KEY MANAGEMENT IN WIRELESS SENSOR NETWORKS

By

Firdous Kausar

A thesis submitted to the Faculty of Information Security Department, Military College of Signals, National University of Sciences and Technology, Pakistan in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Information Security

Oct 2009

ABSTRACT

Wireless sensor networks(WSNs) consist of a large number of low power nodes, with limited processing, communication, and storage resources. Large scale wireless sensor networks (WSNs) are highly vulnerable to attacks because they consist of numerous resource constrained devices communicating via wireless links.The standard security requirements in WSNs include confidentiality, authentication and availability. These security requirements can be provided by encryption and authentication services which in turn demands a comprehensive key management scheme. The goal of key management is to pre-distribute cryptographic keys or keying materials among the nodes prior to the deployment, revoke keys if nodes leave the network, assign new keys to the nodes joining the network and periodically refreshing the keys. However, providing key management in WSNs is difficult due to the unknown network topology prior to deployment, intermittent connectivity and resource limitations of the sensor network environment.

Key management schemes consider hierarchical HSN consisting of a small number of high-end sensors ($\mathcal{H}$-node) and a large number of low-end sensors ($\mathcal{L}$-node). A key generation process is incorporated, where instead of generating a large pool of random keys, a key pool is represented by a small number of generation keys, in order to address storage overhead problem in the constraint sensor nodes. For a given generation key and a publicly known seed value, a keyed-hash function generates a key chain; these key chains collectively make a key pool. In the scheme proposed, after discovering the shared pairwise keys with neighbors, all $\mathcal{H}$-node and $\mathcal{L}$-node destroy their initial key rings and generate new key rings by applying one-way hash function on node's ID and initial key ring. As a consequence, new nodes can still be added in the network beyond the initial deployment even after the deletion of initial key rings from nodes memory. In addition, a self-healing group key distribution scheme is developed for secure multicast commu-

nications in HSN environment. This scheme presents a strategy for securely distributing rekeying messages and specifies techniques for joining and leaving a group. Access control in multicast system is usually achieved by encrypting the content using an encryption key, known as the group key (session key) that is only known by the group controller and all legitimate group members. In proposed scheme, all rekeying messages, except for unicast of an individual key, are transmitted without any encryption using one-way hash function and XOR operation. Further, nodes are capable of recovering lost session keys on their own, without requesting additional transmission from the group controller. Also the time-limited node revocation is achieved without any intervention from the GC.

This research reports the implementation and the performance of the proposed schemes on Crossbow's MicaZ motes running TinyOS and evaluates the computation and storage costs of two keyed-hash algorithms for key chain generation, HMAC-SHA1 and HMAC-MD5. The results show that proposed scheme can significantly reduce the storage requirements as compared to other random key pre-distribution schemes. The performance analysis of the collusion resistant mechanism shows that even if a large number of nodes are compromised, an adversary can only exploit a small number of keys nearby the compromised nodes, while other keys in the network remain safe. Also, the resiliency against node capture is better than previous key pre-distribution schemes. The security analysis of secure group key distribution scheme shows that the proposed scheme is computationally secure and meets the security requirements for forward and backward secrecy.

DEDICATION

Dedicated to My beloved family and to all those, whose prayers always paved the way to success for me

ACKNOWLEDGMENT

Praise to Almighty for bestowing upon me strength and knowledge, to conclude this aspiration in time and craft a substantial contribution.

I owe a special debt of gratitude to Dr. Ashraf Masood who offered his expertise, constructive critic, advice and guidance which made the present work a reality.

Dr. Sajid Hussain englightened me about the proper ways of writing papers, and more generally about the various skills required for research. If not because of his encouragement, I would never have taken the possibly once-in-a-lifetime opportunity to visit the Acadia University for a 5-month research collaboration. I owe him dearly.

I gratefully acknowledge my committee members Dr. Muhammad Akbar, Dr. Noman Jafri and Dr. Shamim Baig for generously sharing their ideas and enlightening and steering me to achieve this landmark.

I am gratified to my colleague Ms. Ayesha Naureen for her valuable encouragement and suggestions. Finally, I am most grateful to my parents and husband for their help and patience throughout the research.

