The Pennsylvania State University
The Graduate School

SECURITY AND PRIVACY ASPECTS IN WIRELESS AND MOBILE NETWORKS

A Thesis in
Computer Science and Engineering
by
Heesook Choi

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2007

The thesis of Heesook Choi was reviewed and approved* by the following:

Thomas F. La Porta
Distinguished Professor of Computer Science and Engineering
Thesis Advisor, Chair of Committee

Guohong Cao
Associate Professor of Computer Science and Engineering

Patrick D. McDaniel
Hartz Family Career Development Assistant Professor of Computer Science and Engineering

George Kesidis
Professor of Computer Science and Engineering

Carleen Maitland
Assistant Professor of College of Information Sciences and Technology

Raj Acharya
Professor of Computer Science and Engineering
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

The goal of ubiquitous computing is to integrate communication into our physical environments. Wireless and mobile networks play an important role in realizing the vision of ubiquitous computing. We address three types of wireless and mobile networks: cellular, mobile ad hoc (MANETs), and wireless sensor networks.

Wireless links have high packet loss rates and delay variation which affect the end-to-end performance. User devices (or nodes) have power constraints, low computational capability, and small memory space. In wireless and mobile networks, performance and resource-aware communication has received significant research attention. As mobile and wireless network services become more ubiquitous, security and privacy manifest into major concerns. Security and privacy services require additional computation and communication overhead. Due to resource limitations, it is a challenge to provide security and privacy services in wireless and mobile networks. We address security and privacy aspects in these environments, inspecting trade-offs between communication and computation costs and security services.

We analyze the effectiveness of an adversary with probabilistic approaches and demonstrate that the proposed solutions mitigate the effectiveness of the adversary. We also explore communication and computation costs via thorough implementation and simulations of solution systems. We show that it is possible to effectively trade-off security level with performance.

Table of Contents

List of Figures

List of Tables

Acknowledgments

"So do not fear, for I am with you; do not be dismayed, for I am your God I will strengthen you and help you; I will uphold you with my righteous right hand." [Isaiah 41:10]

I have walked through this journey since God has provided me with everything through many hands. Here, I would like to express my deep appreciation to people who have supported and advised me during my Ph.D. study.

I am filled with gratitude for my advisor, Dr. Thomas F. La Porta, for his advice and encouragement. He has guided my research direction with valuable and challenging questions. He also has inspired me to persevere and courageously go forward in every situation. This will be kept forever deep in my heart.

I am also honored and thankful to be guided by my Ph.D. committee: Dr. Guohong Cao, Dr. Patrick D. McDaniel, Dr. George Kesidis, and Dr. Carleen Maitland. Dr. Patrick McDaniel has given many advices to direct my research on anonymity and privacy in MANETs. Dr. Sencun Zhu has also provided valuable assitance in the work on security in wireless sensor networks.

My parents (Bongho Choi and Yeanhee Park) have always been by my side. I believe that this has been an essential element for me to stand strong in any situation. In addition, I am very thankful to my sisters (Heeja and Yoomi) and brothers (Insoo and Daesoo). They have shown their pride and love for me. I have reached this milestone with the unfailing love and support from my family. I am also grateful to my cousin (Jeongsook Kim) for her support.

I cordially appreciate the prayers and generous support from Yousung Church members. They have been my shelter in God. The fact that I am a member of this church is one of my most cherished blessings.

I have also been a member of "International Life Group" which is led by Bill Saxton ("Uncle Bill"). I thank all the life group members for their prayers and friendship [1]. I have also been involved in a regular Friday fellowship meeting. Members of this meeting have changed over time. I would like to express my thankfulness to all of them [2]. We have prayed for and encouraged each other. I also want to express my gratitude to my friends and people who have been there for me and supported me [3].

During this journey at the Pennsylvania State University, I have had the fortune of being surrounded by many friends [4]. I met them in classes, research lab, etc. I want to thank the previous and current members of Korean Student Association in the Computer Science and Engineering Department. I earned the most precious treasure (all of my friends here) in my life. I would like to express deep gratitude to them all.

[1]Bill Saxton and Barb Saxton, HenSiong Tan and Joanie Tan, Tze-wei Chu, Hien Nguyen, Maria Mercedes

[2]Kwang Y. Lee and Sangwoel Lee, Kyungna Kim, Hyunju Jeong, Yoongsik Park and Heejin Lee, Seokil Bae, Yoonhee woo, Seunghoon Bang and Mijeong Cho

[3]Myungah Park, James Lee, Sujin Yean, and Seonhee Yang

[4]Jyotsna Kasturi, Suzanne M. Shontz, JaeSheung Shin, Patrick Traynor, William Enck, Patrick C. Hicks, Kevin Buttler, San Rueda Rodriguez, Hui Song, Raju Kumar, Hosam K. Rowaihy, Shiva Kasiviswanathan, Andrew J. Ricketts, Jing Zhao, Liang Xie, Yan Sun, Kameswari Kotapati

To my parents

Chapter 1

Introduction

The goal of ubiquitous computing is to integrate communications with our physical environment. Wireless and mobile networks play an important role in realizing this environment.

A wireless and mobile network can be classified based on whether it relies on infrastructure for its operation or not. One representative wireless and mobile network is a cellular network in which mobile phones communicate over a wired backbone network via a radio air interface. Figure 1.1 illustrates a general cellular network architecture. A service area is composed of a group of cells. Mobile phones in a cell are connected to a base station (BS) through a radio interface. One base station controls all mobile phones in a cell. Each cell covers a few kilometers in diameter. Each BS is connected to a mobile switching center (MSC) which controls user mobility.

At first, the application of a cellular network was limited to voice service. As multimedia and data services became popular on the Internet, the wireless service providers started to evolve cellular networks to accommodate these services as well. The cellular networks rely on infrastructure for control.

Wireless LANs are another fast growing type wireless and mobile network [1], an example of which is given in Figure 1.2. Compared to a cellular network, WLANs have a short transmission range, but high performance. User devices having a wireless LAN

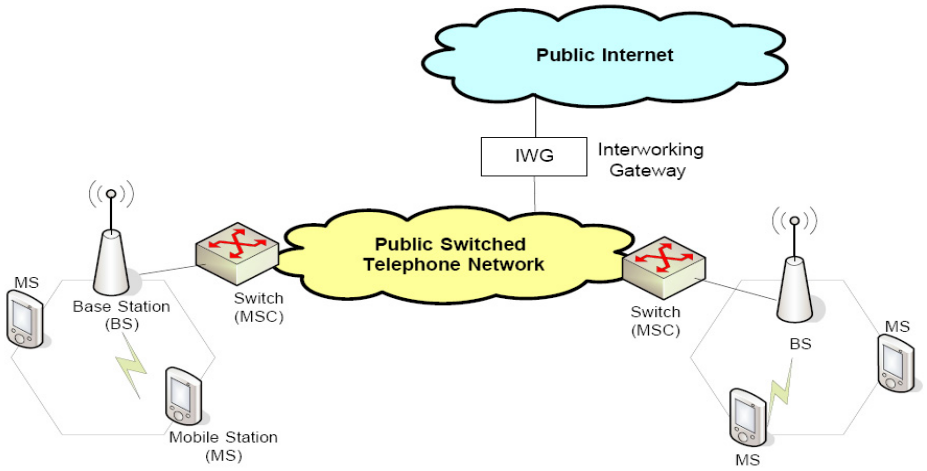


Figure 1.1. Cellular Network Architecture

card are connected to an access point (AP). The AP connects wireless user devices to a local router which provides the connectivity to the Internet. As in the cellular network, WLANs rely on infrastructure for control and mobility management.

To provide seamless service in the face of mobility in wireless LANs, Mobile IP [2] is used. Cellular code-division multiple access (CDMA) networks also adapt Mobile IP to support data services [3]. Figure 1.3 shows the Mobile IP network architecture. In Mobile IP, a mobile host (MH) is assigned a care-of-address (CoA) in a visiting foreign network, while keeping its home address. A home agenet (HA) encapsulates packets for the MH by adding an outer IP header which has HA address and CoA in the outer IP source and destination address fields, respectively. The HA forwards the encapsulated packets to a foreign agent (FA) based on the CoA of the MH. The FA in the visiting network decapsulates the packets from the HA and then forwards them to the MH.

Unlike infrastructure-based wireless and mobile networks, nodes in mobile ad hoc networks (MANETs) bootstrap and operate their network without using infrastructure. Due to the independence from infrastructure, mobile nodes can form a network anywhere at anytime as necessary. A MANET is a self-configuring network because it configures itself without any centralized control. Due to the limited transmission range of the

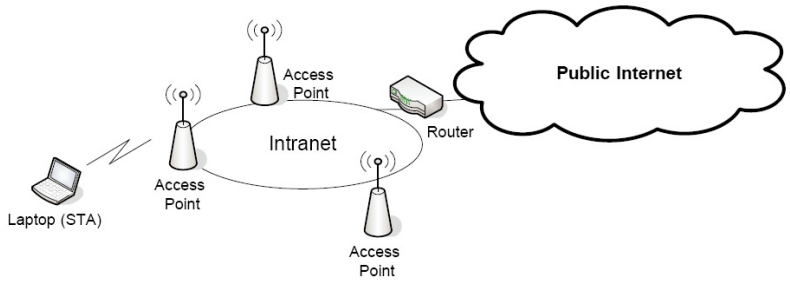


Figure 1.2. Wireless LAN Architecture

typical nodes in a MANET, mobile nodes function as routers to relay traffic for other nodes. Energy aware communication is an important issue in MANETs because mobile nodes are battery powered.

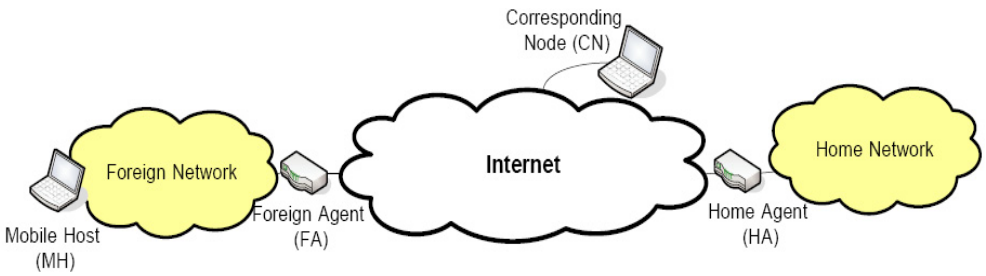


Figure 1.3. Mobile IP Architecture

Wireless sensor networks have similar characteristics to MANETs. Sensors also self-configure a network and communicate with each other over multiple hops by relaying packets for others. A sensor network provides communication in environments for many useful purposes such as disaster recovery and habitat monitoring. Compared to mobile nodes in MANETs, sensors are quasi-static and have more limited resources (memory, computation capability, and energy).

As mobile and wireless network services become more ubiquitous in our world, one major concern is security and privacy. Due to the ubiquity of the service, users may be open to more vulnerabilities. Security and privacy require additional computation and communication, which are important and limited resources in the wireless and mobile

networks. Hence, we aim at providing security and privacy with low communication and computation costs in the networks.

In this thesis, we address security and privacy issues in wireless and mobile networks, while exploring trade-offs between communication and computation costs and security. We show that it is possible to provide security and privacy services that effectively trade-off performance and security. Specifically, we address:

- end-to-end data protection integrated with performance enhancements in networks that use Mobile IP
- anonymity and security of network usage reports (privacy of control points) in civilian MANETs
- privacy preserving communication in MANETs
- defense against clone attacks in wireless sensor networks

Mobile IP provides seamless network service when a user roams. When a mobile node moves to a foreign network, an agent in the foreign network needs to allow the mobile node to use the local network resources. Mobile IP uses the AAA model [4] specified in the IETF for authentication and authorization. This model may be used in cellular or WLAN data networks. Mobile IP, however, does not support data security.

In generic WLAN environments, two types of authentication are supported: Open System and Shared Key. Open system authentication is specified as the default in 802.11 and actually means NULL authentication. Wired Equivalent Privacy(WEP) is the required protocol for shared key authentication. After the authentication and association finishes successfully, a mobile device can transmit data through the AP. However, many research results [5, 6] have shown the vulnerabilities of WEP. To address the WEP vulnerabilities, a new Wi-Fi security specification named as IEEE 802.11i [7] was ratified in 2004. However, it is discovered that there still exist vulnerabilities [8, 9]. Therefore, many WLAN users rely on IPsec [10] to provide an end-to-end secure channel.

While IPsec provides adequate data protection, it creates other limitations. It is a well known fact that the communication on wireless links has a high packet loss and high packet delay variance. In a TCP connection, if packet loss occurs, a corresponding node may misinterpret the loss to congestion in the core network. This reduces its throughput. To address this performance problem in wireless networks, various solutions [11, 12, 13] have been proposed. Most proposed solutions use a router (base station or foreign agent) to execute a performance enhancement by using information in the TCP/IP header. Most of these solutions are not compatible with the end-to-end security service of IPsec, because they cannot access the encrypted header.

One appropriate starting point to overcome this problem is to use Multi-layered IPsec [14] in which packets are divided into multiple zones. A different key is assigned to each zone. This way, some routers may be given the header key so that they may execute performance enhancements.

However, Multi-layered IPsec (ML-IPsec) uses manual keying which raises scalability and flexibility issues in mobile environments. We simplify the Multi-layered IPsec structure such that only two zones (TCP/IP header and payload) are defined. We propose a model to integrate ML-IPsec with Mobile IP. To support dynamic mobile environments, we propose efficient key distribution protocols using network topology information such as neighboring foreign agents or previous foreign agent. We implement the key distribution protocols based on Internet Key Exchange protocol (IKE) [15, 16] and provide detailed performance measures.

In MANETs, mobile nodes bootstrap their own network without any infrastructure. MANETs are applied to mission-oriented applications such as battle field operation and emergency response. Since mobile nodes in these applications have the same mission, they cooperate for achieving a goal. However, in civilian applications of MANETs, mobile nodes are rational and pursue their own interest. To encourage nodes to cooperate, many solutions have been proposed. These existing solutions require mobile nodes to monitor the activities of other nodes and send the monitored reports to the source, destination

or a centralized server (off-line).

There is no existing work to address the reporting privacy of forwarding activities in MANETs. We propose to have mobile nodes monitor themselves. In our protocol, a source selects a reporting node randomly, preventing other nodes en route from discovering which node is sending a report. This provides privacy and security of a report. We further define a scheme to ensure nodes report their activity truthfully.

Privacy of user identity in the Internet has attracted much research attention. This is a challenging objective in wireless networks, because communication is easily overheard. MANET routing protocols use location information to find an efficient route. For an adversary, the location information may be used to trace users and launch attacks. In addition to identity privacy, location privacy in ad hoc networks is also an important issue.

In MANETs, however, anonymity issues have not received as much research attention. There is no satisfactory privacy solution based on private-key cryptosystems suitable for MANETs. In this thesis, we propose a Privacy Preserving Communication System (PPCS) which provides a comprehensive solution to anonymize communication end-points, keep the location and identifier of a node unlinkable, and mask the existence of communication flows. We present an analysis of the security of PPCS against passive internal attackers, provide a qualitative discussion on its strength against external attackers, and characterize its performance trade-offs.

Wireless sensor networks have attracted a great deal of attention because they may be used to support a variety of applications: disaster and safety monitoring, military surveillance, wildlife habitat monitoring, etc. Sensor nodes that are deployed in hostile environments are vulnerable to capture and compromise. An adversary may obtain private information from these sensors, clone and intelligently deploy them in the network to launch a variety of insider attacks. This attack is termed as a *clone attack*.

Currently, the defenses against clone attacks are not only very few, but also suffer from selective interruption of detection and high overhead (computation and memory).

In this thesis, we propose a new effective and efficient scheme, called *SET*, to detect such clone attacks. The key idea of *SET* is to detect clones by computing set operations (intersection and union) of exclusive subsets in the network.

This thesis is organized as follows: Chapter 2 reviews previous work on different aspects of security and privacy in wireless and mobile networks. Chapter 3 proposes an integration architecture of Multi-layered IPsec and Mobile IP, and efficient key distribution protocols that allow good end-to-end connection performance and support dynamic mobility, while providing end-to-end data security. Chapter 4 presents solutions for anonymous and secure reporting of forwarding activities in MANETs. In Chapter 5, we propose a privacy preserving communication system for MANETs. In Chapter 6, we propose a new effective and efficient scheme, called *SET*, to detect clone attacks. We conclude the thesis with final comments in Chapter 7. We discuss our plan for future work in Chapter 8.

Chapter 2

Related Work

There are two models to provide secure communications: public-key cryptosystems and private-key cryptosystems [17]. In the public-key cryptosystem, communicating parties do not need to share any key information before their communication. Two different keys (public key and private key) are used to encrypt and decrypt packets. The private key is secret, while the public key is open. On the other hand, parties using a private-key cryptosystem need to share a key before they communicate. Public key cryptosystem are much more expensive than private-key systems in terms of computation.

As wireless and mobile environments become more prevalent, security research faces new challenging issues. For example, Mobile IP [2] provides only authentication, not data privacy. For providing end-to-end security of data, IPsec must be used between endpoints. The efficient integration of Mobile IP and IPsec is an open problem.

Most security solutions in MANETs depend on the private-key cryptosystems because of computational and energy limitations of mobile nodes. Since there is no infrastructure in MANETs, it is an important and challenging issue to provide key management functions to establish a symmetric key between nodes. There have been many solutions to address the key management [18, 19, 20, 21] in these environments. The symmetric key is used to provide confidentiality, integrity and authenticity of communications in MANETs.

To combat various routing attacks, secure routing protocols have been proposed [22, 23]. However, the existing MANET routing protocols do not consider privacy issues. In addition to key management and secure routing protocols, other security issues to address include motivating cooperation, secure multicast, and secure data aggregation [24, 25, 26]. In this chapter, we review the existing work related to security in infrastructure-based networks, malicious behavior detection and cooperation in MANETs, privacy in MANETs, and clone detection in wireless sensor networks.

2.1 Security in Infrastructure-Based Networks

Several studies have shown that the performance of classic data communication protocols can be quite poor when used over wireless links. In particular, the performance of TCP, the reliable Internet transport protocol, can be degraded by the loss and delay characteristics of a wireless link. Consequently, there have been several efforts aimed at improving the performance of TCP on wireless links. Two of the more promising works do not require any modifications to TCP, but instead perform smart processing (forwarding, filtering, scheduling) on TCP/IP packets based on information gleaned from observing packet flows. In [12], the authors show that by snooping on TCP/IP packets at the wireless edge, determining when packet loss has occurred by detecting duplicate acknowledgments, and performing fast local retransmissions, TCP performance can be greatly improved.

More recently in [11], it was uncovered that TCP performance is adversely affected by the highly variable delay experienced on the 3G wireless links. The effect is that TCP acknowledgments sent on the uplink tend to be compressed causing them to arrive at transmitters back-to-back. The compression of acknowledgments results in transmitters sending bursts of data. These bursts of data can overflow buffers at the wireless edge resulting in high packet loss. The solution proposed is to regulate the flow of acknowledgments to ensure that buffers do not overflow.

In both of these proposals, the node at the wireless edge must observe information in the TCP header to execute their algorithms. The need to have nodes inside the network examine packet payloads to perform smart packet processing or perform packet classification is in direct conflict with the current Internet model of security implemented by the IPsec protocol suite [27, 28, 29, 16]. IPsec supports a variety of operational modes including packet authentication, packet encryption, or both. In the most secure mode, tunnel mode, the entire IP packet is encrypted and encapsulated with a new IP packet header. Therefore, intermediate nodes in the network do not have access to the original IP header information, nor the information contained in any of the transport layer or application layer protocols. This precludes the network from performing smart packet processing and packet classification to improve end-to-end performance.

There are several possible solutions to this problem. Protocols such as TLS [30] and SSL [31] provide security above the transport layer. With their use, user payloads are encrypted, but the TCP and IP headers are in the clear. Therefore, intermediate nodes may access required information to perform many of the performance enhancements discussed above. The main drawback is that the TCP and IP header information is in the clear throughout the entire network allowing for possible eavesdropping to determine communication patterns and traffic characterizations. Also, these protocols do not enable application layer packet classification for protocols such as RTP.

A more flexible solution is defined in the ML-IPsec protocol [14]. This protocol allows a user to define zones within an IP packet. Each zone is encrypted and authenticated with its own security association (SA). Each zone may be accessed (decrypted) by different network elements. This requires SAs to be established between a client and several nodes in a network, each of which can decrypt a certain portion of the IP packet while being unable to view the entire packet.

2.2 Detection of Malicious Behavior in MANETs

The Watchdog/Pathrater [32] scheme proposes the use of a watchdog for detecting misbehaving nodes, and a pathrater to help the routing protocol avoid detected misbehaving nodes. The design uses intermediate nodes along the routing path, wherein a node sends a packet to an intermediate downstream node and verifies that this node forwards the packet. If the node does not forward the packet within a predefined period, it is declared as misbehaving, and the monitoring node notifies the source.

Zhang and Lee [33, 34] propose a general architecture in which all nodes participate in the monitoring of data transmission. Each node is responsible for monitoring a transmission range and cooperating with neighbors in order to detect intrusions. Zhang and Lee later proposed a second scheme to reduce the number of nodes involved in monitoring [35]. In this cluster-based scheme, a cluster head (CH) is elected for monitoring data traffic within the transmission range. The elected CH is responsible for monitoring all neighboring nodes and checking statistics.

AODVSTAT [36] implements an intrusion detection system (IDS) within the AODV [37] routing protocol. The system monitors for routing message drops, data-packet drops, MAC/IP spoofing, and resource depletion attacks. In AODVSTAT, an IDS monitors all observable transmissions from neighbors. Note that all of the above schemes require some level of communication eavesdropping. These solutions are not feasible in our target environments because reliable eavesdropping is not possible.

Awerbuch et al. [38] propose an alternate scheme that uses intermediate nodes on the data path. If a source does not receive an ACK from a destination, the source begins probing all intermediate nodes. This causes each node along the path to send an ACK back to the source. Unfortunately, due to the dynamic characteristics of MANETs, data paths can change frequently, possibly before the failed link is found.

2.3 Cooperation in MANETs

Many times, cooperation between nodes cannot always be expected without incentives. A node may be paid via a credit for behaving cooperatively, or excluded/penalized for misbehaving. We classify the existing solutions into three classes: credit-based, penalty-based, and utility function based schemes.

In the first class, relay nodes are remunerated for forwarding packets for others. Zhong et al. [25] proposed an incentive system, named Sprite, in which selfish nodes are encouraged to cooperate. In Sprite, each node is motivated to honestly report its actions, even in the presence of selfish node collusion. Intermediate nodes retain receipts of received messages. The receipt is then sent to the CCS (Credit Clearance Service) connected to the Internet as proof of forwarding, and the CCS charges/credits based on the received reports. CORE [39] uses a collaborative reputation mechanism to encourage nodes to cooperate. The reputation is calculated via both direct and indirect observation by a node and its neighboring nodes, respectively, within the transmission range. Buttyan and Hubaux [40] proposed a stimulation scheme in which a node can transmit its packet only when it has enough credit count, called a *nuglet*. Nodes in the network can earn nuglets by forwarding packets for others. The authors presented four rules that a node might take and showed that the node could receive the best performance when it follows a cooperative rule.

In the second category, selfish and misbehaving nodes are penalized for their non-cooperative behavior by being excluded from the routing and the community. In CONFIDANT [41], each node monitors one-hop neighbor nodes. If a node detects and concludes malice, it generates an ALARM message to either a source or a friend. This, in turn, causes misbehaving nodes to be excluded from the community. The Watchdog/Pathrater [32] scheme also belongs to this group.

In the third class, a utility function is defined at each node such that nodes can maximize the utility function (their benefit) by cooperatively forwarding packets for others.

These solutions are based on the mathematical framework of game theory. Srinivasan et al. [42] proposed a distributed acceptance algorithm. In this algorithm, relay nodes maintain the history of services that they receive and provide. The authors proposed a game theoretic strategy (Generous TIT-FOR-TAT) which is used for each node to decide whether it accepts a new relay request based on the past history. Anderegg and Eidenbenz [43] proposed a routing protocol, called Ad hoc-VCG, based on game theory. In the Ad hoc-VCG, during route discovery, each node makes its energy cost and transmission power known to other nodes, and a destination chooses the most cost-efficient path based on the collected information. In the Ad hoc-VCG mechanism, nodes cannot receive higher payments by cheating, which encourages nodes to relay packets of others.

In our system, we assume that there is no centralized controller or trusted third party as there is in Sprite. The reports of traffic forwarding activities are collected by the endpoints of a flow, i.e., the source and destination. These nodes can use the reports to detect misbehaving nodes on their own active flows, and thus take immediate actions to protect their flows. In addition, the reports may be used with some of the reputation based systems discussed above.

2.4 Privacy in MANETs

A great deal of previous research has focused on providing confidentiality, integrity, and authenticity of data in MANETs, but anonymity remains an open problem. Pfitzman and Hansen [44] define general terminologies of anonymity. In their article, anonymity is defined as "state of being not identifiable within a set of subjects, the anonymity set."

Chaum's [45] pioneering anonymity solution introduces a mix or a series of mixes (mix network) into a network for hiding communicating endpoints [46] in the Internet. A source selects the route (set of mixes) and encrypts data packets with the public key of each mix in reverse order (from last mix to the first mix). Each mix peels off one layer by decrypting the received packet with its private key and forwarding it to the next hop.

The last mix processes the packet in the same way and transmits it to the destination.

Onion routing [47] is built on a mix-net approach. An onion consists of next hop information and an onion for the next hop. Each intermediate onion router decrypts the received message with its private key to get the next hop and onion for the next hop. The last onion peels off its layer and transmits the encrypted data to the destination. Tor [48] extended onion routing with features that provide forward secrecy.

Mix-nets are not applicable to MANETs, because the resource demands of the underlying public key operations are too expensive for mobile nodes with energy and computation limitations. Moreover, with high mobility, it is not easy to maintain the full path from the source.

In Crowds [49], groups of users (called *crowd*s) cooperate to ensure client anonymity in web systems, e.g., web-browsing. *Jundos* run by each client decide randomly if they should relay the packet to another jundo or transmit it to the web server directly. All users in the group share their symmetric keys to encrypt the relayed packet. Hordes [50] is based on Crowds and proposes to use multicast routing to provide initiator anonymity. Brent [51] proposes receiver anonymity based on incomparable public keys and multicast. In MANETs, however, the maintenance cost of multicast is known to be high.

Most solutions proposed for the Internet use a proxy function (Mix, Jundo, and Onion Router) to provide anonymity. In MANETs, Jian et al. [52] propose a dynamic mix method that accommodates dynamic topology changes. Blaze et al. propose WAR [53], in which anonymous routing is combined with a key distribution protocol and an onion routing structure. However, in MANETs, it is not feasible to form a set of proxy functions since mobile nodes all play an equal role. In civilian applications of MANETs, in particular, mobile nodes may not cooperate to play the larger role of a proxy.

Kong and Hong [54] apply MIX-Net to MANETs by using symmetric key cryptography to provide anonymity. This approach uses a cryptographic trapdoor within a broadcast message to hide the identifiers of local intermediate nodes and the destination. However, in a situation in which adversaries are located on each link, they may simply

monitor the transmission to determine who is broadcasting and how many packets are being broadcast.

Recently, Zhang et al. proposed MASK [55] in which a Trusted Authority (TA) assigns a large number of random identifiers and a set of corresponding secret points to each node sufficient for the lifetime of a node.

2.5 Clone Detection in Wireless Sensor Networks

Due to the unattended operation of sensor networks, sensors are vulnerable to a variety of insider and outsider attacks. To defend against these attacks, researchers have inspected various threat models and addressed many security issues in wireless sensor networks. Walters et al. surveyed these security issues in wireless sensor networks [56].

In this section, we review the existing research efforts to combat clone attacks. Capkun et al. [57] proposed a secure positioning and distance measuring scheme. In particular, they considered an attack in which cloned nodes appear to be a single node that is changing location. To address this attack, the base station uses a unique fingerprint to identify each device. However, this may not be applicable to communication among peer sensors since each sensor will be required to keep the unique fingerprint information of other sensors. If a node is captured and cloned, it will then have the fingerprints it requires to go undetected.

Zhu et al. [58] proposed a key management protocol for sensor networks that provides a defense against the clone attack. The idea is to remove the master key once a sensor established pairwise keys so that although the adversary may generate clones of a node and deploy them, the clones cannot establish pairwise keys with the new neighbors.

A Sybil attack [59] is orthogonal to the clone attack. An adversary compromises nodes and deduces important information of different identifiers which may be unique in the network. Newsome et al. [59] explored the Sybil attack in sensor networks and proposed several defenses. In this thesis, we address the case in whcih there exist a

node's multiple replicas (clones) which have the same identifier.

In addition to efforts on preventive technologies, it is imperative to provide an efficient clone detection system. Few works, however, have addressed clone detection. Parno et al. [60] proposed random multicast and line-selected multicast schemes in which neighbor nodes of a sensor select random multiple witness nodes in the network and send a report (identifier and location) of the node to them. The distributed nature of this solution is interesting. This scheme incurs higher communication overhead ($O(N\sqrt{N})$) and memory usage to accommodate public keys of enough other nodes to achieve a high probability of detection.

Chapter 3

Mobile Multi-Layered IPsec

3.1 Background

To achieve high throughput in wireless networks, smart forwarding and processing of packets in access routers is critical for overcoming the effects of the wireless links. However, these services cannot be provided if data sessions are protected using end-to-end encryption as with IPsec, because the information needed by these algorithms resides inside the portion of the packet that is encrypted, and can therefore not be used by the access routers. A previously proposed protocol, called Multi-layered IPsec (ML-IPsec) modifies IPsec in a way so that certain portions of the datagram may be exposed to intermediate network elements, enabling these elements to provide performance enhancements.

