Node B also knows it is not the original sender of M_{E1} since the key-chain key R_{EB1} is not the expected one (i.e., R_{BB1}) generated by itself, without knowing the sender's identity as proved in Lemma 6.1.

Node B buffers R_{EB1} (node E thus secretly assigns a key chain R_{EB} to node B without disclosing its identity). It uses K_{E1} to decrypt the rest key vector slots and the encrypted payload, replaces the first key vector slot with $f(N_1)$, rotates the slots one step leftwards (in order to keep the message size unchanged), removes the previous signature, and signs the changed message (i.e., M_{E2}).

Node B processes the other $n-1$ received messages similarly and forwards the message batch (i.e., the processed messages) to the next hop in a random order. It is worth noting that one of the n received messages must originate from node B itself, i.e., the decrypted key-chain key should equal R_{BB1}. Node B replaces this message with a new one. If there is no such a message, it means node B's previous initialization packet was maliciously replaced by some node in the ring. Node B should stop processing the message batch and immediately start the detection of packet replacement which will be discussed later.

After that, node B generates a commitment of assigned key chains. This commitment is signed by node B. It contains the current session ID, the current batch ID, a batch counter whose value is set to 0, and the n decrypted key-chain keys $\{R_{AB1}, R_{BB1}, ..., R_{HB1}\}$. Node B broadcasts the commitment locally. Once a node receives node B's commitment, it checks whether this commitment includes the key-chain key it assigns to node B. If true, the node buffers this commitment. Otherwise, it means the node's previous initialization packet/message was replaced by some node. The victim node starts the detection of packet replacement.

Detection of Packet Replacement: The victim node, e.g., node B, asks all nodes in the ring to broadcast their buffered packets/messages. By emulating the propagation of its previous initialization packet/dummy message and comparing the emulation results with the broadcast packets/messages at each hop, node B can locate the adversary. It next reports the adversary by broadcasting its input packet/message at the adversary and the decryption result of the first slot of this input packet/message. For instance, if P_{B1} was maliciously replaced by node C in figure 6.2, i.e., node C sent P'_{B2} instead of P_{B2} to the next hop, node B broadcasts P_{B1}, K_{EB1}, S_{ID} and MAC_{B1}. The other nodes can verify the authority of the report as presented previously. They calculate the correctly forwarded packet P_{B2} based on P_{B1} and K_{EB1}. If they cannot find P_{B2} in node D's received packets, the nodes revoke node C from the ring and restart the transmission initialization.

We notice that the key chain used by a node to identify its own messages might collide with the one assigned by another node, e.g., $R_{BBi} = R_{EBi}$. Such a collision may mean that a sender cannot correctly recognize its own messages and thus breaks the anonymous communication. We address that the probability of the collision is negligible according to the birthday paradox. For instance, if the length of a key-chain key R_{XY_i} is large ($\geq$160 bits) and the number of nodes in a ring is small (< 100), the probability of collision will be less than 10^{-20}. Furthermore, the sender can use the encryption key or the MAC embedded in the slot to identify its own message in the case of a key chain collision, at a cost of increased storage.

When the starting node (i.e., node A) receives the message batch once more and no adversary is reported, the transmission initialization is done. The resulting message batch contains exactly one message from each node in the ring, and all nodes have gotten the assigned key chain correctly. The nodes flush their previously buffered initialization packets (i.e., they only buffer the most recently received message batch). The stating node forwards the message batch to its next hop. The nodes in the ring thus can start the anonymous data transmission that will be discussed next.

Lemma 6.2 *If a message batch contains more than one message originating from honest nodes then a polynomial time adversary cannot link an assigned key chain to a specific (honest) node or vice versa.*

Proof Given an assigned key chain (i.e., a key-chain key R_0) decrypted from a received (honest) dummy message M. Since R_0 (and later key-chain key R_i) does not contain any information about the sender, a polynomial time adversary $\mathcal{A}$ has to use M to identify the corresponding sender. As proved in Lemma 6.1, if a message batch contains more than one message originating from honest nodes then $\mathcal{A}$ cannot link M with a specific (honest) node or vice versa. Therefore, $\mathcal{A}$ also cannot link R_0 with a specific (honest) node with a probability significantly better than random guessing. □

Message Transmission along the Ring

In this part, we show how a sender uses the message batch constructed as above to anonymously send a message to a node in the local ring. When a node receives the message batch from its previous hop, it first increases the local counter (i.e., t, which is used to record the number of message batch as received in current session) by one. The node then decrypts the first key vector slot of each received message using its private key. It next checks whether the hash values of the decrypted key-chain keys (R_{XY_i}) one-to-one match with the previously buffered ones (i.e., whether the message batch contains exactly one message from each node in the ring). For instance, $\mathcal{H}(R_{EB2})$ in figure 6.3 should equal the previously buffered R_{EB1}. Such a checking also prevents the modification or replay attacks. If the checking result proves negative, it means there is an adversary who either maliciously modified/replayed the transmitted packets or set a trap for the node, e.g., node E in figure 6.3 encrypts $R^{'}_{EB2}$ instead of the correct R_{EB2} in message M_{E5}. In this circumstance the node starts the detection of adversary procedure.

Detection of Adversary: when the hash value of decrypted key-chain key R_{XY_i} in a received message, M, does not match with any (previously) buffered ones, the node broadcasts a report about M. This report contains M and the decryption of $M's$ first key vector slot. The other nodes can verify the authority of the received report, i.e., the decryption is correct based on the node's public key and $\mathcal{H}^t(R_{XY_i})$ should but does not equal any element in the node's (previously broadcasted) key chain commitment. Next, the previous hop node should prove that M is a forwarded form of its received message M^*, i.e., broadcasting a report about M^*. If the previous hop node cannot show a valid report, it means it is the adversary. Otherwise, the detection continues until one upstream node cannot show a valid report. This upstream node is identified as the adversary who is the sender of M, recalling that each key vector slot contains a MAC and thus any malicious modification of a transmitted message will be discovered by the next hop node of the adversary. The nodes revoke the adversary from the ring and start a new session. Similarly, if multiple received messages have the same decrypted key-chain keys, the node broadcasts theses messages and the decryption results of their first key vector slots. The other nodes detect the adversary as just presented, i.e., the upstream nodes need to prove they forwarded the received message batch correctly.

If the hash values of decrypted key-chain keys one-to-one match with the buffer values, the node updates its buffered key-chain keys with the new ones. It next verifies the integrity of each received message using the decrypted MAC. If the MAC of a message proves incorrect, it means either the message was modified by the previous hop or the message's sender has set a trap. The node starts the detection of packet modification as presented above. Otherwise, the node updates its previously buffered messages with the received ones and continues processing the received messages.

For each message in the received message batch, the node knows whether it is the original sender of this message based on the key-chain key R_{XY_i} decrypted from the first key vector slot. If the node is the sender, it replaces the message with a new one as illustrated by M_{E5} in figure 6.3. Otherwise, the node uses the decrypted secret key (K) to decrypt the rest key vector slots and the encrypted payload. If the RF flag in the first key vector slot is set, the node extracts the payload, as it is the targeted receiver. The node then replaces the first key vector slot with $f(N_i)$, rotates the slots one step leftwards, removes the previous signature, and re-signs the changed message. After that, the node sends the processed messages to the next hop at a random order.

