Key Management for Wireless Sensor Networks in Hostile Environments

Michael W. Chorzempa

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Masters of Science

in

Computer Engineering

Dr. Jung-Min Park, Chair

Dr. Y. Thomas Hou

Dr. Dong S. Ha

April 28, 2006

Blacksburg, Virginia

Keywords: Wireless Sensor Networks, Hostile Environments, Key Management,

Administrative Keys

Key Management for Wireless Sensor Networks in Hostile Environments

Michael W. Chorzempa

Abstract

Large-scale wireless sensor networks (WSNs) are highly vulnerable to attacks because they consist of numerous resource-constrained devices and communicate via wireless links. These vulnerabilities are exacerbated when WSNs have to operate unattended in a hostile environment, such as battlefields. In such an environment, an adversary poses a physical threat to all the sensor nodes. An adversary may capture any node, compromising critical security data including keys used for encryption and authentication. Consequently, it is necessary to provide security services to these networks to ensure their survival. We propose a novel, self-organizing key management scheme for large-scale and *long-lived* WSNs, called *Survivable and Efficient Clustered Keying (SECK)*. SECK provides administrative services that ensures the survivability of the network. SECK is suitable for managing keys in a hierarchical WSN consisting of low-end sensor nodes clustered around more capable gateway nodes. Using cluster-based administrative keys, SECK provides five efficient security administration mechanisms: 1) clustering and key setup, 2) node addition, 3) key renewal, 4) recovery from multiple node captures, and 5) re-clustering. All of these mechanisms have been shown to localize the impact of attacks and considerably improve the efficiency of maintaining fresh session keys. Using simulation and analysis, we show

that SECK is highly robust against node capture and key compromise while incurring low communication and storage overhead.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Providing security for Wireless Sensor Networks (WSN) presents unique challenges. In addition to the unknown topography present in ad hoc networks, sensors have very high computation, storage, and battery constraints. Mature solutions for key management are too complex for WSNs, as the resources required would quickly exhaust the small sensors. The suitability of these WSNs for military applications and the deployment of these networks in hostile environments have brought the challenge of securing the communication between these extremely resource constrained devices. In addition to battlefield deployment, there are a number of future applications that will require a high level of security. These include emergency response information, energy management, and medical monitoring [21].

1.1 Problem Description

Mature wired and wireless networks make use of public key cryptography. Public key cryptography consists of a secret key owned by one node and its corresponding public key known to everyone. This scheme relies on the fact that if a node uses the public key to encrypt a message, the message can be decrypted using only the corresponding private key. Also, the private key is impossible to derive using the public key. It is important to note that this goal is only accomplished through the use of large keys (relative to symmetric keys), and complex encryption and decryption arithmetic. Symmetric key cryptography uses the same key for encryption and decryption, which allows for smaller keys and less complex encryption arithmetic. However, because the same key is used for both encrypt and decrypt operations, the management of these keys are much more critical than public keys. Since WSNs do not have the storage or computational resources to utilize public key cryptography, symmetric key cryptography must be used.

In this thesis we identify the need for an administrative key management layer for existing session key management solutions for WSNs in order to extend the lifetime of these networks. Below in Table 1.1 we describe the typical key properties associated with session key management schemes. Popular key management schemes assume that global or static administrative keys are sufficient to re-configure the network or encrypt session keys.

Session key management schemes make the simplifying assumption that these administrative keys cannot be compromised either because they are rarely used or are stored in tamper-resistent hardware [6][8]. These assumptions limit the lifetime of the network to that of the time it takes for the static administrative keys to be compromised. We found the need to study a two-tiered key management scheme with dynamic session keys that support network re-configurations so that long-lived WSNs can be supported without such assumptions.

Table 1.1: Typical key properties.

Administrative Keys	Session Keys
Encrypt session keys and control traffic	Encrypt data traffic
Static	Dynamic
Assumed cannot be compromised	May be compromised

1.2 Motivations

Key management is crucial to the secure operation of Wireless Sensor Networks, WSNs. Existing key management schemes focus on the efficiency of bootstrapping session keys and, to a large extent, ignore issues regarding key reconfigurations (e.g., key updates).

Hence, those schemes are more appropriate for short-lifetime WSNs rather than long-lived WSNs. In many key management schemes [6][19][27][9][30] static administrative keys are adequate to manage administrative events such as membership management or re-clustering, but in long-lived networks, the survivability of these keys can not be assumed. We propose an efficient solution for administrative key management called *Survivable and Efficient Clustered Keying*, SECK, that directly addresses the problem of reconfiguration events and membership management. We also show how SECK can efficiently be utilized as an administrative layer for many of the existing session key management schemes proposed for WSNs.

In SECK, we make a distinction between administrative keys and session keys—the latter is sometimes called traffic encryption keys, TEKs. While session keys are provided to encrypt and/or authenticate normal data packets, administrative keys are used to encrypt session keys and for for security administration events. Previous approaches for WSN key management adequately address the session key management issue, but to a large extent, did not consider security administration events. We define a security administration event as any event that requires the network to reconfigure its node membership and/or keys due to an attack or a potential attack.

In a hostile environment, a long-lived WSN operates unattended and its nodes are highly prone to capture. Therefore, supporting reconfigurations such as node additions and revo-

cations, re-clustering, and administrative key updates are critical to maintain the WSN's security and survivability. SECK is a self-organizing scheme that sets up key associations in the network clusters, establishes administrative keys, provides efficient methods for distributing and maintaining session keys, efficiently adds and revokes nodes, provides efficient mechanisms to recover from multiple node captures, and enables location-based re-clustering of nodes. Using analytical and simulation results, we show that SECK is robust against the attacks that we identify in the threat model described in Chapter 3.2. Moreover, our results show that SECK incurs low communication and storage overhead on the sensor nodes.

1.3 Contributions

The contributions of this thesis lie in the establishment of a lightweight key administration layer. SECK provides such a layer through the use of administrative keys. Existing schemes assume static (and sometimes global [6]) administrative keys for security administration events. Since these keys are static, as they are used more often they become increasingly vulnerable to attacks that take advantage of keystream reuse [1]. In SECK, we maintain dynamic administrative keys that are utilized during security administration events. The two most notable features of SECK are: (1) It provides an efficient mechanism

to re-establish secure group communication after multiple node captures have been detected; and (2) It provides a re-clustering scheme that utilizes neighboring clusters to notify and absorb nodes when their cluster-head node has either been compromised or expired due to energy depletion. Existing group key maintenance schemes [10][16][17][20][44][45], can maintain secure communication when a number of nodes have been captured, but once a critical number of nodes have been captured, secure group communication cannot be re-established.

1.4 Organization

In Chapter 2, we describe related research. In Chapter 3, we describe the design framework consisting of the WSN architecture and threat model assumed throughout the thesis. We give full details of SECK in Chapter 4. In Chapter 5, we discuss SECK's robustness against node captures and present a thorough energy analysis of SECK. Finally, we summarize our findings in Chapter 6.

Chapter 2

Related Work

2.1 Static verses Dynamic Key Management

Key management schemes in sensor networks can be classified broadly into dynamic or static solutions based on whether re-keying (update) of administrative keys is enabled pre or post network deployment. Schemes can also be classified into homogeneous or heterogeneous schemes with regards to the role of network nodes in the key management process. All nodes in a homogeneous scheme perform the same functionality; on the other hand, nodes in a heterogeneous scheme are assigned different roles. Homogeneous schemes gen-

erally assume a flat network model, while heterogeneous schemes are intended for both flat as well as clustered networks. Other classification criteria include whether nodes are anonymous or have pre-deployment identifiers and if, when (pre-, post-deployment or both), and what deployment knowledge (location, degree of hostility, etc.) is imparted to the nodes.

While static schemes primarily assume that administrative keys will outlive the network and emphasize pair-wise communication keys, dynamic schemes advocate re-keying to achieve attack resiliency in long-lived networks and emphasizes group communication keys. Table 2.1 shows the primary differences between static and dynamic keying in performing key management functions. Moharram and Eltoweissy [18] provide a performance and security comparison between static and dynamic keying.

Table 2.1: Key management functions in static and dynamic keying.

(Admin. Keys assumed)	Static keying	Dynamic keying
Key assignment	Once at pre-deployment	Multiple times
Key generation	Once at pre-deployment	Multiple times
Key distribution	All keys are pre-distributed to nodes prior to deployment	Subsets of keys are re-distributed to some nodes as needed
Re-keying	Not applicable	Multiple times; requires a small number of messages
Handling node capture	Revealed keys are lost and may be used to attack other nodes	Revealed keys are altered to prevent further attacks

2.1.1 Pairwise Keys

Recently, numerous static key management schemes have been proposed for sensor networks. Many of them are based upon the seminal random key pre-distribution scheme introduced by Eschenauer and Gligor [8]. In this scheme, each sensor node is assigned k keys out of a large pool P of keys in the pre-deployment phase. Neighboring nodes may establish a secure link only if they share at least one key, which is provided with a certain probability based on the selection of k and P. A major advantage of this scheme is the exclusion of the base station in key management. However, successive node captures enable an attacker to reveal network keys and use them to attack other nodes. Chan et al. [25] extended this idea to provide localized attack resiliency. Liu and Ning [27] also extend this idea and pre-distribute t-degree polynomials which are used by neighboring nodes to generate a pairwise key. By using the polynomials, this scheme maintains security up to t colluding compromised nodes. Du et al. [26] pre-deploy pairwise keys based on known deployment points. By choosing keys shared with nodes likely to be in close proximity, the probability of key sharing can be increased. Most WSN applications require additional levels of key sharing among nodes beyond just pair-wise keys—specifically, group keys.

2.1.2 Group Keys

Park and Shin [6] propose a dynamic group key management protocol called LiSP. LiSP addresses the vulnerability of key stream ciphers caused by key reuse. It addresses the problem by frequently and synchronously updating the group key. LiSP utilizes broadcast transmission to distribute the group keys and uses one-way key chains to recover lost keys. While this scheme is very efficient, LiSP requires the use of static administration keys to perform periodic administrative functions. This leaves those keys vulnerable to disclosure.

Carman et al. [24] conducted a comprehensive analysis of various group key schemes. The authors conclude that the group size is the primarily factor that should be considered when choosing a scheme for generating and distributing group keys in a WSN. Wong et al. [44] propose a scalable group key management protocol using key graphs. They utilize keys of multiple granularity to reduce the re-keying overhead associated with membership management. They also investigate multiple approaches for constructing re-keying messages. While efficient, this approach requires a centralized key distribution center. Zhu et al. [19] propose a comprehensive dynamic key management scheme called LEAP that establishes multiple keys for supporting neighborhood as well as global information sharing. Although LEAP includes several promising ideas, it does not adequately address scalability issues concerning the distribution and re-keying of group keys.

2.2 Hierarchical Network Architectures

In this thesis we utilize both a physical and hierarchical network architecture. In this section we discuss the state of some important research being done relating to such a network architecture. We describe promising schemes that take advantage of the inherent benefit of such an architecture to show the motivation for utilizing a hierarchical network architecture design approach.

2.2.1 Key Management

To address the difficult problem of scalability, many have proposed hierarchical network architectures. In [9][10][17], authors utilize a clustered and hierarchical network architecture for key management. Jolly et al. [9] employ a hierarchical network organization to establish *gateway-to-sensor* keys. The clustering technique used by Jolly et al. was originally developed by Gupta et al. [28]. By using GPS signals, Gupta et al. propose to form a cluster in which all nodes are within one hop of the cluster head. Eltoweissy et al. [10][17] present another hierarchical key management scheme based on the Exclusion Basis System to efficiently maintain group and session key information. This approach supports key recovery only if node captures can be immediately detected.

2.2.2 Aggregation

The issue of coordination and aggregation has prompted the need for more capable aggregation techniques. Heinzelman et al. [12][34] propose LEACH, a cluster-based routing and aggregation scheme that periodically elects fresh aggregation nodes to combine sensed data from groups of sensors by eliminating redundancies. This approach was shown to increase system lifetime by an order of magnitude compared to general purpose approaches. This approach necessarily places a higher computational burden on the elected aggregation sensors. Madden et al. [14] present a scheme that makes use of overheard downstream messages to reduce redundancy in sensed data on the fly. They show that aggregation methods can be made efficient for low-end sensors to carry out. Estrin at el. [29] address the need for scalable coordination in large-scale sensor networks. This early paper addresses the need for application specific and localized algorithms. They propose using clusters to propagate relevant sensed data using an approach called directed diffusion.