For example, consider the wireless network of Figure 3.1. In this example, the corporate firewall acts as a Mobile IP Home Agent (HA). Foreign Agent (FA) 1 requires access to TCP/IP header information to perform smart packet processing. Using IPsec, secure communication would entail running an IPsec tunnel between the HA and Mobile Host (MH), in which case FA1 would not have access to the TCP/IP header information. Using ML-IPsec, this header information would be included in Zone A which is accessible to FA1. However, the user payload would be placed in Zone B which is not accessible to

FA1. In this way, the user information is protected end-to-end and the TCP/IP header information is protected from all nodes except FA1 which may perform smart packet processing.

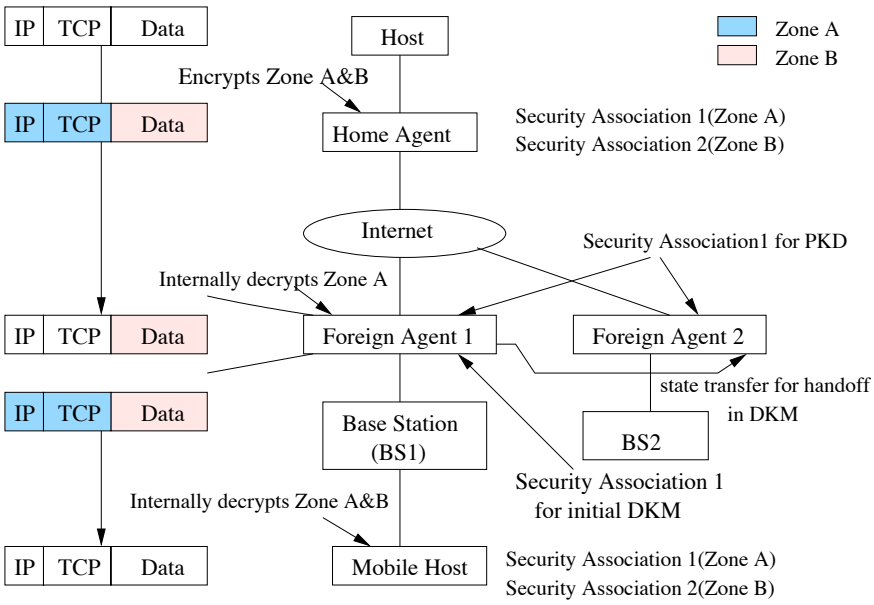


Figure 3.1. Mobile Multi-Layered IPsec Example

While ML-IPsec is a promising start, it has limitations and several unknowns. First, it requires that SAs (secret keys, algorithms, parameters, etc.) be established between multiple elements for a single data session. This requires an efficient key distribution algorithm which has yet to be defined. Second, mobility is not supported. The mobility requires that new SAs be established as a mobile host moves during a data session. For example, in Figure 3.1, if the mobile host moves from base station 1 (BS1) to BS2, SA1 must move from FA1 to FA2. These modifications must be performed quickly so that sessions are not disrupted during a handoff. This is more complex than mobility in basic Mobile IP [61] because in this MML-IPsec system, multiple SAs exist that operate on the user data.

Third, there is no data available on the performance trade-offs between the overhead of supporting multiple zones versus the benefits of packet classification or smart packet

processing. Specifically, mobile access routers, e.g., FAs, will have hundreds of flows passing through them, so the overhead of the key distribution and initialization, handoffs, and per packet processing, must be kept low to achieve high performance.

IKE [15, 16] supports key distribution and mutual authentication between two nodes but requires extensions to support the multiple SAs used in ML-IPsec and is not suitable for mobility. Support for key distribution in mobile networks is the focus of [62]. An efficient method of key distribution and authentication between a home network, security server in a foreign network, and a mobile host is presented. However, this work does not address the distribution of multiple keys required for a ML-IPsec, does not account for mobility, and does not provide any implementation or performance insights.

The remainder of this chapter is organized as follows. In Section 3.2, we discuss our design of MML-IPsec, our model for integrating Mobile IP with IPsec and MML-IPsec, the software platform on which we base our implementation, and the test bed used to evaluate the performance of the protocols. In Section 3.3, we present our key distribution protocol, our two mobility protocols, and characterize their performance. In Section 3.4, we conclude this chapter with a short discussion.

3.2 Mobile Multi-Layered IPsec - Services and Software Platform

In the following subsections we present background on ML-IPsec and how to make it suitable for wireless networks; then we propose solutions to integrate IPsec and mobile IP; the last two subsections describe the software platform on which we base our implementation, and our test bed.

3.2.1 Protocol Services

The original ML-IPsec [14] is defined to allow network layer packets to be segmented into zones, each of which is protected, i.e., encrypted, authenticated, or both, indepen-

dently. Corresponding hosts have access to all zones and can therefore authenticate and decrypt the entire packets. Selected intermediate nodes are given access to one or more selected zones, and may therefore decrypt and authenticate only these portions of the packet. Before communication can commence, a set of SAs, called a composite security association (CSA), must be established, one for each zone in each node for which access to the zone is permitted. As defined in [14], the number of zones and the number of intermediate nodes with access to at least one zone are not limited. Also, there is no limit on the number of zones in a packet, and zones are not required to cover contiguous bits in a packet.

We have somewhat restricted the definition of ML-IPsec to meet the needs of the known methods of enhancing wireless system performance, while keeping the processing complexity low. First, we limit the number of allowed intermediate nodes to a single node, specifically the Mobile IP FA. We choose this node because the vast majority of wireless enhancements operate on a node close to, or supporting a wireless link, and do not require changes to any other portion of the network. Second, we limit the number of zones to two, one for the packet header and one for the payload. The rationale is that most algorithms require access to TCP/IP header information, and not packet payload. This restriction can be easily relaxed.

Finally, we define zones as contiguous portions of the packet to ease processing. Certainly we can let a zone consist of a collection of sub-zones (i.e., continuous blocks) distributed over the whole datagram. We define each zone as one continuous block for two reasons. First, in our settings, it is nature to have only two continuous blocks: one is the IP head and the other is the payload. Second, it eliminates the need for maintaining sub-zone information for each zone. With reduced processing complexity, our protocol is more practical and efficient at core devices such as routers.

In addition to these changes, we have also defined a key distribution protocol for MML-IPsec and two mobility protocols (Section 3.3).

3.2.2 Integrating Mobile IP, IPsec, and MML-IPsec

We assume that if the basic IPsec is used, the IPsec tunnel extends between the HA and the MH. If MML-IPsec is used, the MML-IPsec tunnel includes the HA and the MH, while the FA has access to the header part of the TCP/IP packet. In addition, we assume reverse tunneling [63] is used for data transmitted from the MH so that packets in both directions are consistently encrypted.

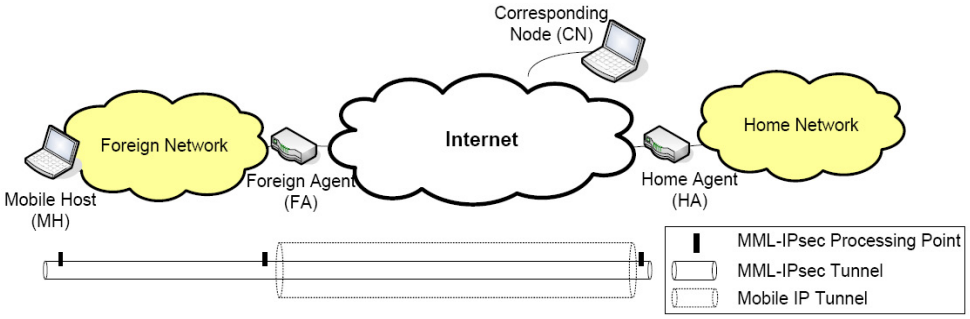


Figure 3.2. Integrating Mobile IP and MML-IPsec/IPsec

Several research projects [64, 65] have proposed solutions to integrate Mobile IP and IPsec. SecMIP [65] is based on the configuration in which the MH uses the DNS/DHCP service to get the COA (Care-of-address). Without using the FA, the MH gets a new collocated COA using DHCP. SecMIP uses the IPsec tunnel to protect the Mobile IP tunnel. While this is a simple scheme to provide security in Mobile IP, the handoff delay is high because the MH must re-establish the IPsec tunnel on every handoff.

Secure Mobile networking (SMN) project [64], by Portland State University, establishes an IPsec tunnel between the HA and the MH. When the MH moves to the foreign network, it has to register its COA to the HA. However, the established IPsec tunnel prevents the Mobile IP registration message from reaching the FA because every packet, including the Mobile IP registration packet, is encrypted with IPsec. In order to solve this problem, the client software is changed so that the Mobile IP packet is not encrypted. The re-establishment of IPsec tunnel on every handoff will degrade the end-to-end performance because it incurs high handoff latencies.

We propose a different integration model for Mobile IP and IPsec which is then largely re-used for integrating Mobile IP and MML-IPsec. This model is similar to SMN, but does not require changes to the client software, and does not require IPsec tunnels to be re-established after each handoff. To enable Mobile IP registration messages to be received by the FA when an IPsec tunnel is in place between the MH and HA, we add an additional routing entry in the MH and leverage the fact that route selection chooses the route having the longest prefix match among multiple matched entries. When an agent advertisement is received by a MH, it adds a route entry for the FA. The new route entry specifies the FA address as the gateway for all packets destined to the FA. After adding this route, the Mobile IP registration message addressed to the FA will match the new route, and therefore be sent directly to the FA, instead of using the old entry through which packets are encrypted.

To eliminate the need to re-establish IPsec tunnels after each handoff, we leverage the fact that when using Mobile IP, while the COA of the MH changes after each IP layer handoff, the IP addresses of the MH and HA remain constant. Therefore, the IPsec tunnel may remain intact. In our model, during Mobile IP registration, the HA simply updates the routing entry for IPsec packets destined to the MH to be forwarded through the new Mobile IP tunnel. This does not require any message beyond the standard Mobile IP registration.

Figure 3.2 depicts this integration model which can be applied to both IPsec and MML-IPsec. The MML-IPsec tunnel is established between the HA, FA and MH, within the Mobile IP tunnel. An advantage of our model is that it does not restrict the scheme of obtaining a COA as that in SecMIP, and provides a seamless integration between Mobile IP and ML-IPsec. Table 3.1 shows the overall handoff flow between the HA, FA, and MH. This illustrates that our proposed model seamlessly integrates Mobile IP with ML-IPsec.

Next, let us consider the the data traffic flow from the MH to the correspondent node (CN). The FA receives the encrypted traffic through the MML-IPsec tunnel or

Table 3.1. Overall Handoff Flow of Integrated Mobile IP and MML-IPsec

MH - Receive an Agent Advertisement (from a new FA) 1. change routing table to set the new FA as a default router 2. send Mobile IP Registration message to the new FA - Receive a Mobile IP Registration Reply message 1. Finished
FA - Receive a Mobile IP Registration message 1. *initiate key distribution** 2. send Mobile IP Registration message to the HA - Receive a Mobile IP Registration Reply message 1. add a new Mobile IP tunnel into routing table for the MH 2. activate MML-IPsec for the MH 3. forward the Mobile IP Registration Reply to the MH
HA - Receive a Mobile IP Registration message 1. reset the previous configuration (MML-IPsec and Mobile IP tunnel) 2. change local configurations to map MML-IPsec into a new Mobile IP tunnel 3. send a Mobile IP Registration Reply message to the FA

*: Optional based on key distribution protocol

IPsec tunnel. The outer IP header has the source address as the MH IP address, and the destination address as the HA IP address. The FA routes this data traffic into the Mobile IP tunnel based on the reverse tunnel of the Mobile IP. Next, let's consider the traffic from the CN to the MH. This traffic is intercepted by the HA. The HA encrypts the traffic and encapsulates it within the Mobile IP tunnel, which has the outer header source address as the HA IP address, and the destination address as the FA IP address. If IPsec is used, when receiving the encapsulated traffic, the FA decapsulates the Mobile IP outer header and transmits the encrypted traffic to the MH. If MML-IPsec is used, the intermediate node decrypts the first zone (packet header), performs some processing, re-encrypts this zone, and forwards the data.

3.2.3 Software Platform

Our implementation is based on Linux FreeS/WAN version 1.99, an open source IPsec implementation available free on the web (under GNU license term), on Linux kernel version 2.4.21. The FreeS/WAN system has two major components: the Pluto Daemon and the Kernel IPsec Support (KLIPS). The Pluto Daemon implements the IKE protocol [15, 16]. We modified portions of the Pluto Daemon to implement our key distribution protocols and to support the interface between the key distribution protocols and the MML-IPsec transport module. We made major modifications to KLIPS to integrate IPsec and SNOOP.

3.2.4 Test Bed

To evaluate the performance of our protocols, we set up a test bed as shown in Figure 3.3. The base stations are DELL Pentium desktops (P4 2.4 GHz), and the MHs are DELL Pentium laptops (Mobile P4 2.4 GHz). These are equipped Orinoco Prism 802.11b wireless cards configured in ad hoc mode so that the desktop machines act as base stations. All these machines are running RedHat 9.0 with Linux kernel version 2.4.21.

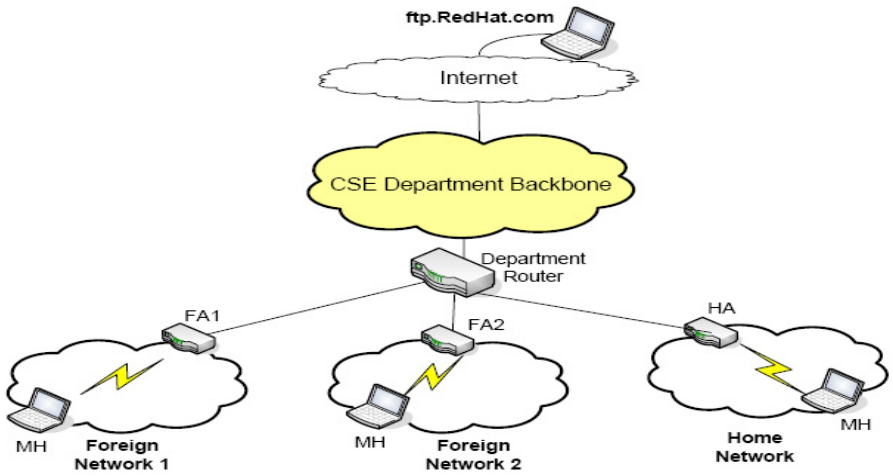


Figure 3.3. Mobile Multi-Layered IPsec Test Bed

3.3 Key Distribution and Mobility Management

In this section we present efficient automatic key management protocols for MML-IPsec integrated with Mobile IP. We include procedures for session initialization and mobility management. Our goal is to enable fast handoffs while maintaining MML-IPsec sessions. In our model, the nodes involved in the MML-IPsec session are the MH, HA, and FA. The MH and HA have access to both zones in the MML-IPsec packets; the FA serves as the intermediate node and has access to the first zone of the packet containing the TCP/IP header. The key management protocols are responsible for establishing the required SAs between these nodes, and for enabling mobility.

The key management protocols have two phases. In the first phase, a MML-IPsec session is established using the initialization procedure. This includes determining if a FA will be involved in the secure session, and hence requires the use of MML-IPsec. The second phase of the protocols supports mobility. We propose two protocols for this purpose. The first, called *Proactive Key Distribution (PKD)*, pre-establishes SAs with not only the current FA, but its neighbors as well. Therefore, when a MH moves to a new FA, the SA already exists. The second, called *Dynamic Key Migration (DKM)*, requires SAs to migrate between FAs as a user moves.

In the following subsections we first discuss MLIKE Initialization, PKD, and DKM. Our description is based on IKE version 2.0 (IKEv2) [16]. We implemented key distribution protocols in both IKE version 1.0 (IKEv1) and IKEv2. These implementations are separate packages since IKEv1 and IKEv2 are not compatible. We present our implementation and performance results.

3.3.1 Initialization

When a MH leaves its home network, it executes Mobile IP registration procedures. In addition, Initialization is invoked. Figure 3.4 shows the Initialization flows between the HA, FA and MH, with the Mobile IP registration.

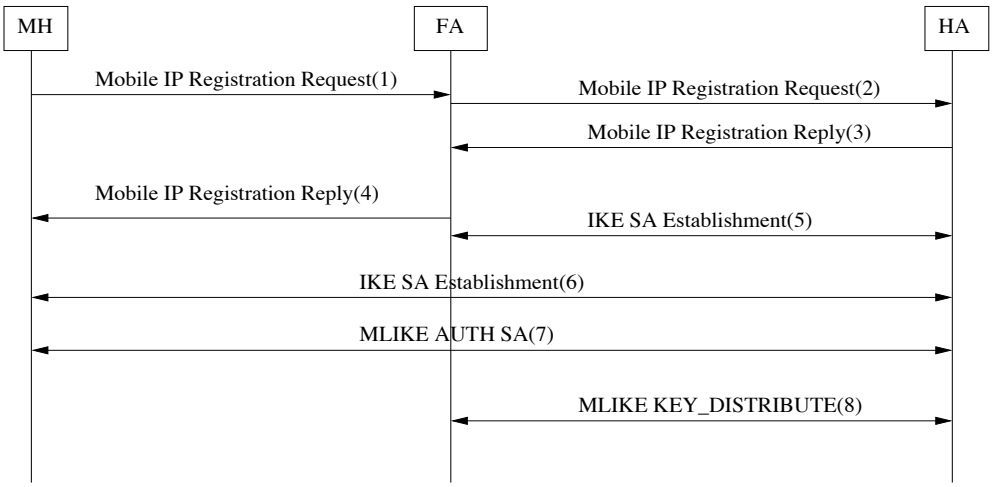


Figure 3.4. Mobile IP Registration and MML-IPsec Key Initialization

The Initialization phase begins after the HA has sent the Mobile IP registration reply to the FA. First, the HA establishes an IKE SA [66, 15, 16] with the FA and MH so that session key information may be exchanged securely. Note that the establishment of the IKE SA between the FA and HA occurs in parallel with sending the Mobile IP registration reply to the MH.

The second step of the Initialization is to establish the CSAs in the MH, HA and FA. The CSA has two elements, a zone map and a zone list. The zone map states the start and stop positions of the zones in the IP datagram. The zone list contains the SAs for all the zones. The HA, FA, and MH create and keep an instance of the CSA. The source and destination (HA and MH) store a complete list of SAs. The FA has a non-null SA in the zone list for the zone that it supports, and a null SA for the zone that it does not support.

The HA and MH setup a MML-IPsec CSA using the MLIKE AUTH SA Exchanges (flow (7) in Figure 3.4). During these exchanges, the MH and HA exchange the complete zone map and SAs to compose a CSA. We define a new payload type called “Zone Map” which delivers the zone map information. In addition, we modify the key exchange protocol to allow for multiple SAs to be included. The secret key values for all zones are

decided in the MML-IPsec AUTH SA Exchanges.

Once the CSA is established between the MH and HA, the HA delivers the CSA to the FA, using the IKE SA with the FA. For the zone to which the FA has access, i.e., the zone covering the TCP/IP header, the HA sends the corresponding non-null SAs to the FA (flow (8) in Figure 3.4) with the corresponding symmetric key values. A new payload, called "SECRET", delivers the symmetric key values.

Upon completion of the Initialization procedure, data transmission using MML-IPsec may take place.

3.3.2 Proactive Key Distribution (PKD)

The goal of PKD is to enable a fast handoff by pre-distributing keys in FAs that are neighbors of the current FA, so that very little overhead is incurred during the real-time handoff. For example, in Figure 3.1, SA1 is placed in both FA1 and FA2 when the session is established. The distribution of the CSA information to these neighboring FAs is performed after the Initialization exchange is complete, so the Initialization overhead will not be increased by PKD. The disadvantage of this approach is that the active key information must be stored in more nodes than are actively being used, thus creating a higher chance of the session key being compromised.

Figure 3.5 shows the PKD protocol flow. The FA, when finished Initialization, notifies the HA of its neighbor FAs (flow (1) in Figure 3.5). The HA establishes an IKE SA to each neighboring FA to transmit ML-IPsec CSA securely (flows (2) and (4) in Figure 3.5). The HA distributes the MML-IPsec CSA information established via IKE SA to the neighbor FAs (flows (3) and (5) in Figure 3.5).

PKD can be performed in two ways: (a) point-to-point sequential key distribution; (b) multicast key distribution. In this thesis, we use the former to simplify the implementation.

When the MH moves to a new FA, the handoff latency is low because the MML-IPsec CSA information is already loaded in the new FA. When the new FA receives a Mobile

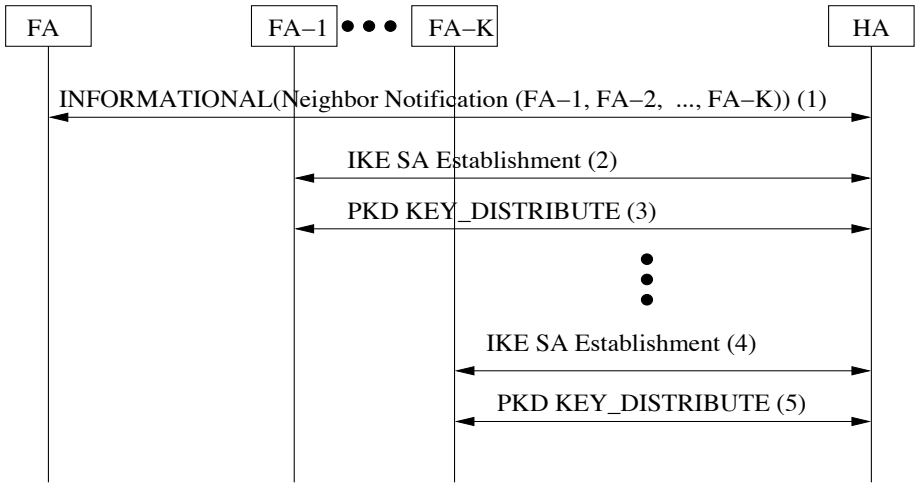


Figure 3.5. Proactive Key Distribution Protocol Flow

IP registration reply from the HA, the FA internally activates the MML-IPsec CSA. The HA only changes the internal binding of the MML-IPsec tunnel to the new Mobile IP tunnel with the neighbor FA.

3.3.3 Directed Key Migration (DKM)

Unlike PKD, in DKM the CSA information is only stored in the FA that is actively serving the MH. Therefore, when a MH changes FAs, the CSA must be migrated from the old FA to the new FA in a secure manner. For example, in Figure 3.1, SA1 must be moved from FA1 to FA2. This method only requires that the CSA be stored in a single intermediate node, but incurs a higher latency than PKD because more signaling is required during the handoff. After a handoff, rekeying may take place as described in Section 3.3.4, so that only one FA has the current CSA.

Figure 3.6 shows the DKM protocol flow. When a MH moves to a new FA, it detects the movement using standard Mobile IP techniques. We modify the Mobile IP Registration [61] by adding an extension to include the previous FA address, as is done in Mobile IP with Route Optimization [67].

First, the MH transmits a Mobile IP registration message with the previous FA

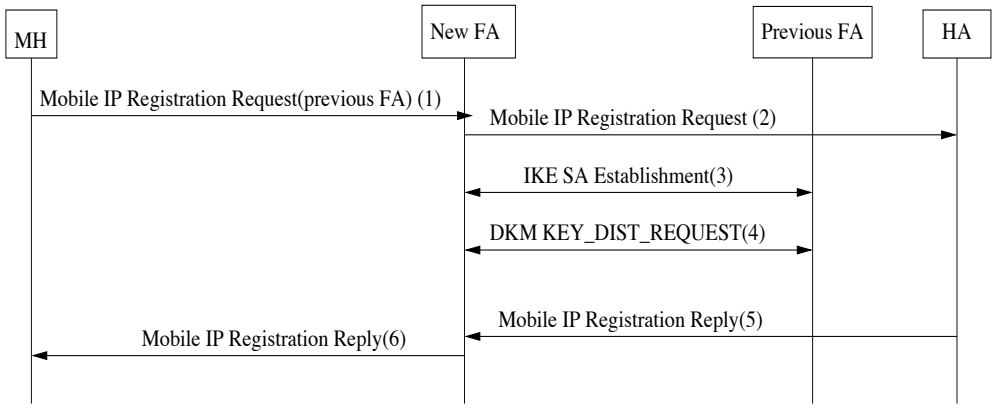


Figure 3.6. Directed Key Migration Protocol Flow

information to the new FA (flow (1) in Figure 3.6). The new FA uses the previous FA information to decide where to retrieve the MML-IPsec CSA information. The new FA initiates the DKM protocol, and relays the Mobile IP registration message to the HA simultaneously.

In DKM, if there is no IKE SA established between the previous FA and the new FA, the new FA establishes an IKE SA with the previous FA so the key information is transferred securely. Once this IKE SA is established, the new FA transmits the MML-IPsec CSA information request to the previous FA (flow (4) in Figure 3.6). The previous FA authenticates the new FA and sends the response to the new FA. The response message includes the MML-IPsec CSA including the secret key values.

Note that the DKM protocol is processed in parallel with the Mobile IP registration between the new FA, HA and MH.

3.3.4 Rekeying and Revocation

There are several reasons why rekeying or key revocation may take place when using MML-IPsec. For example, if a CSA lifetime expires, or a CSA is determined to be insecure, it may be revoked, or if secure communication is still desired, rekeying may take place during which a new CSA is established. Further, key revocation may take place when a Mobile IP tunnel is deleted, for example when a MH returns to its home

network or powers off. Finally, rekeying may take place after a handoff, if the number of FAs that share the CSA exceeds a threshold value. Note that in DKM the threshold value is one since only the current FA has the CSA.

IKEv2 [16] defines rekeying procedures so that the peers can initiate the establishment of a new IPsec SA, while the old IPsec SA is active. This minimizes the interruption of data transmission. In the same way, a new MML-IPsec CSA is established while the old one is being used. Once the new CSA is established, the MH, FA and HA change to use the new CSA for data transfer, and the old CSA is deleted.

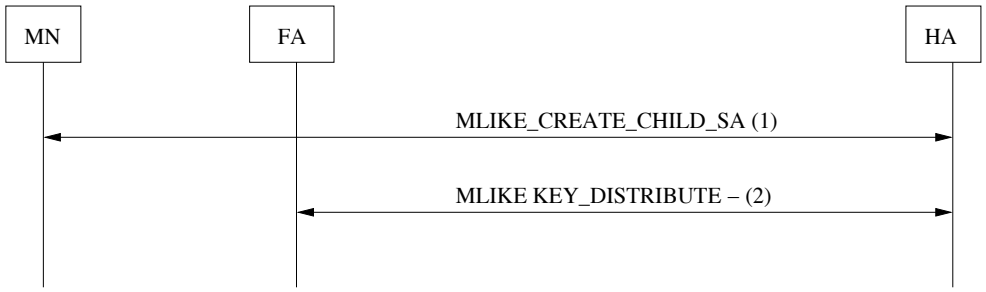


Figure 3.7. MML-IPsec Rekeying by lifetime expiration

Figure 3.7 shows the CSA rekeying which is triggered when MML-IPsec CSA lifetime expires. Either a HA or a MH initiates the rekeying (flow (1) in Figure 3.7). The HA distributes the new CSA to the current FA in which the MH resides (flow (2) in Figure 3.7).

In DKM, a new FA retrieves a MML-IPsec CSA from the previous FA. If the secret information remains with the previous FA, it increases the vulnerability of an attack. To address this vulnerability, the HA and the MN reestablish a new MML-IPsec CSA, where only the key values for the header zone are changed. The HA distributes this new CSA to the new FA. The new FA uses the old CSA until the new CSA is established. After establishing the new CSA, the HA revokes the old CSA from the previous FA.

Figure 3.8 shows the rekeying and revocation procedures in DKM. After retrieving the MML-IPsec CSA from the previous FA, a new FA requests rekeying to the HA (flow (2)) The HA initiates rekeying (flow (3)) and, in the same way as Figure 3.7, distributes

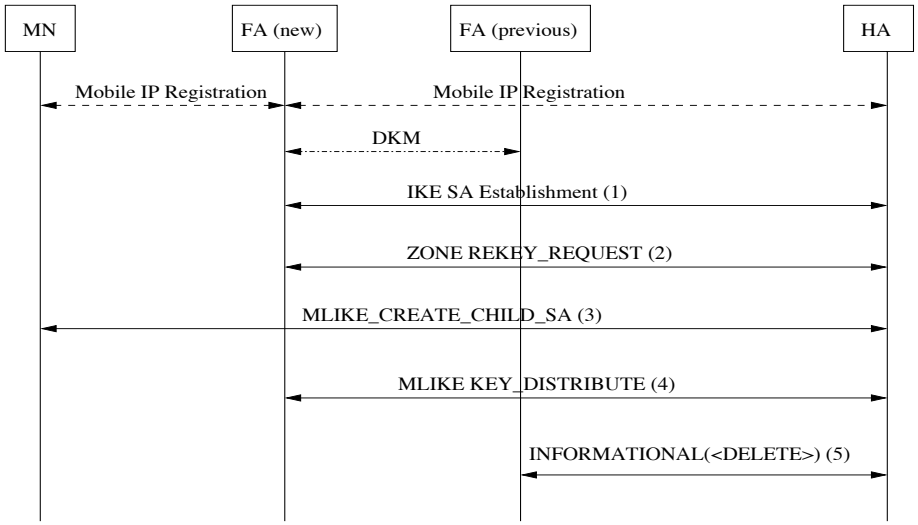


Figure 3.8. MML-IPsec Rekeying/Revocation in DKM

the new CSA to the new FA (flow (4)). The old CSA is deleted from the previous FA (flow (5)).