Lemma 6.3 *Given all nodes in the ring publish their key chain commitments correctly, it is computationally infeasible for an adversary to successfully modify/replay an (honest) node's message. As a result, the message batch contains exactly one message from each node in the ring.*

Proof Assume an adversary $\mathcal{A}$ modifies/replays an (honest) node's message $\mathcal{M}_0$ in the received message batch, i.e., it forwards $\widetilde{\mathcal{M}_1}$ instead of the correct one $\mathcal{M}_1$ to the next hop (honest) node $\mathcal{B}$, and node $\mathcal{B}$ accepts $\widetilde{\mathcal{M}_1}$. Due to the one-to-one relationships between the decrypted key-chain keys and the previously buffered ones, the first key vector of $\widetilde{\mathcal{M}_1}$ must contain the key-chain key R_j that matches with previously buffered R_{j-1}. The order of key-chain keys in the one-way key chain ensures the received $\widetilde{\mathcal{M}_1}$ is a new message, i.e., not a replayed one. The one-way property of the one-way function $\mathcal{H}$ makes it computationally infeasible to calculate R_j from R_{j-1}, even if $\mathcal{A}$ knows R_{j-1}. Also, R_j and the other variables are encrypted using $\mathcal{B}$'s public key, it is computationally infeasible for $\mathcal{A}$ to calculate R_j from $\mathcal{M}_1$ without knowing the corresponding private key, recalling the semantic security of the public key encryption. Thus, the first key vector of $\widetilde{\mathcal{M}_1}$ must be same as that in $\mathcal{M}_1$. As this first key vector contains a message authentication code of $\mathcal{M}_1$, the remaining key vectors and the encrypted payload in $\widetilde{\mathcal{M}_1}$ must be same as those in $\mathcal{M}_1$, assuming the authentication mechanism is secure. As a result, $\widetilde{\mathcal{M}_1}$ must equal $\mathcal{M}_1$, otherwise, $\mathcal{B}$ will detect the violation of message authentication. This contradicts with the assumption. It, therefore, proves an adversary cannot succeed in message modification and replay. As $\mathcal{B}$ has an assigned key chain from each node in the ring, the received message batch thus contains exactly one message from each node in the ring, if it passes the verification. □

Lemma 6.4 *The message replacement made by the sender is computationally indistinguishable, if the ring has more than one honest nodes.*

Proof Suppose an honest node S receives a message batch $\mathfrak{B}$ and $\mathfrak{B}$ passes all verifications. As proved in Lemma 6.2 and Lemma 6.3, $\mathfrak{B}$ contains exactly one message from each node in the ring, and a polynomial time adversary $\mathcal{A}$ cannot link S with S's input message M_o (in $\mathfrak{B}$) with a probability better than random guessing. Given S replaces M_o with M_n in its forwarded message batch, it is computationally infeasible for $\mathcal{A}$ to tell whether M_n is the decryption of M_o using some unknown secret key K, even if $\mathcal{A}$ may be able to link M_o with M_n as the messages originated from the same sender, assuming the semantic security of symmetric key encryption and the security of one-way function $f()$. Therefore, $\mathcal{A}$ cannot link S with M_n with a probability better than random guessing either. The message replacement made by S is thus computationally indistinguishable. □

6.4.3 Anonymous Communication between Two Arbitrary Nodes

In previous subsection, we present the mechanism which allows a node to anonymously send messages to another node in the same ring. In this subsection, we show how to use this mechanism to support the anonymous communication between two arbitrary nodes in the network.

To anonymously communicate with a recipient that may reside in a different ring, the sender embeds the ID of the *destination ring* into the anonymizing payload of its message. The destination ring is the ring in which the recipient resides. The sender then anonymously sends this message to a (randomly selected) local super node as presented in previous subsection, i.e., by setting the flag RF in the corresponding key vector slot.

Once receiving the message, the selected local super node extracts the destination ring ID. If the destination ring is the local ring, i.e., the sender and the recipient are in the same ring, the super node broadcasts the message locally. i.e., sends each member node in the local ring a copy of the message. Otherwise, it forwards the message to a super node in the destination ring. The latter broadcasts the received message to all nodes in its ring. The recipient thus receives the message. As the sender's message is indistinguishable from the messages sent by other nodes in the message batch and the local super node that forwards the message is not necessary the sender, the sender is thus hidden among other nodes in its ring. Similarly, the transmitted message is received by all nodes in the recipient's ring, the recipient is thus indistinguishable from the other honest nodes in its ring.

Anonymous File Sharing

A variant of anonymous communication is anonymous file sharing in which the sender and the recipient not only need to be hidden from the other nodes in the network, but also need to be concealed from each other. The proposed protocol does support such anonymous file sharing, as the sender only needs to know the ID of the ring in which the recipient resides, instead of the recipient's identity.

The sender, i.e., file requester, sends its query to a local super node using the anonymous message transmission described above. This query contains the information about the desired file, the sender's ring ID and a random query ID. Once upon receiving the query, the local super node broadcasts this query in its local ring. It also forwards the query to all other super nodes in the network. The other super nodes broadcast the received query in their local rings. As the result, all nodes in the network receive the query. But they cannot link this query with a specific node.

When the recipient, i.e., file provider, receives the query, it sends a reply message to a local super node along the ring anonymously. This reply message includes the sender's ring ID, the query ID, the recipient's ring ID, and a random reply ID generated by the recipient. The super node forwards the reply message to a corresponding super node based on the sender's ring ID in the reply message. The latter broadcasts the reply message in its local ring. The file requester thus receives the (broadcast) reply message without disclosing its identity and knowing the identity of file provider.

Transmission Rate Adjustment

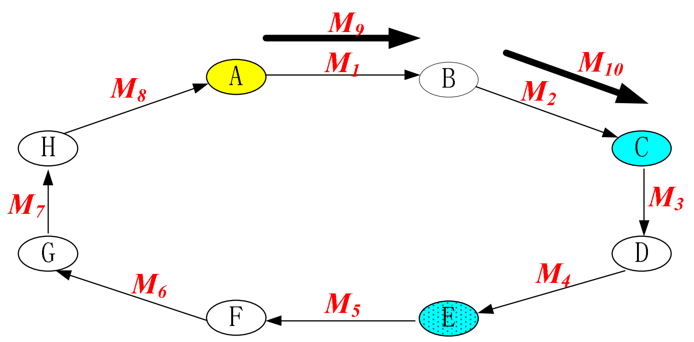
A node may want to change the transmission rate, once it has an amount of data to transmit or after it finishes the data transmission. The proposed protocol supports such a transmission rate adjustment.

To increase the transmission rate, a node first anonymously sends a request to the local super node that is responsible for transmission initiation. This request contains a random request ID (e.g., r). The anonymous message transmission mechanism in the ring ensures only the node and the corresponding super node know this random ID. Once receiving the request, the super node buffers the request ID r and initiates another message batch, if the number of message batches in the ring does not reach a (predetermined) maximum value. As a result, the transmission rate increases.

Once the node finishes data transmission, it sends another request to the corresponding super node in order to decrease the transmission rate. This request contains the previous random request ID r. After receiving the request, the super node checks whether the received r matches the previously buffered one. Such a check prevents other nodes from maliciously decreasing the transmission rate. If the check result proves correct, the super node broadcasts a message to stop one message batch. The transmission rate thus decreases. The super node can also actively decrease the transmission rate, if it finds no data message forwarded/broadcasted in the local ring for a while, i.e., the node may "forget" to send a request to decrease the transmission rate. Analogously, the message size in a ring can also be adjusted dynamically at a cost of increased communication overhead.