2.2.3 Clustering

To address the problem of clustering sensors in a hierarchical network architecture, Wadaa et al. [15] require directional broadcasts of variable power to partition the neighbors of

a cluster head into wedges and distances in order to assign a coordinate identifier. This scheme supports nodes that contain no unique identifier prior to deployment, but does not guarantee a unique identifier for each node. Gupta and Younis [22] propose a fault-tolerant clustering scheme that clusters sensors around a more capable cluster head based on the perceived ranges from the sensors to the cluster head. Re-clustering is then performed through the use of maintained backup information. While simple, this scheme does not address authentication issues.

Chapter 3

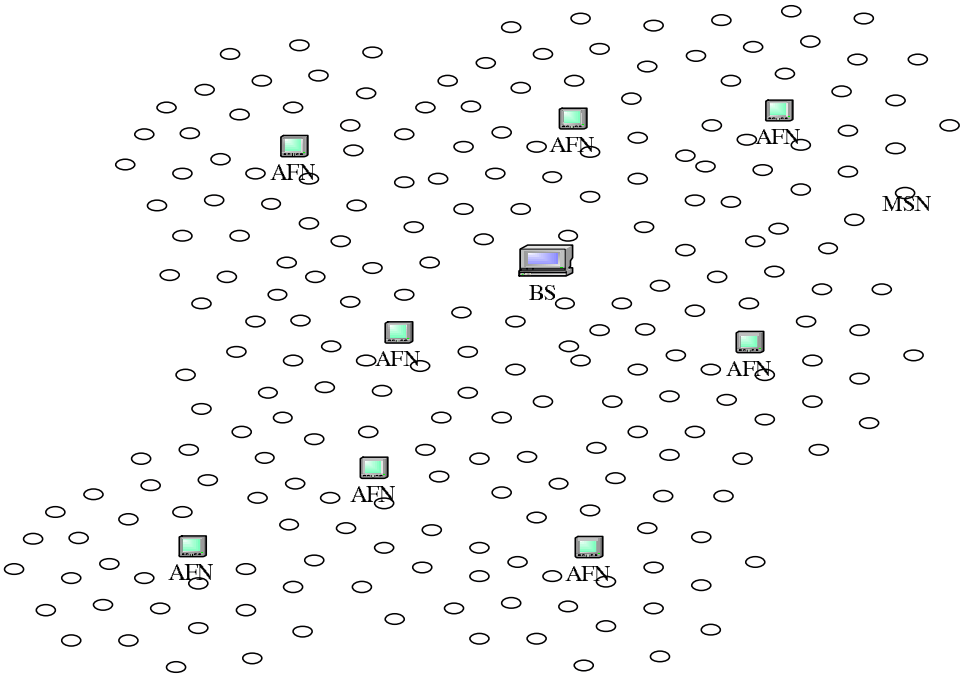
Design Framework

3.1 Network Model

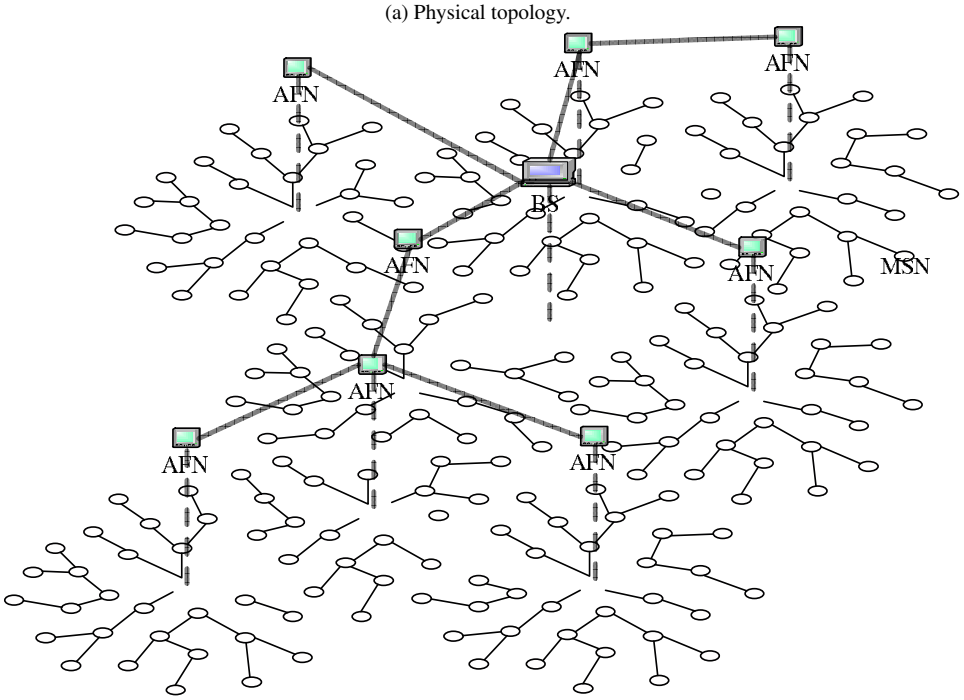
SECK is appropriate for a long-lived network with a physically hierarchical architecture deployed in a hostile environment. We assume that such a hierarchical network's bottom tier consists of numerous clusters of sensor nodes, each cluster consisting of many low-end nodes and a more capable cluster head node. The scheme presented in this thesis focuses on robust administrative key management within a cluster.

In a flat network, all nodes are identical and there is no predetermined architecture. Although simple and efficient for small network sizes, the flat network architecture lacks scalability. A multi-tiered architecture provides scalability, notable energy efficiency and security benefits [12][33][34][46]. Recent data aggregation techniques [14], which remove redundancy in collected data, lend themselves to this hierarchical architecture. Also, WSN routing research has shown that using a multi-tiered architecture for routing can prevent premature battery depletion among nodes near the base station, since, in a flat network, these nodes receive significantly higher traffic volume than remote nodes [35][43]. A multi-tiered architecture can also improve a network's robustness against node or key captures by limiting the effects of an attack to a certain portion of the network. For example, in a multi-tiered WSN, nodes are deployed in clusters, and each cluster can establish keys independently of other clusters. Thus, a key compromise in one cluster does not affect the rest of the network. In this chapter, we describe a two-tiered network architecture that is suitable for large scale WSNs.

Figures 3.1(a) and 3.1(b) show the *physical* and *hierarchical* network topologies for such a network, respectively. In this architecture, a small number of high-end nodes, called Aggregation and Forwarding Nodes, AFNs, are deployed together with numerous low-end sensor nodes, called Micro Sensor Nodes, MSNs. In addition, the network includes a globally trusted base station, BS, which is the ultimate destination for data streams from



(a) Physical topology.



(b) A hierarchical view.

Figure 3.1: A two-tiered wireless sensor network.

all the AFNs. The BS has powerful data processing capabilities, and is directly connected to an outside network. Each AFN is equipped with a high-end embedded processor, and is capable of communicating with other AFNs over long distances. The general functions of an AFN are: (1) data aggregation for information flows coming from the local cluster of MSNs, and (2) forwarding the aggregated data to the next hop AFN toward the BS. A MSN is a battery-powered sensor node equipped with a low-end processor and mechanisms for short-range radio communications at low data rates. The general function of the MSN is to collect raw data and forward the information to the local AFN. The bottom tier of the network consists of multiple clusters, where each cluster is composed of numerous MSNs clustered around an administrative AFN. Within each cluster, a set of keys needs to be deployed and managed to maintain secure communications between the MSNs and the AFN. SECK was designed for this purpose.

Since SECK is capable of managing events that require the network to reconfigure its membership and/or keys due to a network intrusion, an intrusion detection system, IDS, must also be part of the network architecture. SECK assumes the existence of an IDS [2][3][4] to monitor for anomalies in the network. Specifically, we assume the following: (1) Each AFN is equipped with a router-based IDS module that detects anomalous behavior among MSNs; and (2) AFNs and the BS are equipped with a cooperative IDS module, similar to the model proposed in [2], for detecting anomalous behavior among AFNs. These IDS

modules are organized into a hierarchy where the BS ultimately receives all pertinent detection data, and is responsible for initiating reconfigurations. We do not discuss the technical challenges in implementing such an IDS as it is out of the scope of this thesis.

3.2 Threat Model

We consider an attack scenario where an adversary is able to compromise one or more nodes of a WSN. Specifically, we consider three different cases with differing degrees of severity. These cases are illustrated in Figure 3.2. Throughout the remainder of the thesis, when we state that a node has been compromised or captured, we assume that all key information held by that node is known to the attacker. In the first scenario, illustrated by the black hat labelled one in the figure, an adversary may be able to target, and capture an AFN. A less severe attack would be when an attacker captures MSNs within the same cluster, this is illustrated by the black hats labelled two in the figure. In the third scenario, incurring the least degree of damage, an adversary would simply capture nodes at random throughout the network, illustrated by the black hats labelled three in the figure. In each of these attack scenarios we assume attackers are colluding [11] that is, any keys compromised are assumed to be shared among all attackers.

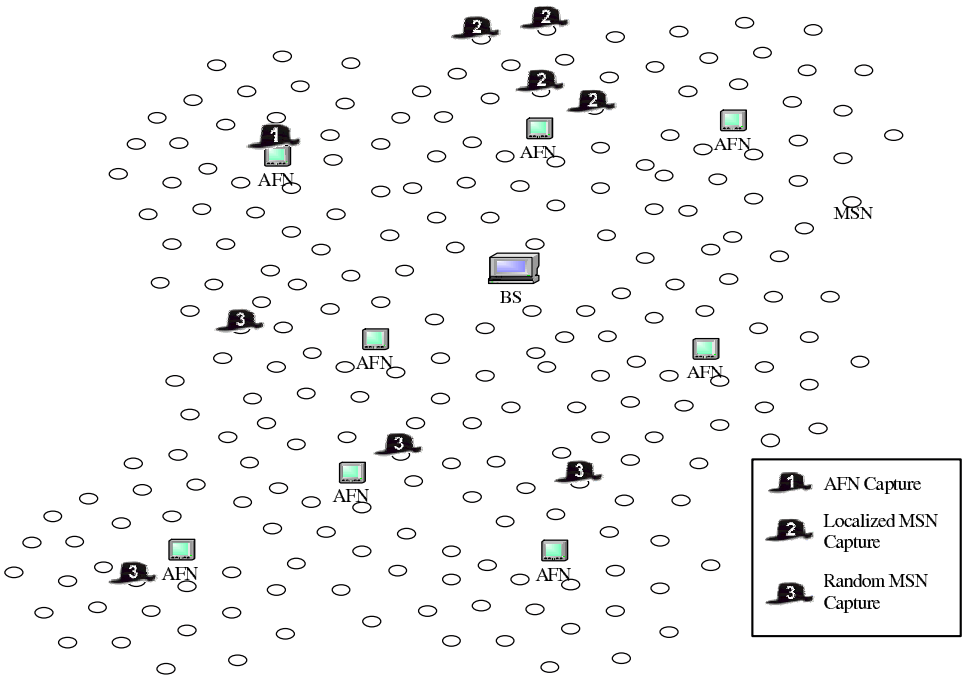


Figure 3.2: Illustration of threats.

One threat that has not been addressed in group key management schemes using administrative keys is the realistic possibility that multiple nodes may be captured before any node capture is detected. Researchers have pointed out that there really is no sure and efficient way to readily detect a node capture [9][19]. Therefore, for a key management solution to be truly effective in a hostile environment, it must recover when notified of multiple node captures.

The attack scenario that possesses the greatest threat to the bottom tier of the network is the compromise of an AFN. This requires an adversary to locate, and visually distinguish an AFN from a MSN. Then an adversary must extract the sensitive contents of the AFN

(e.g., keys). If an AFN capture is not immediately detected, all data collected by MSNs in that cluster will be compromised. After the detection of the AFN capture, the following steps need to be executed to restore normal operations of the cluster: (1) notify MSNs of the capture, (2) establish a new AFN for each MSN, and (3) establish a new security relationship between the MSN and the AFN in the second step. Note that in the second step, MSNs are "re-clustered" or absorbed into other existing clusters in their vicinity. If re-clustering is not supported by the network, all MSNs within the affected cluster are considered off-line until a new AFN is deployed. The AFN that is captured contains a full set of administrative keys that will need re-keying. In order to localize the necessary re-keying operations, it is necessary for administrative keys to be independently replaced. If the administrative key sets were globally calculated and distributed to all AFNs, all keys for all clusters would be compromised as a result of a single AFN capture.