TABLE OF CONTENTS

LIST OF FIGURES

LSIT OF TABLES

ACRONYMS AND ABBREVIAIONS

Notation	Definition
WSN	Wireless sensor network
HSN	Heterogeneous sensor network
BS	Base station
CH	Cluster head
$PRNG$	Pseudorandom number generator
PRF	Pseudorandom function
GC	Group controller
id_{L_i}	Identity of L-node i
id_{H_i}	Identity of H-node i
K	A pool of keys
M	Total number of key chains
C_i	Key chain i
gk_i	Generation key of i-th key chain
R_{L_i}	Set of the keys in L-node i initial key ring
R_{H_i}	Set of the keys in H-node i initial key ring
$R^{'}_{L_i}$	Set of the keys in $\mathcal{L}$-node i new/update key ring
$R^{'}_{H_i}$	Set of the keys in $\mathcal{H}$-node i new/update key ring
r	A number of keys in L-sensor key ring
S	A number of keys in H-sensor key ring
K_M	Master key
CK_i	Cluster key of i-th cluster

ACRONYMS AND ABBREVIAIONS

Notation	Definition
K_{M,L_i}	Authentication key of L-sensor i
$K_{X,Y}$	A shared key between X and Y
AK_X	Authentication key of node X
SK_i	Session key of $i-th$ session
KEK	Key encryption key
K^B	Backward key
K^F	Forward Key
S^F	Forward key seed
S^B	Backward key seed
B_i	Broadcast message of $i-th$ session
$\langle m \rangle_K$	An encryption of message m with key K
$nonce_{L_i}$	A random number string generated by L-sensor i
$MAC_K(m)$	A MAC of message m calculated using key K
$\mid$	concatenation symbol

Chapter 1

INTRODUCTION

1.1 Overview

A WSN typically consists of a potentially large number of extremely resource constrained sensor nodes. Each sensor node is usually battery powered, and has a low-end processor, a limited amount of memory, and a low power communication module capable of short-range wireless communication. Their lifetime is determined by their ability to conserve power. The sensor nodes form an ad-hoc network through the wireless links. There are many technological hurdles that must be overcome for ad hoc sensor networks to become practical though. All of these constraints require new hardware designs, software applications, and network architectures that maximize the motes capabilities while keeping them inexpensive to deploy and maintain. WSNs are ideal candidates for a wide range of applications, such as military, security, health-care [1, 2], industry automation, environmental and habitat monitoring [3, 4].

Secret communication is an important requirement in many sensor network applications [5], so shared secret keys are used between communicating nodes to encrypt data. Some of the major constraints like ad hoc nature, intermittent connectivity, and resource limitations of the sensor networks prevent traditional key management and distribution schemes to be applicable to WSN.

1.2 Background

Due to limited resources of WSNs, it is challenging to incorporate basic security functions, such as authentication, access control, data integrity, privacy, and key distribution. For instance, asymmetric cryptography such as RSA or Elliptic Curve cryptography (ECC) is unsuitable for most

sensor architectures due to high energy consumption and increased code storage requirements. To avoid the use of asymmetric cryptography, several alternative approaches have been developed to perform key management on resource-constrained sensor networks, such as random key pre-distribution schemes, plain text key exchange schemes, and transitory master key schemes.

In WSNs, hierarchical clustering provides scalability, self-organization, and energy efficient data dissemination [6]. A number of cluster formation protocols have been proposed but most existing protocols assume benign environments, and are vulnerable to attacks from malicious nodes.

Most existing research mainly considers homogeneous sensor networks, where all sensor nodes have identical capabilities in terms of communication, computation, sensing, and reliability; however, homogeneous WSNs are not scalable. Several recent works, on the other hand, investigate HSNs.

1.2.1 Key Management

Secret communication is an important requirement in many sensor network applications, so shared secret keys are used between communicating nodes to encrypt data. Some of the major constraints like ad hoc nature, intermittent connectivity, and resource limitations of the sensor networks prevent traditional key management and distribution schemes to be applicable to WSN.

It is not possible to setup an infrastructure to manage keys used for encryption in the traditional internet style because of factors such as unknown and dynamic topology of such networks, vagaries of the wireless link used for communications, and lack of physical protection. A practical solution given these constraints is to pre-load the keys on the sensors before the sensor nodes are deployed. Thus the node has some secret information in it before being deployed, and using this information, they need to setup a secure communication infrastructure for use during operation.