6.4.4 Key Chain Update

A node needs to update its key chain assigned to another node (or itself) and inform all other nodes about the key chain update, once the old key chain runs out. Such a key chain update should be anonymous, i.e., without disclosing the node's identity. A simple solution is that the node requests the local ring to start a new session. However, it is not a good solution as it breaks down other nodes' normal communication. We introduce an alterative solution that allows a node to update its key chains without interrupting others' communication or compromising the anonymity. Figure 6.4 illustrates this solution where we assume node A wants to update its key chain assigned to node C.



$$\begin{aligned}
M_1 = B_{ID} \parallel\ & PK_B\{B, S_{ID}, RF_0, KF_0, R_{ABi}, CK_0, N_1, K_1, MAC_1\} \\
& \parallel K_1\{PK_C\{C, S_{ID}, RF_0, \boxed{KF_1}, R_{ACi}, \boxed{<C, R^n_{AC0}>}, N_2, K_2, MAC_2\}\} \\
& \parallel K_1\{K_2\{PK_D\{D, S_{ID}, RF_0, \boxed{KF_2}, R_{ADi}, <C, R^n_{AC0}>, N_3, K_3, MAC_3\}\}\} \\
& \parallel K_1\{\ldots K_3\{PK_D\{E, S_{ID}, RF_1, KF_2, R_{AEi}, <C, R^n_{AC0}>, N_4, K_4, MAC_4\}\}\} \\
& \parallel \ldots \\
& \parallel K_1\{\ldots K_7\{PK_A\{A, S_{ID}, RF_0, KF_2, R_{AAi}, <C, R^n_{AC0}>, N_8, K_8, MAC_8\}\}\} \\
& \parallel K_1\{K_2\{\ldots K_4\{ Data\}\ldots\}\}
\end{aligned}$$

$$\cdots\cdots$$

$$\begin{aligned}
M_9 = B_{ID} \parallel\ & PK_B\{B, S_{ID}, RF_0, KF_2, R_{AB(i+1)}, <C, R^n_{AC0}>, N_9, K_9, MAC_9\} \\
& \parallel K_9\{PK_C\{C, S_{ID}, RF_0, \boxed{KF_0}, \boxed{R^n_{AC1}}, CK_0, N_{10}, K_{10}, MAC_{10}\}\} \parallel \ldots \\
& \parallel K_9\{\ldots K_{11}\{PK_D\{E, S_{ID}, RF_1, KF_0, R_{AD(i+1)}, CK_0, N_{12}, K_{12}, MAC_{12}\}\}\} \\
& \parallel \ldots \\
& \parallel K_9\{\ldots K_{15}\{PK_A\{A, S_{ID}, RF_0, KF_0, R_{AA(i+1)}, CK_0, N_{16}, K_{16}, MAC_{16}\}\}\} \\
& \parallel K_9\{\ldots K_{12}\{ Data\}\ldots\}\}
\end{aligned}$$

Figure 6.4: An illustration of key chain update, where node A updates the key chain assigned to node C and simultaneously sends data to node E. $R^n_{XY_i}$ denotes a key-chain key from the new key chain.

To update the key chain assigned to node C, in its outgoing message M_1, node A sets the flag KF (in the key vector slot that will be decrypted by node C using its private key) to one, sets the KFs in the following slots to two, and fills the corresponding CK fields. Once node C receives M_2, it extracts the new key chain commitment CK from the first key vector. It then broadcasts a new commitment of assigned key chains. This commitment contains the current session ID and batch ID, the value of node's batch counter, the $n-1$ key-chain keys decrypted from currently received message batch, and $R^n_{AC_0}$ and the last key-chain key in the old key chain which are specially marked. Node C then forwards the processed message batch to the next hop, noting that node C cannot successfully modify any forwarded message as stated previously.

After receiving the broadcast commitment, a node can verify if the other $n-1$

key-chain keys (except the marked $R^n_{AC_0}$) one-to-one match with the ones in node C's previously broadcast key chain commitment, based on the embedded batch counters in these two commitments, the nodes' positions in the ring, and also its local batch counter. If false, it means node C is an attacker that maliciously broadcasts a wrong commitment for assigned key chains. The nodes revoke node C from the ring and start a new session.

Otherwise, the node buffers this newly received commitment, waits for the coming message batch to verify $R^n_{AC_0}$, and then replaces its buffered node C's (old) commitment with the new one. That is in the coming message batch, the KF flag decrypted from the first key vector slot of a message should equal 2, the ID and the key-chain key in the corresponding CK should equal C and $R^n_{AC_0}$ respectively. Otherwise, either the original sender (i.e., node A) set a trap or node C maliciously broadcasted a wrong commitment, recalling it is computationally infeasible to successfully modify a forwarded message. The node needs to detect and revoke the adversary from the ring.

Detection of Malicious Key Chain Update: If a node (suppose node F) finds the decrypted CK from message M_5 does not match with the newly broadcast commitment (either wrong ID or wrong key-chain key), while the KF flag is set to 2, it broadcasts a report that contains M_5 and the decryption result of M_5's first key vector slot. The other nodes can verify this report and then start the detection of the malicious sender, i.e., the upstream nodes need to prove they forwarded messages correctly as presented previously while the decrypted KFs in their received messages are equal to 2. Similarly, if the node finds no decrypted KF equals 2, it has to ask its previous hop from which message it should get the decrypted KF equaling 2, and then it starts the detection of malicious sender. For instance, node F in figure 6.4 finds no decrypted KF equals 2. It sends an inquiry to previous hop node E. The latter points out message M_5, as the decrypted KF in its received message M_4 equals 2. Node F then broadcasts a report that contains the feedback from node E and the decrypted result of the first key vector slot of M_5, showing the decrypted KF in M_5 does not equal 2. The upstream nodes then need to prove they forwarded the message correctly as presented above.

A node X may find the decrypted KF in a received message M equals 2 while the corresponding node Y did not broadcast a updated commitment. In such a scenario, the attacker can be node Y who refuses to broadcast the new commitment or the original sender of M who intently sets a trap, recalling it is computationally infeasible for an adversary to successfully modified a forwarded message. Node X starts the detection of attacker by broadcasting a report that contains M and the decryption result of M's first key vector slot. The upstream nodes then have to prove they forwarded the message correctly. As a result, the nodes in the ring will catch the attacker.