The next threat is the compromise of MSNs within the same cluster. It is possible for an adversary to capture not only some but all the administrative keys of a cluster with only a few MSN captures. This is possible if the adversary is able to pick the nodes in a strategic manner; We will show in Chapter 4.3, the strategic choice would be to compromise N_1 and N_6, which would reveal all five administrative keys. Therefore, if capture detection is not possible, or not prompt, the entire cluster will likely be rendered insecure unless a method of recovery is developed. SECK provides a recovery method to salvage uncompromised

MSNs within a cluster when some or even all administrative keys are compromised.

In the third attack scenario, an attacker may compromise nodes randomly throughout the network. A clustered architecture's main advantage in terms of security is its ability to localize the effects of attacks on randomly chosen nodes. We have discussed some of these advantages in Chapter 2. A secondary advantage is that multiple decentralized attacks may not have increased effect compared to a single attack instance. If two adversaries located randomly throughout the network compromise a node each, combining the information obtained from these nodes provides no added benefit, assuming the nodes are not within the same cluster. Of course, this is true only if each AFN generates its administrative keys independently of others.

Chapter 4

Survivable and Efficient Clustered Keying (SECK)

We start our discussions by giving an overview of the components that comprise SECK. We follow this overview with a detailed description of each component of SECK. First we describe a location training scheme that sets up the network clusters, and the system used to establish administrative keys. We then describe the system used to manage administrative keys within a cluster. Then, we describe the full set of keys maintained by each MSN to support SECK. We next describe techniques for replacing administrative keys that have been compromised by multiple MSN captures. Next, we describe a re-clustering scheme

for salvaging MSNs within a cluster after the AFN of that cluster has been captured. Finally, we describe a method to dynamically add a MSN to the network.

4.1 Overview

SECK is fundamentally an administrative key management scheme for clustered WSNs. The goal of SECK is to set up an administrative infrastructure that allows a centralized AFN to maintain dynamic administrative keys and recover from security administration events. SECK accomplishes these goals through clearly defined administrative tasks. Administrative tasks that SECK supports are; network clustering, route setup, membership management, session key renewal, and re-clustering.

Location training is central to the setup of SECK. In addition to network clustering and route setup, critical information that is used for other mechanisms within SECK is gathered during location training. In this scheme, MSNs partition themselves into appropriate clusters by discovering a primary AFN. Next, MSNs not within direct communication range of its primary AFN locate the minimum energy cost route to this AFN. These two steps setup the communication infrastructure necessary for SECK. Also during location training, the BS obtains primary and backup membership information for each MSN in the network.

This information is needed for re-clustering MSNs in the event that an AFN is captured or depleted.

There are three types of keys used in SECK. Administrative keys, K_a are used for administrative purposes within the session key management scheme. These keys are set up using a technique called the exclusion basis system, EBS. Using an EBS, keys are distributed in such a way that both revoking a node and refreshing a group key are very efficient. Two other types of keys are used for administrative tasks within SECK. A BS pairwise key, Kp_i, is pre-loaded into each MSN and is needed for the re-clustering process after the capture of an AFN has been detected. A tree administrative key, Kt_i, is distributed to a small group of MSNs within a cluster and is used to replace compromised administrative keys after *multiple* MSN captures have been detected.

In the event that a threshold number of MSNs have been compromised, the complete set of session and administrative keys may be compromised. Within SECK, we are able to recover from such a crippling situation where all communication has been rendered insecure. The administrative key recovery scheme utilizes tree administrative keys to salvage remaining MSNs in a cluster. Since tree administrative keys are distributed to a small group of nodes in close proximity to each other, the impact of such a network attack is localized to only those trees containing a compromised MSN.

In a hierarchical network architecture the cluster head nodes, AFNs, carry out several key tasks that are essential to its cluster. Because the loss or capture of an AFN can incapacitate the entire cluster, giving the MSNs the ability to recover from such a situation greatly improves the survivability of the cluster by removing the single point of failure [21]. Session key management schemes point out the importance of such a mechanism, and SECK provides that functionally through a reactive re-clustering scheme.

Finally, it is important to provide support for the addition of new MSNs post-deployment. For security reasons, all of the administrative keys are maintained dynamically and it is impossible for a new MSN to be pre-deployed with any keys that will enable authentication with a MSN or AFN directly. However, each new MSN will contain a unique pairwise key, Kp_i, shared with the BS. SECK provides a mechanism that allows the BS to utilize Kp_i and act as a TTP between the new MSN and the existing AFN. This allows an MSN to authentically join an existing cluster.

Below, in Figure 4.1, we show the overall operation and components that complete SECK. Events are represented in italics and SECK components are represented in non-italics.

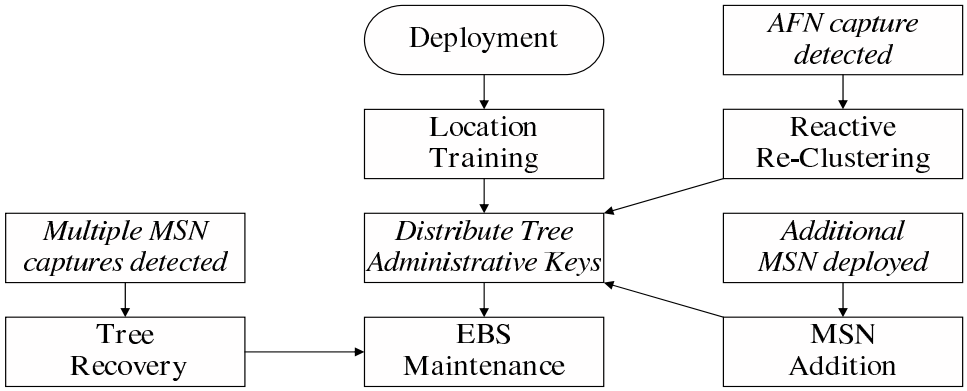


Figure 4.1: Components of SECK.

4.2 Location Training

Location training is critical to the setup of the network and the successful operation of SECK. For SECK to be successful, each MSN must be clearly identified within one primary cluster. Also, it is necessary to setup a routing infrastructure that allows MSNs to forward messages to its primary AFN. Once this cluster membership information has been recorded, each network node will have the information necessary to begin key distribution.

This location training scheme will discover a primary and backup cluster for every MSN, discover the routes to these cluster's AFNs, and notify the BS of MSN cluster membership. We will describe each of the three steps in the subsequent subsections. These steps are; 1) ranging, 2) route setup, and 3) cluster membership notification. The information obtained during each of these steps during location training is illustrated in Table 4.1.

Table 4.1: Information obtained during steps of location training.

Step	MSN	AFN	BS
Ranging	Distance Measurement to AFNs and MSNs in range.		
Route Setup	1)Primary and backup AFN, 2)Next hop MSN in direction of these AFNs.		
Cluster Membership Notification		Membership list of MSNs in its cluster.	Membership list identifying primary and backup AFN for every MSN in the network.

4.2.1 Ranging

Ranging technology is used by MSNs to identify primary and backup AFNs, and for distance vector route setup. Distance estimations are computed after a single broadcast message. This can be accomplished through either Received Signal Strength, RSS, Ultra Wideband, UWB, ranging techniques, or time of flight, ToF, measurements for tightly time synchronized nodes [36][37]. We assume our network nodes are equipped with the resources necessary to establish such distance estimates.

Each AFN broadcasts a single ranging message to every MSN within transmission range. Each MSN is then able to obtain a unique distance measurement for every AFN in range based on the ranging messages sent from these AFNs. This information will be used by MSNs during route setup to select their primary and backup AFNs. Next, MSNs each

broadcast their own ranging message to all neighboring MSNs. Neighboring MSNs calculate their range to this MSN using the received ranging message. These distance measurements will be used as link-costs for the distance vector route setup.

4.2.2 Route Setup

To discover a primary and backup cluster, each MSN wants to discover the ID of the closest two AFNs. The closest of which will become that MSN's primary AFN, and the furthest of which becomes that MSN's backup AFN. Each MSN selects its primary AFN as the AFN with the smallest distance measurement and selects its backup AFN as the AFN with the next smallest distance measurement.

To enable multi-hop communication from MSNs to AFNs, SECK uses an efficient distance vector routing approach to setup static routes. Our approach will use the neighbor distance measurements to calculate energy-efficient route costs, similar to the approaches presented in [38] [39]. We provide a brief summary of the operation of this simple distance vector routing scheme and direct the reader to [40] for more details on distance vector routing. In our scheme each MSN initially generates an entry in its routing table for its discovered primary and backup AFN. Each entry in an MSN's routing table consists of the ID of the destination, the next hop node ID in the direction of the destination, and the distance to the

destination. An entry in MSN x's routing table for node y looks like this:

$$\{id_y, nexthop_y, D_x(y)\},$$

where $D_x(y)$ is the total route cost to node y using $nexthop_y$ as the next hop node in the direction of y. Initially $nexthop_y$ is unknown and $D_x(y) = \infty$. For the first communication round of the algorithm, each MSN broadcasts their distance vector $\mathbf{D}_x$ to each node in their neighborhood, where $\mathbf{D}_x = D_x(AFN_p)||D_x(AFN_b)$ where AFN_p and AFN_b are MSN x's primary and backup AFN respectively. When a MSN, x, receives a new distance vector from any of its neighbors, v, it uses the Bellman-Ford equation [40] to update its own distance vector. During each subsequent round of the algorithm, MSN x only broadcasts $\mathbf{D}_x$ when an update has been made. This approach is proven to converge to the actual least cost route in finite time [41].

4.2.3 Membership Notification

Once the MSN to AFN routes have been generated, each MSN must finally handshake with its primary AFN using a *Membership Notification Message* that will identify this MSNs primary and backup AFN. A *Membership Notification Message* contains the ID of the orig-

inating MSN, N_i, the ID of N_i's primary AFN, AFN_p, and the ID of N_i's backup AFN, AFN_b.

Upon receipt of the *Membership Notification Message*, N_i's primary AFN must record the ID of the MSN who last forwarded N_i's *Membership Notification Message* i.e., the last intermediate MSN on the route from N_i to AFN_p, and a 1-hop neighbor of AFN_p. The primary AFN records this ID so it can identify all MSNs in the same $tree$. We will discuss the importance of $tree$s in section 4.5.

Each MSN will forward all *Membership Notification Messages* to the BS. The BS generates membership lists for all MSNs by storing all the *Membership Notification Messages* received. The BS uses the membership lists to identify MSNs that need to be re-clustered in the event of AFN capture or failure. We describe this re-clustering process in section 4.6.

At this point, each MSN is established in a single AFN's cluster. Also, every AFN has a list containing the ID and $tree$ of each MSN in its cluster. Finally, the BS contains a list identifying the primary and backup AFN for every MSN in the network.

4.3 Exclusion Basis System Description

SECK assigns a set of administrative keys to each node in a network to assist the session key management scheme. To manage administrative keys within a cluster, SECK employs the *Exclusion Basis System*, EBS [16]. EBS essentially provides a mechanism for establishing the administrative keys held by each node. An EBS-based key management system is defined by EBS(n, k, m) where n is the number of nodes supported in the system, k is the number of keys within each key subset, and m is the number of keys from the global key set not held within a subset. We represent the administrative key set for all nodes in the system as U, where $|U| = k + m$ and the total number of keys is $k + m$. A given EBS supports ${}_{k+m}\mathrm{C}_k$ unique subsets of k key assignments. For full details of the EBS, see [10][16]. The notation N_i denotes the i-th node, and S_i denotes its assigned administrative key subset, with $|S_i| = k$. The notation T_i denotes the set of keys not held by N_i, where $T_i = U - S_i$. The following property is the main motivation for utilizing the EBS: a subset of the global key set is uniquely assigned to each node such that the remaining nodes each have at least one of the keys not assigned to that node, that is, for all $j \neq i$, $S_j \bigcap T_i \neq \emptyset$. We will show that this property makes node revocations and session key replacement very efficient.

The AFN serving as the key management entity for its cluster must store all $k+m$ keys, and each MSN must store k keys. Note that the key subset held by each MSN is unique. This

feature is utilized by the AFN to distribute session keys, that is, keys of this subset are used to encrypt the session keys before distribution. We illustrate an instance of EBS(10,3,2) in Table 4.2. The i-th $(1 \leq i \leq 10)$ node in the cluster is denoted as N_i, and the j-th $(1 \leq j \leq 5)$ administrative key is denoted as Ka_j. An entry marked with a "1" indicates that the node in the corresponding column possesses the administrative key of the corresponding row.

Table 4.2: Sample administrative key subsets using EBS(10,3,2).