In PKD, as a MH moves, the MML-IPsec CSA is distributed to more FAs. As the number of FAs with the CSA increases, the less secure MML-IPsec is. To address this problem, the HA initiates rekeying when the number of FAs having the MML-IPsec CSA exceeds a threshold (a system parameter). First, the HA rekeys the MML-IPsec CSA with the MH, where only the keys for the header zone are changed (flow (2) in Figure 3.9). Second, the HA distributes the MML-IPsec CSA to a new set of neighboring FAs (flows (3) - (6) in Figure 3.9). Finally, the HA exchanges INFORMATIONAL (flows (7) and (8) in Figure 3.9) to delete the previously distributed CSA from S = {a set of FAs | A - B}, where A = {a set of existing neighboring FAs} and B = {a new set of current FA's neighboring FAs}.

3.3.5 Implementation

We implemented the key distribution protocols (MLIKE Initialization, PKD, and DKM) on the test bed shown in Figure 3.3. We implemented MML-IPsec key distribution protocols based on IKEv1 and IKEv2. We describe the IKEv2 implementation. The Dynamic

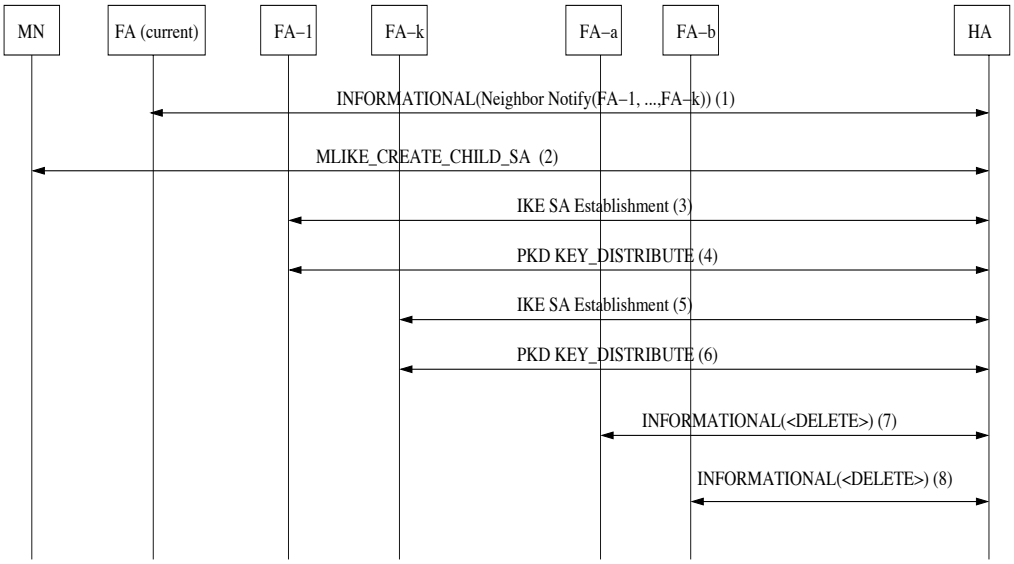


Figure 3.9. MML-IPsec Rekeying/Revocation in PKD

Mobile IP Linux implementation developed by Helsinki University of Technology (HUT) [68] was used in the test bed. Each subnetwork has a different set of wireless configuration parameters such as *essid* and *channel number*. The handoff occurs when the wireless configuration parameters are changed. We manually trigger handoffs through a shell script to run controlled experiments.

The implementation consists of several blocks: ML-IKE, PKD, DKM, Key Management, and the interface to MML-IPsec transport. The communication between these blocks is via the Unix Domain Socket in Linux.

ML-IKE manages, negotiates, and establishes the MML-IPsec CSA for Initialization, while PKD and DKM support mobility. ML-IKE also manages the state machine of the protocols.

The Key Management keeps the establishment state of both IKE SA and MML-IPsec CSA based on the source and destination addresses. Using the connection status, it coordinates Mobile IP and ML-IKE. If there is already an established MML-IPsec CSA in the case of PKD, the key management activates MML-IPsec through adding the route entry and IPsec binding. Otherwise, it initiates the establishment of the IKE SA

if necessary.

To implement the interface between the key exchange protocols and the MML-IPsec module, we modified the Pluto Daemon by adding a new user interface to construct the CSA. The interface constructs a newly-defined zone message and sends the message to the PF_KEYv2 [69] socket. PF_KEYv2 is a new socket protocol family used by trusted privileged key management applications (e.g., ML-IKE, PKD and DKM) to communicate with the operating system's key management internals (i.e., FreeS/WAN's Security Association Database (SADB)). We have modified the PF_KEYv2 source code to handle the newly-defined zone message. The PF_KEYv2 socket will construct the CSA when it receives a zone message.

To implement these protocols based on IKEv1, we extended the IKE of FreeS/WAN [70], which is running in the Pluto Daemon, to include the new exchange types described previously. Furthermore, we implemented IKEv2 and key distribution protocols, using only a small part of Free/SWAN since FreeS/WAN does not support IKEv2.

3.3.6 Performance

In this section, we discuss the performance of the key management protocols measured on our test-bed. We implemented and tested MML-IPsec key distribution protocols based on IKEv1 and IKEv2. In this section, we show the performance of IKEv2-based implementation. We tested the performance of our integrated IPsec/Mobile IP solution as a baseline.

In Table 3.2, we show the message processing time measured by the *gettimeofday* system call in Linux. The results in Table 3.2 measure the processing time on receiving a message or event, excluding the pre-processing, post-processing and transmission time over the media. The node initiating the message flow is the "Initiator," and the recipient is called the "Responder". For example, in DKM, the Initiator is the new FA and the Responder is the previous FA providing the CSA information. In Initialization, the MH is the Initiator and the HA is the Responder.

Table 3.2. Message Processing Time

	Processing Time		
Message Flow	Initiator	Responder	Total
IKE SA Establish	13.3 ms	13.42 ms	26.72 ms
MLIKE CSA Establish	53.2 ms	29.1 ms	82.3 ms
MLIKE CREATE(rekey)	1.3 ms	1.35 ms	2.65 ms
MLIKE Key Distribution	0.4 ms	0.2 ms	0.6 ms
PKD Key Distribution	0.3 ms	0.2 ms	0.5 ms
DKM Key Distribution	0.3 ms	0.2 ms	0.6 ms
Neighbor Notification	2.0 ms	7.3 ms	9.3 ms

In order to measure the handoff latency, we evaluate the time delay from the point that the MH sends a Mobile IP registration message until it finishes establishing the MML-IPsec CSA or IPsec SA. Table 3.3 shows the handoff delay for the pure Mobile IP, Mobile IP integrated with IPsec, and Mobile IP integrated with MML-IPsec.

The Mobile IP registration is necessary for all cases. The pure Mobile IP handoff comprises only the Mobile IP registration overhead and shows almost the same delay for the initialization and handoff phases. In the case of Mobile IP with IPsec, the initialization time consists of the Mobile IP registration and IPsec secure connection establishment between the HA and MH.

Similarly, the initialization of MML-IPsec takes between 539 - 542 milliseconds depending on if PKD or DKM is used. This includes the Mobile IP registration and MML-IPsec establishment with the key distribution to the FA or FAs. The MML-IPsec initialization time is made up of three components. First, the message processing time, including IKE SA establishments, MLIKE AUTH SA Exchanges, MLIKE Key Distribution, and Mobile IP registration procedures, is measured at approximately 220 milliseconds. Second, the FA runs four shell scripts to create local connections and bind the local connections into MML-IPsec interfaces when receiving the Mobile IP registration reply. The four shell scripts take 320 milliseconds to execute. Finally, there are transmission and internal communication latencies which are responsible for the remaining of the delay. These shell commands, and their additional overhead, may be eliminated if

the source code for the various initialization procedures are modified to interact directly, a change planned for our next version.

In Table 3.3, we show the total latency for the Initialization (latency in parenthesis if the shell script overhead is eliminated), and handoff delay.

Table 3.3. Handoff Delay

Phase	Pure MIP	MIP with IPsec	MIP with MML-IPsec	
			PKD	DKM
Initialization	26 ms	197 ms	539(219) ms	542(222) ms
Handoff	25 ms	54 ms	52.7 ms	100.0 ms

Note: MIP is an abbreviated form of Mobile IP.

The MIP with IPsec column in Table 3.3 shows the initialization and handoff delay without performance enhancement functions in the FAs. The initialization of Mobile IP with IPsec takes 197 milliseconds, which includes Mobile IP registration (26 milliseconds), processing time of IKE SA and session SA establishment (109 milliseconds in the total shown in Table 3.2), IPsec message transmission delay, and internal configuration in the HA and MH. Here, the handoff delay takes only 54 ms.

However, if IPsec is integrated into FAs to deliver the performance enhancement and security functions together, the handoff may take at least as long as the initialization (197 ms). The performance enhancement functions require FAs to use the partial information (e.g. TCP/IP header) of the encrypted packet. In order to use the partial information, the FA must access the information in plaintext. Therefore, whenever a MH moves to a new foreign network, IPsec should be re-established between the HA and new FA, and between the new FA and MH.

On the other hand, the handoff in the MML-IPsec only takes 52.7 ms and 100 ms in PKD and DKM respectively. Thus, the MML-IPsec improves the overall handoff delay in an integrated setting of performance and security functions. The MML-IPsec Initialization in IKEv1 takes about 110 milliseconds more than in IKEv2. The handoff delay measurements show that a handoff using PKD incurs an additional 28 milliseconds of delay, while a handoff using DKM incurs an extra 75 milliseconds of delay. The handoff

delays with IKEv1 are similar to IKEv2. These results are encouraging when we consider that these are well within the range of a TCP time-out value.

3.4 Discussion

In this chapter we have presented a simplified version of ML-IPsec, an efficient key distribution protocol for initializing secure wireless sessions, and two protocols for managing mobility for these secure sessions. We call this suite of protocols MML-IPsec. We showed through extensive performance testing of our implementations of these protocols that MML-IPsec successfully enables performance enhancing algorithms to be introduced into wireless networks. Specifically, we also support revoking keys in nodes that are no longer in an active route and rekeying without disrupting data transfer. We find the resulting MML-IPsec protocol, when coupled with SNOOP, greatly increases throughput over scenarios using standard TCP over IPsec (165% on average) [71].

Chapter 4

Anonymous and Secure Reporting of Traffic Forwarding Activities in Ad Hoc Networks

4.1 Background

The establishment of a wireless infrastructure is non-trivial, especially in volatile environments in which node mobility dominates. Occasionally, erecting fixed infrastructures is not feasible due to location or temporal validity. In the absence of a fixed infrastructure, mobile ad hoc networks (MANETs) can be used. By not requiring a fixed infrastructure or centralized control for communication, MANETs are well suited for both mission oriented and civilian applications. Within the network, multi-hop paths are created between nodes that formerly could not communicate. Ideally, each node selflessly forwards each packet to the next node in the path. As nodes move, they leave and join various communication links, thus promoting many ephemeral paths.

Reliable operation in a MANET requires explicit cooperation between nodes. While this is feasible to assume for mission-oriented scenarios, careful consideration needs to take place when applying MANETs to civilian applications. In a civilian mobile ad hoc

network, communicating nodes will use any relay points available. It is conceivable that selfish or malicious nodes exist in these networks. Therefore, there is a need to detect selfish or malevolent behavior and promote cooperation between nodes.

To deal with the selfishness of nodes in civilian ad hoc networks, researchers have proposed many solutions. These solutions encourage nodes to cooperate either by remunerating cooperative behavior or by penalizing malicious and selfish behavior [25, 39, 40, 41, 32]. Most of these solutions are based on information gathered through monitoring of neighbor's behavior.

One method for detecting malicious behavior is to generate reports on traffic flow between nodes. This information can be used to not only detect misbehavior, but also to indicate good network citizens. By identifying nodes that play fairly or are malicious, nodes can better choose with whom to cooperate.

In order for these techniques to function properly, it is imperative to securely and reliably collect traffic reports. Previously proposed monitoring and reporting solutions rely on neighboring nodes to eavesdrop on data transmissions of other nodes in order to generate reports. While this may work well in networks with trusted nodes, e.g., in military settings, it is not feasible for civilian ad hoc networks. Considering the selfish nature of nodes in a civilian network, we cannot expect that they are willing to monitor others and collect the information to report. Furthermore, such reporting techniques assisted by neighboring nodes may not work well if directional antennas are used. Unlike the omnidirectional antennas, the coverage area of directional antennas is of a sector shape and this limits the ability of eavesdrop.

To address these problems, we propose an anonymous and secure random reporting protocol (ASR) in which contributions of intermediate nodes on data traffic forwarding are securely and reliably collected. In ASR, intermediate nodes on a path generate a self-report of traffic forwarding: every delivered data packet initiates a report from one intermediate node that is randomly chosen by a source node; the chosen node then integrates its *self-report* into the data packet. We use a symmetric-key construction for

selecting the reporting node that efficiently prevents disclosure of the selected node's identity. Note that reports may become lost due to mobility and congestion. In order to provide robustness in the face of loss, the report is sent to the either source, destination, or both.

While the basic secure random reporting protocol provides secret node selection, as well as integrity and authenticity of reports, it does not guarantee that the self-report is accurate. Although nodes cannot manipulate others' reports, they may not be trusted to generate accurate reports. To rectify this inadequacy, we propose a forgery detection scheme that provides proofs of delivery implemented by secure network layer acknowledgments.

Because only the selected node modifies the report field, eavesdropping nodes may observe a node's incoming and outgoing packets to determine whether it has generated a report. To thwart this attack, an efficient report wrapping scheme is proposed along with the secure random reporting protocol.

We qualitatively analyze the security that ASR affords against single node misbehavior and colluding nodes. We have simulated our approach using ns-2 [72]. Our results show that ASR accurately monitors packet forwarding activity even in lossy networks. We further simulate malicious packet dropping to determine the effectiveness of ASR. Finally, we analyze the performance and overhead of ASR.

The remainder of this chapter is organized as follows: Section 4.2 describes possible threats in civilian ad hoc networks and assumptions that we use throughout this chapter. Section 4.3 presents an overview of the proposed random reporting protocol. This scheme is then strengthened in Section 4.4 as we extend it to provide report integrity, node selection confidentiality, and prevention of falsified reports. We qualitatively analyze security of ASR in Section 4.5. Section 4.6 provides simulation results and overhead of the secure random reporting protocol. In Section 4.7, we discuss the trade-offs of ASR.

4.2 Model and Assumptions

4.2.1 Threat Model

Civilian ad hoc networks are prone to self interest and malicious behavior. The most straightforward threat to civilian networks is a *denial of service* (DoS). For example, a misbehaving node may simply refrain from participating in routing. However, in such a case, the node is never placed on a path. A possibly more damaging attack occurs when a node acquires a position on a path, but selectively drops packets to degrade performance. We address this latter attack. Our protocol does not aim to prevent or detect attacks on the routing protocol (e.g., the former case), but rather focuses on secure reporting of packet forwarding (e.g., the latter case). Since the forwarding activities are dictated by reports, nodes may attempt to misrepresent themselves by creating, manipulating, or dropping reports in order to veil their misbehavior. For example, a node that maliciously drops packets will specifically wish to drop packets reporting its behavior.

Malicious or selfish nodes can also collude in creating or dropping reports. This collusion exists in different forms: collusion between nodes not on a routing path, collusion between nodes on a path and nodes not involved in forwarding, collusion between non-adjacent nodes on a path, and collusion between adjacent nodes on a path. In the first type, colluding nodes eavesdrop communications and claim that they cooperatively have forwarded packets. In the second form, nodes on a path may give collected traffic information to other nodes not on a path and generate a report. Similarly, non-adjacent nodes on a path can collude in exchanging information to hide their misbehavior. Finally, adjacent selfish or malicious nodes on a path can easily collude in hiding the existence of their packet drops.

More subtle attacks also exist. To gain an advantage, a malicious node may inject fake packets. This expends the energy of all forwarding nodes, thereby rendering them incapable of forwarding future legitimate packets. The known defense for this attack is to use interleaved hop-by-hop authentication schemes [73, 74], in which fake packets are

filtered mid-transmission. We do not address this attack.

4.2.2 Assumptions

We assume that there is no centralized server or Internet connection in MANETs. In this setting, the source and destination must protect their own flows from being disrupted by malicious nodes. Therefore, we assume that the source and destination police their own flows. They collect reports and determine which node is behaving maliciously for their flow. Based on the collected information, they can change to a new routing path which does not include misbehaving nodes, refrain from being using paths containing these nodes in the future, or refuse to forward packets for these nodes. The source and destination's actions, once they discover a malicious node, are out of the scope of this work.

For the purposes of this work, we assume the use of a source routing protocol such as dynamic source routing (DSR) [75] so that the full path information is available. The full path information is used as a basis to verify which nodes participate in forwarding packets. In this thesis, we propose a secure random reporting scheme in which intermediate nodes on a path generate reports of their cooperation on forwarding data traffic. We assume that routing protocol security is provided by some other means [76]. We also assume that good-behaving nodes follow the rules of the control and routing protocols.

Our protocol is based on symmetric key cryptosystems. We assume that there exists an efficient key management scheme to establish pairwise keys. Symmetric cryptography is appropriate for ad hoc networks, due to the limited computational power and battery power of mobile devices. Key management has been actively studied in ad hoc and sensor networks, such as probabilistic key pre-distribution [77, 78, 79] and symmetric polynomial based key establishment [80]. The key management is a separate important issue and out of scope of this work.

4.3 Overview of the Random Reporting Protocol

In this section, we present a basic random reporting protocol. The key idea is that each intermediate node only needs to keep track of its own contribution, instead of observing the actions of other nodes. This use of intermediate nodes is appropriate for a civilian ad hoc network. By introducing randomness, it is more difficult for an adversary to discover, delete, or modify reports. Furthermore, to address packet manipulation and network dynamics, we propose three random reporting protocols: Random Reporting Node Selection (RRNS), Random Reporting Node and Direction Selection (RRNDS), and Random Bidirectional Reporting (RBR). We present these protocols here to describe the basic reporting operations and motivations. We use this protocol as the basis of the secure random reporting protocol of Section 4.4. We detail these protocols in the following subsections.

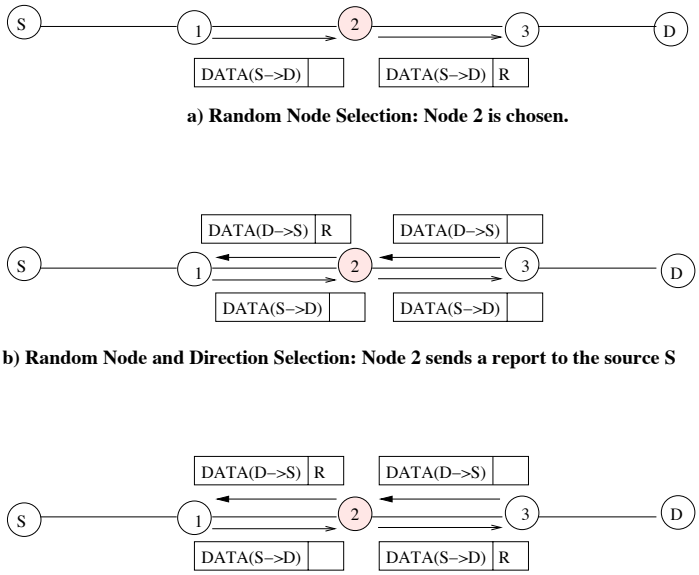
4.3.1 Random Reporting Node Selection (RRNS)

For every data packet, the source randomly chooses one intermediate node to generate a report to the destination. This is accomplished by coupling each data packet with a report, therefore when the destination receives the packet, the relaying activity of intermediate nodes can be dynamically observed.

In RRNS, if the path consists of n intermediate nodes, any node can be chosen with probability $1/n$. Figure 4.1-(a) illustrates RRNS where node 2 has been randomly chosen. In general:

1. For every data packet p, source S randomly chooses (uniform distribution) an intermediate node n_i to generate a report for this flow. S attaches the identifier of randomly chosen node to the packet.

2. For a packet p with selected node n_i, n_i attaches report R to p and forwards p carrying report R towards destination D.



a) Random Node Selection: Node 2 is chosen.

b) Random Node and Direction Selection: Node 2 sends a report to the source S

c) Random Bidirectional Reporting: At node 2, report is transmitted to both S and D

Figure 4.1. Random Reporting Protocol: source S and destination D: node 2 is selected to generate a report

3. Destination D receives p, maintains and periodically analyzes the forwarding activity of all intermediate nodes involved in forwarding, looking for traffic deviations.

The idea of choosing a random node is similarly used by [81, 82] for micro-payment, in which a randomly chosen transaction is used for a merchant to deposit some amount of money.

Since the intermediate node is selected randomly, other nodes are unable to predict the selection schedule. While the randomness provides better reports, the described scheme is vulnerable to attack. Without taking precautions, reports may be manipulated by downstream nodes with selfish intentions. Section 4.4 addresses this by introducing a secret node selection scheme.

4.3.2 Random Reporting Node and Direction Selection (RRNDS)

In RRNS, if malicious intermediate nodes are located close to the destination, they may drop packets which contains reports from upstream nodes. In this case, the destination

may receive reports only from these malicious nodes and misinterpret the location of the problem.

Random Reporting Node and Direction Selection (RRNDS) is proposed to make RRNS more robust. RRNDS extends Step 2 of RRNS by allowing the chosen node to decide the direction to send the report. If the report is sent towards the destination, it is attached to the data packets, just as in RRNS. On the other hand, if a source-bound direction is chosen, the report may be piggybacked on traffic on the reverse path, or a separate report message is transmitted. Figure 4.1-(b) shows this scheme.

4.3.3 Random Bidirectional Reporting (RBR)

The report in RRNDS is transmitted to either the source or the destination. Unfortunately, this reduces the amount of report information received by the source or destination. This shortage of reports may cause the source or destination to imprecisely analyze the relaying activity of intermediate nodes.

We address this problem by modifying Step 2 of RRNS to transmit the report to both the source and destination. This technique, shown in Figure 4.1-(c), is referred to as Random Bidirectional Reporting (RBR). In the figure, node 2 sends a report to the destination and source node. Simulation results reported in Section 4.6 show that bidirectional reporting improves effectiveness in the face of mobility. Finally, just as in RRNDS, if the communication between source and destination is bidirectional, source-bound reports are attached to data packets destined for the source. This reduces communication overhead.

4.4 Secure Reporting Protocol

The random reporting protocols discussed in Section 4.3 are based upon random node selection. If intermediate nodes (selfish or malicious) discover a packet including a report and the selected node, the information may be manipulated or dropped. This section

proposes efficient constructions that conceal the node selection from other intermediate nodes, and provide a forgery detection scheme.

The following notation is used in the following subsections to describe the anonymous secure reporting protocol.

- ID_i: Identifier of node n_i.
- K_{ij}: A pair-wise key between node n_i and n_j.
- $hash(x)$: Cryptographic hash function computation for x
- σ: $HMAC(K_{SD}, DATA|ID_i)$ computation result for the data and ID_i.
- $DATA$: Data transmitted between the source and destination.
- R_f: Report for the forward traffic.
- R_b: Report for the backward traffic.
- H_R: HMAC result over forward and backward reports of node n_i, H_R=$HMAC(K_{iD}, R_f|R_b)$.
- $E_{K_{ij}}(X)/D_{K_{ij}}(X)$: Encryption/Decryption for X with a symmetric key K_{ij}

4.4.1 Secure and Random Reporting Protocol

Based on the path information, the source node chooses one intermediate node n_i uniformly at random, and computes $Token$, which is added to the data packet. The $Token$ contains the node selection information which is not disclosed. Using an $HMAC$ in the computation of the $Token$ provides both randomness and secrecy in the node selection. The selected node identifier is contained in the encrypted data so that a destination easily discovers it by decrypting the data packet, while other nodes cannot. The source performs the following operations: choose one intermediate node n_i and encrypt data with the selected node identifier, $E_{K_{SD}}(DATA|ID_i)$; compute $\sigma = HMAC(K_{SD}, DATA|ID_i)$; compute $H_i = hash(K_{Si}|\sigma)$; generate $Token = \sigma \oplus H_i$; send a packet, $[DATA, \sigma, Token]$, to the first intermediate node.

When a node receives a packet, it needs to determine if it is the randomly selected node. Upon receiving a data packet, $[DATA, \sigma, Token]$, an intermediate node n_j computes $H_j = hash(K_{jS}|\sigma)$ and XORs it with the received $Token$. If the result of the XOR operation is equal to the received σ, the node knows it was chosen. This is only satisfied at node n_i since the source uses a pairwise key K_{Si} to compute the Token. Since the above test in other intermediate nodes is not satisfied, they do not generate reports.

Table 4.1. Random and Secure Reporting

Source: - Choose one intermediate node n_i and encrypt, $E_{K_{SD}}(DATA\|ID_i)$ - Compute $\sigma = HMAC(K_{SD}, DATA\|ID_i)$ - Compute $H_i = hash(K_{Si}\|\sigma)$ - Compute $Token = \sigma \oplus H_i$ - Send the packet $(DATA, \sigma, Token)$
Intermediate Node n_i: - Compute $H_i = hash(K_{iS}\|\sigma)$ - XOR H_i with $Token \rightarrow H_i \oplus Token = H_i \oplus \sigma \oplus H'_i$, where H'_i is received - Check if $XOR(Token, H_i) == \sigma$ - If n_i is chosen, **o** Generate $Report = [R]$ and attach it to the data packet $\rightarrow$ next hop node: $[DATA, \sigma, Token, Report]$
Destination: - Check the integrity of data packet and decrypt it to find out the chosen node. - Save the report and check whether there exists a misbehavior. - If the report is not valid, ignore the report.

The chosen intermediate node n_i sends its report by attaching it to the data packet. The report R includes the number of packets the node has forwarded for the flow. This scheme is resilient to report manipulation in which node n_k replaces $Token = \sigma \oplus H_i$ with $\sigma \oplus H_k$, because the resulting $HMAC(K_{SD}, DATA|ID_k)$ is not equal to the received $\sigma = HMAC(K_{SD}, DATA|ID_i)$. Without knowing K_{SD}, the node n_k cannot change σ to masquerade as a selected node. When receiving a data packet, the destination checks that σ and the received report R are valid, i.e. if $\sigma = HMAC(K_{DS}, DATA|ID_i)$ is satisfied.

If the selected node generates an extra-packet to send the report to the source,

this will be visible to neighboring nodes. To deal with this problem, the report field consists of two subfields: forward report (R_f) and backward report (R_b) which indicate the number of forwarded packets for destination-bound and source-bound traffic flows, respectively. A destination-bound packet selects a node to generate both forward (source to destination) flow and backward (destination to source) flow reports. Similarly, a source-bound packet is processed such that *Report* contains $[R_f, R_b]$.

The key idea of node selection is to conceal the node selection from other nodes. Table 4.1 summarizes the secure random reporting protocol.

4.4.2 Report Forgery Detection Scheme

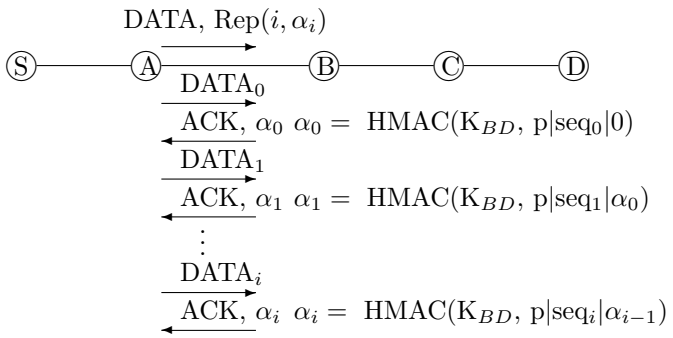
Even if the reporting node selection is secure, we still need to assure that a report is truthful. To address this, a report forgery detection scheme is proposed.

In many wireless networks, such as 802.11, link level acknowledgments (ACK) are used to help overcome the losses on the wireless links due to transmission errors or mobility. We used the link level acknowledgments to provide information for the forgery detection.

The ACK allows a receiving node to confirm to a sending node that a packet has been received successfully. If the sending node does not receive an ACK from the receiver after several retransmissions, the link is considered broken. In many common protocols, such as DSR, an error message, called Route Error, is sent back to the source node. If this occurs, the source node will change the data path.

Each data packet is sent in a link layer frame. For each data packet i, successfully received in frame j_f, the receiver sends an ACK(j_f) consisting of seq_i, α_i, and β_i. α_i is an HMAC for path p, the packet sequence number (seq_i), and α_{i-1}, while β_i indicates an HMAC for path p, the packet sequence number (seq_i), and β_{i-1}.

The receiver r uses pairwise keys K_{rS} and K_{rD} as the symmetric key for α and β respectively, since the report is transmitted to both the source and destination. The forgery detection scheme does not use a key shared between two neighboring nodes in



Node ID	Reported Chained HMACs
A	$\alpha_i = HMAC(K_{BD}, p\|seq_i\|\alpha_{i-1})$
B	$\alpha_{i+1} = HMAC(K_{CD}, p\|seq_{i+1}\|\alpha_i)$
C	$\alpha_{i+2} = HMAC(K_D, p\|seq_{i+2}\|\alpha_{i+1})$

Flow(S, D) Table at destination D

Figure 4.2. Chained HMACs to Detect Report Forgery: Suppose that A, B, and C are selected to generate a report in sequence for packets seq_{i+1}, seq_{i+2}, and seq_{i+3}, respectively.

order to prevent these nodes from colluding and allowing one node to manipulate the HMAC. This chained HMAC requires intermediate nodes to only maintain current state information, not the history of packet transmissions.

Figure 4.2 provides an example flow of chained HMACs for the report forgery detection scheme. Suppose that data transfer begins with initial packet $DATA_0$. For this first packet, node B computes $\alpha_0 = HMAC(K_{BD}, p|seq_0|0)$ and $\beta_0 = HMAC(K_{BS}, p|seq_0|0)$. Node B then sends to A a link layer ACK carrying the computed α_0 and β_0. After the initial packet, node B uses the previous HMACs, α_{i-1} and β_{i-1}, in the computation of α_i and β_i. That is,

$$\alpha_i = HMAC(K_{BD}, p|seq_i|\alpha_{i-1})$$

$$\beta_i = HMAC(K_{BS}, p|seq_i|\beta_{i-1})$$

In the case of Figure 4.2, since the report direction is towards the destination, the report is attached to a DATA transmission.