Lemma 6.5 *The update of key chains does not help an adversary to identify the sender of a transmitted message, given the ring has more than one honest node and each honest node independently updates its key chains assigned to itself and other nodes.*

Proof Suppose there are only two honest nodes, B and C, in the ring. Node C broadcasts an updated key chain commitment which is triggered by an honest node. A polynomial time adversary $\mathcal{A}$ receives a data message M (even if the KF flag in M is set to 2). Let $Pr\{Trigger = A\}$ denote the probability that this key chain update is triggered by node B and $Pr\{M = B\}$ denote the probability that $\mathcal{A}$ successfully links M with its sender B. Given node B and node C independently update their key chains, $\mathcal{A}$ will get $Pr\{Trigger = A\} = Pr\{Trigger = B\} = 1/2$, as we have proven $\mathcal{A}$ cannot link node B (or node C) with a specific key chain with a probability significantly better than random guessing. Since an (honest) node's key chain update is independent from $\mathcal{A}$'s guess about the message sender, we can get $Pr\{M = B\} = Pr\{M = B|Trigger = B\} + Pr\{M = B|Trigger = C\} = 1/2 + 0 = 1/2$ (i.e., random guessing). Similarly, we can prove if there are k honest nodes in the ring and there are multiple rounds of key chain updates, the adversary $\mathcal{A}$ still cannot link an honest node with a specific message or vice versa with a probability significantly better than random guessing, given that each honest node independently updates its key chains. □

6.5 Analysis

6.5.1 Anonymity

Theorem 6.1 *If a ring has at least k honest nodes, the proposed protocol provides sender k-anonymity.*

Proof As proved in Lemma 6.3, a transmitted message batch contains exactly one message from each node in the ring, and the one-way key chains secretly assigned by the nodes prevent an adversary from maliciously modifying/replaying a transmitted message in the received message batch. Recall that we use semantically secure public key encryption and secret key encryption. Given that a node randomly puts its own message into $n-1$ other forwarded messages, if there are at least $k-1$ other honest nodes, an adversary and the nodes in the local ring (including the local super node that is selected to forward the node's message), cannot distinguish the node's message from the other $k-1$ messages originating from honest nodes. Within a session, a node is able to link two messages transmitted in two different message batches to the same sender by using the assigned one-way key chains. However, the node cannot identify the sender, as it cannot link an assigned one-way key chain with a specific node as proved in Lemma 6.3. Therefore, a node cannot distinguish the sender of a message from the other $k-1$ honest nodes, even it accumulates a set of transmitted message batches. Lemma 6.4 and Lemma 6.5 further prove that the adversary cannot identify the sender by observing message replacements and key chain updates. $\square$

Theorem 6.2 *If every ring has at least k honest nodes, the proposed protocol provides recipient k-anonymity.*

Proof The message is broadcasted locally in the ring where the recipient resides. All nodes in the ring receives the message. Since there are at least k honest nodes in ring, the adversaries cannot distinguish these nodes as the recipients. $\square$

Anonymity Degree k and Ring Size

The value of anonymity degree k is an important parameter in the proposed protocol. In general, k can be any number between 1 and $\mathcal{N}$, while k equals $\mathcal{N}$ means the proposed protocol provides full anonymity. However, as pointed out in [8], 2-anonymity would be enough to cast reasonable doubt, thus invalidating a criminal charge in the United States legal system, and 3-anonymity would be enough to invalidate a civil charge in the absence of other evidence. A small value of k, therefore, is sufficient enough.

The minimum ring size ϑ is determined by the anonymity degree k. It can be $\frac{2k}{1-\beta}$ as presented in [8], where β is the fraction of nodes that are compromised by the adversary. Each ring thus will have at least k honest nodes with a very high probability.

6.5.2 Security

Adversaries may attempt to trace the senders or the recipients by watching traffic patterns or message encodings. Such attacks are referred to as the traffic analysis attacks. A survey on traffic analysis attacks can be found in [11] and [99]. In the proposed protocol, the nodes' traffic patterns are publicly known. That is, their messages propagate around the rings. Since a transmitted message batch contains exactly one message from each node in the ring and the messages are re-encrypted hop-by-hop using secret keys extracted from the messages themselves, adversaries cannot link an honest node with a specific message in the message batch or vice versa, as shown in Lemmas 6.2 to 6.5. Therefore, even given that the traffic patten and message encoding algorithm are publicly known, adversaries cannot distinguish the sender and the recipient of a transmitted message from the other $k-1$ honest nodes in the local ring.

The adversaries may maliciously modify, replay or tag a transmitted message in order to locate the sender or the recipient. As shown in Lemma 6.3, the proposed protocol effectively prevents these attacks by employing message pre-authentication and one-way key chains. Also, the data payload of a transmitted message is encrypted using a one-time secret key. Therefore, even if the same data payload is transmitted multiple times, the adversary cannot link them without knowing the encryption keys.

An adversary may drop some messages which is supposed to forward. In the proposed protocol, a message batch contains exactly one message from each node in the ring. A node knows how many messages in a message batch it should receive from its previous hop. If a node drops some messages, it will be immediately identified by the next hop node. We notice that the adversary who drops a message batch can deny the receipt of message batch. To thwart such an attack, the super node can re-organize the ring structure once the drop of message batch is identified, i.e., by changing the nodes' positions in the ring (e.g., to put the node who denies receipt of message batch as its next hop). If a suspect has been reported by a fraction of nodes (e.g., over 1/3 nodes, considering the Byzantine fault tolerant method only allows less than 1/3 malicious nodes in a group) after the ring re-organization, this suspect then can be identified as an attacker. If there is a message board system in the network (all nodes sequentially extract the message batches from the board and then put their forwarded message batches back to the board) such a message dropping attack can be prevented completely.

As a node's joining and leaving will interrupt current anonymous communication in the ring and trigger a new round of transmission initialization, an adversary may attempt to destroy the proposed protocol by frequently joining and leaving the network. The proposed protocol cannot prevent such an attack. However, the influence of the attack is localized to a ring, and it is relatively easy to identify the attacker, as the attacker will join in the same ring with high probability due to the node admission rule.

An adversary that is elected as a super node may refuse forwarding messages across the rings and thus block the anonymous communication between the sender and the recipient. It is hard to detect such an attack if the sender does not have the capability to monitor all network traffic. The proposed protocol employs multiple super nodes in a ring to mitigate such an attack. The sender can try another super node, if it suspects its previous message might not be forwarded by a given super node. Therefore, unless the adversary compromises all super nodes in the ring, the sender still be able to communicate with the recipient anonymously. As the adopted selection mechanism of super node [28] ensures all nodes have equal chance to be elected, it will be a high probability of at least one honest super node in the ring.

We notice that a malicious super node may send faulty ring configuration information to each node to ruin the establishment of the ring and the anonymous communication. This attack can be prevented by adopting the Byzantine method discussed in [103] at a cost of increased communication overhead and the constrain that less than 1/3 nodes in a ring can be malicious. Furthermore, a malicious super node may refuse to accept a new node. The proposed protocol cannot prevent such an attack. However, as there are multiple super nodes in a ring, the new node can contact another super node as discussed previously.

6.5.3 Efficiency

The proposed protocol requires a sender's message to sequentially traverse through all nodes in the local ring. It thus increases the communication overhead and the average data latency. Furthermore, the key vector in each transmitted message and the dummy messages transmitted by the nodes in the ring present another sources of communication overhead. In terms of message (or bit) complexity, i.e., the messages (or bits) transmitted in the network for every anonymous message (bit) sent, as stated in previous work [8, 13], the communication overhead of the proposed protocol is $O(n^2)$, where n is the number of participants in a ring.

In the proposed protocol, each node has to buffer the latest message batch it received, and save n key chain commitments and n key-chain keys. It also needs to store n one-way key chains that it assigns to itself and other nodes in the ring, if it does not want to calculate the key-chain keys online when sending messages. Jakobsson [62], and Coppersmith and Jakobsson [35] proposed schemes to improves the performance of one-way key chain, which requires $O(log(L))$ storage and $O(log(L))$ computation to access an element, where L is the length of the key chain. The storage overhead at each node, therefore, is $O(n \times log(L))$ stored key-chain keys and $O(n)$ buffered messages.