	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}
Ka_1	1	1	1	1	1	1	0	0	0	0
Ka_2	1	1	1	0	0	0	1	1	1	0
Ka_3	1	0	0	1	1	0	1	1	0	1
Ka_4	0	1	0	1	0	1	1	0	1	1
Ka_5	0	0	1	0	1	1	0	1	1	1

The administrative keys effectively serve as key encryption keys. Popular session key management schemes utilize a static administrative key when the session key needs to be updated, revoked, or re-configured. Our administrative keys would replace those static keys to provide a higher level of security as the increased lifetime of the network requires fresh administrative keys. Another advantage of EBS is the separation of administrative keys from session keys. A node can possess any number of session keys that it may share with different sub-groups of nodes.

As stated earlier, to initially distribute a session key to a specific node, the AFN sends that key encrypted with the unique key combination that the node possesses. However, to

initially distribute a cluster-wide session key, to all members of the cluster, one message is broadcast by the AFN to all members of the cluster for each administrative key. This requires $k + m$ short broadcasts by the AFN. At any time, to update a session key Kg with Kg', and distribute to any number of members less than or equal to n, the basic scheme executes a similar procedure using a maximum of $k + m$ keys. A session key, Kg, would be updated using the following procedure:

$$AFN \Rightarrow N_1, \ldots, N_{10} : \qquad \mathbf{E}_{Ka_1}(\mathbf{E}_{Kg}(Kg' \parallel ID_{AFN})).$$

$$AFN \Rightarrow N_1, \ldots, N_{10} : \qquad \mathbf{E}_{Ka_2}(\mathbf{E}_{Kg}(Kg' \parallel ID_{AFN})).$$

$$AFN \Rightarrow N_1, \ldots, N_{10} : \qquad \mathbf{E}_{Ka_3}(\mathbf{E}_{Kg}(Kg' \parallel ID_{AFN})).$$

$$AFN \Rightarrow N_1, \ldots, N_{10} : \qquad \mathbf{E}_{Ka_4}(\mathbf{E}_{Kg}(Kg' \parallel ID_{AFN})).$$

$$AFN \Rightarrow N_1, \ldots, N_{10} : \qquad \mathbf{E}_{Ka_5}(\mathbf{E}_{Kg}(Kg' \parallel ID_{AFN})),$$

where $\Rightarrow$ represents broadcast transmission, $\mathbf{E}_K(M)$ represents encrypting message M with key K, and $\parallel$ represents bit-wise concatenation. These broadcast messages ensure that only previous holders of Kg will be able to successfully receive the new Kg'. If N_1 is captured, or if its keys are compromised, it is necessary to revoke all administrative and session keys held by N_1 and thus evict it from all future secure communications ensuring forward secrecy. To do this, the AFN needs to replace the administrative keys and

the session keys known to N_1. From our running example, evicting N_1 would require the following two transmissions ($m = 2$).

$$AFN \Rightarrow N_1, \ldots, N_{10} :$$

$$ID_{AFN} \parallel \mathbf{E}_{Ka_4}(\mathbf{E}_{Ka_1}(Ka'_1) \parallel \mathbf{E}_{Ka_2}(Ka'_2) \parallel \mathbf{E}_{Ka_3}(Ka'_3)),$$

$$AFN \Rightarrow N_1, \ldots, N_{10} :$$

$$ID_{AFN} \parallel \mathbf{E}_{Ka_5}(\mathbf{E}_{Ka_1}(Ka'_1) \parallel \mathbf{E}_{Ka_2}(Ka'_2) \parallel \mathbf{E}_{Ka_3}(Ka'_3)),$$

From Table 4.2, it can be seen that all remaining nodes will be able to decipher at least one of these messages.

If more than one node is captured within a cluster, two cases need to be considered: (1) non-colluding node captures (e.g., attacks carried out by different adversaries); and (2) colluding node captures. In the latter case, colluding attackers may compromise all administrative keys by capturing only a few nodes (e.g., by capturing nodes N_1 and N_6 shown in Table 4.2, all administrative keys are revealed). In the former case, a maximum of m^y broadcast messages will be needed to evict y nodes at once. In our running example, to evict non-colluding nodes N_1 and N_6, four messages are needed to distribute the five new keys. One message will be doubly encrypted with Ka_2 and Ka_4, the second message with Ka_2 and

Ka_5, the third message with Ka_3 and Ka_4, while the fourth message will be encrypted with Ka_3 and Ka_5.

4.4 Key Establishment

Once all *Membership Notification Messages* have been received by an AFN, that AFN generates a list of unique *key subset identifier*s for every MSN established in its cluster and broadcasts this list. Upon receipt of this message each MSN will contain a unique *key subset identifier*. This *key subset identifier* identifies that MSN's unique key subset of EBS administrative keys.

Each AFN is also responsible for generating and distributing the tree administrative keys for each $tree$ in its cluster. This requires n messages transmitted by the AFN, one for each of the unique *key subset identifiers* used in the cluster. The first step is to generate d tree administrative keys, where Kt_j is the tree administrative key for all nodes in the j-th $tree$. Recall that a $tree$ consists of all nodes that utilize the same forwarding route. From Figure 4.2, we see that N_1 and N_2 are members of $tree_0$. Each Kt_j is distributed to each N_i in

$tree_j$ as follows:

$$\text{For all } N_i \in tree_j, \text{AFN} \rightarrow N_i : \qquad \mathbf{E}_{Ka_1}(\mathbf{E}_{Ka_2}(\ldots \mathbf{E}_{Ka_{|S_i|}}(Kt_j)\ldots)),$$

where $\rightarrow$ represents unicast transmission, and $S_i = \{Ka_1, Ka_2, \ldots, Ka_{|S_i|}\}$. Recall that S_i is N_i's administrative key subset. The encryptions are nested to ensure only N_i successfully receives Kt_j on this transmission.

It is important for the administrative key set, U, to be independently updated within each cluster after all tree administrative keys have been established. If the same set of administrative keys are maintained globally, the compromise of a single cluster's keys would compromise the entire network's keys. We described in Chapter 3.2 that in order to isolate an attack, the administrative keys must be independently generated in each cluster and not shared among clusters.

4.5 Administrative Key Recovery

SECK provides a mechanism to recover compromised administrative keys from multiple MSN captures within an isolated set of $trees$. The MSNs in the remaining trees can be salvaged by reestablishing their administrative key subsets using their tree administrative

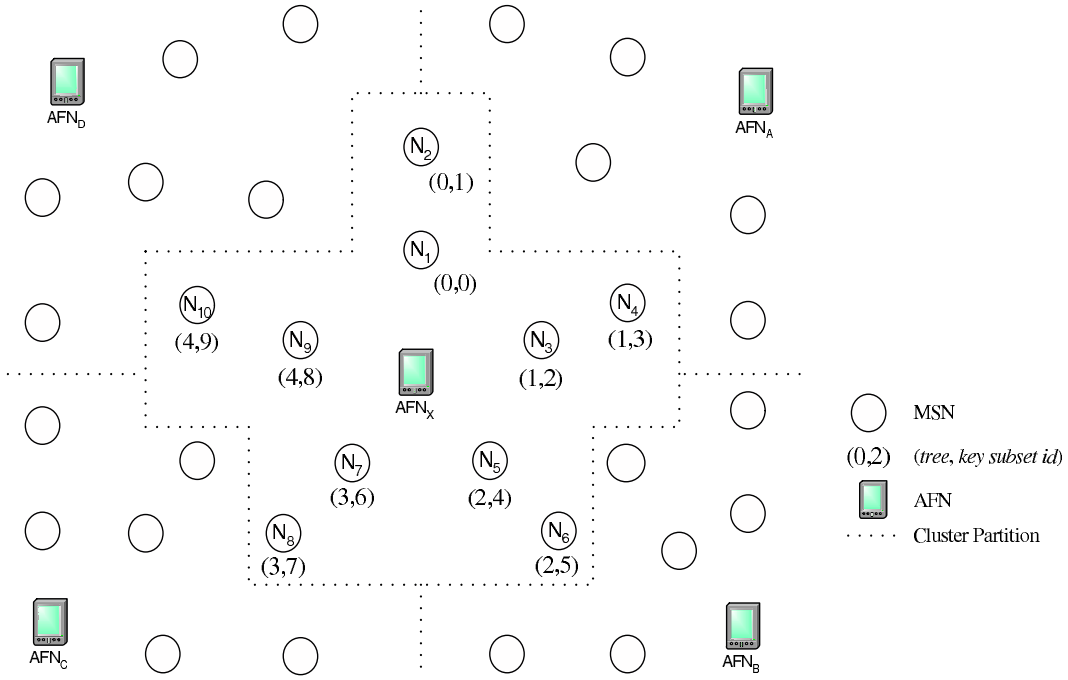


Figure 4.2: Cluster after location training and key distribution.

keys.

In Section 3.2, we stated that the capture of a group of MSNs can compromise most or all of the administrative keys using an EBS. When all of the administrative keys of the cluster have been compromised, even the MSNs that are not captured are excluded from any further communications with the rest of the network. Hence, it is necessary to distribute new session keys to those MSNs. We assume a scenario where an AFN is notified by the BS that a set of nodes, which we denote as S_c, has been compromised, resulting in the compromise of all administrative keys. In response, the AFN computes the set of trees not containing any node from S_c, which we denote as S_{ut}, as

$$S_{ut} = \{tree_i : tree_i \bigcap S_c = \emptyset, \forall tree_i\}.$$

The AFN then creates $|S_{ut}|$ messages, each containing a set of new administrative keys and transmits the messages to the appropriate $trees$ as shown in the following expression.

$$\text{For all } tree_i \in S_{ut}, \text{AFN} \Rightarrow tree_i :$$

$$\mathbf{E}_{Kt_i}(\mathbf{E}_{Ka_1}(Ka'_1) \parallel \mathbf{E}_{Ka_2}(Ka'_2) \parallel \ldots \parallel \mathbf{E}_{Ka_{k+m}}(Ka'_{k+m})).$$

Note that the technique described above cannot salvage MSNs that belong to a $tree$ in which some of its nodes have been compromised. In Chapter 5.1.3, we will show that the technique described above can salvage a greater percentage of MSNs when the attack is more localized (i.e., concentrated in a specific region of a cluster).

4.6 Reactive Re-clustering

Unlike a MSN, an AFN carries out several key tasks that are essential to its cluster. Because the loss or capture of an AFN can incapacitate the entire cluster, giving the MSNs the ability to recover from such a situation greatly improves the survivability of the cluster by removing the single point of failure [21]. To keep the MSNs operational after the

type of AFN capture described in Chapter 3.2, we need to re-cluster the MSNs into backup clusters and establish new security relationships between the re-clustered MSNs and their respective backup AFNs, AFN_b. We assume that the event of a successful AFN capture is infrequent enough as to warrant a reactive approach so as not to require MSNs to maintain security associations with backup clusters. This second approach is common of re-clustering approaches such as [22]. We must also assume that the backup AFNs have overlapping coverage, that is, are necessarily able to directly transmit to all MSNs joining its cluster. If not, re-clustering would not be feasible.

In SECK's approach, the BS acts as a trusted third party, TTP. Once notified of an AFN capture by the IDS, the BS consults its membership lists generated during location training to first identify the MSNs belonging to the captured AFN's, AFN_c, cluster, and second to identify the necessary backup AFNs. The BS must then notify these backup AFNs as to which MSNs will need to be absorbed into their cluster. These backup AFN's then absorb the MSNs by issuing them updated administrative keys using a shared secret generated by the TTP.

Using the membership lists (generated during location training) that identify the MSNs within AFN_c's cluster, the BS generates tickets for each of these MSNs using that MSN's

pair-wise key, Kp_i, as follows:

$$Ticket_{N_i} = \mathbf{E}_{K_{pi}}(K_{AFN_b-N_i}||ID_{AFN_b}||ID_{N_i}||nonce),$$

where N_i is an MSN from AFN_c's cluster. Note that ID_{AFN_b} is contained in the membership lists. The key, $K_{AFN_b-N_i}$, must be uniquely generated for each N_i, to be used by AFN_b for securely distributing updated keys to N_i. The $nonce$ is attached to prevent replay attacks.

Once all of the tickets for the nodes in AFN_c's cluster have been generated, they are distributed to the corresponding backup AFNs as follows:

$$\text{BS} \rightarrow AFN_b : \qquad \mathbf{E}_{K_{AFNb}}(K_{AFN_b-N_i}||ID_{N_i})||Ticket_{N_i},$$

where AFN_b is N_i's backup AFN, and K_{AFN_b} is a pair-wise key shared between AFN_b and the BS.

When AFN_b receives this message, it assigns an unused key subset identifier to N_i and generates an absorption message as follows;

$$Absorb_{N_i} = \mathbf{E}_{K_{AFNb-Ni}}(S_i||subset\_identifier_{N_i}),$$

where S_i is N_i's newly assigned set of administrative keys and $subset\_identifier_{N_i}$ identifies which keys from AFN_b's administrative key set are included in S_i. This message will securely establish N_i's membership within AFN_b's cluster.