Mobility is fundamental to MANETs. When a path changes, nodes may not have the sequence number for a particular path. Moreover, a node on a path may selectively drop packets. Hence, it is necessary to include a sequence number map (*smap*) in the report

so that the source and destination receiving the report can determine if it is correct. The sequence number map is a bit representation of forwarded packets, which starts with the sequence number (seq_s) of the first packet received after the previous report has been generated. For example, suppose that after generating a report, node n_i receives packets having sequence numbers 3 (seq_s = 3), 4, 5, 8, and 10. The sequence number map is "11100101" (left to right).

The report generated by an intermediate node is composed of the number of forwarded packets ($Pkts$=5), the sequence number (seq_s=3), the sequence number map ($smap$), and the latest α_i. In summary, the reports have the following format:

$$[Pkts, seq_s, \text{ smap}, \alpha_i].$$

Since the destination knows the key, K_{BD}, it can verify α_i was created correctly. The most recent HMAC, α_i, can then be verified by computing the HMAC chain from the initial α with the initial sequence number and sequence map information.

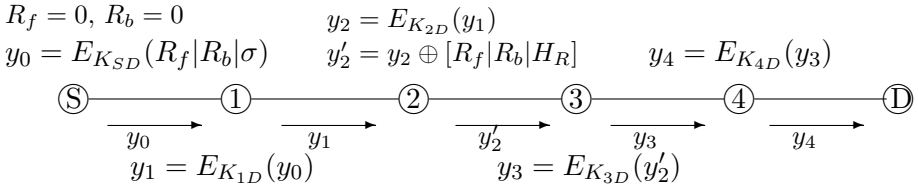
4.4.3 Report Wrapping Scheme

While the secure random reporting protocol protects the identity of the selected node from in-path adversaries, it is susceptible to traffic analysis. If an adversary can overhear traffic going in and out of node n_i, determining whether or not n_i was selected is trivial. If the incoming and outgoing data does not match, n_i has attached a report. Therefore, if this adversary is downstream from n_i, it can selfishly strip out the report.

A great deal of research has been done to provide anonymous communication in the Internet using a proxy (Mix, Jundo, and Onion Router) [45, 49, 47]. Different approaches [54, 55] have been proposed in ad hoc networks, considering features such as mobility, congestion, and energy and computation limitations. These existing solutions aim to provide anonymous communication for data packets. Our goal is to provide anonymity of the reporting node from eavesdropping nodes. We propose an efficient transformation scheme in which eavesdropping nodes may not discover whether a node

generates a report.

Every node on the routing path encrypts the received report, $y_i = E_{K_{iD}}(y_{i-1})$ where y_{i-1} is the received report generated by the previous node. In this way, the report field is altered by every node on a path. The encrypted report field appears random to other nodes. Unlike other intermediate nodes, the selected node first encrypts y_{i-1} and XORs the encrypted value with its report value. Figure 4.3 shows how each intermediate node processes the report field. Nodes use a pairwise key with the destination as the encryption key. Although the initial value of R_f and R_b at a source node is 0, σ, which is a message authentication code, makes the encrypted result random.



(a) Report transformation in each intermediate node

$$y_2 = E_{K_{2D}}(E_{K_{1D}}(E_{K_{SD}}(0|0|\sigma)))$$
$$y_2' = D_{K_{3D}}(D_{K_{4D}}(y_4))$$
$$[R_f|R_b|H_R] = y_2 \oplus y_2' = y_2 \oplus y_2 \oplus [R_f|R_b|H_R]$$

(b) Operations at destination node D

Figure 4.3. Report transformation against traffic analysis attack: node 2 is selected to generate a report

A destination knows all the nodes en route and the selected node (n_i). The destination can generate the encrypted report field value ($y_i = E_{K_{iD}}(y_{i-1})$) by repeatedly encrypting the report field, from a source to a selected node in sequence. It decrypts the received report field until it recovers the report field transmitted by the selected node, R_i'$= y_i \oplus [R_f|R_b|H_R]$. The destination node XORs the two values ($y_i \oplus R_i$') which outputs the report $[R_f|R_b|H_R]$ generated by the selected node n_i.

The above scheme explains how the destination-bound report field is processed in every node. For the source-bound report field, the operations are the same. Let z_i denote the source-bound report field that a node n_i generates. The report field is $z_i = E_{K_{iS}}(z_{i-1})$.

4.5 Security Analysis

In this section, we qualitatively analyze the security that ASR provides against the misbehavior of both single node and colluding nodes on reporting of traffic forwarding activities. Primarily, we analyze selfish behavior that harmfully affects the reporting scheme. We begin by looking at attacks from a single node, and then progress to situations of multiple node collusion.

4.5.1 Single Node Misbehavior

The goal of a single malicious node is to be placed on a forwarding path, and then to drop packets to degrade the performance of the flow. The malicious node will attempt to evade discovery as long as possible. To do this, it may drop, manipulate, or fabricate reports to hide its behavior. In this subsection, we present how ASR combats against these single node attacks.

In ASR, intermediate nodes cannot determine which other node on a path is chosen to generate a report, but can only determine whether they are chosen or not. A greedy node may use this knowledge to drop all packets but those that contain its own reports. In this case, either the source or destination will change the path because they will only receive reports from a single node on the path.

Consider the cases in which an intermediate node manipulates the reporting node selection or a report to hide its behavior. For instance, the intermediate node may replace the token in a packet with a random value, which results in no report in the packet. Barring a link break, the packet containing the replaced token will be transmitted to the destination. The destination checks packet integrity and decrypts the packet to determine the selected node's identifier. Next, it checks the token integrity and determines that the token was manipulated. Similar to the first case, the destination or source can establish a new path since it determines that one of intermediate nodes on the current path is not trustful.

Let us suppose that in Figure 4.2, node A forwards a packet and node B drops the

packet. In this case, node B may purposefully choose to not send an ACK to node A to hide its malicious behavior. If this occurs, however, node A will send a route error message to the source. The source will change to a new data path.

Instead of withholding ACKs, node B may attempt to hide its selfish behavior by sending an ACK to node A with a random value for the chained HMAC and subsequently dropping the packet. In ASR, the destination can determine if α is incorrect. As shown in Figure 4.2, the destination retains knowledge of the state by keeping a table consisting of the node identifier and the most recently received chained HMAC. The destination determines the expected α by calculating the chained HMAC. Hence, to protect the flow, either the source or the destination can change to a new path which does not include nodes A and B. Likewise, node A may drop packets and yet generate a report in which it inserts bogus α. This case is similarly detected as the one discussed above. In both cases, however, the destination cannot distinguish whether the fallacious α is generated by A or B, i.e., which node implicates the other. This is a limitation of ASR.

A malicious node can use different techniques than the aforementioned ways, such as dropping packets or manipulating reports, to veil malicious selfish behavior. The node may fabricate a report claiming that it forwarded more packets than it has. For example, in Figure 4.2, node A may send a report saying it forwarded 100 packets when it really only forwarded 50 packets. In order for A to cheat under the chained HMAC scheme, it must provide α_{100} and β_{100} to the destination and source, respectively. Since the generation of α_{100} and β_{100} requires knowledge of K_{BD}, only known by node B and D, A cannot fake the report. The only way for A to learn α_{100} and β_{100} is for node B to tell it. This only occurs after A forwards 100 packets to B.

Lastly, nodes which once forwarded packets may attempt to replay reports. Since the report validation scheme uses the sequence number field to compute α_i and β_i, this prevents report replay. The proposed secure random reporting protocol collects reports from intermediate nodes en route, keeping track of the routing path of packets.

4.5.2 Nodes in Collusion

Now we discuss collusion attacks by multiple nodes. We first categorize different colluding scenarios based on participating nodes. Let us define an *out-of-path* node to represent a node which is not on a path of a flow and an *in-path* node to denote a node which is on a path and forwarding traffic for a flow. There exist four different colluding forms: collusion between out-of-path nodes, collusion between out-of-path and in-path nodes, collusion between non-adjacent in-path nodes, and collusion between adjacent in-path nodes.

Out-of-path nodes within transmission range of in-path nodes can eavesdrop traffic flows. These nodes may claim that they have forwarded packets for a flow. In ASR, however, to forge a report, out-of-path nodes need to know the path, selected node, and pairwise keys that the selected node shares with the source and destination. Unless these nodes collaborate with intermediate nodes, this is not possible.

Similarly, suppose that an out-of-path node is within transmission range of intermediate nodes on a path and collaborates with in-path nodes. The out-of-path node may strive to collect information to determine which node is selected by eavesdropping and inform the collaborating in-path node of the information. However, due to the hop-by-hop wrapping scheme in ASR, the out-of-path nodes cannot determine which node has generated a report.

Non-adjacent in-path nodes may forge reports by exchanging information to conceal their selfish behavior. For example, suppose that nodes 1 and 3 in Figure 4.3 (a) collude and node 1 drops packets. Node 1 may attempt to provide information for node 3 to generate a false report. In this case, however, the colluding in-path nodes are adjacent to cooperative nodes on a path. The colluders do not have the chained HMACs generated by the cooperative nodes adjacent to them since they did not forward packets.

In a similar and more powerful way, adjacent nodes, e.g. nodes 2 and 3 in Figure 4.3 (a), may collude to drop packets and generate reports as if they forwarded all the packets. Since they may release their keys to each other, this colluding attack may result in forged

report packets, i.e., the chained HMACs may appear to contain valid information. The last colluding node (3 in the example), however, which is adjacent to non-colluding node (node 4), cannot generate a report stating that it forwarded all the packets to node 4. In order to generate this report, the node (node 3) needs α_i that can be generated only by the next hop cooperative node (node 4) when it receives i packets. Hence, the last colluding node will be detected by the forgery detection scheme. This way, nodes in collusion will gradually be excluded from routing. However, identifying all colluding adjacent nodes at once is a challenging and difficult issue. To limit a colluding attack by adjacent nodes, other approaches such as game theoretic schemes [25] may be used such that nodes behave fairly to maximize their utility.

4.6 Performance and Overhead Analysis

In order to analyze the reliability of the proposed reporting schemes, we simulated our protocols using ns-2. These simulations illustrate the robustness of three random reporting protocols (RRNS, RRNDS, and RBR). Additionally, we explore the overhead of packet size and memory, and the cost of the underlying cryptographic constructions based on empirical data.

4.6.1 Simulation Environment

Table 4.2 shows the parameters of the ns-2 simulations. Mobile nodes use IEEE 802.11 MAC with a transmission range of 250m. Additionally, the CMU scenario generation tool [72] was used to create a network consisting of 50 mobile nodes in an 1500m X 300m range. A random waypoint mobility model with speeds of 20 m/second was used, in which each node moves to random location in the specified network area with an average moving speed which ranges from minimum speed to maximum speed. Once a node locates to the target location, it remains in that position for a time (pause time) before moving to another random location.

The effects of mobility and traffic loads on three secure random reporting protocols

Table 4.2. Simulation Parameters

Simulation Time	900 seconds
Number of nodes	50
Packet Size	512 bytes
Mobility	Random waypoint mobility model (20 meter/second)
Routing Protocol	Dynamic Source Routing (DSR)
Data Rate	4 packets/second
Transport Protocol	UDP

are analyzed by running simulations with 4 different pause times. We include 25 and 30 CBR background traffic flows in our simulations. In each case, the target source and destination are randomly selected from 50 nodes.

The period in which the destination and source observe the reports is fixed to allow proper analysis. We heuristically determined a basic observation period of ten seconds based on the speed (average 10 m/s) and transmission range (250 m). In order to adapt to the dynamic characteristic of path changes, reports were collected based on the flow and path. Traffic flows are defined by the source and destination addresses. As paths change, the source and destination keep track of the flow state, the current path state, and the active path list consisting of paths transmitting the traffic during the observation period.

When a node encounters a link failure, it sends a Route Error to the source as defined in the DSR routing protocol. Until the source receives the Route Error, it continues to use the current path. Therefore, packets transmitted to this path before the source changes a path may be lost due to a broken link. In the detection algorithm presented in the next subsection, the Route Error message helps the source estimate the number of lost packets due to link breakage, not a malicious node on a path.

4.6.2 Simulation Results and Discussion

Packet loss occurs due to mobility, congestion, and malicious dropping. Identifying the source of packet loss is difficult due to the random nature of its occurrence. For the flow of the designated source and destination, we implement an adversarial setting in which if

a path has more than one intermediate node, one intermediate node on the path is set to behave maliciously by dropping packets. The effectiveness of the protocol (shown in the following figures) is the percentage of experiments that correctly identify the malicious nodes. In this conservative model, the adversary behaves only slightly different than well behaved nodes: nodes that more aggressively drop packets will be detected more easily, and the protocols will be more effective.

The simulation was performed with two different attack strategies for dropping packets: fixed dropping rate (Case 1) and by matching the malicious dropping rate with the naturally occurring average dropping rate for the network under its current conditions (Case 2). For each case, we devised a simple detection algorithm which is described below. Here, the detection algorithms themselves are not important; they are for illustrative purposes only to quantify the impact of the report protocol variants.

Case 1: The dropping rate of a malicious node is set a fixed value. We measured the average packet loss rate over the simulation time, excluding malicious nodes. We had ten background traffic sources generate 4 packets/sec and a target source generate about 28 packets/sec. Results showed a 12% average packet loss (caused by congestion and mobility) from this baseline test. Using the measured average packet loss as a guideline, a second battery of experiments simulated an adversary that dropped 17% of the received packets at a different speed; we chose a slightly higher value (17%) than the average (12%) to accommodate the variance of packet loss rate. We designated any node shown to have a loss rate greater than 12% as anomalous, during the observation period (ten seconds).

Figure 4.4 shows the impact of mobility on the effectiveness of the three protocols under this simple attack. All points are the averaged value over 5 runs. In the static case, all three protocols showed an almost 100% detection rate. However, as mobility is introduced, the effectiveness of RRNS decreases sharply since the reports embedded in data packets are lost. By contrast, the RBR protocol remains highly effective in all experiments. In RBR, the report is transmitted to both the source and destination. The

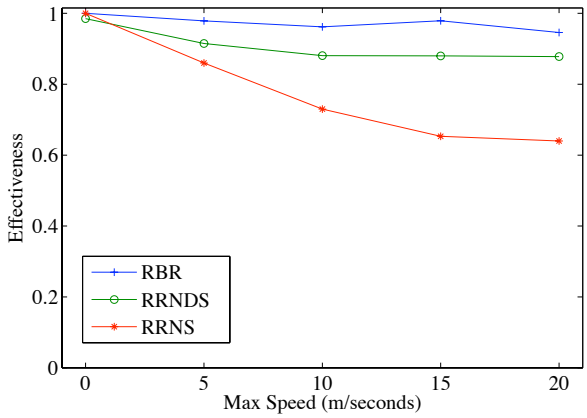


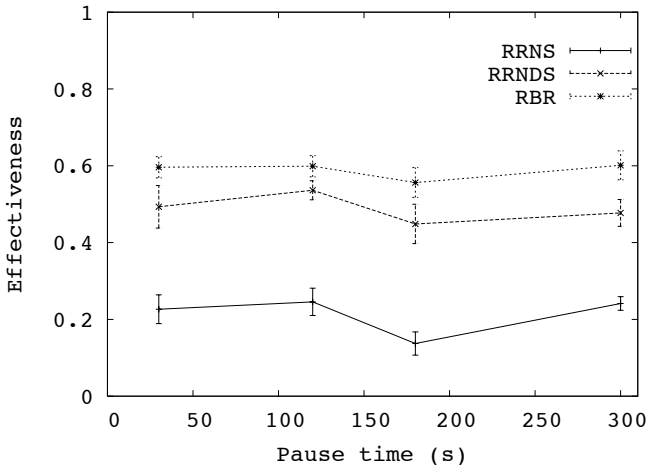
Figure 4.4. Effectiveness vs. Mobility in a fixed attacking rate

redundant transmission improves the robustness of reports in the presence of mobility. We discuss the effectiveness of RRNDS below.

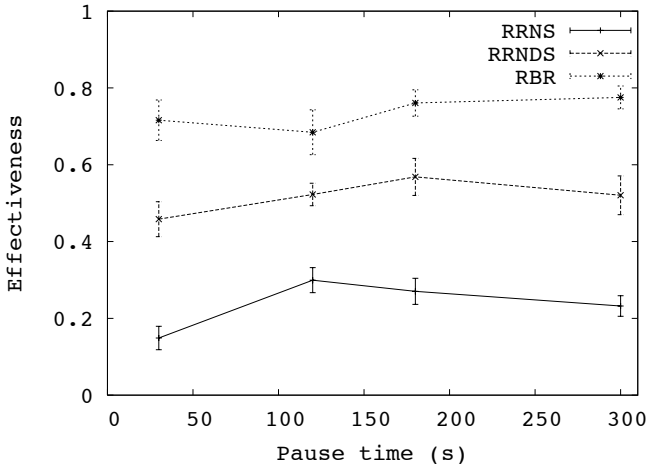
Case 2: With low traffic load and low mobility, if a malicious node drops too many packets, a source or destination may easily detect malicious behavior. Since paths change frequently due to high mobility or congestion, it is not easy to estimate how long a path will be used for data transmission. Considering this, we assume that an adversary may know the network statistics and use it to schedule packet drops. For example, an adversary may drop packets at the rate of the average packet drops due to natural congestion or mobility. This way, an adversary may reduce the possibility of being detected.

To simulate this adversary model, we evaluated average packet loss rate and its standard deviation at each scenario, leaving out malicious drops. The average packet loss rate has a different value at four pause times (30, 120, 180, and 300 seconds) under 25 and 30 CBR background traffic sources. Every traffic source generates 4 packets/sec. A designated malicious node en route dropped packets at the average packet loss rate measured at a specific condition. The detection algorithm in the source and destination used the average packet loss rate plus its standard deviation to detect malicious dropping. Although more sophisticated algorithms can be used to detect the malicious dropping, we picked this simple detection algorithm to evaluate the robustness of three random

reporting protocols for illustration purpose.



(a) 25 CBR Sources



(b) 30 CBR Sources

Figure 4.5. Effectiveness vs. Mobility/Loads

Figure 4.5 shows the effectiveness of three random reporting protocols under the intelligent attack strategy, using 25 and 30 CBR background traffic sources respectively. The figure also shows 95% confidence interval and average effectiveness where all points are the averaged value over 10 runs. The effectiveness is degraded as mobility decreases. In a stable network (low mobility and low traffic load), the average packet loss rate is very low, and therefore the adversary's dropping is also very low. Under this condition, the subtle packet dropping is not easily distinguishable from the normal packet loss. By

the same reasoning, the effectiveness under 30 background traffic sources is better than under 25 background sources.

In both attack strategies, RRNDS showed higher effectiveness than RRNS even though the source and destination receive only about half of the reports, i.e., the same total number of reports is received. This can be explained by the additional information that a source node has. In addition to source-bound reports, the source node knows how many packets it transmitted and gains useful information from Route Errors, which allows the source to evaluate the number of packets lost due to link failure on the path. Conversely, the destination only receives reports included in the successfully received packets.

To evaluate the effect of the extra information, we simulated a scenario with a 120 second pause time. In this case, the source received almost half of the reports and 137 Route Errors in RRNDS. RRNDS was 1.4 times as effective as RRNS. To confirm the effect of the additional information, we also simulated a special case in which all reports are transmitted only to the source. In this case, the source receives successful reports and Route Errors. This improves the effectiveness of RRNDS and RRNS by 56% and 280% respectively, but is still lower than RBR's effectiveness. While sending all reports to the source appears to be advantageous, it is vulnerable to a report dropping attack by downstream nodes. When the communication is unidirectional (source-to-destination), the selected node has to generate an extra packet to send a report to the source node. A node downstream of the selected node knows that the packet is a report and drops the packet. Moreover, in the case of bi-directional communication, as the effectiveness of RBR shows, the reports transmitted in both directions can help the detection algorithms in the source and destination.

Case 3: Finally, we conducted simulations to compare resiliency of ASR with a reference scheme in which intermediate nodes periodically send a separate report of their contribution to a source. Because we do not assume an Internet connection, we have the report sent to the source.

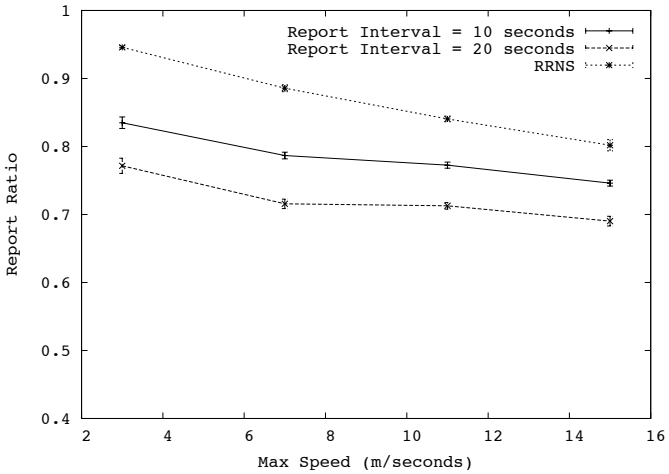


Figure 4.6. Reporting Ratio vs. Mobility

In these simulations, there are 10 CBR connections in which each source generates a CBR packet (512 bytes) per second, and mobile nodes move at a speed uniformly distributed from 0 m/s to a maximum without pause. The simulation was run for 500 seconds.

The report ratio is defined as the ratio of total reported number of forwarded packets to total number of packets forwarded by intermediate nodes. We compared the reference scheme with RRNS which is the least robust version from three random reporting protocols. Figure 4.6 shows the simulation result of the averaged report ratio and 95% confidence interval, where all points are the averaged value over 10 runs. As the report interval becomes short, the amount of reported information in the reference scheme gets close to RRNS, while frequent report transmission incurs communication overhead. From this simulation, the piggybacking of a report in ASR with the random node selection provides an efficient and effective solution to collect reports.

4.6.3 Overhead

In MANETs, mobile nodes have limited batteries, small memory, and limited computational power. Therefore, it is imperative to analyze the overhead caused by ASR. In this subsection, we explore the overhead of ASR: packet size, memory overhead, and

computational overhead.

Packet Size: The secure random reporting protocol requires two new fields: *Report* and $Token$. The *Report* field consists of two reports: forward report (R_f) and backward report (R_b). Each report is composed of an initial sequence number (4 bytes), a chained HMAC (16 bytes for MD5), a sequence number map (*smap* 4 bytes), and the number of packets (4 bytes) that the selected node forwarded. The $Token$ is a cryptographic hash output (16 bytes for MD5). The total overhead incurred by the secure random reporting protocol is 68 bytes which is only 4.5% of maximum data packet size. On the other hand, existing eavesdropping schemes require a separate packet transmission of the report to other specific nodes or a centralized server. This separate packet transmission incurs additional transmission delay and energy consumption. Our reporting protocol removes these extra costs by embedding a report in data packet with small overhead in packet size.

Memory Overhead: In ASR, each intermediate node on the path maintains only current state information of a specific flow. The state consists of path information (20 bytes if a path has 5 nodes), sequence number map (4 bytes), the number of forwarded packets (4 bytes), α and β (32 bytes). Here, the path information is used for both the forward and backward traffic, while other state information should be maintained separately for the forward and backward flows, which requires 100 bytes in total. The source and destination should keep the information of intermediate nodes on a path which consists of sequence number, the number of forwarded packets, and α and β during a monitoring period. The monitoring period is decided by the source and destination. Let us assume that the source generates one packet per second, the monitoring period is 10 seconds, and a source route has five intermediate nodes. In this setting, the source and destination only need about 400 bytes, respectively. The chained HMAC scheme enables the source and destination to keep the information of each intermediate node, and intermediate nodes to maintain current state information of a flow, independent of the number of received packets.

Computation Overhead: One concern of the Secure Random Reporting Protocol is the computational overhead. Depending on the chosen cryptographic function, the overhead will vary. To test the performance of HMAC and encryption/decryption, we wrote a module for the Linux 2.6 kernel, using the available cryptographic API. Tests were performed on a Pentium 3 800MHz, 192MB RAM system using a stock Linux 2.6.11 kernel compiled by GCC 3.3. The tests covered MD5, SHA1, and SHA256 digest functions with varying key sizes (64bit, 128bit, 160bit, and 256bit). AES CBC encryption/decryption is also evaluated with varying key size (128bit, 192bit, and 256bit). Results were averaged using the raw cycle count over 1000 test runs to ensure accurate results.

In our simulation, the longest path has ten intermediate nodes. For this path, we estimate the computational overhead of the secure reporting protocol. As expected, there was no performance difference for varied key sizes, therefore, the average raw cycle count for the key sizes was used. Finally, using the `cpu_khz` value of 647894, actual time latencies were calculated.

Table 4.3. HMAC Computational Overhead (647,894K cycles/sec)

Algorithm	HMAC Chain	HMAC Data	Total
MD5	7.037 μs	29.950 μs	271.01 μs
SHA1	19.108 μs	86.614 μs	746.468 μs
SHA256	20.308 μs	95.606 μs	800.452 μs

Table 4.3 shows the computation time for individual calculations, which emulated the actual data used by the protocol. This shows the results of the case in which a report field includes forward and backward report values and the report is transmitted to both the source and destination. As described earlier, the input data for each stage of the HMAC chain consists of a sequence number and the output of a previous HMAC. Thus, the total input data size for the HMAC chain is four bytes for the sequence number and the number of bytes outputted by each cryptographic digest function (MD5 = 16 bytes, SHA1 = 20 bytes, SHA256 = 32 bytes). The HMAC data column in the table represents the overhead for performing the HMAC on the Maximum Transmission Unit (MTU),

1500 bytes.

In the report wrapping scheme, every intermediate node encrypts the report field, and a destination performs encryption and decryption of that field. The report field (40 bytes) consists of 2 reports (forward and backward: $2 * 4$ bytes) and a hash computation result of two reports (32 bytes in SHA256). AES encryption/decryption of the report field takes 2.7905 μs. This results in 55.81 μs overhead.

The total computational cost consists of four factors: two HMACs of data packets at the source and destination, two chained HMACs (α_i and β_i) at each intermediate node, a hash computation (*Token* checking) at each of the intermediate nodes and the destination, and encryption/decryption of the report field at each node and the destination. According to the results of computational overhead above, the secure random reporting protocol, report wrapping and forgery detection schemes have less than 856 μs overhead.

To measure end-to-end transmission delay, we set a condition in which packets does not experience link failure and congestion. In this optimal setting, data transmission from the source to the destination takes 65 milliseconds on a path which has 10 intermediate nodes (6 milliseconds per hop transmission). The total HMAC, hash, and AES computation takes only 1.3% of the total time.

If we map the results to a PDA, we still find that the computational overhead is low. For example, PDA processor speeds range from 200 MHz to over 600 MHz. For the 200 MHz PDA, the clock cycles are four times longer than our laptop. Suppose that we choose to use SHA256. The total computation of ASR in a 200 MHz PDA will take approximately 3.42 milliseconds.

4.7 Discussion

A mobile ad hoc network does not require a fixed infrastructure and a centralized control to enable communication between mobile nodes. Most applications target mission oriented scenarios such as battle fields and emergency rescue. In these scenarios, mobile nodes actively cooperate with each other to achieve a goal. This is different than civilian

mobile ad hoc networks, where nodes are not necessarily cooperative.

We propose a secure random reporting protocol for a civilian ad hoc network, in which the source and destination collect reports from intermediate nodes on the routing path. Every data packet initiates a report from one intermediate node that is randomly chosen by a source node. Through a symmetric cryptographic construction, we ensure that the node selection is not disclosed to other intermediate nodes.

Although the report is securely transmitted to the destination, we cannot assure that it is accurate, since nodes may cheat in order to get credit. We devise a chained HMAC scheme on the link layer acknowledgments to verify the validity of the received report.

From both security and performance perspectives, the secure random reporting protocol is advantageous for gathering the forwarding activities of mobile nodes in civilian ad hoc networks. The protocol has a small communication overhead due to the increase in packet size caused by including the real time reports with data transmission. Our simulation results demonstrate the promising possibility of the reporting protocol.

We discussed a traffic analysis attack which tries to discover who is generating a report. Mobile nodes around the reporting node can overhear the received and transmitted packet of the selected node. Only the selected node will change the report field and other intermediate nodes transmit packets without transformation. Since the current scheme does not include the solution to combat with the traffic analysis attack, we will address this problem in future research.

If collaborating compromised nodes are not adjacent on the path our protocol can discover which node is falsifying reports with the help of the reports from honest nodes. However, if they are located next to each other, it is difficult to know which node is the first compromised node. We leave the colluding problem of adjacent compromised nodes as future work.

Although most application protocols are bidirectional, there exist several unidirectional protocols, e.g. multimedia services using UDP. If the traffic flow is unidirectional, the selected node cannot attach self-report to the reverse directional data packet. To

decide if the traffic flow is unidirectional, the intermediate node may inspect packet headers, or use timeouts to detect source-bound packets. Once the intermediate node detects the unidirectional flow, it has two choices: either it generates a separate report or does not send a report towards the source.

With the first choice, the selected node sends a separate packet carrying a reportto the source. As simulation results in Section 4.6 show, this bidirectional reporting improves the robustness of reports. However, the selected node's neighbors may eavesdrop on the communication and determine that the selected node has generated a report. By choosing this option, the node selection will not be anonymous. Moreover, the separate report packet incurs communication overhead.

With the second choice, the intermediate node does not generate a separate report towards the source, which becomes RRNS. This reduces the robustness of reports as shown by the simulation results. However, this way the selection of the reporting node remains anonymous to neighbors, without incurring extra communication overhead.