The proposed protocol also imposes heavy computation overhead on each node. A node has to decrypt every received message using its private key. It also has to verify the digital signature of each received message and generate a digital signature for every forwarded message (by signing the whole message batch instead of each individual message, the computation overhead caused by this part can be significantly reduced). As the message batch contains exactly one message from each node in the ring, the computation overhead at each node is $O(n)$ public key operations.

6.5.4 Comparison with Some Existing Schemes

We briefly compared the proposed protocol with some existing anonymous communication protocols, e.g., Onion routing [100], BUS [13], Crowds [101], Mixminion [39] and $k-$anonymous message transmission [8]. The comparison results can be summarized as follows.

Compared with the mixnet-liked protocols, Onion Routing, Crowds and Mixminion, the proposed protocol provides provable anonymity to both the sender and the recipient. It does not rely on the statistical properties of background traffic. It can hide the sender and the recipient from each other, and support the detection of active adversaries without compromising the anonymity. However, the proposed protocol may have larger communication latency, since a node has to wait until receiving a message batch before being able to send a data message out, while the above mixnet-liked protocols allow a sender to independently decide when to send a message.

Compared with BUS, the proposed protocol is more efficient, $O(n^2)$ vs $O(N^3)$, where N is number of nodes in the network. It allows the recipient to reply without

Table 6.2: A Brief Comparison with Some Existing Protocols

	Sender anonymity			Recipient anonymity		Provable Anony-mity	Overhead$^{⋇}$	Latency
	External Nodes†	Internal Nodes††	Recipient	External Nodes†	Internal Nodes††			
OR [115]	Yes	Maybe[1]	No	Yes	Maybe[2]	No	$O(d)$	$O(d)$
BUS [13]	Yes[3]	Yes[3]	?[4]	Yes[3]	Yes[3]	Yes	$O(N^3)$	$O(N)$
Crowds [101]	Yes	Maybe[1]	No	Maybe	No	No	$O(d)$	$O(d)$
Mixminion [39]	Yes	Maybe[1]	Optional	Yes	Maybe[2]	No	$O(d)$	$O(d)$
KAMT [8]	Yes	Yes	Yes	Yes	Yes	Yes	$O(n^2)$	#
Proposed	Yes	Yes	Yes	Yes	Yes	Yes	$O(n^2)$	$O(n)$

where d is the number of enroute onion routers or intermediate nodes; N is the number of nodes in the network; n is the number of nodes in the subgroup or the ring.

† Nodes not en route. †† Nodes en route

⋇ Measured in terms of message complexity at network level, assuming nodes always have data to transmit.

# The latency depends on the implementation of broadcast. It requires 3 rounds of broadcast.

[1] The first mix may guess the sender with some confidence.

[2] The last mix may deduce the recipient with some confidence.

[3] Not satisfy the anonymity defined by [94]. Adversaries only cannot conclude if a message contains data.

[4] The authors suggested using random seating to achieve it. This prevents a recipient from replying.

[5] The latency depends on the implement of broadcast.

knowing the sender's identity and supports anonymous broadcast. Furthermore, it provides a method for users to detect active adversaries without compromising the anonymity.

Compared with $k-$anonymous message transmission, the proposed scheme provides the same degree of anonymity with the same level of communication overhead. But, it avoids the transmission collision.

To summarize, compared with the existing anonymous communication protocols, the proposed one provides better anonymous protection, or provides the same degree of anonymity with the same or less communication overhead, under a strong adversary model that colluding adversaries can eavesdrop all network traffic and even modify/replay the transmitted packets.

6.5.5 Implementation Evaluation

We have implemented the basic communication module of the proposed protocol in C on Linux planform in order to validate the feasibility, assuming the rings have been constructed and the nodes know each other's public key. We adopted RSA public key encryption algorithm with 1024-bit key size and Blowfish secret key encryption in CBC mode with 160-bit key size and 64-bit initialization vector. We did not use one-way key chains in this implementation. The cryptographic operations were implemented by the OpenSSL [3] library and no assembly language was used.

We tested this basic component on the PlanetLab [4]. In the tests, we assume the rings have been established, the super nodes have been decided, the nodes know each other's public key (for simplicity we manually let all nodes use the same public/private key for the purpose of reducing code length. Such a simplification will not effect the evaluation result of the proposed protocol), and the nodes agree on the parameters used in the local ring, e.g., ring topology. Each node establishes a TCP connection with its previous hop node and its next hop node. The sender and the recipient are located in different rings, where each ring has variant number of nodes. The broadcast is simulated by unicast, i.e., the local super node unicasts the data message to the recipient directly. Table 6.3 shows the average processing time of a message batch, the estimated maximum bandwidth for anonymous communication, and the average end-to-end communication delay, for 100 rounds of test.

Table 6.3: Processing Time of Message Batch and Latency (Across Rings)

Number of Nodes in the Ring	Payload Size	Processing Time of a Message Batch†	Estimated Maximum Bandwidth	End-to-End Latency with anonymity††	Resulting Throughput
4	128Bytes	23.31ms	42.9Kbps	62.39ms	15.7Kbps
	512Bytes	23.34ms	171.4Kbps	71.09ms	56.3Kbps
5	128Bytes	29.17ms	34.3Kbps	76.73ms	13.0Kbps
	512Bytes	29.46ms	135.8Kbps	85.90ms	46.6Kbps
6	128Bytes	35.12ms	28.5Kbps	97.16ms	10.3Kbps
	512Bytes	35.14ms	113.8Kbps	104.81ms	38.2Kbps

† Run on a P4 2.2Ghz machine with 512M memory under Redhat Fedora 2.

†† The sender is one-hop away from its super node, i.e., total 3 hops between the sender and the recipient.

As illustrated by table 6.3, the proposed protocol introduces a significant communication latency if compared with that of direct communication between sender and recipient without anonymity. Our tests showed that the latency of direct point-to-point communication was around 17.1ms, while the end-to-end latency in the proposed protocol was several times larger (the value of this latency depended on the number of nodes in the ring and the sender's position in the ring).

The bandwidth for anonymous communication in the proposed protocol is only tens of Kbps. The throughput degrades as the number of nodes in the ring increases. Such a performance even is not as good as the one of low-latency anonymous communication services like Freedom [21] or Tarzan [49]. Furthermore, like the previous DC-net family protocols, a node in the proposed protocol has to wait (in this case, until receiving a message batch) before being able to send a data message out, while the previous mixnet-liked protocols allow a sender to independently decide when to send messages. These are tradeoffs between performance and anonymity.

6.6 Summary

Privacy has become a critical issue in electrical communication, especially in the dynamic network environments where no trusted server is available. In this chapter, we introduced a k-anonymity communication protocol for overlay networks. This protocol is based on the construction of logical rings and the deployment of multicast, instead of relying on any trusted server. It ensures the sender and the recipient are indistinguishable from the other $k-1$ honest nodes and also hides them from each other, if each ring has at least k honest nodes. Compared with the existing protocols, the proposed protocol provides better anonymous protection, or provides the same degree of anonymity with the same or less communication overhead. The analysis shows the proposed protocol is secure even under a strong adversary model.

Chapter 7

Conclusion and Future Work

This dissertation has investigated multiple research problems related to securing communication in dynamic network environments. Dynamic network environment means the set of network participants may change unpredictably over time and the network topology may evolve consequentially. How to protect users' communication in such dynamic network environments is a challenging issue, if compared with that in a static system, in which the fixed network topology and the predetermined user memberships can be used to secure users' communication.