N_i is first notified of AFN_c's capture when it overhears the following message from AFN_b.

$$AFN_b \rightarrow \text{neighbors} : \qquad ID_{N_i}||ID_{AFN_b}||Absorb_{N_i}||Ticket_{N_i}.$$

Upon hearing this message from AFN_b, N_i must first authenticate this message using the BS pair-wise key and $Ticket_{N_i}$. Next, N_i re-assigns its administrative key subset to the one specified in $Absorb_{N_i}$. Finally, N_i sends an acknowledgement message to AFN_b via the next hop node identified during location training, signalling the successful absorption into AFN_b's cluster. This message can be expressed using the following notation, assuming that the next hop node is N_x and $S_i = \{Ka_1, Ka_2, Ka_3\}$:

$$N_i \rightarrow N_x : \qquad ID_{N_i}||\mathbf{E}_{Ka_1}(ID_{N_i})||(\mathbf{E}_{Ka_2}(ID_{N_i})||(\mathbf{E}_{Ka_3}(ID_{N_i}).$$

Note that if an EBS reaches its maximum number of nodes, i.e., $n = {}_{k+m}\mathbf{C}_k$, adding a new node will require the expansion of the EBS by adding a new key. For brevity, we do not

describe the process of extending an EBS in this thesis; we simply assume that the size of a cluster's initial EBS will be sufficient for node additions. We refer the reader to [16] for an EBS expansion mechanism.

4.7 MSN addition

Throughout the lifetime of a WSN it may be necessary to deploy additional MSNs. We propose a way of adding nodes to an existing WSN. We assume that additional MSNs are randomly deployed, and these MSN's resulting clusters are unknown. In SECK, the keys maintained by an MSN are generated post-deployment. For this reason it is impossible for a new MSN to be pre-deployed with any keys that will enable authentication with a MSN or AFN directly. However, each new MSN will contain a unique pairwise key, Kp_i, shared with the BS. This key allows the BS to act as a TTP between the new MSN and the existing AFN.

To determine which AFN's cluster is most appropriate to join, a new MSN, N_{new}, conducts a ranging and route setup procedure nearly identical to those in location training. First, N_{new} broadcasts its own ranging message to each neighboring MSN. These neighboring MSNs calculate their range to N_{new} and reply to N_{new} with their range calculation and

distance vector as follows:

$$\text{For all neighbors } N_i,\ N_i \rightarrow N_{new}: \qquad id_{N_i} \parallel range_{N_i - N_{new}} \parallel \mathbf{D}_{N_i}.$$

where $range_{N_i - N_{new}}$ is the range calculated by N_i to N_{new}. N_{new} uses these received distance vectors and ranging information to execute the Bellman-Ford equation and determine the most appropriate AFN's cluster to join, AFN_x, and the next hop node ID in the direction of AFN_x (N_x). Once the next hop node to AFN_x is known, N_{new} constructs a short authentication request message destined for the BS. Node N_{new} sends the following message to N_x:

$$N_{new} \rightarrow N_x: \qquad (ID_{N_{new}} \parallel ID_{AFN_x} \parallel nonce) \parallel MAC_{Kp_i},$$

where MAC_{Kp_i} represents a message authentication code, MAC, of this message generated with N_{new}'s base station pairwise key, Kp_i. Node N_x then routes this message to AFN_x. After receiving this message, AFN_x attaches its own MAC using its pair-wise key shared with the BS, K_{AFN_x}.

$$AFN_x \rightarrow \text{BS}: \qquad (ID_{N_{new}} \parallel ID_{AFN_x} \parallel nonce) \parallel MAC_{Kp_i} \parallel MAC_{K_{AFN_x}}.$$

When the BS receives this message, it authenticates N_{new} and AFN_x using the MACs. If both nodes are authenticated, the BS generates the following ticket for N_{new}:

$$Ticket_{N_{new}} = \mathbf{E}_{Kp_i}(K_{AFN_x - N_{new}} || ID_{AFN_x} || ID_{N_{new}} || nonce).$$

Node N_{new} is then able to establish a security association with AFN_x in a manner that is identical to how a node is re-clustered as described in Section 4.6.

Chapter 5

Evaluation of Robustness and Efficiency

In this chapter we provide a thorough discussion of the robustness of the components that comprise SECK. By robustness, we mean the ability of the network to continue to operate after an attack has been detected. Also in this chapter we discuss the efficiency of each component of SECK. We calculate the pre and post-deployment storage overhead, as well as the communication energy consumption. Finally, we provide a comparative analysis of our scheme with two other well-known key management schemes to highlight the communication efficiency benefits of SECK.

5.1 Robustness

Node captures in hostile environments are inevitable. An effective key management scheme should be able to recover from such attacks to be effective. We describe some of the inherent security advantages of utilizing a clustered and hierarchical network architecture. Then, using the threats identified in Chapter 3.2, we analyze how well SECK recovers from those attacks.

5.1.1 Clustered Architecture

A clustered and hierarchical framework for WSNs provides many beneficial security properties. Isolation is the primary benefit of a clustered key management scheme. Each AFN is responsible for independently calculating and periodically distributing new administrative keys. Hence, an attack that yields keys within one cluster will not impact any other cluster in the network. An adversary must perform a global attack in order to completely compromise the network. This is not the case for most non-hierarchical key management schemes. In non-hierarchical key management solutions, a localized attack often has a global impact on the network's keys [8][23][25][27].

5.1.2 MSN Capture

In the example given by Table 4.2, it is possible for an adversary to capture all the administrative keys of a cluster with only a small number of node captures by strategically selecting the nodes to capture. For instance, the capture of N_1 and N_6 would reveal all five administrative keys. Fortunately, such attacks are difficult to carry out—to be successful, an adversary needs to know which administrative keys are stored in each MSN! Of course, the adversary can always randomly choose the nodes to capture and hope that a large proportion of administrative keys are revealed by those captures.

In Figure 5.1, the expected ratio of keys captured is plotted as a function of the number of nodes captured for several instances of SECK, with all instances supporting approximately 50 nodes. We are interested in instances of SECK that support about 50 nodes, as that is the expected size of a typical cluster. The figure shows that the ratio k/m has a direct impact on the robustness of SECK against node captures—decreasing the ratio improves robustness against node captures and increasing the ratio has the opposite effect. However, setting the ratio k/m to a low value incurs a cost—as the ratio decreases, the communication overhead required to distribute session keys increases. One can see that there exists a communication overhead versus node capture resiliency tradeoff. Further discussions on this issue are continued in the next section.

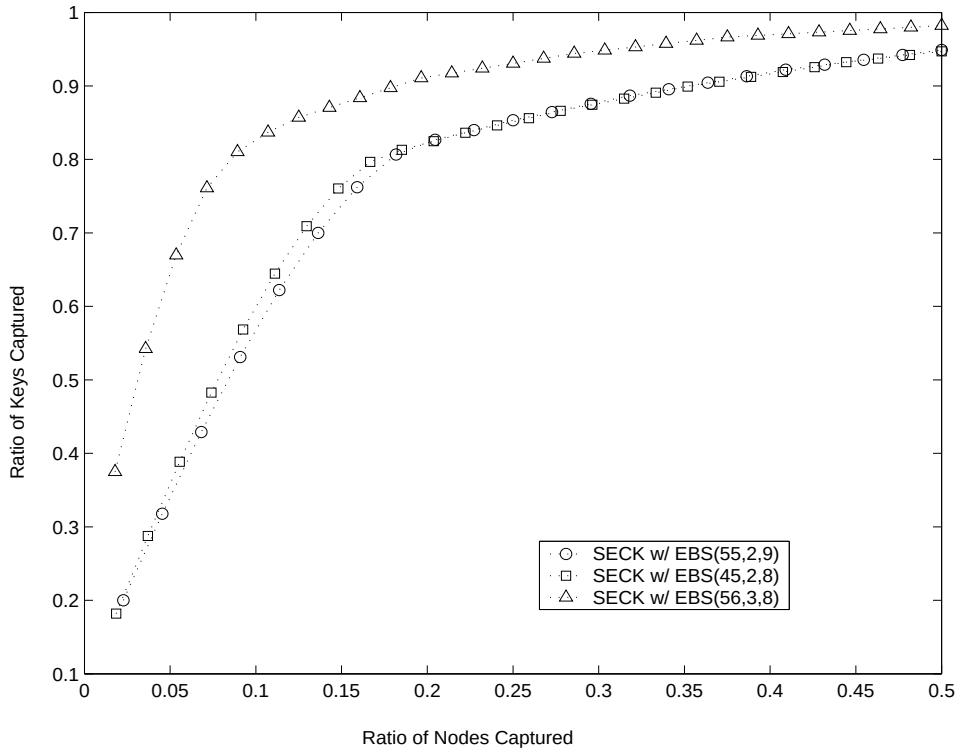


Figure 5.1: Expected ratio of keys captured vs. ratio of nodes captured.

5.1.3 Administrative Key Recovery

Recall that SECK generates d tree administrative keys for a cluster consisting of n MSNs in which every MSN is within h hops of its primary AFN. Figure 5.1 shows that SECK, using EBS(55,2,9), provides the most resiliency against node captures. If all the administrative keys of a cluster are compromised, the network needs to execute the MSN administrative key recovery procedure. In the following paragraphs, we discuss the effectiveness of the administrative key recovery procedure in two distinct cases—best and worst case

scenarios—assuming that x MSNs have been captured.

The worst case is when each of the x captures occurs in separate trees. This leaves $d - x$ trees unaffected, and $(d - x) \cdot h$ nodes can be recovered. If we approximate d with n/h, the ratio of nodes that can be recovered is $(n - h \cdot h)/(n - x)$. Suppose that every MSN in the cluster is within two hops of the AFN (i.e., $h = 2$) and EBS(55, 2, 9) is used. Then our procedure recovers 0.66 of the uncompromised nodes in the worst case.

The best case occurs when the attack is completely localized. That is, all nodes within a single tree are captured before the attacker moves on to the next tree. This will affect $\lceil x/h \rceil$ trees, leaving $d - \lceil x/h \rceil$ trees unaffected. If we again approximate d with n/h, the ratio of nodes that can be recovered is $(n - h \cdot \lceil h/x \rceil)/(n - x) \approx 1$. From the above analyses, we can observe that SECK's recovery procedure performs ideally in localized attacks.

In Figure 5.2, we have plotted the ratio of recoverable nodes in the best and worst case attack scenarios. We assume that EBS(55, 2, 9) is employed. In the figure, we have highlighted the case where fourteen nodes have been captured. In practice, the actual recovery ratio is expected to be somewhere between 0.6 and 1.0 when fourteen nodes are captured. As expected, as the ratio of nodes captured increases, the ratio of recoverable nodes drops.

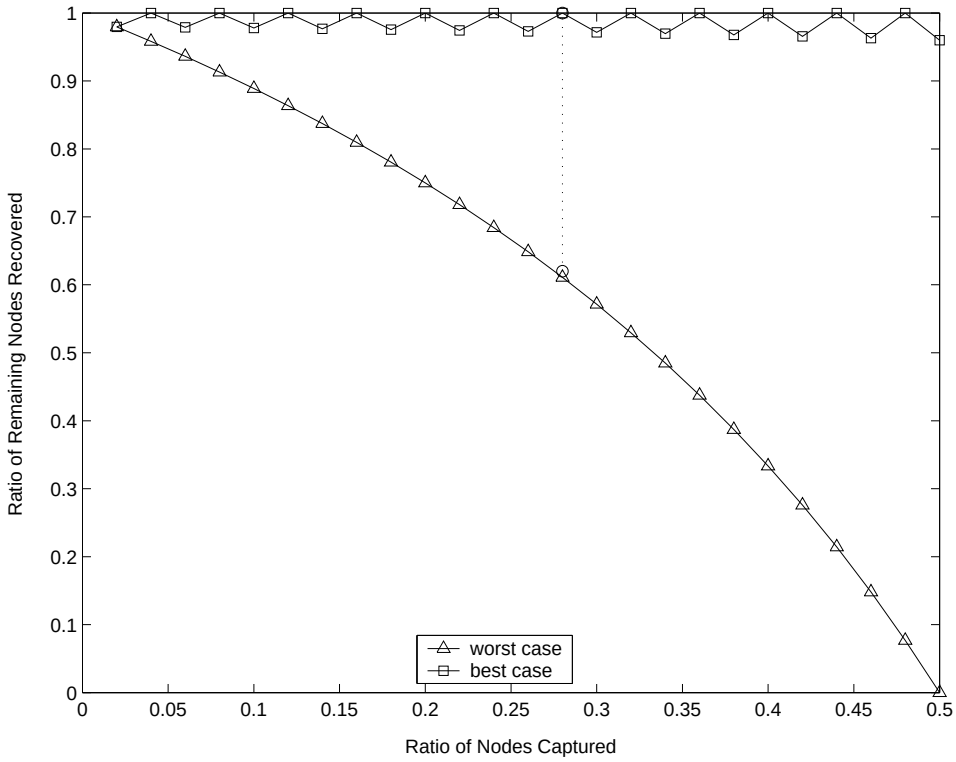


Figure 5.2: Administrative key recovery procedure evaluation.