Chapter 5

Privacy Preserving Communication in Ad Hoc Networks

5.1 Backgound

In MANETs, mobile nodes cooperate to forward data on behalf of each other. Typical protocols used for self-organizing and routing in these networks expose the node identifiers (network and link layer addresses), neighbors, and the end-points of communication. Some modes of operation further mandate that the nodes freely divulge their physical location. In short, nodes must advertise a profile of their online presence to participate in the MANETs. This is, in many cases, highly undesirable.

Both military and civilian MANETs may find the mandated exposure of information unacceptable. For example, in a military setting, identities of officers and soldiers, their locations, and their communication patterns are critically sensitive intelligence. Civilian applications have similar concerns. Consider students communicating on campus: it is neither desirable nor appropriate for students to expose who they are or where they are to the larger campus community.

Ideally, a node should be able to keep its identity, its location and its correspondents private, i.e., remain *anonymous* [45, 47, 49]. Any solution providing anonymity must overcome the broadcast nature of wireless environments (which enables eavesdropping) and operate under often tight resource constraints. Past "wired world" privacy solutions

do not map well to MANETs because of the processing requirements they place on the nodes. Simple solutions like packet encryption are also largely ineffective because of ease of traffic analysis over a broadcast media. Hence, supporting privacy in MANETs is enormously challenging.

In this chapter, we propose a Privacy Preserving Communication System (PPCS) which provides a comprehensive solution to anonymize communication end-points, keep the location and identifier of a node unlinkable, and mask the existence of communication flows.

To realize this level of privacy, we propose a series of lightweight cryptographic techniques. These are effective at combating eavesdropping by individual nodes. To further defend against more sophisticated collaborative attacks via traffic analysis, we introduce a resilient packet forwarding scheme. To evaluate the effectiveness of PPCS, we define the optimal guessing strategy that may be used by one or more adversaries in cooperation and show that with PPCS, the probability of correctly guessing the source or destination of a flow is independent of the number of compromised nodes on the path. Even in this case, the adversary cannot confirm that it has guessed correctly, and it cannot learn the real identifier of the source or destination. To quantify the overhead of this solution, we perform extensive simulations that show that there is minimal impact on packet delivery.

This chapter is organized as follows: Section 5.2 describes the network model and examines passive attacks. Section 5.3 presents an anonymous communication system (PPCS). In Section 5.4 we present an optimal guessing strategy, and, based on this strategy, inspects the effectiveness of an adversary in PPCS. In Section 5.5, we evaluate the performance impact of PPCS. In Section 5.6, we discuss the trade-offs of PPCS.

5.2 Network and Threat Model

5.2.1 Network Model

We assume that the wireless interface between nodes is bidirectional, i.e., if node i hears the transmission of node j, then node j is also able to hear node i.

We assume that there exists a symmetric key management service to establish pairwise keys between nodes, and that the source and destination establish symmetric keys prior to communications. Such services are well studied in ad hoc and sensor networks [77, 58, 78, 83, 80], and their design is explicitly outside the scope of this work. The source and destination know each other's real identifier. Non-compromised nodes in the network do not disclose any information to compromised nodes beyond what is required for the normal operation of the network.

Note that the privacy services we propose operate solely on the network layer; we assume that the contents of the communication have been duly masked, e.g., via end-to-end encryption.

We use the following notation throughout:

- S: Identifier of a source
- D: Identifier of a destination
- n: Average number of neighboring nodes in transmission range
- P_{Si}: Source S's i-th flow pseudonym
- P_{Di}: Destination D's i-th flow pseudonym
- K_{ij}: Symmetric key between nodes i and j
- $E_{K_{SD}}(.)$: Encryption with a key K_{SD}
- $D_{K_{SD}}(.)$: Decryption with a key K_{SD}

5.2.2 Threat Model

We adopt Diaz et al.'s [84] classification of adversaries based on the following characteristics: Internal-External, Passive-Active, and Local-Global. We define an internal adversary as a node that is compromised and on the routing path. An external adversary

is a compromised node not on the path, or an external node not directly participating in the MANET, i.e., it only eavesdrops on traffic between nodes.

This thesis only considers passive attacks, i.e., attacks that consist of eavesdropping on communications to collect private data. A local adversary can see and launch attacks in a limited range. A global adversary covers the entire path or the network. A set of colluding local adversaries may form a global adversary by sharing information. We defer the active attacks to future work.

Traffic analysis is often used to subvert anonymity [84, 85, 86]. In this attack, adversaries monitor packet transmission to infer important information such as a source, destination, and source-destination pair. We consider the following traffic analysis attacks in this work:

Packet Tracing Attack: A packet may be traced from source to destination by eavesdropping the transmission of the same packet as it traverses the network. Note that the adversary need not be able to recover the packet content to infer the source and destination of the flow.

Packet Counting Attack: Eavesdropping nodes collaborate to discover a path by overhearing and simply "counting" packets that traverse nodes. In a network with low load, this is a straight-forward way to discern data paths.

Timing Attack: Adversaries may analyze the time correlation between packets passing through nodes to discover a flow [87]. If two adversaries perform this analysis and compare results, they may infer a source-destination pair.

TTL Attack: Adversaries exploit the packet time-to-live (TTL) field to discover the destination. The value of the TTL field in a packet is set by a source to limit the number of hops a packet takes in the network. Every intermediate node decreases the TTL by 1 before it forwards the packet. Because this information is sent in the clear, adversaries may determine the relative position of a node on a path, and perhaps the source or destination if they are located near these nodes.

Adversaries may also try to discover information about paths of which they are a part.

Many routing protocols expose control information, such as the source and destination or the other nodes on the path, to all nodes on a path. Nodes can also typically overhear the next-hop node on a path as it forwards a packet. Combining this information, adversaries on a path can learn source-destination pairs, next hop nodes, and the entire path of a flow.

Mobile nodes may obtain their own location information using global positioning system (GPS) or other similar techniques. If a node knows the identifiers of its neighboring nodes, it also may estimate their locations. An adversary may also use location information to launch various attacks by tracing an object's location. Therefore, dissociation of location and identity is an important issue.

5.3 Privacy Preserving Communication

In the following subsections, we present a privacy preserving system which is composed of three mechanisms to anonymize the communication in MANETs. *Dynamic Flow Identification* is aimed at preventing identification of source-destination pairs. *Random Node Identification* dissociates the identity and location of nodes. *Resilient Packet Forwarding* is targeted at thwarting sophisticated traffic analysis attacks.

5.3.1 Dynamic Flow Identification

Traditional MANET routing protocols require each control and data packet to contain the source and destination addresses to find a route and identify a flow. With this general approach, an adversary close to the source or destination, or an adversary on the communication path between the two, will be able to link the correspondents, and perhaps learn their location.

To define a flow without releasing the source and destination addresses, we propose a dynamic flow identification scheme based on forward chaining. In the dynamic flow identification scheme, two flow pseudonyms, P_{Di} and P_{Si}, are defined for the forward and backward flows respectively. The flow pseudonym replaces the source and destination

addresses in the packets. A source broadcasts a RREQ packet which contains these flow pseudonyms, <RREQ, P_{Si}, P_{Di}, $E_{K_{SD}}(.)$>.

Intermediate nodes receive a RREQ packet and check if they are the destination by attempting to successfully decrypt and interpret the flow pseudonyms, i.e., "open the trapdoor" [54], which conceals the source and destination address as decribed below. If they are not the destination, they add a routing table entry for the backward flow identified by the flow pseudonym P_{Si} in the RREQ. A destination receives a RREQ and determines that it is the destination by checking the received P_{Di}.

Since each node must perform the trapdoor check, it is important for the check to be efficient. The initial flow pseudonyms, P_{D0} and P_{S0}, of the forward and backward flows are generated by using the symmetric key and real identifiers of the source and destination. Either a source or a destination can change the flow pseudonym at anytime. To do this, subsequent flow pseudonyms are generated based on the previous flow pseudonym using *forward chaining* as follows:

$$P_{S0}{=}f_{K_{SD}}(S) \rightarrow P_{S1}{=}f_{K_{SD}}(P_{S0}) \ldots \rightarrow P_{Sn}{=}f_{K_{SD}}(P_{Sn-1})$$

$$P_{D0}{=}f_{K_{SD}}(D) \rightarrow P_{D1}{=}f_{K_{SD}}(P_{D0}) \ldots \rightarrow P_{Dn}{=}f_{K_{SD}}(P_{Dn-1})$$

, where f is a cryptographic keyed one-way hash function (HMAC [88]). The results of function f appear random to the intermediate nodes. The trapdoor check is very lightweight, consisting only of computing a hash and a simple search for a matching node. Also note that the trapdoor check only occurs when processing the RREQ message; once the flow has been routed, the check is not required for forwarding subsequent packets. To further improve the efficientcy of the trapdoor check in each node, an optimal data structure such as binary search tree can be used.

5.3.2 Random Node Identification

Location privacy requires node identity and location to be unlinkable and untraceable. We propose to use a random node identifier to dissociate a real node identifier from

location information. In normal operation, a mobile node has two addresses: a layer 2 address (MAC address) and a layer 3 address (node identifier).

Every node in the network generates random layer 3 and MAC addresses, referred to as random node identifiers (RNI), and advertises itself using its RNI via a message such as a HELLO message in AODV [89]. Neighboring nodes know each other only through their RNIs. The RNI is *locally* used for routing and communicating with neighboring nodes.

Each node changes its RNI after a random interval to prevent an adversary from learning its location and then starts advertising itself with the new RNI. The protocol to change RNI is the same as for an update due to mobility.

Since the source and destination associate with one another using end-to-end flow psuedonyms described in the previous subsection, they do not have to know each other's RNI. This has two benefits. First, the RNI may be changed without end-to-end coordination. Second, since the source and destination do not know each other's RNI, the communication between a source and destination does not disclose the location of either party to the other.

Due to the randomness and independence of the new and old RNI, an adversary cannot trace the changes of node RNI. One risk with this approach is identifier collision, in which two nodes choose the same RNI, might occur. However, the probability that two nodes generate the same RNI (MAC address (48 bits) and layer 3 address (32 bits)) is statistically insignificant, ($\frac{1}{2^{48}}\frac{1}{2^{32}} = \frac{1}{2^{80}}$).

5.3.3 Resilient Packet Forwarding

To combat traffic analysis attacks by eavesdropping nodes we propose a resilient traffic forwarding scheme which is composed of multi-path random forwarding (MPRF), Hint, and random TTL (RTTL).

Multi-Path Random Forwarding (MPRF): In a relatively stable network (mild traffic load and low mobility), a path between a source and destination may be used for

an extended period of time. This type of path, in particular, is susceptible to a traffic analysis attack. To thwart attacks on a single path, MPRF establishes multiple paths between the source and destination. For each packet, an intermediate node en route randomly selects a next hop from its local list of possible next hop nodes, and forwards the packet to the selected node. Thus, a path that a packet takes is decided dynamically at each intermediate node.

Multi-path routing protocols have been proposed for improving reliability and providing quality of service in ad hoc networks [90, 91, 92]. These multi-path routing protocols establish link/node disjoint paths to distribute traffic to avoid congestion. However, node/link disjoint paths are also vulnerable to traffic analysis attacks. Collaborating eavesdroppers may easily obtain exact packet counts and reconstruct the end-to-end paths. To resolve these vulnerabilities and establish a sufficient number of multiple paths, we relax the node/link disjointness condition present in most multi-path routing protocols. By allowing non-disjoint paths, MPRF diffuses traffic in an irregular manner making traffic analysis more difficult, i.e., requiring a larger number of colluders. In addition, when a node selects multiple paths, the most recently joined node is not be chosen since compromised nodes can continuously change their identifiers to hamper the communication (Denial Of Service).

Hint: Although a packet is encrypted by a source, if the encrypted packet is transmitted without any modification on each link, it is vulnerable to traffic analysis attacks which determine a data path by observing the incoming and outgoing packets of nodes. To address this problem, the encrypted packet is transformed on a hop-by-hop basis.

To make the hop-by-hop transformation more efficient and anonymous, we propose an HMAC [88] based scheme, called *Hint*. An intermediate node randomly selects a next hop node according to MPRF. It encrypts a packet using a shared key with the selected node and computes an HMAC over the encrypted packet. This HMAC result is called the **Hint**. Then it broadcasts the packet which consists of the Hint and encrypted packet. As an example, the following shows **Hint** operations of each intermediate node:

N_S: $C = E_{K_{SD}}(Data)$
$MC =< P_{S0}, P_{D0}, TTL, C >$
$EL_S = E_{K_{N_S N_1}}(MC)$, Hint $= HMAC(K_{N_S N_1}, EL_S)$
Broadcast $< Hint, EL_S >$
N_1: $MC = D_{K_{N_S N_1}}(EL_S)$
$EL_1 = E_{K_{N_1 N_2}}(MC)$, Hint $= HMAC(K_{N_1 N_2}, EL_1)$
Broadcast $< Hint, EL_1 >$
N_2: $MC = D_{K_{N_1 N_2}}(EL_1)$
$EL_2 = E_{K_{N_2 N_D}}(MC)$, Hint $= HMAC(K_{N_2 N_D}, EL_2)$
Broadcast $< Hint, EL_2 >$
N_D:
$MC = D_{K_{N_2 N_D}}(EL_2) \rightarrow MC =< P_{S0}, P_{D0}, TTL, C >$
$Data = D_{K_{SD}}(C)$

Neighboring nodes check if a received packet is for a flow which they serve by simply computing the HMAC for the received packet. If the check results in success, it decrypts the received packet with the corresponding key and forwards it according to MPRF. The HMAC calculation takes a few micro seconds as shown in [93]. Only the corresponding local receiver decrypts the packet. If $D(.)$ denotes the overhead for packet decryption, and n is the average number of neighbors in transmission range, Hint reduces the average computation overhead at a node from $\frac{1}{2}n^2D(.)$ to $\frac{1}{2}n^2HMAC(.)$ when compared to schemes that encrypt and broadcast a packet.

Due to the transformation on each link combined with broadcast transmission, eavesdroppers are not able to learn the relationship between incoming and outgoing packets of a node. Although a compromised node en route may see several control fields like TTL in clear text, it cannot discover which node will be the next hop of its neighboring next hop. For each traffic flow, since there is no relation between flows, an adversary will have difficulty in discovering the flow. Furthermore, when a destination receives a packet, it broadcasts a random packet as a response, hiding its role from neighboring nodes. This random packet is not distinguishable from a transformed packet by Hints. Neighboring nodes discard the packet.

During route discovery, Hints are used to transform a RREP in the same way. Oth-

erwise, an adversary may discover a route through tracing RREP messages.

Random Time-To-Live (RTTL): The TTL field is used for discarding packets which have not found a destination and circulated through the network. In MANETs, the TTL is set to the length of a path by a source node. Each node on the path decreases the value by 1. Thus, the TTL value reveals the position of a node on a path from a source or a destination. The receiver anonymity set may be reduced to a set of nodes neighboring a compromised node from a set of all possible receivers.

To prevent compromised nodes from learning their position on a path, we propose a Random Time-To-Live (RTTL). A source node generates a random value and sets the TTL field with the sum of this random value and path length, RTTL. The RTTL should be less than the maximum hop count (Network diameter). The source includes the initial random value in the encrypted data packet. Intermediate nodes decrease the TTL value of a packet by 1 as they do in the normal packet forwarding. This TTL field does not release the absolute position of a node due to the random value. A destination decrypts the received packet and checks if the received RTTL is valid by subtracting its initial random value.

5.4 Security Analysis

In Section 5.2.2, we presented a classification of attackers. In this section, we characterize the anonymity provided by PPCS against attacks by internal compromised nodes and then argue informally about the anonymity provided by our system against eavesdropping attacks. To support this analysis, we present an optimal guessing strategy to be used by an adversary for each attack.

5.4.1 Internal Attackers

In this subsection we examine the effectiveness of PPCS against collaborating internal adversarial nodes. Intermediate nodes on the path can see the flow pseudonym and TTL field of a packet. Intermediate nodes also know the previous and next hop nodes of a

packet on the routing path. Using this information, the compromised intermediate nodes on a path collude to make an educated guess as to the source and destination of a flow.

To characterize the probability that a set of internal compromised nodes collaborate on successfully discovering anonymity we first derive a general equation which can be applied to each case of anonymity (source/destination and communicating pair).

The following notation is used in the remainder of our analysis.

- N: Total number of nodes
- C: Number of compromised nodes in the network
- L: Average path length
- T: Number of uncompromised nodes disclosed by intermediate compromised nodes en route
- W: Number of intermediate nodes on multiple paths established between the source and destination
- G: $(N - C) - T$
- p: probability that a node is compromised
- $P_{f,s}$=$P_{l,r}$: probability that the first/last hop node guesses a source/destination correctly, respectively
- $P_{i,s}$=$P_{i,r}$: probability that an intermediate node guesses a source or a destination correctly
- $P_{i+f,l}$=$P_{i+l,l}$: probability that the first/last hop node and intermediate nodes together guess linkability of the source correctly and destination
- $P_{f+l,l}$: probability that the first and last hop nodes together guess linkability of the source and destination correctly
- $P_{i+i,l}$: probability that intermediate nodes together guess linkability of the source and destination correctly

Let $P(A = s)$ and $P(A = r)$ denote the probability that an adversary discovers a source or a destination. Note that the adversary can determine only which node is a source or a destination, not the identifier due to the random node and flow identification schemes. Since the values, $P(A = s)$ and $P(A = r)$, are the same, we discuss the probability $P(A = s)$ below. Let $P(A = (s, r))$ denote the probability that an adversary discovers the source and destination pair.

5.4.1.1 Generalization

Without loss of generality, we assume that the probability of a compromised node being able to exploit a vulnerability is dependent on its position on a path. In particular, the first and last hop nodes on a path may have a higher probability of finding a source or destination, respectively, than an intermediate node on the path depending on the characteristics of the security solution. To this end we derive the probability of four cases of node compromise as in Table 5.4.1.1. We determine the probabilities of $P(CH)$,

Table 5.1. Classification of node compromise

CH	the first hop of a source is compromised and zero or more other compromised nodes are on the path, but not the last hop.
HC	the last hop is compromised and zero or more other compromised nodes are on the path, but not the first hop node.
CC	the first and last hop nodes are compromised, as well as zero or more compromised nodes on the path.
HH	the first and last hop nodes are not compromised nodes, but one or more compromised nodes are on the path

$P(HC)$, $P(CC)$, and $P(HH)$ for a path that has k compromised nodes in each case.

$$\begin{aligned} P(CH) &= (1-p)^{L-k}p^k\tbinom{L-2}{k-1} \\ P(HC) &= (1-p)^{L-k}p^k\tbinom{L-2}{k-1} \\ P(CC) &= (1-p)^{L-k}p^k\tbinom{L-2}{k-2} \\ P(HH) &= (1-p)^{L-k}p^k\tbinom{L-2}{k} \end{aligned}$$

Let $P_{CH}|P_{HC}|P_{CC}|P_{HH}$ denote the probability that an adversary discovers target anonymity in each case.

$$\begin{aligned} P_{CH} &= P(A|CH)P(CH) \\ P_{HC} &= P(A|HC)P(HC) \\ P_{CC} &= P(A|CC)P(CC) \\ P_{HH} &= P(A|HH)P(HH) \end{aligned}$$

In these equations, $P(A|X)$ is the probability that anonymity is discovered given that the compromise scenario X has occurred.

The probability that an adversary discovers target anonymity is defined

$$P(A) = P_{CH} + P_{CC} + P_{HC} + P_{HH} \tag{5.1}$$

This is a measure of the effectiveness of compromised nodes. In disjoint multi-paths environments, the probability that an adversary discovers anonymity is

$$P_m(A) = 1 - (1 - P(A))^R \tag{5.2}$$

where R is the number of disjoint paths established between the source and destination.

5.4.1.2 Optimal Guessing Strategy

We now present the optimal strategy that an adversary may use to discover flow endpoints (a source, a destination, or both). First, consider an optimal anonymity solution in which no information is leaked. In this case a compromised node does not know its previous or next hops, or its position on a path. It only knows of other compromised nodes. In this situation, the best an adversary can do is to guess the source from the set of uncompromised nodes. The probability of guessing correctly is $\frac{1}{(N-C)}$.

Now consider a non-ideal anonymity solution in which an adversary can identify its position on the path, but not other nodes on the path except for its direct previous and next hops. If the node is the first hop (information learned by seeing the TTL in the reverse path), it knows its previous hop is the traffic source. If a node is not the first hop on a path, its best guess is a random choice of all nodes in the network not counting the nodes it knows to be compromised or the nodes that compromised nodes can rule out as the source, such as their next hop nodes or previous hop nodes if they are not the first on the path. We call this set U, which has $G = N - C - T$ members. Thus the probability of an intermediate node guessing correctly is $\frac{1}{G}$.

Finally, consider the situation when RTTL is used within PPCS. In this case an

adversary knows it is on the path, but cannot tell its position on the path. Therefore, a different guessing strategy will be used. The adversaries have two choices. First, they can make a random guess of all nodes in set U, in which case their chance of guessing correctly is $\frac{1}{G}$. A better strategy is simply to guess its previous hop as being the source. Although the adversary does not know its place on the path, it has a $\frac{1}{L}$ chance of being the first hop node and thus guessing correctly. Even if several nodes on the path are compromised and collaborate, the only information they can learn is which adversary is closest to the source, and guess the previous hop to that node, i.e., they will all guess the same node. This strategy results in a probability of guessing the source that approaches $\frac{1}{L}$, independent of the number of compromised nodes on the path. The only way that the random guess strategy will be better for an individual node is if $G \leq L$, i.e., the average path length is greater than the number of uncompromised nodes in the network which is an unlikely scenario.

Based on the discussion above, we assume the following three strategies to guess the source node on a path: (1) In an ideal environment, adversaries make a random guess from the set of non-compromised nodes; (2) If an adversary is on a path, and it knows its position on the path, it will guess its previous hop as the source if it is the first hop node, otherwise it will make a random choice from the set U; (3) If an adversary is on a path, and it does not know its position on the path, it will always guess its previous hop on the path as the source.

5.4.1.3 Source/Destination Anonymity

Compromised internal nodes collaborate to determine a source using explicit information such as the flow pseudonym, TTL value, and next and previous hop nodes.

Let us suppose that there is more than one compromised node on a routing path. These nodes conspire to discover a source of traffic. $P_{f,s}$ and $P_{i,s}$ are the probabilities that the first hop and intermediate nodes guess a source, respectively. The probability

$P(A = s)$ is

$$\begin{aligned} P(A = s) &= P_{CH} + P_{HC} + P_{CC} + P_{HH} \\ &= P_{f,s} \sum_{k=1}^{L-1} (1-p)^{L-k} p^k \binom{L-2}{k-1} \\ &+ P_{f,s} \sum_{k=2}^{L-2} (1-p)^{L-k} p^k \binom{L-2}{k-2} \\ &+ P_{i,s} \sum_{k=1}^{L-1} (1-p)^{L-k} p^k \binom{L-2}{k-1} \\ &+ P_{i,s} \sum_{k=1}^{L-2} (1-p)^{L-k} p^k \binom{L-2}{k} \end{aligned} \tag{5.3}$$

The first two terms correspond to the first two terms in equation 5.1. The last two terms correspond to the last two terms in equation 5.1. Note that we do not need to account for intermediate nodes compromised in the scenarios covered by the first two terms in equation 5.1 because of the manner in which compromised nodes will collaborate. That is, if two nodes on a path are compromised and collaborate, they can compare the TTL field of the packets they receive and determine who is closer to the source. This is the only node that can correctly guess the source if an optimal guessing policy is used as discussed.

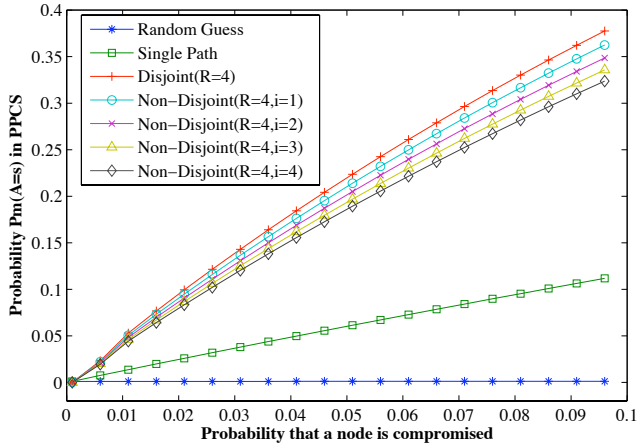
Based on the optimal guessing strategy discussed in the previous subsection, we can now evaluate $P_{f,s}$ and $P_{i,s}$ and determine the impact of PPCS. If an adversary knows its position on the path, $P_{f,s} = 1$ and $P_{i,s} = \frac{1}{G}$. In cases in which an adversary does not know its position on the path, such as if RTTL is used with PPCS, $P_{f,s} = 1$ and $P_{i,s} = 0$. This is because all adversaries will always guess the previous hop of a first adversary (the same guess), so in cases in which the first hop node is an adversary, all guess will be correct, and in cases in which the first hop node is not an adversary, all guesses will be incorrect.

We now extend this analysis to consider the impact of MPRF on security. PPCS establishes multiple paths between the source and destination. With the assumption

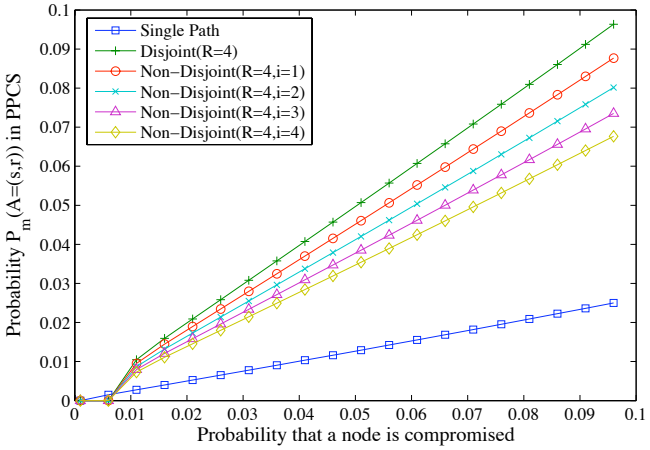
that each path of the R paths is disjoint, the probability an adversary discovers a source or destination is

$$P_m(A = s) = 1 - (1 - P(A = s))^R \tag{5.4}$$

In a disjoint multi-path environment, intermediate nodes have only one previous and next hop nodes. Since intermediate nodes do not know their position on a path, compromised nodes have the same probability $P_{f,s}$=1 and $P_{i,s}$=0 which is used to compute $P_m(A{=}s)$.



(a) Source Anonymity



(b) Source and Destination Linkability

Figure 5.1. Probability of an adversary

Figure 5.1 shows the effectiveness of compromised nodes in a disjoint multiple-path

environment. An adversary has a higher probability of guessing the source in a multiple disjoint path environment since more information may be open to more compromised nodes.

However, MPRF uses multiple non-disjoint paths. Thus every intermediate node may have multiple forward and backward hops for a flow. Furthermore, the first hop node on one path may be a non-first hop node on a different path of which it is a part. These multiple incoming links increase the number of choices for guessing, and hence reduce the probability of an adversary guessing correctly as shown in Figure 5.1 (a). In fact, the probability of guessing correctly is insignificant, when the adversary uses a random guessing strategy.

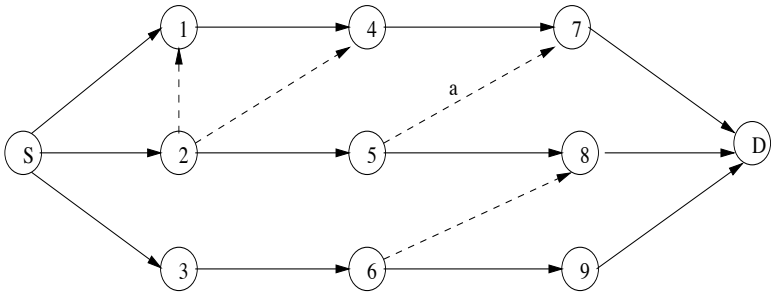


Figure 5.2. Non-Disjoint Multi-Paths

In Figure 5.2, the addition of each dotted link increases the incoming degree of the corresponding nodes(1, 4, 7, and 8). From this, we can compute the average incoming degree of a node, $\frac{W+i}{W}$, where W is the number of nodes on disjoint multipaths and i is the number of added directed links.

Table 5.2. Impact of PPCS on Probability

Probability	Perfect Anonymity	No PPCS	PPCS (Previous Hop Policy)		
			Single Path	Disjoint Multipath	Non-Disjoint Multipath
P_{fs}	$\frac{1}{N-C}$	$\frac{1}{G}$	1	1	$\frac{W}{W+i}$
P_{is}	$\frac{1}{N-C}$	$\frac{1}{G}$	0	0	0

i: number of directed links added to disjoint multipaths

Hence, the probability that an intermediate node determines a node from candidate

previous hop nodes is $\frac{W}{W+i}$. $P_{f,s}$ becomes $\frac{W}{W+i}$. $P_{i,s}$ is still 0 since intermediate nodes beyond the first hop will always guess wrong. Figure 5.1 (a) compares the probability that an adversary may guess a source in disjoint multi-path and non-disjoint multipath environments where 4 disjoint multipaths exist and the average path length is 5. This result demonstrates that MPRF in PPCS reduces the effectiveness of an adversary.

In summary, Table 5.2 shows the effect of using PPCS on the probability that intermediate and first hop nodes guess a source correctly. For destination anonymity, the analysis and equations are similar.