A number of solutions have been proposed in the area of securing communication, e.g., the Diffie-Hellman key exchange protocol and some secure group communication protocols. However, these solutions are not sufficient in dynamic network environments, considering the dynamics of user memberships and network topology.

This thesis focuses on two particular classes of dynamic networks, i.e., mobile ad hoc networks and overlay networks, and concentrates on three security concerns, i.e., (i) confidentiality and integrity, (ii) access control, and (iii) communication privacy. It provides new solutions to address these security concerns separately.

This dissertation first introduces an IP address auto-configuration scheme for mobile ad hoc networks, since a precondition of network communication is that each user should have a unique identifier (address). In the proposed address auto-configuration scheme, each node first randomly generates a public/private key pair. It then uses the hash value of the public key as its IP address, if this address is not being used by another node. A node thus can authenticate itself using its public key. Such a self-authentication greatly thwarts the attacks to which the previous address auto-configuration schemes were vulnerable.

Next, this thesis presents two storage-efficient stateless group key distribution schemes. These two schemes are developed to efficiently distribute a common group key to a dynamic group, e.g., nodes in an overlay network, with less storage requirement at each user. The nodes can use the common secret key to protect the secrecy and integrity of their communication against illegitimate users.

Third, this thesis investigates the method using the proposed stateless group key distribution scheme to enforce network access control in mobile ad hoc networks. A proof-of-concept implementation and the test results on some wireless devices demonstrate the effectiveness of the proposed method.

Finally, this thesis introduces an anonymous communication protocol to protect the privacy of users' communication relationships in dynamic network environments, in which users cannot trust each other. In the proposed anonymous communication protocol, network participants (nodes) are organized with a set of logical rings. Each node uses message batching to make its own message indistinguishable from other transmitted messages. If each ring has at least k honest nodes, this proposed anonymous communication protocol will provide provable $k-$anonymity for both the sender and the recipient. Compared with the existing schemes that either cannot provide provable anonymity or suffer from transmission collision, this proposed anonymous communication protocol provides better protection, or provides the same degree of anonymity with the same or less communication overhead, under a strong adversary model that colluding adversaries can eavesdrop all network traffic and modify the transmitted packets.

However, the schemes presented in this dissertation, especially the proposed anonymous communication protocol, heavily rely on the public key operations. They may not be suitable in some dynamic network environments where each participant has constrained computation power, e.g., wireless sensor networks. How to develop a lightweight alternative, therefore, will be one direction for future work. A possible starting point is to relieve the assumptions. If we can assume all network participants come from the same organization, we may be able to use (pre-distributed) pairwise keys and other lightweight authentication methods (such as Merkle Hash Tree) to protect the users communication.

Furthermore, as stated at the beginning of this dissertation, the proposed schemes are based on current Internet protocols. They cannot provide an all-in-one solution protecting users' communication in dynamic network environments. How to integrate these schemes into the design of next generation Internet, in order to provide a secure and comprehensive solution, will be another direction for future work.

Bibliography

[1] *http://user.it.uu.se/∼henrikl/aodv.*

[2] `http://www.netfilter.org`.

[3] Openssl. `http://www.openssl.org`.

[4] Planetlab. `http://www.planet-lab.org`.

[5] The homepage of Gnutella. `http://gnutella.wego.com`.

[6] The homepage of KaZaa. `http://www.kazza.com`.

[7] The homepage of Skype. `http://www.skype.com`.

[8] L. Ahn, A. Bortz, and N. Hopper. K-anonymous message transmission. In *Proceedings of the 10th ACM conference on Computer and Communications Security*, pages 122–130, Washington D.C., USA, 2003.

[9] Y. Amir, C. Danilov, D. Dolev, and *et al.* "steward: Scaling byzantine fault-tolerant systems to wide area networks". In *Proceedings ofthe International Conference on Dependable Systems and Networks*, Philadelphia, PA, 2006.

[10] K. Chan ans S. Chan. Key Management Approaches to Offer Data Confidentiality for Secure Multicast. *IEEE Network*, 17:30–39, Sept-Oct 2003.

[11] A. Back, U. Möller, and A. Stiglic. Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems. In *Procings of 4th Information Hiding Workshop*, Pittsburgh, PA, 2001.

[12] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti. Secure Pebblenets. In *Proceedings of the Internation Symposium on Mobile Ad Hoc Networking and Computing*, Long Beach, CA, 2001.

[13] A. Beimel and S. Dolev. Buses for Anonymous Message Delivery. *J. Cryptology*, 16:25–39, 2003.

[14] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA.

[15] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A System for Anonymous and Unobservable Internet Access. *Lecture Notes in Computer Science*, pages 115–129, 2001.

[16] S. Blake, D. Black, M. Carlson, and *et al.* An Architecture for Differentiated Services. *RFC 2475, IETF*, 1998.

[17] C. Blundo and A. Cresti. Space Requirements for Broadcast Encryption. *Advances in Cryptology-EUROCRYPTO'94*, LNCS950:287–298, 1994.

[18] C. Blundo, P. D'Arco, and et al. Design of Self-Healing Key Distribution Schemes. *Design, Codes, and Cryptography, N. 32, 2004.*

[19] C. Blundo, L. A. Frota Mattos, and D. R. Stinson. Trade-offs Between Communication and Storage in Unconditionally Secure Schemes for Broadcast Encryption and Interactive Key Distribution. *CRYPTO'96*, pages 387–400, 1996.

[20] J. Boleng. Efficient Network Layer Addressing for Mobile Ad Hoc Networks. In *Proceedings of the International Conference on Wireless Networks*, Las Vegas, NV, 2002.

[21] P. Boucher, A. Shostack, and I. Goldberg. Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., December 2000.

[22] M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. *Advances in Cryptology-EUROCRYPT'94*, LNCS950:275–286, 1995.

[23] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communication & Mobile Computing: Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):83–502, 2002.

[24] R. Canetti, P. Cheng, F. Giraud, D. Pendarakis, J. R. Rao, P. Rohatgi, and D. Saha. An IPSec-based Host Architecture for Secure Internet Multicast. In *Proceedings of the Network and Distributed Systems Security Symposium*, San Diego, CA, 2000.

[25] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited. In *Proceedings of 30th Annual ACM Symposium on the Theory of Computing*, Dallas, TX, 1998.

[26] R. Canetti, T. Malkin, and K. Nissim. Efficient Communication-Storage Tradeoffs for Multicast Encryption. In *Proceedings of EUROCRYPTO*, pages 459–474, 1999.

[27] R. Canetti, D. Micciancio, and O. Reingold. Perfectly One-Way Probabilistic Hash Functions. In *Proceedings of 30th Annual ACM Symposium on the Theory of Computing*, Dallas, TX, 1998.

[28] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002.

[29] A. Cavali and J. Orset. Secure Hosts Auto-Configuration in Mobile Ad Hoc Networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops*, Tokyo, Japan, 2004.

[30] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha. Key Management for Secure Internet Multicast Using Boolean Function Minimisation Technique. In *Proceedings of IEEE INFOCOM*, New York, NY, 1999.

[31] D. Chaum. Untraceable Electrical Mail, Retrun Address, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[32] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *J. Cryptology*, 1:65–75, 1988.