5.2 Efficiency

SECK is designed to be a *lightweight* administrative key management solution. In order to motivate the need for such a robust administrative key management layer, we must demonstrate the storage and energy efficiency of our scheme. This section demonstrates the storage requirements and energy consumption requirements needed to support SECK.

5.2.1 Storage Overhead

Prior to deployment, each MSN needs to store a key subset matrix and the complete administrative key set. The key subset matrix is a $(k + m) \times n$ bit matrix that identifies the keys associated with each subset. Table 4.2 is a sample 5×10 key subset matrix. The key subset matrix requires $n \cdot (k + m)$ bits to store, and the complete administrative key set requires $128 \cdot (k + m)$ bits to store (here, we assume that 128 bit AES keys are used). Additionally, one 128-bit base station pair-wise key is stored at each MSN for a total initial storage requirement of $((128+n)\cdot(k+m)+128)$ bits. With this formula, one can calculate that each MSN would need to store 266 bytes of initial keying material if EBS(55,2,9) is employed.

After deployment, each MSN deletes most of its initial keying material immediately after the conclusion of the location training processes. Specifically, each MSN deletes its key subset matrix and the unused administrative keys. Recall that each MSN is assigned one 128-bit tree administrative key after deployment. Assuming that EBS(55,2,9) is employed, each MSN would need to store 64 bytes of keying material or four 128-bit keys, at the end of the location training process. Figure 5.3 illustrates the pre and post deployment storage overhead required by SECK.

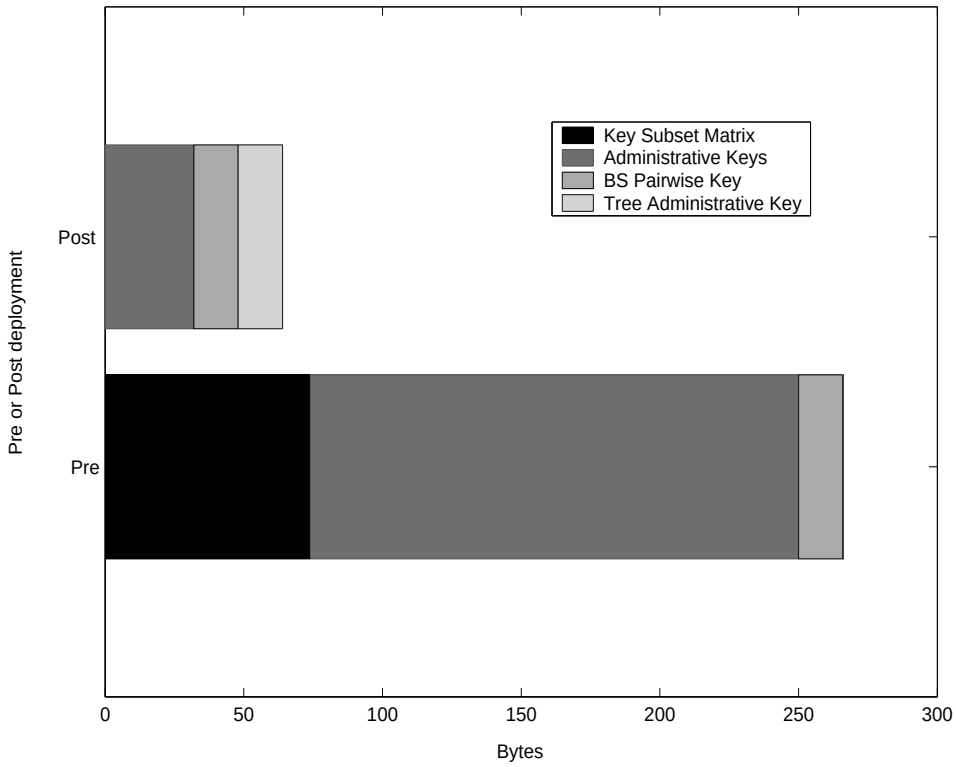


Figure 5.3: Storage overhead.

5.2.2 Communication Energy Overhead

In key management schemes, the energy required for computation is three orders of magnitude less than that required for communication [21]. Moreover, the amount of energy consumed for computation varies significantly with hardware. Hence, we only consider energy costs associated with radio signal transmission and reception, and do not consider energy costs associated with computation. To calculate the amount of energy dissipated during the execution of SECK, we use the power usage specification of Sensoria's WINS

NG: the RF radio component consumes $0.021\text{mJ}/\text{bit}$ for transmission, and $0.014\text{mJ}/\text{bit}$ for reception when operating at 10kbps [32]. We make the same assumptions as in [21] with regard to the following message element sizes:

- All node IDs are 32 bits.
- All nonces are 64 bits.
- All symmetric keys are 128 bits.
- The $tree$ identifier is 32 bits.
- The $hopcount$ is 32 bits.
- All MACs are 128 bits.
- The RSA modulus used for digital signatures is1024 bits.

Table 5.1 shows the amount of energy dissipated by a single node to complete one instance of each process (five processes are listed). We assume that EBS(55,2,9) is employed in a cluster consisting of 50 MSNs. Each AFN has a neighbor degree of 20, that is $|S_{h_1}| = 20$. In the following, we discuss the major factors that contribute to the energy consumption for each of these processes.

Table 5.1: SECK communication energy consumption.

		Transmission (mJ)	Reception (mJ)	Total (mJ)
Location Training	AFN	0.672	67.2	67.87
	MSN	809.1	539.4	1348
Distribute Tree Administrative Key	AFN	336.00	N/A	336.00
	MSN	N/A	4.48	4.48
EBS Maintenance	AFN	44.35	N/A	44.35
	MSN	N/A	29.56	29.56
Tree Recovery	AFN	29.57	N/A	29.57
	MSN	N/A	19.71	19.71
Reactive Re-clustering	AFN	117.6	140	257.6
	MSN	6.05	9.86	15.91
MSN Addition	AFN	20.16	11.2	31.36
	MSN	16.80	12.55	29.35

The energy required for location training is determined by the maximum number of control packets needed to complete the distance vector route setup. [41] shows that this algorithm converges in $O(n^3)$ packet per node where n is the number of nodes in a cluster, and assuming a path exists between the sources and destination, unicast transmission is used, and distance estimates are non-negative. Since we are operating in a wireless medium, broadcast transmission can be used to distribute a distance vector and the convergence reduces to $O(n^2)$. This upper bound can be further reduced by considering the worst case cluster diameter.

The current upper bound, $O(n^2)$, assumes a worst case cluster diameter of n. In a generic network, nodes may be connected linearly, thus having a worst case network diameter of n. However, clustered networks generally cover a circular area so the worst-case cluster

diameter can be improved by assuming nodes are distributed as to cover a circular area. We relate the problem of finding the largest cluster diameter with the known circle packing problem [42]. This circle packing problem specifies the most efficient arrangement of non-overlapping circles to occupy the smallest total space. Placing nodes exactly on the outer edge of their transmission range and arranging them in the manner identified in [42] and illustrated in Figure 5.4, we can obtain the maximally connected, minimally dense cluster that will provide us with a realistic upper bound of cluster radius.

The maximum number of communication hops from opposite edges of a cluster of n nodes arranged in the manner described above is D/d, where D is the diameter of the circle covered by the cluster and d is the transmission radius of a single node. For our purposes, the network diameter for an MSN is the number of MSN communication hops from its primary AFN to its backup AFN or simply D/d.

Figure 5.4 shows an illustration of the network described above. The center of each smaller circle represents the location of the nodes, and the circle's diameter represents the transmission radius of that node (radius of circle $= 1/2$ transmission radius). D can be expressed by summing the area of all the covered regions, $\mathbf{A}$, and uncovered regions, $\mathbf{B}$, then estimate this total area of the cluster as a circle with diameter D. We assume that the area of the covered regions outside the cluster circle would offset the omitted outer uncovered regions. The total area is then expressed as;

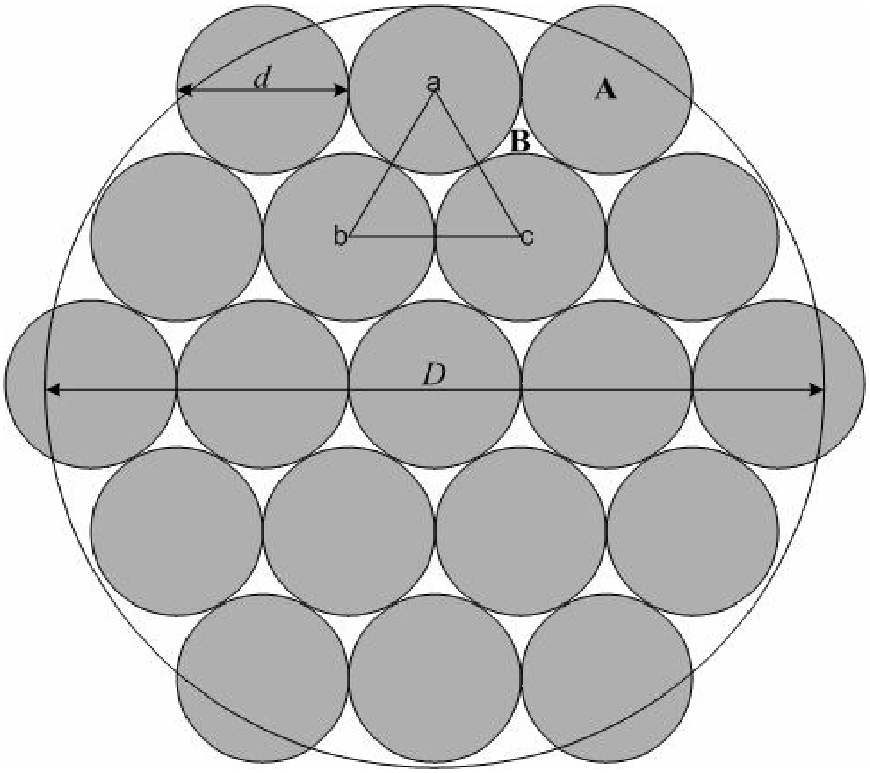


Figure 5.4: Circle packing representation of a cluster.

$$D^2(\pi/4) = n \cdot \mathbf{A} + (\text{number of uncovered regions}) \cdot \mathbf{B}. \tag{5.1}$$

The areas of $\mathbf{A}$ and $\mathbf{B}$ is then;

$$\mathbf{A} = d^2 \cdot \pi/4$$

$$\mathbf{B} = (\text{area of } \Delta abc) - (\text{area of } 60^\circ \text{sector}) \cdot 3$$

$$\mathbf{B} = (d^2 \cdot \sqrt{3}/4) - (d^2 \cdot \pi/24) \cdot 3.$$

Each internal covered region ($\mathbf{A}_{internal}$) is adjacent to six uncovered regions. Each external covered region, $\mathbf{A}_{external}$, is adjacent to three uncovered regions except for six who are adjacent to two uncovered regions.

$$\text{number of uncovered regions } = \frac{6(\mathbf{A}_{internal}) + 3(\mathbf{A}_{external} - 6) + 2(6)}{3}$$

$$\text{number of uncovered regions } = 2 \cdot \mathbf{A}_{internal} + \mathbf{A}_{external} - 2.$$

We define $RINGS$ as the number of broadcast hops away from the central MSN broadcast region an MSN is. The values for $\mathbf{A}_{internal}$, $\mathbf{A}_{external}$, and n are derived from this value;

$$n = 1 + \sum_{i=1}^{RINGS} 6 \cdot i$$

$$n = 3 \cdot RINGS^2 + 3 \cdot RINGS + 1$$

$$\mathbf{A}_{external} = 6 \cdot RINGS$$

$$\mathbf{A}_{internal} = 3 \cdot RINGS^2 - 3 \cdot RINGS + 1.$$

To find $RINGS$ in terms of n we use the quadratic equation as follows;

$$RINGS = \frac{-3 \pm \sqrt{9 - 12(1-n)}}{6}.$$

Finally, Plugging in all of the derived values into Equation 5.1 yields the following equation;

$$D^2(\pi/4) = n \cdot (d^2(\pi/4)) + (6 \cdot RINGS^2) \cdot (d^2(\sqrt{3}/4 - \pi/8))$$

$$D/d = \sqrt{n + RINGS^2(6\sqrt{3}/\pi - 3)}$$

$$D/d = \sqrt{\frac{(2\sqrt{3} - \pi) + 4\sqrt{3}n - (2\sqrt{3} - \pi)\sqrt{12n - 3}}{2\pi}}. \tag{5.2}$$

From our assumed cluster size $n = 50$, Equation 5.2 gives $D/d \approx 7.42$. We can now say that for MSN's uniformly deployed around a central location the cluster diameter is bounded by eight communication hops for a cluster size of less than 50. Using this in our computation of upper bound, we have a communication cost of $8n$ packets per node until convergence.