The need to form a group might be driven by the requirement that a query being propagated through any node and in turn, such a node need to define a multi cast group to make the query

initiated in those nodes and then collect the result over a time efficiently and securely, and also modify such queries effectively over the time. One example of such multi cast group could be a region defined with a geometric shape.

1.2.2 Secure Group Communication

Group communication is an internetwork service that provides efficient delivery of data from a source to multiple recipients. Secure group communication enables each user to determine/obtain the same session key (SK) without permitting unauthorized users to do likewise and securely update keys to prevent the leaving/joining user from accessing the future/prior communications, which is referred to as forward and backward secrecy.

Secure group communication is increasingly used as an efficient communication method for group-oriented applications in WSNs. Group communication in WSNs are more susceptible to unauthorized access because of the open nature of broadcast channel. One method for enabling secure group communication is the periodic distribution of a new key (called a session key) to group members. All messages exchanged within the group during a fixed interval of time, or session, are communicated securely through encryption under this session key. Group controller is responsible for distributing session key to the nodes in the group securely by encrypting them using the Key Encrypting Key.

Moreover, considering the dynamic node topology due to nodes attachment and detachment, it is necessary to refresh the session key to prevent the detached node from accessing future communications and the newly attached node from accessing previous communications. The updation of session key after some fixed interval of time is called re-keying process, which ensure that newly added nodes can not decrypt previous group communication and prevents the detached node from eavesdropping future group communication. Self healing is the property which makes nodes' enable to recover the missed session keys on their own from the latest re-keying message without

contacting the group controller.

1.3 Problem Statement

The key management scheme employed in a sensor network for secure application must integrate authenticity, confidentiality, integrity, scalability, availability and flexibility. In addition to it some other requirement for key management scheme include supporting in-network processing and facilitating self-organization of data, among others. Key management is crucial to the secure operation of WSNs and must also meet certain other criteria for efficiency in light of vulnerability to adversaries, including resistance, revocation, and resilience.

Key management in most of the wired and wireless network is achieved through public key cryptography. WSNs have a limited resource,which can make using public key cryptography, impractical as they use much more energy for their integral complex mathematical calculations and increased code storage requirements. This constraint is mitigated by making use of more efficient symmetric techniques that involve fewer computational procedures and require less energy and memory to function. The symmetric key management schemes used in WSNs can be classified as single network-wide key, pairwise key establishment,trusted base station, and random key pre-distribution.

Key pre-distribution schemes are a favored solution for establishing secure communication in sensor networks. Often viewed as the safest way to bootstrap trust, the main drawback is seen to be the large storage overhead imposed on resource-constrained devices and also these schemes are quite insecure because pre-loading global secrets onto exposed devices strengthens the incentive for attackers to compromise nodes.

Random key pre-distribution schemes are vulnerable to collusion attacks. Colluding attackers mainly take advantage of the pairwise secret keys stored by each sensor node as these keys are selected from same key pool and can be used throughout the network, yet ordinary sensors can

only communicate with the small fraction of nodes within radio range. An attacker can readily exploit this lack of coordination between nodes and can now share its pairwise keys between compromised nodes, enabling each to present multiple authenticated identities to neighboring nodes while escaping detection.

Most existing research mainly considers homogeneous sensor networks, where all sensor nodes have identical capabilities in terms of communication, computation, sensing, and reliability; however, homogeneous WSNs are not scalable. As large-scale homogeneous networks suffer from high costs of communication, computation, and storage requirements. The HSNs are preferred because they provide better performance and security solutions for scalable applications in dynamic environments.

Efficient solutions for the problem of key distribution are essential for the feasibility of secure group communication in sensor networks. Secure group communication needs a secret shared by all the group members for group oriented applications in WSNs. The shared key provides group secrecy and source authentication. Secure group communication protocols should provide efficient rekeying and self healing property. In rekeying, the session keys are updated periodically, when new users join or old users leave the group. Another important problem in multicast communication is reliability. Since multicasting is an unreliable mode of communication, packets may be lost during the communication. If a packet containing key updating information is lost, authorized receivers may not be able to calculate the session key. This may influence rekeying and so the rekeying system must be self-healing if packet loss occurs.A user who has been off-line for some period is able to recover the lost session keys immediately after coming back on-line. Thus self-healing approach of key distribution is stateless.