[33] W. Chen and L. Dondeti. Performance Comparison of Stateful and Stateless Group Rekeying Algorithms. In *Proceedings of the 4th International Workshop on Networked Group Communication*, Boston, MA, 2002.

[34] W. Cheswick and S. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker.* Addison-Wesley, 1994.

[35] D. Coppersmith and M. Jakobsson. Almost Optimal Hash Sequence Traversal. In *Proceedings of International Conference on Financial Cryptography*, Southampton, Bermuda, 2002.

[36] F. Dabek, M. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area Cooperative Storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, Chateau Lake Louise, Banff, Canada, 2001.

[37] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer-Verlag, ISBN 3540425802, 2002.

[38] W. Dai. `http://www.eskimo.com/~weidai/benchmarks.html`.

[39] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, 2003.

[40] Y. Desmedt and Y. Frankel. Threshold Cryptosystems. *Advances in Cryptology - Crypto '89*, LNCS 435:307–315, 1990.

[41] T. Dierks and C. Allen. The TLS protocol - Version 1.0. *IETF RFC2246*, Jan 1999.

[42] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[43] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, 2004.

[44] D. Dolev, N. Lynch, S. Pinter, E. Stark, and W. Weihl.

[45] L. R. Dondeti, S. Mukherjee, and A. Samal. Survey and Comparison of Secure Group Communication Protocols. Technical report, University of Nebraska-Lincoln, June 1999.

[46] R. Droms. Dynamic Host Configuration Protocol. *RFC 2131, IETF*, 1997.

[47] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, vol. IT-31(no. 4):469C472, 85.

[48] A. Fiat and M. Naor. Broadcast Encryption. *Advances in Cryptology – CRYPTO'93*, LNCS773:480–491, 1993.

[49] M. Freedman and R. Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, Washington, D.C, USA, 2002.

[50] X. Fu, Y. Zhu, B. Graham, R. Bettati, and W. Zhao. On Flow Marking Attacks in Wireless Anonymous Communication Networks . In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pages 493–503, Columbus, OH, 2005.

[51] T. E. Gamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Trans. on Information Theory*, 31(4):469–472, July 1985.

[52] F. Garcia, I. Hasuo, W. Pieters, and P. Rossum. Provable Anonymity. In *Proceedings of the 3rd ACM Workshop on Formal Methods in Security Engineering*, Alexandria, VA, 2005.

[53] S. Goel, M. Robson, M. Polte, and E. Sirer. Herbivore: A Scalable and Efficient Protocol for Anonymous Communication. Technical Report 2003-1890, Cornell University, Ithaca, NY, February 2003.

[54] S. Goldwasser and S. Micali. Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information. pages 365–377, San Francisco, CA, 1982.

[55] P. Golle, M. Jakobsson, A. Juels, and P. Syverson. Universal Re-Encryption for Mixnets. In *Proceedings of the 2004 RSA Conference, Cryptographer's Track*, San Francisco, CA, 2004.

[56] P. Golle and A. Juels. Parallel Mixing. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, Washingto D.C, USA, 2004.

[57] C. Gülcü and Gene Tsudik. Mixing email with babel. In *Proceedings of the Symposium on Network and Distributed System Security*, San Diego, CA, 1996.

[58] V. Hadzilacos and J. Halpern. Message-optimal Protocols for Byzantine Agreement. In *Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing*, pages 309–323, 1991.

[59] D. Halevy and A. Shamir. The LSD Broadcast Encryption Scheme. *Advances in Cryptology-CRYPTO'02*, LNCS 2442:47–60, 2002.

[60] H. Harney and C. Muchenhirn. Group Key Management Protocol (GKMP) Specification. *RFC 2093, IETF*, 1997.

[61] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a Distributed Firewall. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, Athens, Greece, 2000.

[62] M. Jakobsson. Fractal Hash Sequence Representation and Traversal. In *Proceedings of the IEEE International Symposium on Information Theory*, Lausanne, Switzerland, 2002.

[63] J. Jannotti, D. Johnson, M. Kaashoek, and J O'Toole. Overcast: Reliable Multicasting with an Overlay Network. In *Proceedings of the UNENIX OSDI*, San Diego, CA, 2000.

[64] A. Jerichow, J. Müller, A. Pfitzmann, B. Pfitzmann, and M. Waidner. Real-Time MIXes: A Bandwidth-Efficient Anonymity Protocol. *IEEE Journal on Selected Areas in Communications*, 16(4), 1998.

[65] Robert J. Jenkins Jr. Isaac. In *Fast Software Encryption*, pages 41–49, 1996.

[66] S. Kent and R. Atkinson. IP Authentication Header. *RFC 2402, IETF*, 1998.

[67] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP). *RFC 2406, IETF*, 1998.

[68] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. *RFC 2401, IETF*, 1998.

[69] D. Kesdogan, J. Egner, and R. Büschkes. Stop-and-Go MIXes: Providing Probabilistic Anonymity in an Open System. In *Proceedings of Information Hiding Workshop*, Portland, OR, 1998.

[70] D. Kraft and G. Schafer. Distributed Access Control for Consumer Operated Nobile Ad-hoc Networks. In *Proc of the 1st IEEE Consumer Communication and Networking Conference*, Las Vegas, NV, 2004.

[71] J. Kubiatowicz, D. Bindel, Y. Chen, and et. al. OceanStore: An Architecture for Global-scale Persistent Storage. In *Proceedings of ACM ASPLOS*, 2000.

[72] P.S. Kumar. A survey of Multicast Security Issues and Architectures. In *Proceedings of 21st National Information Systems Security Conferences*, Arlington, VA, 1999.

[73] R. Kumar, R. Rajagopalan, and A. Sahai. Coding Constructions for Blacklisting Problems without Computational Assumptions. *Advances in Cryptology-CRYPTO'99*, LNCS 1666:609–623, 1999.

[74] H. Kurnio, L. McAven, R. Safavi-Naini, and H. Wang. A Dynamic Group Key Distribution Scheme with Flexible User Join. *ICISC 2002*, LNCS 435:478–496, 2003.

[75] L. Lamport. Password Authentication with Insecure Communication. *Communications of the ACM*, 24(11):770–772, 1981.

[76] L. Lamport, R. Shostak, and M. Pease. The Byzantine General Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[77] D. Liu and P. Ning. Efficient Self-Healing Group Key Distribution with Revocation Capability. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, Washington D.C.

[78] V. Lo, D. Zhou, Y. Liu, and et al. Scalable Supernode Selection in Peer-to-Peer Overlay Networks. In *Proceedings of the 2nd International Workshop on Hot Topics in Peer-to-Peer Systems*, La Jolla, CA, 2005.

[79] M. Lubby and J. Staddon. Combinatorial Bounds for Broadcast Encryption. *Advances in Cryptology-EUROCRYPTO'98*, LNCS 1403:512–526, 1998.

[80] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, Boston, MA, 2000.

[81] A. J. Menezes, P. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[82] S. Mittra. Iolus: A Framwork for Scalable Secure Multicasting. In *Proceedings of ACM SIGCOMM*, pages 277–288, New York, NY, 1997.

[83] M. Mohsin and R. Prakash. IP Address Assignment in a Mobile Ad Hoc Network. In *Proceedings of IEEE MILCOM*, Anaheim, CA, 2002.