The remaining communication costs are strait forward to calculate. Each of these remaining components of SECK have a communication cost that is directly related to the number of nodes within a cluster. We next describe the dominating factors contributing to the communication cost of each remaining component within SECK.

To distribute the tree administrative keys to all MSNs in its cluster, an AFN must generate one message for each MSN in its cluster. Hence, the energy consumed by each AFN is directly proportional to the number of MSNs in its cluster. The communication overhead required for the EBS maintenance process is the message transmission needed for refreshing all administrative keys, and therefore this overhead is dependent on the dimension of the EBS that is used. The energy cost of the tree recovery process shown in Table 5.1 is the energy dissipated by the AFN for each $tree$ recovered. The energy cost of reactive re-clustering calculated in Table 5.1 is the energy needed to forward the recovery messages needed to recover one MSN. Therefore, the total energy dissipated during the cluster recovery process depends on the number of MSNs attempting to recover with a specific backup AFN. Finally, the energy cost of node addition calculated is the energy needed to respond to, then to forward the recovery messages of one newly added MSN. It can be seen from Table 5.1 that SECK effectively offloads much of the energy-intensive operations to the more capable AFN.

5.3 Comparison with other Schemes

The keying communication overhead of SECK is incurred during (1) the initial key establishment phase and (2) during periodic key maintenance procedures. We compare the communication overhead incurred in (1) and (2) with those incurred in Localized Encryption and Authentication Protocol, LEAP, [19] and Simple Key Distribution Center, SKDC, [21], respectively. A unique feature of SECK is its use of both administrative keys and session keys. Because of this feature, we cannot compare SECK, in its entirety, with a single scheme. Instead, we decompose SECK into (1) and (2), and compare the two constituent parts with LEAP and SKDC that respectively carry out similar functions. LEAP supports multiple levels of communication through multiple degrees of key sharing. SECK provides a similar level of flexibility within a clustered architecture. SKDC is a session key distribution scheme that utilizes a leader node to distribute the session key. It was shown in [24] that this basic method used by SKDC is the most efficient means to distribute a session key.

5.3.1 Comparison of Key Establishment Overhead

In this section, we compare the communication overhead incurred by SECK during the key establishment process with that of LEAP. LEAP is a well-known key management

scheme that provides communication flexibility by establishing multiple classes of keys. In SECK, a location-training process is executed to establish administrative key sets, then a distribution process is executed to distribute a session key. LEAP goes through similar key establishment processes to establish each node's pairwise and cluster keys.

LEAP restricts each node to three secure communication groups. SECK places no such restriction, and provides a simple mechanism to establish secure communication groups of any degree within a cluster. In LEAP, every MSN has a distinct pairwise key established with each neighbor. In SECK, however, every MSN establishes a pairwise key only with its primary AFN.

In LEAP, each node calculates a single pairwise key to share with each neighbor. This calculated pairwise key is unicasted to each of this node's d neighbors. In addition, establishing a LEAP cluster key requires $n(d-1)^2/(n-1) \approx (d-1)^2$ key transmissions throughout the network [19]. LEAP does not provide a concrete message format. To compute the energy dissipation incurred by LEAP, we assume that LEAP has the same message format as that used in SECK for key distribution. Here, we only consider communication-related energy dissipation. In Figure 5.5, we compare SECK and LEAP by plotting the energy dissipated by the network for key establishment as a function of the network size (for SECK, network size implies cluster size). Note that the size of a network (i.e., number of nodes) has the biggest impact on the energy required to establish keys. We set the

connection degree as 20, which was suggested by the authors of LEAP.

It is shown in Figure 5.5 that SECK is more efficient for small network sizes. Distributing similar amounts of keying material using the method described in SECK is better than that of LEAP when fewer nodes are considered. For the clustered network architecture described in Chapter 3, we assume that clusters consist of approximately 50 nodes—for such network sizes SECK outperforms LEAP.

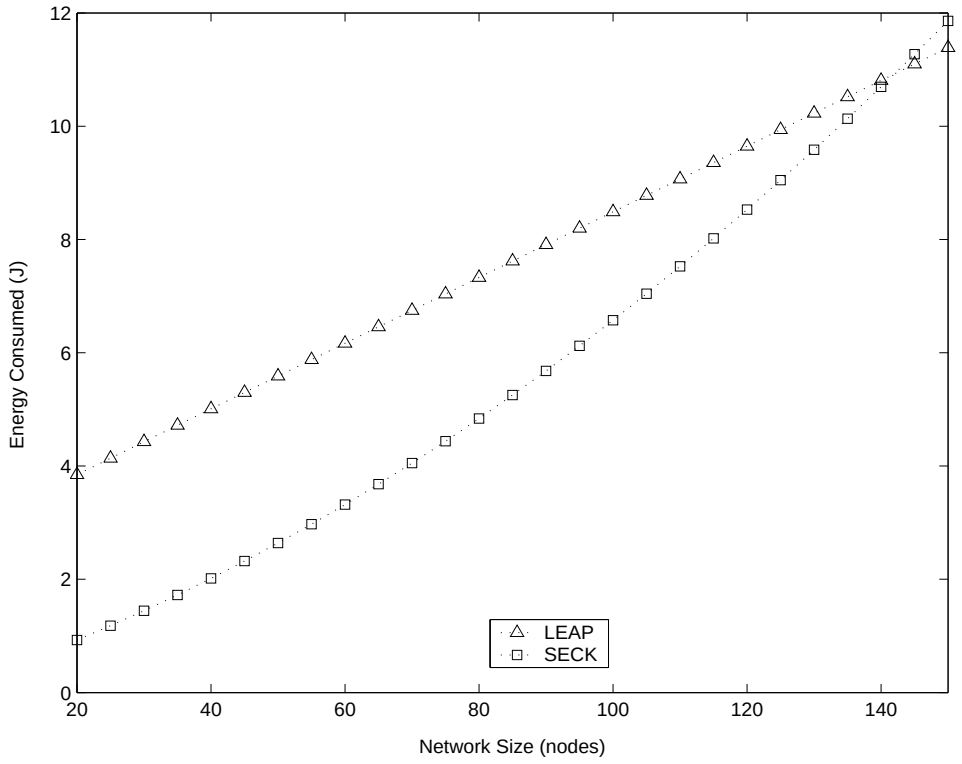


Figure 5.5: Key establishment communication overhead.

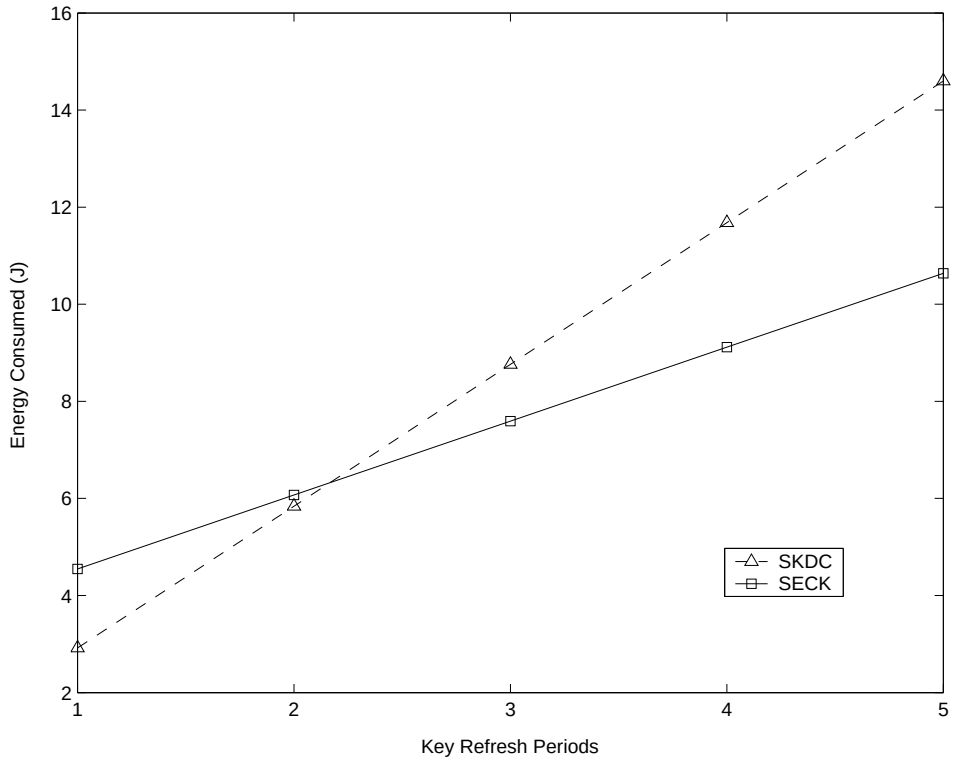


Figure 5.6: Key refresh communication overhead.

5.3.2 Comparison of Maintaining Session Keys

We now switch our attention to the communication overhead incurred by SECK to maintain session keys. Carman et al. [24] show that the straightforward technique of unicasting a session key to each group member incurs the least amount of communication overhead among session key distribution schemes. SKDC is a simple instance of such a method, and is most effective for a single instance of session key distribution. Our emphasis here is on the efficiency of session key distribution over multiple key update periods.

For a network deployed in a hostile environment, where node captures are expected, it is important to be able to continually distribute new session keys efficiently. SKDC creates an individual message for each group member every time that a session key must be updated, while SECK requires one message for each administrative key. In Figure 5.6, we compare SECK and SKDC in terms of the communication energy dissipated by the network due to session key redistributions. The plot shows that SKDC outperforms SECK during the initial session key update periods. This is because of SECK's initial overhead for establishing the administrative keys. However, SECK outperforms SKDC as the accumulated number of session key redistributions increases. This result was expected—the administrative keys employed in SECK reduce the cumulative number of messages that need to be transmitted by the AFN for session key redistributions.

Chapter 6

Conclusions

In a large-scale WSN with clustered formations of sensor nodes, a key management scheme is needed to manage the large number of keys in the system. In this thesis, we have describe a cluster-based dynamic key management scheme, SECK, that was designed to address this specific issue. SECK meets the stringent efficiency and security requirements of WSNs by using a set of administrative keys to manage other types of keys such as session keys. SECK is a comprehensive key management solution that includes: (1) a location training scheme that establishes clusters and the cluster coordinate system used in the MSN recovery procedure; (2) a scheme for establishing and updating administrative keys; (3) a scheme for distributing session keys using administrative keys; (4) a scheme for recovering

from multiple node captures; and (5) a scheme for re-clustering and salvaging MSNs in the event that their AFN has been captured. Through analytical and simulation results, we have shown that SECK is resilient to node and key captures while requiring a low level of communication overhead.

Bibliography

[1] N. Borisov, I. Goldberg, D. Wagner, "Intercepting mobile communications: the insecurity of 802.11," in *Proc. IEEE/ACM MobiCom.* pp. 180–189. Italy, July 2001.

[2] Y. Zhang and W. Lee, "Intrusion detection in wireless ad-hoc networks," in *Proc. of IEEE/ACM MobiCom,* pp. 275–283. Boston MA, August 2000.

[3] R. Zhang, D. Qian, C. Bao, W. Wu, X. Guo, "Multi-agent based intrusion detection architecture," in *Proc. International Conferece on Computer Networks and Mobile Computing,* pp. 494–501. Beijing China, October 2001.

[4] S. Ganeriwal and M. Srivastava, "Reputation-based framework for high integrity sensor networks," in *Proc 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks,* pp. 66–77. Washington DC, October 2004.

[5] D. Johnson, D. Maltz, and Y. Hu, "The dynamic source routing protocol for mobile ad hoc networks (DSR)," (Internet-Draft). Mobile Ad-hoc Network (MANET) Working

Group, IETF, July 2004.