5.4.1.4 Source and Destination Unlinkability

If the path between a source and destination is known, the source and destination pair is also discovered. The probability that an adversary discovers the source and destination pair in a single path environment is

$$\begin{aligned} P(A{=}(s,r)) &= P(A{=}(s,r)|CH)P(CH) \\ &+ P(A{=}(s,r)|HC)P(HC) \\ &+ P(A{=}(s,r)|CC)P(CC) \\ &+ P(A{=}(s,r)|HH)P(HH) \\ &= P_{i+f,l}\sum_{k=1}^{L-1}(1-p)^{L-k}p^k\tbinom{L-2}{k-1} \\ &+ P_{i+l,l}\sum_{k=1}^{L-1}(1-p)^{L-k}p^k\tbinom{L-2}{k-1} \\ &+ P_{f+l,l}\sum_{k=2}^{L-2}(1-p)^{L-k}p^k\tbinom{L-2}{k-2} \\ &+ P_{i+i,l}\sum_{k=1}^{L-2}(1-p)^{L-k}p^k\tbinom{L-2}{k} \end{aligned} \tag{5.5}$$

$P_{*,l}$ denotes the probability that nodes en route guess the source and destination pair.

As discussed in the previous section, if an adversary knows its position on a path, the probability that the first/last hop node determines a source or a destination is 1. The

probability that other intermediate nodes guess a source/destination becomes $\frac{1}{G}$, since intermediate nodes know that their previous/next hop is not the source/destination and may guess one node of a set of possible sources/destinations. Therefore, if intermediate nodes know their position, $P_{f+i,l}$ and $P_{i+l,l}$ are $\frac{1}{G}$, $P_{i+i,l}$ is $(\frac{1}{G})^2$, and $P_{f+l,l}$ is 1.

If the adversary does not know its position on a path because of RTTL, the same guessing strategy as previously discussed is used. Thus, $P_{f+l,l}$ is 1, and $P_{i+f,l}|P_{i+l,l}|P_{i+i,l}$ become zero.

By extending the above single path case to a disjoint multi-path, the probability of discovering the source and destination pair is

$$P_m(A = (s, r)) = 1 - (1 - P(A = (s, r)))^R \tag{5.6}$$

In disjoint multi-path environments, intermediate nodes have the same probability as the single path to guess the source and destination pair. Figure 5.1 (b) shows the probability that an adversary discovers the communicating pair in a disjoint multi-path environment.

In a non-disjoint multi-path environment, we can apply the same reasoning as for the source anonymity case to determine that $P_{f+l,l}$ is $(\frac{W}{W+i})^2$, and $P_{i+f,l}|P_{i+l,l}|P_{i+i,l}$ become zero.

As Figure 5.1 (b) shows, an adversary has a lower probability to discover the communicating pair in non-disjoint multi-path environments than disjoint multi-path environments. This verifies that MPRF of PPCS mitigates the effectiveness of internal compromised nodes, while providing defense against eavesdropping attacks.

5.4.2 Eavesdropping

Since nodes in MANETs share a common broadcast channel, they overhear all communication within transmission range. Hence, an adversary may learn information by collecting and analyzing overheard data without revealing its existence. A set of local eavesdroppers form a global eavesdropper to cover a path. They may have a dedicated communication channel to exchange information.

In PPCS, every node en route uses the *Hint* to prevent correlation between forwarded packets and locally broadcasts the transformed packet. The eavesdroppers may not learn which node is the local sender and receiver of a packet, due to the local broadcasting and hop-by-hop transformation of packets. This limits eavesdroppers from obtaining information about the relationship between the incoming and outgoing packet of a node.

MPRF in PPCS spreads traffic over multiple paths, preventing eavesdroppers from learning the source, destination, or communicating pair by counting broadcast packets. Eavesdroppers located in different areas see different amounts of broadcast traffic with varying delay. Thus, a global eavesdropper is unable to discover significant information about node identity or flows.

To fully characterize eavesdropping requires a model of traffic that encompasses the amount of information an adjacent eavesdropping node can observe, and distribution of information sent through that victim and intermediate nodes, and the frequency and structure of the underlying traffic. We are currently developing a analytical model for this exceedingly complex environment.

5.5 Performance Evaluation

In this section, we evaluate the effect of PPCS on the performance of routing and data transmission. We performed our simulation in the ns2 simulator [72]. Specifically, we evaluate the effect of MPRF in which multiple paths are established and each packet on a flow may take a different path.

Table 5.3. Simulation Parameters

Simulation Time	900 seconds
Number of nodes	50
Area	900X900
Speed	Maximum 20 m/sec
Mobility model	Random Waypoint Model
Packet size	512 bytes
Traffic pattern	10 CBR/UDP connections (4 packets/s)

As a baseline multi-path routing protocol we use ad hoc on-demand multipath distance vector routing (AOMDV) [91]. To implement MPRF, we modified AOMDV to relax the node/link disjointness requirement and to randomly choose a next hop node at each intermediate node. Finally, to determine the impact of randomly changing the node pseudonym during the life of a flow, we modified MPRF to create a version that uses stable node pseudonym, called S-MPRF. Table 5.5 summarizes the simulation environment.

We measured packet delivery ratio (PDR), end-to-end packet delay, and routing overhead with different pause times under a random waypoint mobility model.

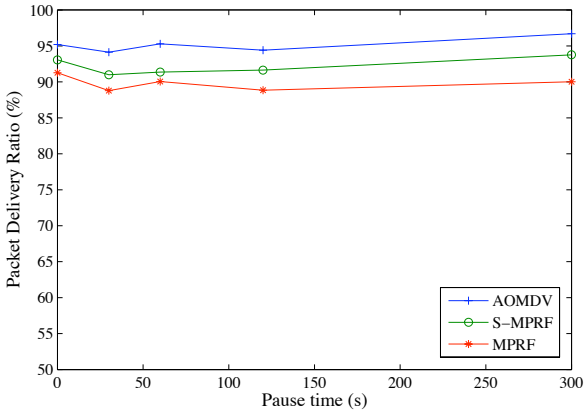
MPRF increasingly degrades the packet delivery ratio as mobility increases. Since each packet takes a different path, packets are more vulnerable to link failure or network congestion. Figure 5.3 (a) shows that the packet delivery ratio is decreased 3% and 5% in S-MPRF and MPRF, respectively. This result shows that the impact of changing node pseudonyms is small. The fact that multiple paths are susceptible to breaking for each flow, increases the routing overhead required to overcome these failures. As shown in Figure 5.3 (c), there is a 42% increase in routing overhead in MPRF over AOMDV.

In traditional routing protocols, packets are transmitted on the shortest path. With MPRF packets are randomly distributed to across multiple paths. Because some paths will be longer than the shortest path, the end-to-end packet delay will increase. Figure 5.3 (b) shows a 51% increase in packet delivery delay in MPRF and S-MPRF. We discuss the trade-offs between the security and performance in the next section.

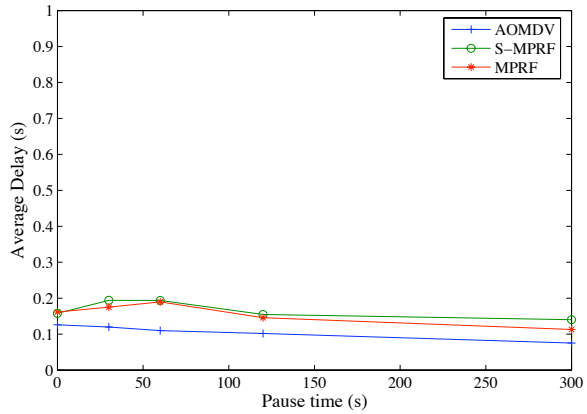
5.6 Discussion

In this section we discuss the trade-offs of MPRF. According to the analysis in Section 5.4.1, as the number of paths increases, the probability of an internal adversary compromising anonymity increases. While using non-disjoint paths is better than using disjoint paths, both are less secure against internal attackers.

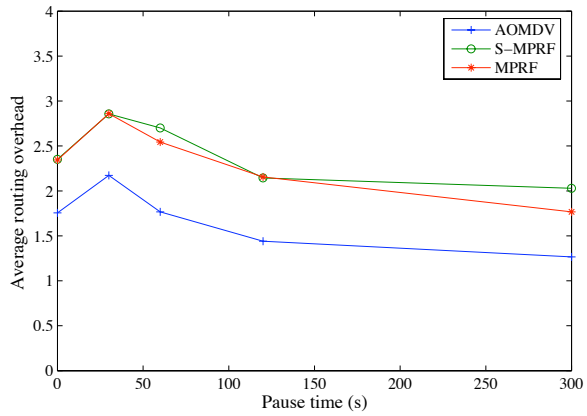
Although a single path solution is more secure against internal compromised nodes,



(a) Packet Delivery Ratio



(b) Packet Delay



(c) Routing Protocol Overhead

Figure 5.3. Performance with different pause times

it is less secure against eavesdroppers. To combat these attacks, it is better to establish more paths to distribute traffic. As an extreme example, if a packet is broadcast over the entire network (the number of multiple paths is infinite), eavesdroppers may not discover a flow at all.

Based on a security perspective alone, the choice of using MPRF should be based on a risk analysis of the network. If an attacker is more likely to be external, MPRF should be used. If the attacker is more likely to be internal, it should not.

If MPRF is to be used, the packet forwarding performance of the network will decrease as discussed in 5.5. Disjoint multi-path forwarding provides better packet delivery ratio (3-5%) than the non-disjoint multi-path forwarding used in MPRF. In non-disjoint multi-path environments, an intermediate node may receive packets of a flow from multiple neighbors which may cause more collisions on the wireless interface. However, given that the difference in performance is small, using MPRF is advisable as it does improve security as shown in Figure 5.1.

Chapter 6

SET: Detecting node clones in Sensor Networks

6.1 Background

In the previous chapters, we address data security along with high performance in Mobile IP, security and privacy of traffic relaying reports in MANETs, and communication privacy in MANETs. In this chapter, we address clone detection, an important, yet overlooked, security issue in the wireless sensor networks. Unlike traditional network equipment, sensors have very limited resources: low processing capability, small memory size, and limited power supplies. Many sensor networks are self-configured with no centralized control. This enables unattended sensor deployment into inaccessible and hostile environments. These environments, however, often make sensors susceptible to capture and compromise. Once a sensor is compromised, the information inside is easily accessible. An adversary may replicate captured sensors and deploy them in the network to launch a variety of insider attacks. This attack process is broadly termed as a *clone attack* [1].

Several researchers have discussed attacks that may be launched using cloned nodes [58, 94, 57]. Since a cloned node has legitimate information (codes and key materials), it may participate in network operations in the same way as a non-compromised node. It can

[1]The adversary may also place the extracted information from multiple sensors into one physical sensor, which may become a simplified sybil. If this sybil node is cloned, we will detect it.

then launch a variety of attacks. For example, it may create a *black hole*, initiate a *wormhole attack* [95] with a collaborating adversary, inject false data [96] or incorrectly aggregate data to bias results [97, 96]. In an environment in which sensed data must be kept private, cloned nodes also leak data. These clones impact the ability of a network to operate securely and maintain privacy.

A simple way to address clone attacks is to have every sensor send an authenticated report of itself and its neighbors to a base station using an existing data forwarding protocol. The symmetric key shared between the base station and sensor is used to authenticate the report. However, this simple method incurs high communication overhead due to the redundant information being reported. Furthermore, this approach is vulnerable to attacks in which a compromised node selectively drops reports on the way to the base station. Researchers have also proposed using witness-based schemes [60], but these rely on public key cryptography which may not be practical for sensor networks.

To overcome the limitations of existing solutions, we propose an effective and efficient scheme to detect clone attacks. We leverage the following observations to form the basis of our detection scheme. A sensor network can be modeled as a set of non-overlapping subregions. All nodes in the network have unique identifiers. Sensor nodes in each subregion form an exclusive subset. Since node identifiers are unique, the intersection of any two subsets should be empty. If an adversary replicates nodes and deploys them in the network, the intersection of subsets including these replicated nodes will not be empty, and hence a clone attack can be detected. We call this solution *SET*.

The first research challenge is to form exclusive subsets in a secure way. To address this challenge, we propose an algorithm in which an exclusive subset is securely formed among one-hop neighbors in a distributed way, while providing the authentication of nodes' subset membership. To efficiently and reliably compute set operations on these subsets, we employ a tree structure to compute non-overlapped set operations and integrate an interleaved authentication technique into the set operations on the tree to preserve the reliability of set operation results. We further fortify *SET* with randomiza-

tion, which makes the exclusive subset and tree formation unpredictable to the adversary as well as makes the system more communication and memory efficient.

We provide a security analysis of *SET* considering both individual and colluding attackers. The analysis shows that attackers may only be effective in a very limited number of scenarios. Further, we show that these scenarios are very unlikely to occur. We also analyze the communication overhead of *SET* and verify the analysis by simulation. The analysis and simulation results confirm that *SET* is an appealing solution in sensor networks due to its efficiency and low communication overhead, while providing trustworthy detection of node replication.

The remainder of this chapter is organized as follows: Section 6.2 describes the network model and assumptions, and examines the threats of an adversary. Section 6.3 details each component of SET. Section 6.4 shows the security analysis of SET in various settings that an adversary may use. In Section 6.5, we further analyze the performance and show the simulation results of SET. In Section refsec:set-diss, we conclude this chapter with a short discussion.

6.2 Network and Threat Model

6.2.1 Network Model and Assumptions

We deal with a sensor network which consists of a base station (BS) and a large number of low-end sensors. We model the wireless sensor network as an undirected graph $G = (V, E)$ where V and E are a set of nodes and edges, respectively. We use a unit disk graph model so that there exists an edge between nodes u and v, $(u, v) \in E$, if the Euclidean distance of u and v satisfies $|u - v| \leq 1$. We assume that the sensor network is a connected graph, i.e., there exists a path between any two sensor nodes.

We assume that the base station has information (e.g., key materials) of all deployed sensors. Each sensor has a unique key shared with the base station which can be used for computing authentication codes or encrypting data to the base station. In addition, sensors are able to establish a pairwise key with other peers to support secure peer-to-

peer communication. We employ an id-based pairwise key establishment scheme such as [98, 83, 80, 58] in which the keying material of a node is bound to its identity. Thus a node cannot lie about its id to neighbors, and a node knows the authenticated ids of all its neighbors. We also assume that a broadcast authentication protocol such as μTESLA [99] can be used to authenticate packets broadcast by the base station. We assume that the communication between the base station and sensors is made reliable by using one of many retransmission techniques.

6.2.2 Threat Model

We consider a setting in which the capability of the adversary is bounded such that only a limited number of sensors are compromised. Compromised nodes are totally in control of the adversary. An adversary may capture a few sensors, copy the information into its own sensors, and deploy the clones in places that are intelligently decided. Since cloned nodes have authenticated information extracted from the compromised sensors, they can be involved in network operations and launch various insider attacks.

The adversary may try to conceal the existence of cloned nodes. To conceal their existence, the adversary may interfere with the detection algorithm. For example, if sensors are required to report their presence (identifier) regularly, cloned nodes may not participate unless there exists a scheme to monitor sensors' regular reports which is difficult in sensor networks. An adversary may also drop or manipulate the reports of others that they are forwarding. Cloned nodes may also collaborate by eliminating cloned identifiers from reports.

6.3 SET: Clone Detection in Sensor Networks

In this section, we present a dependable and resilient defense scheme, *SET*, to detect clone attacks, i.e., to detect duplicated sensors. *SET* consists of five components: exclusive subset construction, authentication of subset covering, distributed set computation and interleaved authentication on subset trees, and verifiable random selection which further

optimizes *SET*.

As described in the introduction, our solution is based on a set model of a sensor network in which exclusive subsets are formed and a report of each subset is transmitted to the base station.

Due to the typical random deployment of sensors, it is challenging to construct exclusive subsets in the network. We first present an *Exclusive Subset Maximal Independent Set* (ESMIS) algorithm by which exclusive subsets are formed in a distributed way in the network (subsection 6.3.1). To ensure secure subset construction in a network with compromised nodes, we propose to integrate the ESMIS algorithm with an authentication scheme (subsection 6.3.2). We optimize *SET* by applying randomization to the exclusive subset formation, without losing security and exclusiveness (subsection 6.3.3). To perform efficient and reliable set computation in the network, we propose a multiple tree based set computation scheme so that intersection and union of subsets can be efficiently computed (subsection 6.3.4). Finally, we present an interleaved authentication scheme, to preserve the reliability of set computation on a tree (subsection 6.3.5). In the following subsections, we detail these components of *SET*.

6.3.1 Exclusive Subset Construction

The first component of our system creates exclusive subsets in the network. Each of these subsets will have a subset leader (SLDR). The SLDR reports the members of its subset and itself; if all reported subsets have no intersection, there are no clones.

A trivial way of constructing exclusive subsets is for each node to be a subset. Each node sends a report of itself either to the base station or to randomly selected nodes using a variety of protection mechanisms [60]. This will incur high communication overhead. To reduce this cost, we can define a subset that covers nodes over multiple hops. However, it is difficult to ensure that every node within a defined number of hops is exclusively in a subset.

To reduce the communication overhead of a single node subset algorithm, and to

ease the security design in a multi-hop covering subset, we propose to form a secure and exclusive subset. The scope of the subset is the transmission range of a node. The algorithm has two parts. First, set leaders (SLDRs) are determined such that no two SLDRs are within transmission range. Second, nodes that are not a SLDR associate with a single SLDR to form exclusive subsets.

This setting exactly matches the maximal independent set (MIS) problem [100]. By definition, in a graph $G = (V, E)$, a maximal independent set is a subset of V such that (1) there is no edge between nodes in the MIS, and (2) other nodes not in the MIS have at least one neighbor in the MIS [100]. However, the existing MIS algorithms are designed without considering security.

Protocol Description. Now, we present a secure distributed MIS approach to form exclusive subsets, preventing a specific node from being the target of an adversary. We use members of the MIS as SLDRs. Property (1) of a MIS guarantees that the SLDRs have no edges between them. The second property allows us to easily form an exclusive subset as we describe below.

We now describe the distributed MIS algorithm, *ESMIS* shown in Algorithm 1. We define three states (*Init*, *Ruler*, *Ruled*) of a node. The base station initiates detection by generating a random *seed* and broadcasting it to the network. On receiving the *seed*, every node sets its initial state as *Init* before starting Algorithm 1.

The basis of the ESMIS algorithm is a hash function $H_1 : (seed|x) \rightarrow y \in [1..d]$, where d is the average degree of a node in the network and x is a node ID. Since each node has a list of neighbors, a node locally computes H_1 for itself and its neighbors. The node having the largest result, H_{max}, among neighbors becomes a SLDR and changes its state to *Ruler*. If there exists more than one node with the same H_{max}, the node having the biggest identifier will be the SLDR. The SLDR announces itself by sending an ANN(ounce) message.

Other nodes that have a value smaller than H_{max} wait for an ANN message. If a node in *Init* state receives an ANN message, the node changes its state to *Ruled* and

Algorithm 1: ESMIS Algorithm (Code of node v)

Input: $N_v = \{v_1, v_2, ..., v_k\}$ // a set of neighbors of v;
$seed$, $State = Init$, $SLDR = 0$
Procedure:

```
1: H_v = H_1(seed|v);
2: H_{n,max} = max{M_i = H_1(seed|v_i)|for all v_i ∈ N_v};
3: H_max = max{H_{n,max}, H_v};
4: if (H_max = H_v) then
5:    State = Ruler; Broadcast an ANN message; Stop;
6: endif
7: Start WAIT timer;
8: while State = Init do
9:    if State = Init and COV messages received from all neighbors then
10:      State = Ruler; Broadcast an ANN message; Stop;
11:   endif
12:   if State = Init and ANN message received from v_i then
13:      State = Ruled; SLDR = v_i; send a COV message to neighbors; Stop;
14:   endif
15:   if WAIT timeout and there are at least one neighbor in Init then
16:      if H_v is the largest among these Init state neighbors then
17:         State = Ruler; Broadcast an ANN message; Stop;
18:      else
19:         Start WAIT timer;
20:      endif
21:   else
22:      State = Ruler; Broadcast an ANN message; Stop;
23:   endif
24: endwhile
```

saves the SLDR identifier in the ANN message. This implies that the node becomes a member of the subset in which the SLDR node is the head. When a node transitions into the *Ruled* state it sends a "covered" message (COV) to its neighbors which includes its identifier and its SLDR identifier. *A subset is composed of a SLDR node and member nodes covered by the SLDR.* The SLDR node identifier is used to identify the subset.

Figure 6.1 shows an example with a small network which consists of 23 nodes and has $d = 4$; the value outside the circle of each node is the H_1 computation result. This shows one example of constructing seven subsets according to Algorithm 1. Nodes 2, 3, 4, 8, 14, 19, and 23 are selected as SLDR nodes of the following subsets respectively:

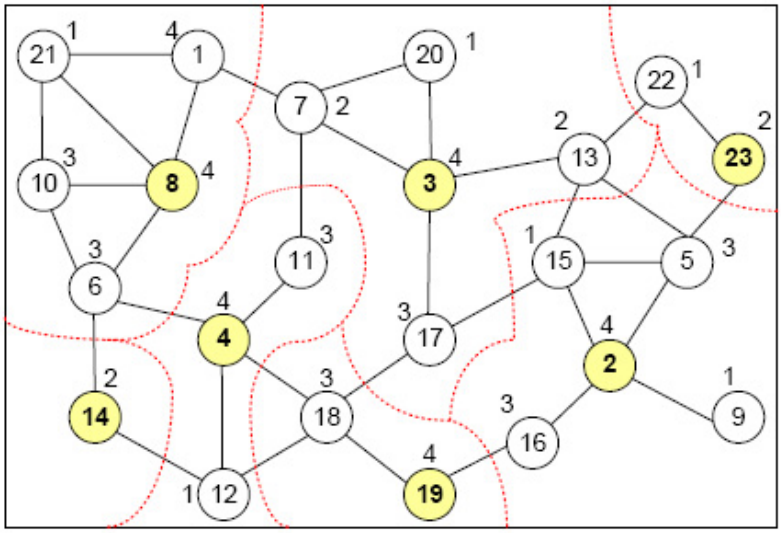


Figure 6.1. An example network with seven subsets generated by the ESMIS algorithm

$\{2, 5, 9, 15, 16\}$, $\{3, 7, 13, 17, 20\}$, $\{4, 11, 12\}$, $\{1, 6, 8, 10, 21\}$, $\{14\}$, $\{18, 19\}$, $\{22, 23\}$.

Since the algorithm is distributed and parallel, there exist race conditions leading to potential conflicts. For example, node 6 is located in an overlapped region and receives ANN messages from nodes 4 and 8. Without loss of generality, node 6 is covered by the SLDR from which it receives the first ANN message and simply ignores any other ANN messages received later.

Now, consider node 14 which is waiting for an ANN message because it does not have H_{max} amongst its neighbors. In this example, 14 only receives COV messages from neighboring nodes 6 and 12, i.e., it does not receive any ANN messages. To resolve this case, if a node that does not have H_{max} receives COV messages from all neighboring nodes, it becomes a SLDR. In this case node 14 becomes a SLDR (lines 9 and 10 in Algorithm 1).

Finally, a deadlock may occur if two or more adjacent nodes are in *Init* state, neither having H_{max}, and SLDR candidate neighbors are covered by other SLDRs out of transmission range of these nodes. Nodes 22 and 23 in Figure 6.1 are in this situation. The same rule is applied to break this deadlock. If a waiting timer ($WAIT$) expires and there exists more than one node in *Init* state within transmission range, a node (23 in

the example) having the largest H_1 result among the nodes in deadlock becomes a SLDR (lines 16 and 17 in Algorithm 1).

Correctness analysis. The ESMIS algorithm takes time proportional to the number of neighbors. Note that this does not depend on the total number of nodes in the network. After executing the ESMIS algorithm, each SLDR knows which neighbors it covers. It then sends a report of its subset to the base station. The following lemma shows that the ESMIS algorithm generates a MIS.

Lemma 1. *A set of SLDRs generated by Algorithm 1 (ESMIS) is a MIS.*

Proof: (*Sketch*) To prove this, we need to show that two statements hold: 1) no two SLDRs have an edge between them and 2) there is no node to be added to the MIS after finishing the ESMIS, i.e., all nodes not in the MIS have at least one neighbor in the MIS.

1*)* Let us assume that there is an edge between two SLDRs in the MIS generated by Algorithm 1, i.e., two SLDRs are selected in transmission range. In Algorithm 1, there are three cases that a node becomes a SLDR: a node has the largest H_1 result within transmission range (line 5); it receives COV messages from all neighbors (line 10); a node has the largest H_1 among two or more *Init* state nodes in *deadlock* (line 17). First, the selection of a SLDR is based on the H_1 result and unique node identifier. Every node computes H_1 of itself and neighbors. For two nodes in transmission range to become a SLDR, the two nodes must have the same largest H_1 result and the same identifier. However, in the network, since every node has a unique identifier, this cannot happen. This contradicts the assumption. The second case does not occur since if there exists at least one SLDR, the other node receives at least one ANN message (i.e., it will not receive COV messages from *all* neighbors). The third case cannot occur for the same reasoning as the first case. Therefore, there cannot be an edge between SLDRs.

2*)* Under the assumption of network connectivity, every node in the network starts Algorithm 1 when it receives the *seed*. Once a node performs the ESMIS, its state is either *Ruled* or *Ruler*. Nodes in *Ruler* state are already in the independent set. Let us assume that there is a *Ruled* node which can be added to the independent set. The *Ruled* node is already connected to a SLDR which covers it. The addition of the *Ruled* node

to the MIS, i.e., the node becomes a SLDR, creates an edge between two SLDRs. This cannot occur as shown in 1 above. Hence, once every node finishes the ESMIS algorithm, there is no more node to be added to the MIS. □

In the ESMIS algorithm, an adversary is unable to predict which nodes will be SLDRs because of the use of the random *seed*. To further strengthen security, each time that the base station initiates the detection of node replication, it broadcasts a different *seed*. Hence, although compromised nodes may, in some specific scenarios, collaborate to successfully hide themselves, they can be detected with a significant probability in other rounds in which the base station generates a different seed. We will examine this probability in the next subsection.

6.3.2 Authenticated Subset Covering

The ESMIS algorithm presented above works well in a network in which no node is compromised. However, sensors may be deployed in a hostile environment in which an adversary may try to hide its existence by not participating in the ESMIS algorithm or by generating a bogus COV message with a fabricated SLDR identifier.

To curb this attack, we present a membership authentication approach. The basis of this approach is to have a *Ruled* node inform all its neighbor SLDRs of its covering SLDR. This way, the neighbor SLDRs can authenticate the veracity of the statement as follows. Suppose that a node i receives the first ANN message from SLDR g and a second from SLDR h. Node i is covered by g. As presented in the previous subsection, node i sends a COV message (ID=i, SLDR=g) to its neighbors. SLDR h determines that i is not a member of its subset. Node i then transmits the identifier of SLDR h to SLDR g so that SLDR g generates a membership authentication of i to SLDR h. In the remainder of this chapter, K_{ij}=K_{ji} denotes a pairwise key between nodes i and j. The message that g sends to h is as follows:

$$g \rightarrow h : [i, g, MAC(K_{gh}, i|g)]$$

The SLDR h can confirm that SLDR g covers the neighbor i by checking this message and saving $MAC(K_{gh}, i|g)$, referred to as *membership MAC*, which is used to vouch for SLDR g's covering of i.

Neither can a node generate a bogus COV message carrying a fraudulent SLDR identifier. For example, consider the case in which node i sends a COV message including SLDR f which does not exist. Node i should be able to generate the membership authentication computed by f to h with a pairwise key which has been established between h and f. However, node i can neither generate this key nor the membership authentication.

During this authentication process, a SLDR collects a set of neighbor SLDRs and shortest path information to them. The maximum distance of shortest paths between two SLDRs is three hops, i.e., at most two covered nodes may be on the path between two neighboring SLDRs.

After this authentication, each SLDR generates a report that includes a set of members, non-member nodes, and a MAC for the report. A report generated by SLDR g is as follows:

$$M_k = [k, S_{gk}]$$

$$g \rightarrow BS : [S_g, M_1, \ldots, M_m, MAC(K_{g,BS}, S_g|M_1|\ldots|M_m)]$$

, where non-member neighbors ($\cup_{k=1}^{m} S_{gk}$) are covered by m neighbor SLDRs, S_{gk} is a set of non-member neighbors covered by SLDR k, and S_g is a set of nodes covered by SLDR g. In summary, the subset report from a SLDR includes non-members and SLDRs covering the non-members, as well as a set of members and a MAC for the report. Otherwise, corrupted SLDR and member nodes may collaborate and not report a member.

6.3.3 Verifiable Random Member Selection

In the basic scheme, every node's identifier is reported to the base station. This may cause the report packet size to become large near the top of the tree. To address this, we further optimize *SET* by using randomization for the subset formation, in which a SLDR generates a report of randomly selected members, instead of all members.

First, according to Algorithm 1, SLDRs are elected and exclusive subsets are constructed in the network. Each SLDR then selects a subset of members at random, based on a threshold σ which is a system parameter in $(0, \ldots, 1]$. For instance, if σ is 0.75, each SLDR generates a subset report of itself and randomly selected 75% of its members. However, this solution raises a question of how to verify whether a node is not selected or is not included intentionally to hide its existence by an adversary, if it is a clone.

To overcome the limitation of the simple randomization approach, we present a verifiable random selection scheme. The key idea of the verifiable random selection is to enable the base station to determine which sensors should be reported. To do this, the base station releases additional information (a bit array of $m + 1$ bits, where m is the maximum value of $y{=}H_1(seed|x)$ computation [2]), referred to as a *mbins*, when broadcasting the *seed*. A SLDR generates a report of members whose y-th bit of the *mbins* is set to 1. For example, if m is seven and the *mbins* is set to "10101010" (left to right), a SLDR selects members whose y is 0, 2, 4, or 6. The base station can control the number of reported nodes by setting more (or less) bits of the *mbins* to "1". Since a SLDR already has a list of neighbors and their H_1 values, the decision does not incur any additional computation or communication. Furthermore, due to the randomness of the *seed*, the adversary cannot predict whether clones will be selected or not.