The taxonomy of attacks and challenges allow us to formulate the research questions investigated in this thesis. It has been identified that key management and secure group communication are

two of the most important security aspects in WSNs. There is a need to design key management schemes that counter the problem of collusion attack and scalability issue in random key pre-distribution schemes.

1.4 Contributions

The objective of this research is to provide partial solution to the research problems given in the previous section. HSN consisting of a small number of high-end sensors ($\mathcal{H}$-node) and a large number of low-end sensors ($\mathcal{L}$-node) is considered. Further, for scalable solutions, the proposed schemes use hierarchical structure, where $\mathcal{H}$-node act as cluster head (CH) and $\mathcal{L}$-node as cluster members.

A scalable and efficient protocol for key management is proposed that is sensitive to the sensor nodes resource constraints, including storage, computation and communication. The proposed key management scheme is based on random key pre-distribution for HSNs.

To address storage overhead problem in the constraint sensor nodes, a key generation process is incorporated, where instead of generating a large pool of random keys, a key pool is represented by a small number of generation keys. For a given generation key and a publicly known seed value, a keyed-hash function generates a key chain; these key chains collectively make a key pool. In pre-deployment phase each $\mathcal{H}$-node is preloaded with the master key and a large number of generation keys. Each $\mathcal{L}$-node is preloaded with authentication key (which is generated by applying one way hash function on $\mathcal{L}$-node id and master key) and a small number of generation keys. As a result, by using generation keys, the proposed scheme significantly reduces the storage requirements. Adversary or malicious nodes are precluded to join the cluster as each $\mathcal{L}$-node is authenticated by CH using $\mathcal{L}$-noders authentication key.

Dynamic network topology is native to WSNs because nodes can fail or be added. As result, the proposed scheme allows dynamic node addition and removal. In the case of node addition,

the proposed scheme is able to distinguish between legitimate and malicious nodes. Further, as adversaries can compromise sensors and acquire all security information, a rekeying scheme is incorporated to update all types of keys periodically. Secure inter-cluster communication is also provided.

This research also reports the implementation and the performance of the proposed scheme on Crossbows MicaZ motes running TinyOS and evaluate the computation and storage costs of two keyed-hash algorithms for key chain generation, HMAC-SHA1 and HMAC-MD5. The results show that proposed scheme can significantly reduce the storage requirements as compared to other random key pre-distribution schemes. Also, the resiliency against node capture is better than previous key pre-distribution schemes.

Random key distribution schemes are identified to be susceptible of collusion attacks. As a consequence, a new random key distribution scheme is proposed to prevent collusion attacks in ubiquitous HSN. The collusion attack on key pre-distribution scheme mainly takes advantage of the globally applicable keys, which are selected from the same key pool. This attack can be prevented if the initial key ring of each node is deleted after setting up the shared pairwise keys with neighbors. But the random key pre-distribution schemes will be unable to add new nodes once the initial key rings has been deleted from nodes memory.

As a result, in proposed scheme, after discovering the shared pairwise keys with neighbors, all $\mathcal{H}$-node and $\mathcal{L}$-node destroy their initial key rings and generate new key rings by applying one-way hash function on nodes ID and initial key ring. Further, in order to keep record of the keys in its initial key ring, these newly generated key rings are assigned the same ids as that were of the original keys. The proposed scheme can also add new nodes in the network beyond the initial deployment even after the deletion of initial key rings from nodes memory.

In collusion attacks, adversary mainly takes advantage of globally applicable keys choosing form

the same key pool. But in proposed scheme, no two nodes in the network have common keys in their key rings after deletion of the old key rings. As no more globally applicable secrets remain in the nodes memory, it is not possible by adversary to launch a collusion attack.

The analysis of proposed scheme shows that even if a large number of nodes are compromised, an adversary can only exploit a small number of keys nearby the compromised nodes, while other keys in the network remain safe. It outperforms the previous random key pre-distribution schemes by considerably reducing the storage requirement, while providing more resiliency against node capture and collusion attacks.