[6] T. Park and K. Shin, "LiSP: A lightweight security protocol for wireless sensor networks," in *ACM Transactions on Embedded Computing Systems* Vol. 3, No. 3, pp. 634-660, August 2004.

[7] M. Bohge and W. Trappe "An authentication framework for hierarchical ad hoc MSN networks," in *Proc. of the ACM Workshop on Wireless Security (WiSe),* pp.79-87. San Diego CA, September 2003.

[8] L. Eschenauer and V.D. Gligor. "A key-management scheme for distributed MSN networks," in *Proc. Of the 9th ACM Conf. on Computer and Communications Security (CCS),* pp. 41-47, Washington DC, November 2002.

[9] G. Jolly, M. Kusu, and P. Kokate, "A hierarchical key management method for low-energy wireless MSN networks," in *Proc. of the 8th IEEE Symposium on Computers and Communication (ISCC),* pp. 335–340. Turkey, July, 2003.

[10] M. Eltoweissy, A. Wadaa, S. Olariu, and L. Wilson "Scalable cryptographic key management in wireless sensor networks," in *Proc. Distributed Computing Systems Workshops,* Tokyo Japan, March 2004.

[11] B. Awerbuch, R. Cortmola, D. Holmer, C. Nita-Rotaru, and H. Rubens, "Mitigating byzantine attacks in ad hoc wireless networks." Johns Hopkins University Technical Report Version 1, March 2004.

[12] W. Heinzelman, "Application-specific protocol architectures for wireless networks," *Ph.D. Dissertation,* Massachusetts Institute of Technology, June 2000.

[13] C. Intanagonwiwat, R. Govindan, and D. Estrin. "Directed diffusion: A scalable and robust communication paradigm for MSN networks," in *Proc. 6th Conference on Mobile Computing and Networking (MobiCom),* pp. 56–67. Boston MA, Aug. 2000.

[14] S. Madden, R. Szewczyk, M. Franklin, and D. Culler. "Supporting aggregate queries over ad-hoc wireless MSN networks," in *Proc. 4th IEEE Workshop on Mobile Computing Systems and Applications,* pp. 49–58. Callicoon NY, June 2002.

[15] A.Wadaa, S. Olariu, L. Wilson, M. Eltoweissy, and K. Jones, "Training a sensor network," *Mobile Networks and Applications*, vol. 10, no. 1, pp. 151–168. Feb. 2005.

[16] M. Eltoweissy, H. Heydari, L. Morales, and H. Sudborough, "Combinatorial optimizations of group key management," *Journal of Networks and Systems Management,* vol. 12, no. 1, pp. 30-50. March 2004.

[17] M. Eltoweissy, M. Younis, and K. Ghumman, "Lightweight key management for wireless MSN networks," in *Proc. IEEE International Conference on Performance, Computing, and Communications,* pp. 813-818. Phoenix AZ, April 2004.

[18] M. Moharram and M. Eltoweissy, "Key Management Schemes in Sensor Networks: Dynamic versus Static Keying," *ACM Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN 2005),* Montreal Canada, October 2005.

[19] S. Zhu, S. Setia, S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed MSN networks," in *Proc. of the 10th ACM Conference on Computer and Communication Security (CCS),* pp. 62-72, Washington DC, October 2003.

[20] R. Moharrum, M. Mukkamala, and M. Eltoweissy, "CKDS: an efficient combinatorial key distribution scheme for wireless," in *Proc. International Conference on Performance, Computing, and Communications,* pp. 630-636. Phoenix AZ, April 2004.

[21] D. Carman, P. Kruus, and B. Matt. "Constraints and approaches for distributed MSN network security," *Network Associates Inc. (NAI) Labs Technical Report No.00010.* September 2000.

[22] G. Gupta and M. Younis. "Fault-tolerant clustering of wireless MSN networks," *IEEE Wireless Communications and Networking,* vol. 3 no. 1 pp.1579-1584. March 2003

[23] B. Matt "A preliminary study of identity-based, group key establishment protocols for resource constrained battlefield networks," *Network Associates Labs Technical Report 02-034,* September 2002.

[24] D. Carman, B. Matt, and G. Cirincione. "Energy-efficient and low-latency key management for MSN networks." in *Proc. of 23rd Army Science Conference,* Orlando FL, December 2002.

[25] H. Chan, A. Perrig, and D. Song. "Random key predistribution schemes for MSN networks." in *Proc. IEEE Symposium on Security and Privacy,* pp. 197-213. Oakland CA, May 2003.

[26] W. Du, J. Deng, Y.S. Han, S. Chen, and P. K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," in *Proc. IEEE INFOCOM'04*, March 2004.

[27] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proc. ACM Conference on Computer and Communications Security (CCS),* pp. 52-61. Washington DC, October 2003.

[28] G. Gupta and M. Younis, "Performance evaluation of load-balanced clustering of wireless MSN networks," in *Proc. 10th International Conference on Telecommunications,* pp. 1577-1581. Papeete, French Polynesia, February 2003.

[29] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks," in *Proc. 5th International Conference on Mobile Computing and Networks (MobiCom),* pp. 263–270. Seattle WA, August 1999.

[30] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, "SPINS: Security protocols for sensor networks," *Wireless Networks,* vol. 8, no. 5, pp.521-534. November 2002.

[31] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM,* vol. 43, no 5, pp. 51-58. May 2000.

[32] Sensoria Corporation, "WINS NG Power Usage Specification: WINS NG 1.0," January 2000. Available: http://www.sensoria.com/

[33] J. Chou, D. Petrovis, and K. Ramchandran, "A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks," in *Proc. IEEE INFOCOM vol. 2,* pp. 1054–1062. San Francisco CA, April 2003.

[34] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks" in *Proc. 33rd Hawaii Int. Conference on System Sciences,* pp. 3005-3014. Maui HI, January 2000.

[35] C. Karl and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad Hoc Networks Journal,* vol. 1 no. 1, pp. 293-315. January 2003.

[36] S. Gezici, Z. Tian, G. Giannakis, H. Kobayashi, A. Molisch, H. Poor, Z. Sahinoglu, "Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks," *IEEE Signal Processing Magazine,* vol. 22, no. 4, pp. 70–84, July 2005.

[37] N. Patwari, A. Hero, M. Perkins, N. Correal, R. O'Dea, "Relative location estimation in wireless sensor networks," *IEEE Transactions on Signal Processing* vol. 51, no. 8, pp. 2137–2148, August 2003.

[38] I. Stojmenovic and X. Lin, "Power-aware localized routing in wireless networks," in *IEEE Transactions on Parallel and Distributed Systems,* vol. 12, no. 11, pp. 1122–1133. November 2001.

[39] J-H. Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," *IEEE Transactions on Networking,* vol. 12, no. 4, pp. 609–619. August 2004.

[40] D. Bertsekas and R. Gallager "Data networks," Chapter 5, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1987.

[41] M. Schwartz, "Telecommunication networks: protocols, modeling and analysis," Chapter 6, Menlo Park, California: Addison-Wesley Publishing Co., 1986.

[42] H. Steinhaus, "Mathematical snapshots," 3rd ed. New York: Dover, p. 202, 1999.

[43] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *IEEE Computer* vol. 35, no. 10, pp. 54-62, June 2002.

[44] C K Wong, M Gouda, and S Lam, "Secure group communications using key graphs," in *Proc. IEEE Transactions on Networking* vol 8, no 1, pp. 16-29. February 2000.

[45] H Harney and C Muckhenhirn, "Group key management protocol (GKMP) Specification," RFC 2093, July 1997.

[46] J. Pan, Y.T. Hou, L. Cai, Y. Shi, and S.X. Shen, "Topology control for wireless sensor networks," in *Proc. ACM International Conference on Mobile Computing and Networking (Mobicom),* pp. 286–299. San Diego CA, September 2003.

Appendix: Pseduocode

MSN Location Training Procedure

Notation
$AFN_i =$ AFN in range of MSN
$AFN_p =$ Primary AFN
$AFN_b =$ Backup AFN
$ranging\_message =$ Message sent to calculate pairwise distance
$membership\_notification\_message = (ID_{MSNi}||ID_{AFN_p}||ID_{AFN_b})$
$timeout =$ appropriate time to allow distance vector route setup to converge
$forward\_to_r(m) =$ forward message m toward node r

MSN Procedure
1. $foreach(AFN_i)\{$
2. $\quad recv(ranging\_message)$;
3. $\quad AFNdistances.insert(AFN_i) = calcDistance(ranging\_message)$;
4. $\}foreach(neighbor_i)\{$
5. $\quad recv(ranging\_message)$;
6. $\quad MSNdistances.insert(neighbor_i) = calcDistance(ranging\_message)$;
7.$\}$
8.$AFN_p = AFNdistances.min()$;
9.$AFNdistances.remove(AFN_p)$;
10.$AFN_b = AFNdistances.min()$;

11. //distance vector route setup
12. $\mathbf{D}_x = setupDistanceVector(AFN_p, AFN_b, MSNDistances)$;
13. $broadcast(\mathbf{D}_x)$;
14. $while(!timeout)\{$
15. $\quad recv(neighborDx)$;
16. $\quad BellmanFord(\mathbf{D}_x, neighborDx)$;
17. $\quad if(DxIsUpdated)\{$
18. $\quad\quad broadcast(\mathbf{D}_x)$;
19. $\quad \}$
20. $\}$
21. //membership management
22. $forward\_to_{AFNp}(membership\_notification\_message)$;

End

AFN Procedure

1. $send(ranging\_message)$;
2. $foreach(MSN_i)\{//MSN_i$ is in this AFN's cluster
3. $\quad recv(membership\_notification\_message)$;
4. $\quad assignTree(MSN_i)$;
5. $\quad send\_to_{BS}(membership\_notification\_message)$;
6. $\}$

End

BS Procedure

1. $foreach(MSN_i)\{$
2. $\quad recv(membership\_notification\_message)$;
3. $\quad insertInMembershipList(MSN_i, AFN_p, AFN_b)$;
4. $\}$

End

Reactive Re-clustering Procedure

Notation
AFN_c = captured AFN
$Ticket_{N_i} = E_{K_{pi}}(K_{AFN_b-N_i}||ID_{AFN_b}||ID_{N_i}||nonce)$
$Absorb_{N_i} = E_{K_{AFNb-Ni}}(S_i||subset\_identifier_{N_i})$
$ACK_{N_i} = ID_{N_i}||E_{Ka_1}(ID_{N_i})||(E_{Ka_2}(ID_{N_i})||(E_{Ka_3}(ID_{N_i})$ where $Ka_1, Ka_2, Ka_3 \in S_i$
$verify_k(m)$ = verify message m using key k
$send\_to_r(m)$ = send message m to node r

MSN Procedure
1. $recv(ID_{N_i}||ID_{AFN_b}||Absorb_{N_i}||Ticket_{N_i})$;
2. $if(ID_{N_i} == my\_id)\{$
3. $verify_{Kp_i}(Ticket_{N_i})$;
4. assign administrative key set to S_i;
5. $forward\_to_{AFN_b}(ACK_{N_i})$;
6. }
End

AFN Procedure
1. $forall(N_i)\{$ //Receive Tickets from BS
2. $recv(E_{K_{AFNb}}(K_{AFN_b-N_i}||ID_{N_i})||Ticket_{N_i})$;
3. generate $Absorb_{N_i}$;
4. $send\_to_{N_i}(ID_{N_i}||ID_{AFN_b}||Absorb_{N_i}||Ticket_{N_i})$;
5. }
6. $while(\exists N_i$ yet to respond with $ACK_{N_i})\{$
7. $recv(ACK_{N_i})$;
8. $forall(K_j \in S_i)\{$
9. $verify_{K_j}(ACK_{N_i})$;
10. }
11. }
End

Base Station Procedure
1. $forall(N_i \in AFN_c$'s cluster)$\{$
2. generate $Ticket_{N_i}$;
3. $send\_to_{AFN_b}(E_{K_{AFNb}}(K_{AFN_b-N_i}||ID_{N_i})||Ticket_{N_i})$;
4. }
End

Vita

Michael W. Chorzempa was born on July 30, 1982 in Charlotte, North Carolina. In 2000, he graduated from Brooke Point High School and in the fall of that same year he enrolled in Virginia Polytechnic Institute and State University. In the spring of 2004, he completed his undergraduate degree and achieved a Bachelor of Science Degree in Computer Engineering and graduated Cum Laude. As an undergraduate he was a member of Tau Beta Pi and Eta Kappa Nu (HKN) honor society.

In the fall of 2004, he began to pursue his Masters degree in Computer Engineering at Virginia Tech under the guidance of Dr. Jung-Min Park.