Since the base station knows the list of all sensors, it can check if the reports include all the sensors to be reported. Except for the selection of members in each exclusive subset, all the operations are performed in the same way. Hence by using the verifiable

[2]The maximum value of $y{=}H_1(seed|x)$ computation indicates the degree of a node as described in 6.3.3. Therefore, the size of m will be only a few bits

random selection, we can reduce the message size of reports, while achieving the verifiable subset report of sensors. The limitation of this approach is that it may take longer (multiple rounds) to detect clones. However, as the analysis in the subsection 6.4.1.3 will show, the probability that a base station may run more than one round to detect clones is very low. We will show the detailed security analysis later in Section 6.4.

6.3.4 Distributed Set Computation on Subset Trees

The ESMIS algorithm outputs a set of unit exclusive subsets in the network. To detect node replication, each SLDR can send its report to the base station. Due to the random deployment of sensor nodes, the base station may not know which nodes are selected as SLDRs. This may enable the adversary to drop reports from subsets on the way to the base station.

To support more efficient and attack-resilient detection of set operations, we employ a random tree structure. A tree enables the hierarchical construction of subsets, guaranteeing exclusiveness because of its no-cycle condition. A subset generated by the ESMIS algorithm is a node of the subset tree.

The tree construction is initiated by a root SLDR. The root sends a final report collected from the tree to the base station. This leaves open a question of which SLDR will be the root. A compromised node may try to be a root to control the set computation. To overcome the vulnerability in a single-rooted and fixed tree, we present a multiple tree based approach in which multiple roots are decided randomly in the network. Each root initiates tree construction so that multiple trees of subsets are constructed independently and in parallel. As a result of multiple tree construction, the network can be divided into non-overlapping regions. The intersection and union of subsets are performed on each tree in a distributed way. If the intersection of all subsets computed in the base station is empty, there are no clones detected in the network.

The *seed* that the base station broadcasts is also used for choosing roots randomly in the network. Roots are determined by computing another hash function, $H_2 : seed \rightarrow M$,

where M is an integer value in $[1, \ldots, N]$. If a subset has at least one node whose identifier is in $[M, M + B)$, where $M = H_2(seed)$, the SLDR of the subset becomes a root. For example, if nodes in $[M, M + B)$ are in different subsets in the network, B trees will be constructed. The value B is a system parameter which decides the maximum number of trees constructed in the network. This way, the adversary is unable to determine roots and the tree structure.

The tree construction starts from each root by discovering the SLDRs neighboring the root; the SLDRs that are neighbors of the root set the root as their parent and become the root's children; for each child, the same tree extension is performed recursively. Discovering neighbors is carried out by sending a message, *CTREE*, carrying the root identifier to neighbors. Table I shows the tree construction pseudo code on a tree. During the tree construction, SLDRs obtain path information on the tree, i.e., the ancestor SLDRs on the tree. We present how this path information is used to authenticate set operations performed on the tree subsequently.

Table 6.1. Pseudo Code of Constructing a Subset Tree

At Root
Send CTREE ($SID_1 = r$) to all neighboring SLDR nodes
Intermediate node i
:: Receive CTREE($\{SID_k\}_{k=1,\ldots,i-1}$) from SLDR SID_{i-1}
if (Root = 0 & Parent = 0) **then**
    Root = SID_1
    Parent = SID_{i-1}
    Ancestors = $\{SID_k\}_{k=1,\ldots,i-1}$
    Add myself (SID_i) to $\{SID_k\}_{k=1,\ldots,i-1}$
    for all neighbor SLDR N_k, except SID_{i-1} **do**
        Send CTREE($\{SID_l\}_{l=1,\ldots,i}$) to N_k
    endfor
    Send CTREE_RESP to SID_{i-1}; done;
endif
Send CTREE_RESP to SID_{i-1}; done;

Unless a SLDR j is a root, it becomes a child of another SLDR i from which it receives the first CTREE message. The SLDR j sets i its parent and sends to i a

CTREE_RESP message (ID=j, parent=i, root=SID_1). When SLDR i receives the CTREE_RESP message in which a parent field is set i, it adds j to its children list. A SLDR may receive multiple CTREE messages which have a different root SLDR or parent. A SLDR becomes a leaf on the tree in the following situations: either it does not have any neighbor SLDR except its parent or all neighboring SLDRs except the parent already joined a tree.

After the tree construction, the leaf SLDR sends its subset report to its parent. The parent collects its children's subsets, and computes the intersection of these subsets to check if a clone exists. If not detected, it generates a union of its children's subsets and sends this new report to its parent. Each root forwards its final report (union of all subsets in a tree) to the base station.

If the intersection in the midst of the tree is not empty, this implies that clones exist in the subtree. The notification of node replication will be transmitted to the base station which will take further action.

6.3.5 Interleaved Authentication on a Subset Tree

An important issue we must address in the tree-based set operation is the dependability of the set operation results (intersection and union of subsets) on the tree. A parent collects its children's subsets, and computes the intersection and union of the collected subsets, with its own subset. If the parent is a corrupted node, it may delete cloned identifiers from the union.

Therefore, the set operations should be verified. To address this, we leverage the previous work [101] to propose an interleaved authentication scheme on the tree. During the tree construction, a SLDR obtains the path information from the root to itself. When a SLDR sends a report to its parent, it computes a keyed MAC, such as an HMAC [102], not only for its parent, but also for its grandparent (interleaved MAC).

If a corrupted parent changes the results of set operations, the grandparent can detect the inconsistency by computing and checking the interleaved MACs. We quantify the

security afforded by this approach in the next subsection.

The report is different based on whether a SLDR is a leaf or an intermediary on the tree. Let us denote i, p, and g as child, parent, and grandparent SLDRs, respectively, and S_i as a subset of child i.

Leaf: Leaf SLDR i computes MACs for its parent and its grandparent. A subset report that leaf SLDR i generates is composed of its subset, MAC of the subset report, and interleaved MAC as follows:

$$i \rightarrow p : [S_i, MAC(K_{ig}, S_i), MAC(K_{ip}, S_i|MAC(K_{ig}, S_i)]$$

A parent SLDR checks the $MAC(K_{ip}, S_i|MAC(K_{ig}, S_i))$s of all children i to verify the reports from its leaves.

Intermediate: An intermediate SLDR on a tree receives subsets from its children and computes the intersection and union of the received subsets and its own subset. It then generates a report to its parent. The report generated by an intermediate node also carries interleaved MACs from the children and an interleaved MAC computed by itself for its grandparent. Suppose that SLDR i has k children. A report that SLDR i generates is as follows:

$$\begin{aligned} M_1 &= (S_i, S_1|| \ldots ||S_k, \{MAC(K_{jg}, S_j\}_{j=1,k}) \\ M_2 &= (S_1|| \ldots ||S_k||S_{k+1}), \text{ where } S_{k+1} = S_i \\ M_3 &= (M_1|MAC(K_{ig_i}, M_2)) \\ i \rightarrow p_i &: [M_1, MAC(K_{ig_i}, M_2), MAC(K_{ip_i}, M_3)] \end{aligned}$$

where p_i and g_i denote the parent and grandparent of SLDR i respectively, and $||$ indicates the concatenation of subsets. The grandparent g_i uses M_2 and $MAC(K_{ig_i}, M_2)$ to check the operation of node i and p_i.

If the child is not a leaf, the intermediate SLDR i checks the concatenated subsets and MACs from its grandchildren ($MAC(K_{jg}, S_j)$ for all grandchildren j), as well as MACs from its children ($MAC(K_{kp}, M_k)$ for all children k). This interleaved authentication

verifies the set operations of intermediate SLDRs and detects if they have removed identifiers. If all subset reports successfully pass the verification, the intermediate SLDR sends its parent a report which consists of a concatenation of children's subsets, its subset, interleaved MACs, and MAC for the report. Figure 6.2 shows an example of how the interleaved authentication works.

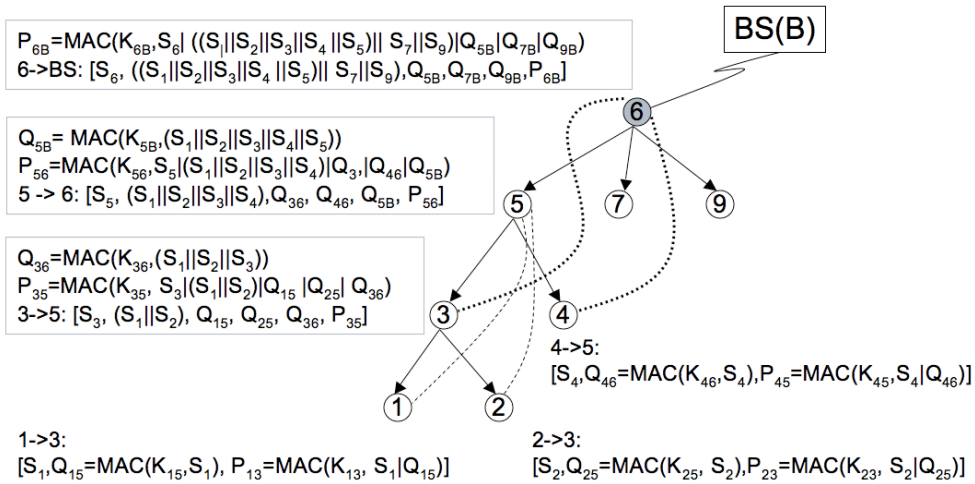


Figure 6.2. An example of interleaved authentication on a tree with height 3: SLDR 3 sends to SLDR 5 a report which includes its own subset S_3, concatenation of S_1 and S_2 ($S_1||S_2$) and their interleaved MACs (Q_{15} and Q_{25}), SLDR 3's interleaved MACs (Q_{36}) and P_{35}. SLDR 5 computes the interleaved MACs for the received subsets (S_1 and S_2) and check with the received MACS (Q_{15} and Q_{25} from SLDR 1 and 2).

6.3.6 Detection at the Base Station

Each root forwards its final report (union of all subsets in a tree) to the base station. The base station verifies the reports from roots by checking MACs generated by roots and interleaved MACs by their children. The base station detects the clone attack by computing the intersection of any two received subsets from roots. If it detects cloned nodes, it may revoke the corresponding nodes by broadcasting the list of cloned identifiers to the network.

A compromised SLDR may lie by inserting a nonexistent neighbor node into the report. The base station will then receive multiple identifiers for this non-compromised node. To deal with this case, the base station may run multiple rounds of *SET* since the

probability that the compromised node consecutively becomes a SLDR is very low. This basic scheme may be augmented with a reputation scheme to determine which nodes are inserting false reports. The details of this issue are outside the scope of this thesis.

Based on the received reports, the base station can detect clones unless compromised nodes collaborate to hinder the detection. There are limited scenarios under which the collaboration of compromised nodes can successfully hinder detection. We quantify the likelihood of these circumstances in the next subsection.

6.4 Security Analysis

In this section, we qualify the impact of compromised nodes on our detection scheme. We find that adversaries can only be effective in limited scenarios. We quantify the effectiveness of an adversary by analyzing the probability that the conditions for these scenarios hold true. As we demonstrate, there are two colluding scenarios where the adversary can hide cloned members with a very low probability. However, in *SET*, the concealed clones in a specific round can be detected in the subsequent rounds since subset and tree structures may change based on the new *seed* broadcast by the base station. Unless compromised, we assume that sensor nodes operate properly according to the defined protocols.

The following notation is used in the remainder of the analysis.

- n: Number of clones that have the same identifier.
- N: Number of nodes in the network.
- T: Number of trees constructed in the network.
- B: Parameter value used to choose multiple roots in the network.
- L: Average number of neighbor subsets.
- P_s: Probability that a node is selected to be reported.

- P_a: Probability that an adversary avoids detection.
- $P(j{=}\text{Comp.}){=}P_c$: Probability that node j is compromised.
- $P(\text{Child}(i) = j)$: Probability that SLDR j becomes a child of SLDR i..
- P_{cchild}: Probability that SLDR j is compromised and becomes a child of a SLDR.
- P_{pc}: Probability that parent and child SLDRs are compromised and the parent is not a root.
- $P_{r,pc}$: Probability that parent and child SLDR nodes are compromised and the parent node is a root of the tree.

6.4.1 Node compromise on the subset construction

6.4.1.1 Single Node Compromise

In performing the subset construction, a compromised node may strive to be a head of the subset by falsely claiming that it has the largest H_1 result. In the ESMIS algorithm, however, every sensor locally computes H_1 for itself and all neighbor nodes. Therefore, a compromised node cannot falsely claim that it has the largest result, since other nodes also have the computation results.

The compromised node may also attempt to avoid being reported as a member of a subset by either claiming that it is covered by a non-existent SLDR or by keeping silent. A covering SLDR provides the membership authentication to verify that it covers a node. Since the compromised node does not have a pairwise key for its non-existent covering SLDR, it cannot provide this membership authentication. This prevents a node from claiming coverage by a non-existent SLDR.

In the ESMIS algorithm, the protocol precludes a node from keeping silent while it has a neighbor relationship with other nodes. If a node attempts to hide itself by not responding an ANN message, the non-compromised SLDR and other neighbors will detect this.

6.4.1.2 Nodes in collusion

In our setting, a node may play one of two roles: member or SLDR. Thus, various colluding scenarios may take place: member nodes may collude (members in the same subset or members in the different subsets), or a member node and SLDR may collude.

Two types of attacks may be attempted by colluding member nodes. First, a colluding node may try to help its partner claim to be the SLDR. As discussed above, given that other non-compromised nodes have the H_1 results of all neighbors, this attack is not effective. Second, colluding nodes may try to be silent, thus going undetected. Since colluding members have neighbor relationship with their SLDRs, they cannot be silent, nor can they claim to be covered by a non-existent SDLR as discussed above.

Now consider the case of a compromised member i and SLDR j in collusion. If they are within transmission range of each other, one possible attack is for node i to claim itself as covered by j, and for j to eliminate i from its subset. In *SET*, SLDRs send their neighbor list (members and *non-members*) to the base station when a subset report is transmitted through the tree as described in Section 6.3.4. Thus, the base station will check if the reported non-members are completely covered by a SLDR when it receives all reports from root SLDRs. Since the corrupted SLDR j does not include i in its subset, the base station will detect this missing node. It will query the legitimate neighbor SLDRs which reported the non-members (including i) for the membership MACs and the SLDRs of the missing non-members.

However, suppose that the compromised member j has only one SLDR within range which is also compromised. Since there is no other SLDR to report j as a non-member, the collusion of these two nodes may work to hide the compromised member j. To investigate the effectiveness of this colluding attack, we performed simulation of a network of 800 nodes in which two random nodes, x and y, are cloned. We generated 2, 6, 12, and 20 clones of each compromised node and deployed them as pairs (x,y) within transmission range of each other in different parts of the network. For the case in which an adversary generates two clones of each compromised node, the attack was successful 0.3% of the

time. In the other cases (6, 12, and 20 compromised pairs), the adversary was never observed to be successful. Note, for this attack to be successful, *all* cloned pairs must be in a covered-SLDR relationship with no neighbor SLDRs; as the number of cloned pairs grows, this becomes a highly improbable scenario. This demonstrates that our subset construction and membership authentication schemes successfully mitigate the effectiveness of this colluding attack.

6.4.1.3 Effectiveness of Verifiable Random Selection

Now, we explore the effectiveness of an adversary in the verifiable random selection scheme. As discussed above, a single compromised node is not able to hide itself in *SET*. The random selection affects only the unit subset formation. Thus as a special case of node collusion, we separately analyze the effectiveness of an adversary to avoid detection during the subset construction in the random selection scheme.

First, we investigate the probability that an adversary successfully avoids detection. We need to consider two scenarios: either the identifier of clones is selected for reporting or not. Assume that n clones exist in the network. For an adversary to be successful in the first case, only a single clone node is reported; the other $(n-1)$ clones must be hidden. To do this, the adversary needs $(n-1)$ compromised SLDRs to collude with and hide the $(n-1)$ clones. For the later case, although the clone identifier is not selected, an adversary can avoid detection only when at most one clone is elected to a SLDR, because if multiple clones become a SLDR, each must generate a report and the base station will receive duplicate identifiers and detect the clone attacks. Let us denote $P_{1,SLDR}$ as the probability that at most one node becomes a SLDR among n nodes located outside transmission range of each other. Since every node within transmission range has the same probability ($\frac{1}{d}$, where d is the average degree) to be a SLDR, the probability, $P_{1,SLDR}$, is defined by

$$P_{1,SLDR} = \binom{n}{1}(\frac{1}{d})(1-\frac{1}{d})^{n-1} + (1-\frac{1}{d})^{n}.$$

Therefore, the probability P_a that an adversary successfully avoids detection using verifiable random selection is

$$P_a = P_s(P_c(1 - \frac{1}{d}))^{n-1} + (1 - P_s)P_{1,SLDR} \tag{6.1}$$

, where $(P_c(1 - \frac{1}{d}))^{n-1}$ indicates that each of the $(n - 1)$ clones that has not been selected as a SLDR is covered by a SLDR that is compromised. Equation 6.1 gives an upper bound of the probability that an adversary successfully avoids detection in the verifiable random selection scheme. This analysis assumes that if a SLDR and member are both compromised, the collusion will always be successful. However, this is not true as discussed in the previous analysis. Figure 6.3 shows that the adversary's effectiveness plummets as the number of clones (n) increases.

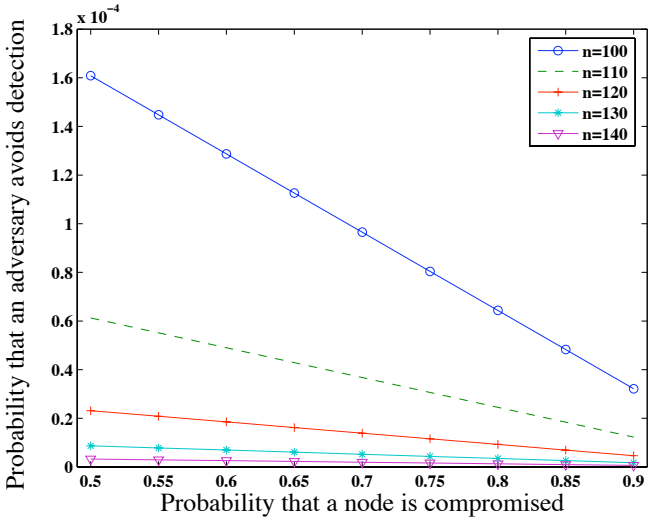


Figure 6.3. The probability that an adversary successfully avoids detection

Now, we analyze the expected number of rounds that the base station must run *SET* for detecting clones. With verifiable random selection, the base station may run multiple rounds to detect clone attacks because in a round, a portion of a total set is reported. Based on the probability distribution (6.1), the expected number of rounds is defined by $E(X{=}k){=}\sum_{k=1}^{\infty} kP_a^{(k-1)}(1{-}P_a)$, which can be simplified to $E(X{=}k){=}\frac{1}{1-P_a}$. Note that the expected number of rounds converges to one quickly as n increases. Thus we can argue

that the verifiable random selection makes *SET* more communication efficient, without sacrificing security. Furthermore, notice that the adversary must compromise at least $(n-1)$ other nodes to conceal clones.

6.4.2 Node compromise on the tree

In order to attack set operations on a tree, the compromised node must be a SLDR. In this subsection, we examine the impact of SLDR compromise.

6.4.2.1 Single Node Compromise

Once a compromised node becomes a SLDR, it also joins the tree construction and participates in set operations on the tree. The compromised SLDR may try to change the results of set operations on a tree by removing cloned node identifiers. The proposed interleaved authentication on a tree enables the detection of the changes by the compromised SLDR: if the corrupted SLDR removes identifiers from the subsets of its children, its parent SLDR will detect inconsistency by computing interleaved MACs for the received subsets and checking with the received interleaved MACs which are computed for the original subsets by the grandchildren.

6.4.2.2 Nodes in Collusion

The main goal of colluding SLDRs is to hamper the set operations so that cloned nodes may not be detected. There are four colluding scenarios: 1) compromised SLDRs belong to different trees; 2) compromised SLDRs are on different paths of the same tree; 3) compromised SLDRs are on the same tree, but are not in an immediate parent-child relationship; and 4) compromised SLDRs are in a parent-child relationship.

For the former two cases, if the compromised SLDRs are on the different paths or different trees, they cannot effectively collaborate. Their misbehavior on its tree or path is detected in the same way as the case of a single compromised SLDR.

As an example of the third case, suppose that a SLDR and its grandparent are compromised. The corrupted SLDR will send its subset to the parent SLDR which is not

compromised. This SLDR collects subsets from all its children. The legitimate parent SLDR sends the result to its parent (children's grandparent SLDR which is compromised). This grandparent may remove elements from the collaborating grandchild. In Figure 6.2, let us assume that SLDRs 2 and 5 are compromised. SLDR 3 sends a report to SLDR 5 which includes subsets S_1, S_2, and S_3. Corrupted SLDR 5 may remove elements from S_2 so that these nodes are not detected. However, the SLDR 6 will detect the inconsistency by computing $M_{36} = MAC(K_{36}, (S_1||S_2||S_3)_{rx})$ and checking if M_{36} is the same as the received MAC (Q_{36}) computed by the SLDR 3, where $(S_1||S_2||S_3)_{rx}$ is the received subsets from 5.

Therefore, an adversary can only be effective if the compromised nodes have a parent-child relationship (case 4). Since the adversary may be able to hide clones in this scenario, during the tree construction, a compromised SLDR may strive to have a parent-child relationship with another compromised SLDR with which it is a neighbor. In *SET*, however, due to the randomness in the subset construction, this cannot be planned, and if it occurs, is unlikely to occur repeatedly in subsequent rounds.

The parent-child scenario can be divided into two cases: the compromised parent is a root or the compromised parent is not a root. Note that a corrupted root can have a significant impact on the detection of clones. For these two operative scenarios, we quantify the effectiveness of the adversary by evaluating the probability that they can be in this relationship.

We first determine several probabilities that are used in the remaining analysis. A SLDR becomes a root if it has at least one identifier in $[M, M + B)$ in its subset. Hence, the probability that a SLDR becomes a root is

$$P_r = 1 - (1 - \frac{B}{N})^d$$

, where d is the average number of neighboring nodes in a subset.

In order for SLDR j to be a child of SLDR i, it should receive the first CTREE message from SLDR i, and it should not be a root because although it receives the first

CTREE message from i, it cannot be a child of i if it is a root. Hence, the probability that SLDR j becomes a child of SLDR i is

$$P(\text{Child}(i) = j) = \frac{1}{L} * (1 - P_r). \tag{6.2}$$

The first term comes from the fact that a SLDR can receive the first CTREE message from L neighbor subsets.

Parent-Child (P_{pc}): We first analyze the probability that SLDR j is compromised and that it is a child of SLDR i which is also compromised and not a root. Since the node compromise and tree construction events are independent, the probability of being a child of SLDR i and compromised is

$$\begin{aligned} P_{cchild} &= P(j = \text{Comp.}, \text{Child}(i) = j) \\ &= P(j = \text{Comp.}) * P(\text{Child}(i) = j) \\ &= P_c * \frac{1}{L} * (1 - P_r) \end{aligned} \tag{6.3}$$

by applying equation 6.2.

For the adversary SLDR i to be effective, it must have at least one compromised child SLDR. Thus, the probability that the adversary is effective is the same as the probability that SLDR i is compromised and it has at least one compromised child. In the case we examine here, SLDR i is not a root. Therefore, the probability is:

$$\begin{aligned} P_{pc}(\text{effective}) &= (1 - P_r) * P(i = \text{Comp.}) * \sum_{k=1} P_{cchild} \\ &= (1 - P_r) * P_c * (1 - (1 - P_{cchild})^{L-1}) \end{aligned} \tag{6.4}$$

Root ($P_{r,pc}$): If the compromised parent is a root of the tree, then the attack may have a large impact on the final set result. Equation 6.4 is the probability that the compromised parent SLDR i is not a root. Now, we examine the case in which SLDR i is compromised and is a root.

The probability that a parent SLDR i is a compromised root, and it has at least one

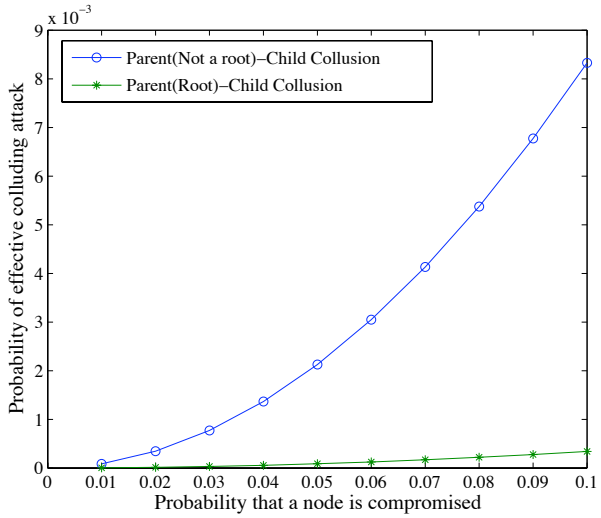


Figure 6.4. Effectiveness of an adversary ($L = 17$, $d = 10$, and $B = 4$)

compromised child is

$$P_{r,pc}(\text{effective}) = P_r * P(i = \text{Comp.}) * (1 - (1 - P_{cchild})^L) \tag{6.5}$$

Figure 6.4 shows the probabilities of P_{pc}(effective) and $P_{r,pc}$(effective). When the compromised nodes occupy 10% of the network, the scenario of collusion of a non-root parent and child SLDR has a probability 0.008 of occurring. For the scenario of a compromised root and its child in collusion, the probability of occurrence is $(0.3 * 10^{-3})$. In these cases, the adversary can hide clones only if all clones belong to the subtree rooted at the colluding parent-child SLDRs. Furthermore, in *SET*, since the base station generates a different *seed* each time, the probability that this attack may occur consecutively is statistically insignificant (0.008^2 and $(0.3 * 10^{-3})^2$ for P_{pc} and $P_{r,pc}$ respectively). We can conclude that our detection scheme based on a set model can support a reliable and secure detection of the clone attack.

6.5 Performance Analysis

Based on requirements of a large sensor network, we performed simulations on networks of 800, 1600, 2400, and 3200 nodes which are deployed randomly. The area of the network

is varied to achieve an average degree of 10. In this section, we study the performance of the ESMIS algorithm and analyze the overhead of *SET*, comparing with existing solutions. The analysis and simulation results show that *SET* outperforms the existing solutions.

6.5.1 Performance of ESMIS algorithm

As a reference point, we used a random MIS algorithm in which a node is randomly selected from V to join a MIS and its neighboring nodes are removed from V. This operation is performed continuously on V until V becomes empty.

We conducted simulations on the ESMIS algorithm and the random MIS algorithm 50 times to get the average size of the MIS. The results are shown in Figure 6.5. The results

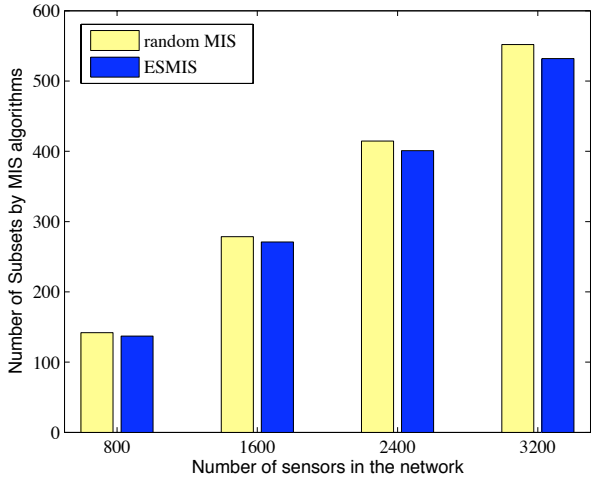


Figure 6.5. Comparison of the number of subsets.

show that the ESMIS algorithm creates fewer subsets than the random MIS algorithm, while it can be executed in a distributed and parallel way. Since communication cost is proportional to the number of subsets, we claim that the ESMIS algorithm is appealing due to its distributed nature and its lower communication overhead, which we quantify below.

6.5.2 Communication Overhead Analysis

In this subsection, we examine the overhead incurred by *SET*. We perform a worst case analysis and simulation.

In *SET*, the subset construction, membership authentication, tree construction, and subset reporting incur packet transmissions. The communication cost can be measured by the expected number of messages transmitted during these operations. Broadcast (μTESLA) by the base station costs O(N) message transmission in the network. During the construction of subsets and membership authentication, selected SLDRs broadcast an ANN message and send membership MACs to the neighbor SLDR if some of its members are overlapped. The covered nodes send a COV message to their neighbors, relay the received neighbor COV messages to their SLDRs, and relay the membership MACs. Therefore, the subset construction and membership authentication costs $O(N)$ message transmissions in the entire network.

In constructing a subset tree, the CTREE message is propagated through to the leaf SLDRs. The CTREE_REP message is transmitted to the neighbor SLDR which transmitted the CTREE message. The transmission of the CTREE and CTREE_REP messages depends on the tree size. Suppose that T trees are constructed in the network and a tree has an average degree T_d. The total message transmission on a tree is

$$\sum_{k=1}^{h} T_d^k = \frac{T_d^{h+1} - 1}{T_d - 1}$$

, where h is the height of the tree. The height h of the tree is $\lfloor log_{T_d}(\frac{N}{ST}) \rfloor$, in which S is the average size of a subset and $\frac{N}{S}$ subsets may be constructed in the network. Therefore, the message transmission of the tree construction on each tree takes $O(\frac{N}{ST})$ which results in total $O(\frac{N}{S})$ message transmissions in the network. The constant factor 2 (at most 2 nodes between two SLDRs) is canceled out in the big O notation.