[84] Bodo Möller. Provably Secure Public-Key Encryption for Length-Preserving Chaumian Mixes. *Proceedings of CT-RSA 2003*, LNCS 2612:244–262, April 2003.

[85] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster Protocol — Version 2. http://www.abditum.com/mixmaster-spec.txt, July 2003.

[86] D. Naor, M. Naor, and J. Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. *Advances in Cryptology-CRYPTO'01*, LNCS 2139:41–62, 2001.

[87] S. Nesargi and R. Prakash. MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network. In *Proceeding of IEEE INFOCOM*, New York, NY, 2002.

[88] U.S. Department of Commerce. National Institute of Standards and Technology. Digital Signature Standard. *NIST FIPS PUB 86*, 1996.

[89] National Institute of Standards and Technology. NST FIPS PUB 186, Digital Signature Standard, U.S. Department of Commerce. 1994.

[90] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Proceedings of EUROCRYPT 1999*, pages 223–238.

[91] M. Pease, R. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *J. ACM*, 27(2):228–234, 1980.

[92] C. Perkins, E. Royer, and S. Das. IP Address Autoconfiguration for Ad Hoc Networks. *Internet Dratf, draft-ietf-manet-autoconf-00.txt*, July 2000.

[93] A. Perrig, R. Canetti, J. Tygar, and D. Song. Efficient Authentication and Signing of Multicast Stream over Lossy Channels. In *Proceedings of the 21st IEEE Symposium on Security and Privacy*, Orkland, CA, 2000.

[94] A. Pfitzmann and M. Köhntopp. Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology. In *Proceedings of Workshop on Design Issues in Anonymity and Unobservability*, pages 1–9, Berkeley, CA, 2000.

[95] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable Communication with Very Small Bandwidth Overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, pages 451–463, Mannheim, Germany, 1991.

[96] A. Pfitzmann and M. Waidner. Networks without User Observability. *Computers & Security*, 2(6):158–166, 1987.

[97] A. Shamir R. L. Rivest and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[98] S. Rafaeli and D. Hutchison. A Survey of Key Management for Secure Group Communication. *RFC 1949, IETF*, 1996.

[99] J. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems. In *Proceedings of Privacy Enhancing Technologies Workshop*, pages 10–29, Berkeley, CA, 2000.

[100] M. Reed, P. Syverson, and D. Goldschlag. Anonymous Connections and Onion Routing. *IEEE J. on Selected Areas in Coomunications, Special Issue on Copyright and Privacy Protection*, 16(4):482–494, 1998.

[101] M. Reiter and A. Rubin. Crowds: Anonymity for Web Transaction. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[102] R. Rivest. Unpublished work. A description of RC4 appears in B. Schneier, Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C, John Wiley & Sons, Inc., 1996.

[103] R. Rodrigues, B. Liskov, and L. Shrira. The Design of a Robust Peer-to-Peer System. In *Proceedings of the 10th ACM SIGOPS European Workshop*, Saint Emilion, France, 2002.

[104] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the 18th IFIP/ACM Internation Conference on Distributed Systems Platforms*, Heidelberg, Germany, 2001.

[105] R. Safavi-Naini and H. Wang. New Constructions of Secure Multicast Re-keying Schemes using Perfect Hash Families. In *Proceedings of the 7th ACM Conference on Computer and Communication Security*, Athens, Greece, 2000.

[106] Y. Sasson, D. Cavin, and A. Schiper. P^5: A Protocol for Scalable Anonymous Communication. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, 2002.

[107] B. Schneier. Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). In *Proceedings of Fast Software Encryption*, 1993.

[108] C. Shields and B. Levine. A Protocol for Anonymous Communication over the Internet. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 33–42, Athens, Greece, 2000.

[109] V. Shmatikov. Probabilistic Model Checking of an Anonymity System. *Journal of Computer Security*, 12(3-4):355–377, 2004.

[110] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, and D. Dean. Self-Healing Key Distribution with Revocation. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA, 2002.

[111] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman Key Distribution Extended to Group Communication. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, 1996.

[112] D. R. Stinson and T. van Trung. Some New Results on Key Distribution Patterns and Broadcast Encryption. *Designs, Codes and Cryptography*, 14:261–279, 1998.

[113] I. Stoica, R. Morris, D. Karger, and et. al. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *Proceedings of ACM SIGCOMM*, pages 149–160, San Diego, CA, 2001.

[114] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. OverQoS: An Overlay Based Architecture for Enhancing Internet QoS. In *Proceedings of the 1st Symposium of Networked System Design and Implementation*, San Francisco, CA, 2004.

[115] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, CA, 1997.

[116] USAToday. `http://www.usatoday.com/tech/news/internetprivacy/2006-08-14-aol-eff-ftc_x.htm?csp=34`.

[117] N. Vaidya. Weak Duplicate Address Detection in Mobile Ad Hoc Networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Lausanne,Switzerland, 2002.

[118] M. Waidner. Unconditional Sender and Recipient Untraceability in Spite of Active Attacks. *Advances in Cryptology: EUROCRYPT'89*, LNCS 434:302–319, 1989.

[119] D. Wallner, E. Harder, and R. Agee. Key Management for Multicast: Issues and Architectures. *RFC2627, IETF*, 1999.

[120] P. Wang, P. Ning, and D. Reeves. Storage-Efficient Stateless Group Key Revocation. In *Proceedings of the 7th Information Security Conference*, Palo Alto, CA, 2004.

[121] P. Wang, P. Ning, and D. Reeves. Network Access Control for Mobile Ad-Hoc Networks. In *Proceedings of the 7th International Conference on Information and Communication Security*, Beijing, China, 2005.

[122] P. Wang, P. Ning, and D. Reeves. A k-Anonymous Communication Protocol for Overlay Networks. In *To appear in the ACM Symposium on InformAtion, Computer and Communication Security*, Singapore, 2007.

[123] P. Wang, D. Reeves, and P. Ning. Secure Address Auto-Configuration for Mobile Ad Hoc Networks. In *Poster, in the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, San Diego, CA, 2005.

[124] B. Waters, E. Felten, and A. Sahai. Receiver Anonymity via Incomparable Public Keys. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, Washington D.C, USA, 2003.

[125] K. Weniger. Passive Duplicate Address Detection in Mobile Ad hoc Networks. In *Proceedings of IEEE Wireless Communications and Networking Conference*, New Orleans, LA, 2003.

[126] P. Wilson, M. Johnstone, M. Neely, and D. Boles. Dynamic Storage Allocation: A survey and Critical Review. In *Proceedings of Internation Workshop on Memory Management*, Kinross, UK, 1995.

[127] C. Wong, M. Gouda, and S. Lam. Secure Group Communications Using Key Graphs. In *Proceedings of ACM SIGCOMM*, Vancouver, B.C, 1998.

[128] H. Yoshino, N. Hayashibara, T. Enokido, and M. Takizawa. Hierarchical Protocol for Byzantine Agreement in a Peer-to-Peer Overlay Network. In *Proceedings of the 16th International Workshop on Database and Expert Systems Applications*, pages 5–9, Copenhagen, Denmark, 2005.

[129] S. Zhu, S. Xu, S. Setia, and S.Jajodia. LHAP:A Lightweight Hop-by-Hop Authentication Protocol For Ad-hoc Networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops*, Providence, Rhode Island, 2003.

[130] P. R. Zimmermann. *The Official PGP User's Guide.* MIT Press, Jun 1995.