The transmission of reports in a tree is the same as the reverse of the tree construction, with an additional cost for each root to transmit a final subset report to the base

station. The message transmission overhead for reports in the network is $O(T\sqrt{N} + \frac{N}{S})$. Therefore, the total message transmission overhead in *SET* takes $O(N)+O(N)+O(\frac{N}{S})+O(T\sqrt{N} + \frac{N}{S}) = O(N)$ in the entire network.

Table 6.2 shows the comparison of the message transmission overhead with existing broadcast and multicast schemes, individual node report, and subset report schemes. The analysis results demonstrate that *SET* is the most efficient in terms of communication.

Table 6.2. Communication Cost Comparisons

Schemes	Communication Cost
Broadcast	$O(N^2)$
Randomized Multicast [60]	$O(N^2)$
Line-Selected Multicast [60]	$O(N\sqrt{N})$
Individual Node Report	$O(N\sqrt{N})$
Individual Subset Report	$O(\frac{N}{S}\sqrt{N})$
SET	$O(N)$

Simulation: To verify the theoretical analysis, we implemented three schemes in a simulation environment: single node reporting, individual SLDR reporting, and *SET*. We conducted our simulations on four different size networks mentioned above. For each network, we generated 50 topologies to get the average message overhead of each scheme in the entire network. The base station is located at a random place in the network.

The shortest path is used between all nodes, including SLDRs, and the base station. Figure 6.6 shows our simulation results including the benchmarking cost $O(N\sqrt{N})$ which is the best performance of the existing solutions. As the network size increases, *SET* shows scalable performance; it is linearly proportional to N as the analysis shown in Table 6.2. *SET* is also advantageous in that it detects clones as the intersection and union of subsets are performed on a tree.

6.5.3 Memory Overhead

Available sensor products have limited memory. For example, Mica 2 motes, have 4 KB RAM [103]. In *SET*, each root maintains approximately $\frac{N}{T}$ identifiers, each of which is 12 bits. In the four networks used in our simulation, if we construct four trees in

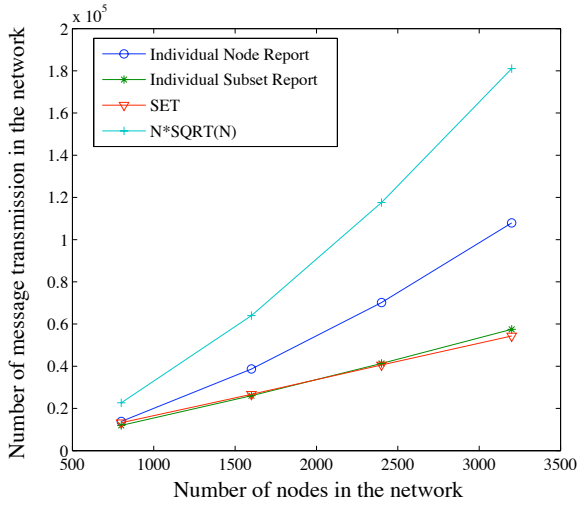


Figure 6.6. Message Transmission Overhead: Comparing *SET* with individual node and individual subset report.

the networks, the subset at a root occupies only 0.3 KB, 0.6 KB, 0.9 KB, and 1.2 KB, respectively. This can be further reduced by using the verifiable random selection.

Moreover, roots do not need to keep this subset continuously. Each time that the base station initiates detection, roots and SLDRs are reselected based on a new *seed*. Hence, during the network lifetime, sensors will share the overhead of being a SLDR and a root.

6.6 Discussion

In the basic scheme of *SET*, every node's identity is reported to the base station. An adversary may successfully avoid detection by hindering the exclusive subset and set operations on a tree such as by eliminating a node identity from the report. In Section 6.4, we analyze the probability that the adversary successfully avoids detection, which is very low. In addition to the report manipulation, a compromised intermediate SLDR on a tree may drop a subset report received from one of its children. Even if the compromised SLDR drops the received report to impede *SET*, the base station can detect clones in other rounds with high probability. Every time the base station generates a different random value of the *seed* so that the exclusive subset and multiple random tree structure

may change to mitigate the effectiveness of the adversary.

However, the basic scheme has a limitation that the report packet size may become large near the tree root. To overcome this limitation, we propose a verifiable random member selection scheme in which the base station releases a selection rule to the network such that SLDRs report only members that satisfy the selection criterion. The base station can also control the number of reported nodes by the selection rule. The base station, however, may have to run SET multiple rounds. Our analytical study shows that the base station can detect a clone in a single round with high probability.

There is a trade-off between the number of rounds and energy consumption. If the base station runs SET more frequently, it can detect clones quickly with very high probability, but will consume more energy. We leave the decision of how often the base station runs SET as a system management policy. For instance, the base station can use a combined strategy of periodic and on-demand execution to detect abnormal behavior.

In this thesis, we focus on detecting clones under the threat model in which the adversary may try to avoid detection. Unlike the efforts to avoid detection, the adversary may attempt to implicate a non-compromised node. A compromised SLDR may insert a non-existent neighboring node into the report, which makes the non-compromised node incorrectly appear as clone. We will address this issue in our future work.

Chapter 7

Conclusion

In this thesis, we present security and privacy solutions in different wireless and mobile environments. MML-IPsec provides end-to-end data protection while allowing a FA to execute performance enhancement functions. Our contribution is to propose efficient key distribution, rekeying, and mobility protocols. We implement these key distribution protocols on our test bed and measure the overhead of key distribution and mobility. From the measurement results, we claim that our key distribution protocols are feasible.

In a civilian application of MANETs, nodes are assumed to be selfish and rational, i.e., they pursue their own self-interest. Hence, the ability to accurately measure traffic forwarding is critical to ensure proper network operation. These measurements are also often used to credit nodes based on their level of participation, or to detect loss. In this thesis, we propose a protocol that uses nodes on the data path to securely produce packet forwarding reports. Reporting nodes are chosen randomly and secretly so that malicious nodes cannot modify their behavior based upon the monitoring point. The integrity and authenticity of reports are preserved through the use of secure link layer acknowledgments and monitoring reports. The robustness of the reporting mechanism is strengthened by forwarding the report to multiple destinations (source and destination). We explore the security, cost, and accuracy of our protocol.

In both military and civilian applications of mobile ad hoc networks, users may find any mandated exposure of information unacceptable. For instance, a user may not

want others to know with whom (s)he is talking or where (s)he is. To address these issues, we present a Privacy Preserving Communication System (PPCS) which provides a comprehensive solution to anonymize communication end-points, keep the location and identifier of a node unlinkable, and mask the existence of communication flows. To evaluate the effectiveness of PPCS, we define the optimal guessing strategy that may be used by one or more adversaries in cooperation and show that with PPCS, the probability of correctly guessing the source or destination of a flow is independent of the number of compromised nodes on the path.

Wireless sensor networks have attracted a great deal of attention because they may be used to support a variety of monitoring applications. However, sensor nodes that are deployed in hostile environments are vulnerable to capture and compromise. An adversary may obtain the private information from the sensors, clone and intelligently deploy them in the network to launch a variety of insider attacks. This attack process is termed as a clone attack. Currently, there are few defenses against clone attacks, and those that exist may suffer from selective interruption of detection and high overhead (computation and memory). In this thesis, we propose a new effective and efficient scheme, called *SET*, to detect such clone attacks. The key idea of SET is to detect clones by computing set operations (intersection and union) of exclusive subsets in the network. We show the reliability and resilience of SET by analyzing the probability that an adversary may effectively obstruct the set operations. Performance analysis and simulations also demonstrate that the proposed scheme is more efficient than existing schemes from both communication and memory cost standpoints.

Chapter 8

Future Work

8.1 Security in wireless sensor networks

Clone detection in mobile and wireless sensor networks: In this thesis, we proposed an efficient and effective clone detection system, called *SET*, in which we assumed that sensors do not have mobility. In other applications, however, sensors move to maximize sensing coverage or re-establish coverage due to the failure of sensors. An adversary then may use mobility to avoid detection. Mobility may also cause the exclusive subset and multiple random tree structure to change based on the new network topology.

As a future work, we will extend SET to address clone detection in mobile environments, while minimizing the impact of sensor mobility on the communication overhead of SET. We will also address an attack in which a compromised SLDR lies by inserting a node that is not in its neighbor list into the report, which will cause a non-compromised node to appear as a clone incorrectly.

Privacy in wireless sensor networks: In many applications of wireless sensor networks, contextual relationship, e.g., the location and sensor identity, should not be exposed to the adversary. In the panda hunting problem [104], the authors addressed source-location privacy by employing a random walk scheme to protect the panda (source) from hunters. This privacy service can not be provided by simply encrypting packets. Moreover, it is not necessary for the adversary to capture key materials. Instead, the

adversary may eavesdrop on communication and analyze the contextual relationship.

In data centric sensor networks (DCS), queries generated from sensors inside the network can be served by sensors which store sensed data, not by the base station. It is important that the location that the sensed data are stored in is protected. Min et al [105] addressed data-location privacy in DCS-based sensor networks.

In many cases, multiple sinks are used for collecting sensed data. If an adversary identifies the sink location, it can interrupt the data collection by the sink by simply jamming the area.

As these examples show, privacy in sensor networks is an important issue to be addressed.

8.2 Security in vehicular ad hoc networks (VANETs)

In vehicular ad hoc networks (VANETs), vehicles (on-board computers) communicate over multiple hops with each other and with roadside base stations connected to the Internet. A variety of useful applications may emerge such as safety, cooperative driving, traffic congestion reduction, and entertainment. Due to this market potential, both industry and academia have started paying attention to VANETs.

Unlike MANETs, vehicles have high mobility which causes communication to be short-lived, and the number of vehicles is very large. These characteristics make it challenging to provide security in VANETs. Raya and Hubaux [106] studied possible attacks in VANETs and proposed a general security architecture based on the public key infrastructure (PKI). In addition to the general security architecture, the authors also proposed a key management scheme with key revocation. Security in VANETs is yet in a fledgling state.

VANETs can be integrated with a wireless infrastructure such as a wireless mesh network which may consist of public mesh routers and private wireless routers. Vehicles may use this infrastructure to be connected to the Internet. From a service perspective,

this may boost the penetration of VANETs. This, however, raises many questions: 1) how roadside base stations authenticate vehicles; 2) as a vehicle which is already authenticated moves, should other roadside base stations trust the previous authentication (efficient), or should every base station authenticate vehicles; and 3) what access control will be used to prevent unauthorized vehicles from using the infrastructure, etc.

In safety applications of VANETs, vehicles collect information of road conditions, traffic congestion, or accident status, and transmit the information to other vehicles or roadside base stations. In this setting, an adversary, which could be an attacker or a selfish vehicle, may inject bogus information or manipulate the collected information. Hence, it is important to protect the information and operation in VANETs.

8.3 Location privacy in location-based services (LBS)

Location and positioning techniques enable users to determine their precise location. Based on the location information, users can receive a variety of handy application services, e.g., find a closest gas station, reserve a nearby restaurant, and find a friend. These services are generally referred to as location-based services (LBS). LBS require users to give their current location information to a service provider. However, this opens a back door for an unreliable service provider to release user location information without agreement. An attacker may use the location information to trace users and launch attacks.

Researchers have addressed location privacy issues in LBS [107, 108, 109]. An idea commonly used in these solutions is to introduce a location anonymizer between the users and LBS server. The location anonymizer takes the query request from users and makes it anonymize such that k-anonymity is satisfied. k-anonymity implies that a user can not be identifiable from k users. There are two ways to satisfy the k-anonymity: wait for $(k-1)$ other users where a querying user locates and extend the area to cover k users including the user. The LBS server may determine the answer for the anonymized query and reply back to the querying user.

This approach may cause the answer either to be imprecise due to the extended area to satisfy the k-anonymity, or to be delayed until k users are around in the specified area. Until this point, there is no breakthrough to overcome the weakness of k-anonymity based approaches. It is challenging and valuable to address this problem from different perspectives such as applying cryptosystems.

Bibliography

[1] Standard, I. (1999) "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *ANSI/IEEE Std 802.11*.

[2] Perkins, C. (1996) "IP Mobility Support," *IETF rfc2002*.

[3] McCann, P. J. and T. Hiller .

[4] Glass, S., T. Hiller, S. Jacobs, and C. Perkins (2000) "Mobile IP Authentication, Authorization, and Accounting Requirements," *IETF rfc2977*.

[5] W. A. Arbaugh, N. S. and Y. C. J. Wan (2002) "Your 802.11 Network has no clothes," *IEEE Wireless Communications*.

[6] Walker, J., "Intercepting Mobile Communications: The Insecurity of 802.11," http://www.issac.sc.berkley.edu/issac/wep-faq.html.

[7] Standard, I. (2004) "Part11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Amendment6: Medium Access Control (MAC) Security Enhancements," *ANSI/IEEE Std 802.11i*.

[8] Falk, M. (2004) "Fast and Secure Roaming in WLAN," *MS Thesis in the Department of Computer and Information Science at Linkoping University*.

[9] He, C. and J. C. Mitchell (2004) "Analysis of the 802.11i 4-Way Handshake," *in Proceedings of the ACM Workshop on Wireless Security (WiSe04)*.

[10] Kent, S. and R. Atkinson (1998) "Security Architecture for the Internet Protocol," *IETF rfc2401*.

[11] Chan, M. and R. Ramjee (2002) "TCP/IP Performance over 3G Wireless Links with Rate and Delay Variation," *Proc. of ACM Mobicom*.

[12] Balakrishnan, S. Seshan, E. Amir and R.H. Katz, H. (1995) "Improving TCP/IP Performance over Wireless Networks," *Proc. of ACM Mobicom*.

[13] Bakre, A. and B. R. Badrinath (1995) "I-TCP: Indirect TCP for Mobile Hosts," *15th International Conference on Distributed Computing Systems*.

[14] Zhang, Y. and B. Singh (2000) "A Multi-Layer IPSec Protocol," *Proc. of 9th USENIX Security Symposium.*

[15] Harkins, D. and D. Carrel (1998) "The Internet Key Exchange Protocol," *IETF RFC 2409.*

[16] Kaufman, C. (2004) "Internet Key Exchange (IKEv2) Protocol," *IETF Internet Draft draft-ietf-ipsec-ikev2-17.txt.*

[17] Stallings, W., "Cryptography and Network Security: Principles and Practice," PRENTICE HALL.

[18] Chan, H., A. Perrig, and D. Song (2003) "Random Key Predistribution Schemes for Sensor Networks," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P).*

[19] Du, W., J. Deng, Y. Han, S. Chen, and P. Varshney (2004) "A Key Management Scheme for Wireless Sensor Networks Using Deployme nt Knowledge," in *Proceedings of IEEE Infocom.*

[20] Eschenauer, L. and V. Gligor (2002) "A key management scheme for distributed sensor networks," in *Proceedings of ACM Conference on Computer and Communications Security (CCS).*

[21] Liu, D. and P. Neng (2003) "Establishing Pairwise Keys in Distributed Sensor Networks," in *Proceedings of ACM Conference on Computer and Communications Security (CCS).*

[22] Hu, Y.-C. and A. Perrig (2004) "A Survey of Secure Wireless Ad Hoc Routing," *IEEE Security and Privacy.*

[23] Hu, Y.-C., A. Perrig, and D. B. Johnson (2002) "Ariadne: A secure On-Demand Routing Protocol for Ad hoc Networks," *ACM/IEEE International Conference on Mobile Computing and Networking.*

[24] Przydatek, B., D. Song, and A. Perrig (2003) "SIA: Secure Information Aggregation in Sensor Networks," *In ACM SenSys (Conference on Embedded Networked Sensor Systems).*

[25] Zhone, S., J. Chen, and Y. R. Yang (2003) "Sprite: A Simpe, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks," *Proc. of IEEE INFOCOM.*

[26] Perrig, A., D. Song, and J. D. Tygar (2001) " Efficient and Secure Source Authentication for Multicast," *Symposium on Network and Distributed Systems Security (NDSS) 2001.*

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

[28] Kent, S. and R. Atkinson (1998) "IP Authentication Header," *IETF RFC 2402.*

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

[30] Dierks, T. and C. Allen (1999) "The TLS Protocol," *IETF RFC 2246*.

[31] Freier, P. Karlton and P.C. Kocher, A. (1996) "The SSL Protocol, Version 3.0," *Netscape Communications Corp.*

[32] Marti, S., T. Giuli, K. Lai, and M. Baker (2000) "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *Proc. of ACM Mobicom*.

[33] Zhang, Y. and W. Lee (2000) "Intrusion Detection in Wireless Ad Hoc Networks," *6th International Conference Mobile Computing and Networks*.

[34] Zhang, Y. and W. Lee (2000) "Intrusion Detection in Wireless Ad Hoc Networks," *Proc. of ACM Mobicom*.

[35] Huang, Y. and W. Lee (2003) "A Cooperative Intrusion Detection System for Ad Hoc Networks," *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks(SASN)*.

[36] Vigna, G., S. Gwalani, K. Srinivasan, E. Belding-Royer, and R. Kemmerer (2004) "An Intrusion Detection Tool for AODV-based Ad hoc Wireless Networks," *20th Annual Computer Security Applications Conference*.

[37] Perkins, C. E. and E. Belding-Royer (2003) "Ad hoc On-Demand Distance Vector (AODV) Routing," *IETF RFC3561*.

[38] Awerbuch, B., D. Holmer, C. Nita-Rotaru, and H. Rubens (2002) "An On-Demand Secure Routing Protocol Resilient to Byzantine Failures ," *ACM WiSe*.

[39] Michiardi, P. and R. Molva (2002) "CORE: A Collaborative Reputation Mechanism to enforce node cooperation in Mobile Ad Hoc Networks," *In Proceedings of The 6th IFIP*.

[40] Buttyan, L. and J.-P. Hubaux (2003) "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks," *Mobile Networks and Applications*.

[41] Buchegger, S. and J.-Y. L. Boudec (2002) "Performance Analysis of the CONFIDANT Protocol(Cooperation Of Nodes: Fairness in Dynamic Ad-hoc NeTworks)," *MOBIHOC*.

[42] Srinivasan, V., P. Nuggehalli, C. F. Chiasserini, and R. R. Rao (2003) "Cooperation in Wireless Ad Hoc Networks," *IEEE INFOCOM*.

[43] Anderegg, L. and S. Eidenbenz (2003) "Ad hoc-VCG: a Truthful and Cost-Efficient Routing Protocol for Mobile Ad Hoc NetworksWith Selfish Agents," *AC Mobicom*.

[44] Pfitzmann, A. and M. Hansen, "Anonymity, Unlinkability, Unobservability, Pseudonymity, and Identity Management -A Consolidated Proposal for Terminology Version v0.23," dud.inf.tu-dresden.de/literatur/.

[45] Chaum, D. L. (1981) "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Communications of the ACM.*

[46] Jerichow, A., J. Muller, A. Pfitzmann, B. Pfitzmann, and M. Waidner (1998) "Real-Time Mixes: A Bandwidth-Efficient Anonymity Protocol," *IEEE Journal on Selected Areas in Communications.*

[47] Reed, M. G., P. F. Syverson, and D. M. Goldschlag (1998) "Anonymous Connections and Onion Routing," *Journal on Selected Areas in Communication Special Issue on Copyright and Privacy Protection.*

[48] Dingledine, R., N. Mathewson, and P. Mathewson (2004) "Tor: The Second-Generation Onion Router," *Proceedings of the 13th USENIX Security Symposium.*

[49] Reiter, M. and A. Rubin (1998) "Crowds: Anonymity for Web Transactions," *ACM Transactions on Information and Systems Security.*

[50] Neil, B. and C. Shields (2002) "Hordes: A Protocol for Anonymous Communication Over the Internet," *ACM Journal of Computer Security.*

[51] Waters, B. R., E. W. Felten, and A. Sahai (2003) "Receiver anonymity via incomparable public keys," *ACM conference on Computer and communications security.*

[52] Jiang, S., N. Vaidya, and W. Zhao (2004) "A Mix Route Algorithm For Mix-net in Wireless Mobile Ad Hoc c Network," *IEEE International Conference on Mobile Ad Hoc and Sensor Systems.*

[53] Blaze, M., J. Ioannidis, and A. D. Keromytis (2003) "WAR: Wireless Anonymous Routing," *Security Protocols Workshop.*

[54] Kong, J. and X. Hong (2003) "ANODR:ANonymous On Demand Routing with Untraceable Routes for Mobile Ad-hoc Networks," *In ACM MOBIHOC.*

[55] Zhang, Y., W. Liu, and W. Lou (2005) "Anonymous Communications in Mobile Ad Hoc Networks," *IEEE INFOCOM.*

[56] Walters, J. P. and Z. Liang (2006) " Wireless Sensor Network Security: A Survey," *Security in Distributed, Grid, and Pervasive Computing.*

[57] Capkun, S. and J.-P. Hubaux (2005) "Secure Positioning of wireless devices with application to sensor networks," *IEEE Infocom.*

[58] Zhu, S., S. Setia, and S. Jajodia (2003) "LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," *ACM conference on Computer and communications security.*

[59] Newsome, J., E. Shi, D. Song, and A. Perrig (2004) "The Sybil Attack in Sensor Networks: Analysis & Defenses," *3rd International Symposium on Information Processing in Sensor Networks*.

[60] Parno, B., A. Perrig, and V. D. Gligor (2005) "Distributed Detection of Node Replication Attacks in Sensor Networks," *IEEE Symposium on Security and Privacy*.

[61] Perkins, C. (2002) "IP Mobility Support for IPv4," *IETF RFC 3220*.

[62] Salgarelli, M. Buddhikot, J. Garay, S. Patel, and S. Miller, L. (2003) "Efficient Authentication and Key Distribution in Wireless IP Networks," *IEEE Communications Magazine*.

[63] Montenegro, G. (1998) "Reverse Tunneling for Mobile IP," *IETF RFC2344*.

[64] Binkley, J. (2001) "An Integrated IPSEC and Mobile-IP for FreeBSD," *Technical Report 01-10*.

[65] Braun, T. and M. Danzeisen (2001) "Secure Mobile IP Communication," *Workshop on Wireless Local Networks at the 26th Annual IEEE Conference on Local Computer Networks*.

[66] Maughan, M. Schertler, M. Schneider, and J. Turner, D. (1998) "Internet Security Association and Key Management Protocol (ISAKMP)," *IETF RFC 2408*.

[67] Perkins, C. and D. Johnson (2000) "Route Optimization in Mobile IP," *Mobile IP Working Group Draft draft-ietf-mobileip-optim-09.txt*.

[68] Helsinki University of Technology (2001), "Dynamics Mobile IP Software," .
URL `http://www.cs.hut.fi/Research/Dynamics/`

[69] McDonald, C. Metz, and B. Phan, D. (1998) "PF_KEY Key Management API, Version 2," *IETF RFC 2367*.

[70] www.freeswan.org (2002), "Introduction to FreeS/WAN 1.99," .
URL `http://www.freeswan.org/`

[71] Choi, H., H. Song, G. Cao, and T. F. L. Porta (2005) "Mobile Multi-Layered IPSec," *IEEE INFOCOM*.

[72] http://www.isi.edu (2000), "The Network Simulator - ns-2," .
URL `http://www.isi.edu/nsnam/ns/`

[73] Zhu, S., S. Setia, S. Jajodia, and P. Ning (2004) "An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks," *In Proc. of IEEE Symposium on Security and Privacy*.

[74] Zhu, S., S. Xu, S. Setia, and S. Jajodia (2003) "LHAP: A Lightweight Hop-by-Hop Authentication Protocol For Ad-Hoc Networks," *In Proc. of the 23rd International Conference on Distributed Computing Systems Workshops.*

[75] Johnson, D. B., D. A. Maltz, Y.-C. Hu, and J. G. Jetcheva (2004) "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," *http://www.ietf.org/internet-drafts/draft-ietf-manet-drIETF draft.*

[76] Hu, Y.-C. and A. Perrig (2004) "A Survey of Secure Wireless Ad Hoc Routing," *IEEE Security and Privacy, special issue on Making Wireless Work.*

[77] Eschenauer, L. and V. Gligor (2002) "A key management scheme for distributed sensor networks," *In Proceedings of ACM Conference on Computer and Communications Security.*

[78] Chan, H. and a. S. A. Perrig (2003) "Random key predistribution schemes for sensor networks," *In Proceedings of the IEEE Symposiumon Security and Privacy.*

[79] Zhu, S., S. Xu, S. Setia, and S. Jajodia (2003) "Establishing Pair-wise Keys For Secure Communication in Ad Hoc Networks: A Probabilistic Approach," *IEEE International Conference on Network Protocols (ICNP'03).*

[80] Liu, D. and P. Neng (2003) "Establishing pairwise keys in distributed sensor networks," *In Proceedings of ACM Conference on Computer and Communications Security.*

[81] Jakobsson, M., J.-P. Hubaux, and L. Buttyan (2003) "A Micro-Payment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks," *In Proceedings of Financial Cryptography.*

[82] Micali, S. and R. Rivest (2002) "Micropayments Revisited," *CT-RSA.*

[83] Du, W., J. Deng, S. Han, and P. Varshney (2003) "A pairwise key predistributionscheme for wireless sensor networks," *In Proceedings of ACM Conference on Computer and Communications Security.*

[84] Back, A., U. Moller, and A. Stiglic (2001) "Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems," *Proceedings of Information Hiding Workshop (IH 2001).*

[85] Serjantov, A. and P. Sewell (2003) "Passive attack analysis for connection-based anonymity systems," *In European Symposium on Research in Computer Security.*

[86] Raymond, J.-F. (2000) "Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems," *Proceedings of International Workshop on Design Issues in Anonymity and Unobservability.*

[87] Levine, B. N., M. K. R. C. Wang, and M. Wright (2004) "On timing attacks in low-latency mix-based systems," *In Proceedings of the 8th International Conference on Financial Cryptography.*

[88] Krawczyk, H., M. Bellare, and R. Canetti (1997) "HMAC: Keyed-Hashing for Message Authentication," *IETF RFC 2104 (http://www.ietf.org/rfc/rfc2104.txt)*.

[89] Perkins, C. and E. Royer (1999) "Ad hoc On-Demand Distance Vector (AODV) Routing," *IETF RFC 3561 (http://www.ietf.org/rfc/rfc3561.txt)*.

[90] Lee, S.-J. and M. Gerla (2001) "Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks," *IEEE International Conference on Communications*.

[91] Marina, M. K. and S. R. Das (2001) "AOMDV: Ad hoc On-demand Multipath Distance Vector Routing Protocol," *IEEE ICNP*.

[92] Ye, Z., S. V. Krishnamurthy, and S. K. Tripathi (2003) "A Framework for Reliable Routing in Mobile Ad Hoc Networks," *IEEE INFOCOM*.

[93] Choi, H., W. Enck, P. Mcdaniel, and T. F. L. Porta (2005) "Secure Reporting of Traffic Forwarding Activity in Mobile Ad Hoc Networks," *Proceedings of The Second Annual International Conference on Mobile and Ubiquitous Systems*.

[94] Becher, A., Z. Benenson, and M. Dornseif (2006) "Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks," *International Conference on Security in Pervasive Computing*.

[95] Hu, Y.-C., A. Perrig, and D. Johnson (2003) "Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks," *IEEE Infocom*.

[96] Yang, Y., X. Wang, S. Zhu, and G. Cao (2003) "SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks," *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*.

[97] Przydatek, B., D. X. Song, and A. Perrig (2003) "SIA: secure information aggregation in sensor networks," *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys (SenSys)*.

[98] Blundo, C., A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung (1995) "Perfectly-Secure Key Distribution for Dynamic Conferences," *Lecture Note in Computer Science*.

[99] Perrig, A., R. Szewczyk, J. Tygar, and etal (2002) "SPINS: Security Protocols for Sensor Networks," *Wireless Networks*, **8**(5), pp. 521–534.

[100] Moscibroda, T. and R. Wattenhofer (2005) "Maximal Independent Sets in Radio Networks," *PODC'05*.

[101] Zhu, S., S. Setia, S. Jajodia, and P. Ning (2004) "An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks," *Proceedings of IEEE Symposium on Security and Privacy*.

[102] Bellare, M., R. Canetti, and H. Krawczyk (1996) "Keying Hash Functions for Message Authentication," *Lecture Notes in Computer Science Vol. 1109*.

[103] Crossbow (2006), "Wireless Sensor Networks (http://www.xbow.com/Products)," .
URL `http://www.xbow.com/Products/`

[104] Kamat, P., Y. Zhang, W. Trappe, and C. Ozturk (2005) "Enhancing Source-Location Privacy in Sensor Network Routing," *IEEE International Conference on Distributed Computing Systems, 2005*.

[105] Min, S., S. Zhu, W. Zhang, and G. Cao (2007) "pDCS: Security and Privacy Support for Data-Centric Sensor Networks," *IEEE INFOCOM 2007*.

[106] Raya, M. and J.-P. Hubaux (2007) "Securing Vehicular Ad Hoc Networks," *Journal of Computer Security, Special Issue on Security of Ad Hoc and Sensor Networks*.

[107] Gedik, B. and L. Liu (2005) "Location Privacy in Mobile Systems: A Personalized Anonymization Model," *IEEE International Conference on Distributed Computing Systems, 2005*.

[108] Gruteser, M. and D. Grunwald (2003) "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," *Proceedings of First ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys)*.

[109] Mokbel, M. F., C.-Y. Chow, and W. G. Aref (2006) "The New Casper: Query Processing for Location Services without Compromising Privacy," *In Proceedings of VLDB 2006*.

Vita

Heesook Choi

Heesook Choi received her B.S. degree in Computer Science and Statistics and M.S. degree in Computer Science from the Chungnam National University, Korea, in 1990 and 1992 respectively. She was a senior research staff in Electronics and Telecommunications Research Institute (ETRI) in Korea before she enrolled in the Ph.D. program at the Pennsylvania State University in August 2002. Her research interests lie in security and privacy in distributed systems and wireless mobile networks, focusing on designing algorithms and conducting systems research.