As a third contribution, this research develops a computationally secure and efficient group key distribution scheme with self-healing property and time-limited node revocation capability for large and dynamic groups over insecure HSNs. This research presents a strategy for securely distributing rekeying messages and specify techniques for joining and leaving a group.

Access control in multicast system is usually achieved by encrypting the content using an encryption key, known as the group key (session key) that is only known by the group controller and all legitimate group members. The session keys are updated periodically, where the update is performed regardless of changes in network (group) topology. Periodic rekeying can significantly reduce both the computation and communication overhead at the GC and the nodes, and thus improve the scalability and performance of key distribution protocols.

The group life cycle is given by m, which determines the total number of sessions for a group. The GC uses the pseudorandom number generator (PRNG) of a large enough period to produce a sequence of m random numbers $(r_1, r_2, \ldots, r_m)$. The GC randomly picks two initial key seeds, the forward key seed S^F and the backward key seed S^B. In the pre-processing time, it computes two hash chains of equal length m by repeatedly applying the same one-way hash function on each seed. During the initial configuration setup, each node assigned a prearranged life cycle

(t_1, t_2) along with forward key at time t_1 and $t_2 - t_1$ number of random numbers corresponding to the sessions in which the node will participate in the group communication.

In each session, the GC compute the broadcast message by XOR of backward key and random number corresponding to that session. when the node receives the broadcast message, it recovers the backward key by Xoring the broadcast message with the random number of that session. It then computes the forward key of that session by applying one-way hash function on its forward key and then computes the session key from the forward and backward keys of that session.

All rekeying messages, except for unicast of an individual key, are transmitted without any encryption using one-way hash function and XOR operation. In proposed scheme, nodes are capable of recovering lost session keys on their own, without requesting additional transmission from the group controller. The time-limited node revocation is achieved without any intervention from the GC. The dual direction (forward and backward) hash chains are used to provide forward and backward secrecy of the session key.

The analysis shows that the proposed scheme is computationally secure and meets the security requirements for forward and backward secrecy. Further, it can tolerate high channel loss rate and hence make a good balance between performance and security which is suitable for WSN applications.

1.5 Organization of Research Work

This thesis has been organized in Chapters. Chapter 2 presents a literature survey of key managements and secure group communication in WSNs. Chapter 3 presents the scalable and efficient key management scheme based on random key distribution for HSNs. Chapter 4 presents the random key distribution scheme preventing collusion attacks in ubiquitous HSNs. Chapter 5 presents a key distribution scheme for secure group communication with self healing and rekeying in HSNs. Lastly, chapter 6 concludes the research and highlights the future work.

Chapter 2

BACKGROUND

2.1 Introduction

A wireless sensor network typically consists of a potentially large number of incredibly resource constrained sensor nodes. Each sensor node is usually battery powered, and has a low-end processor, a limited amount of memory, and a low power communication module capable of short-range wireless communication. Their lifetime is determined by their ability to conserve power. The sensor nodes form an ad-hoc network through the wireless links. There are many technological hurdles that must be overcome for ad hoc sensor networks to become practical though. All of these constraints require new hardware designs, software applications, and network architectures that maximize the motes capabilities while keeping them inexpensive to deploy and maintain.

In the not too distant future, tiny, dirt-cheap sensors may be literally sprayed onto roads, walls, or machine, crating a digital skin that sense a variety of physical phenomenon of interest: monitor pedestrian or vehicular traffic in human aware environments and intelligent transportation grids, report wildlife habitat conditioning for environmental conservation, detect forest fires to aid rapid emergency response, and track job flows and supply chains in smart factories [6].

Sensor networks may consist of many different types of sensors such as seismic, low sampling rate magnetic, thermal, visual, infrared, acoustic and radar, which are able to monitor a wide variety of ambient conditions that include the following :temperature, humidity, vehicular movement, lightning condition, pressure, soil makeup, noise levels, the presence or absence of certain kinds of objects, mechanical stress levels on attached objects, and the current characteristics such as speed,

direction, and size of an object.Typical sensor network platforms have limited bandwidth. For example, the UC Berkeley Mica platform's transmitter has a bandwidth of 10 Kbps. Transmission reliability can be low, which makes the communication of large blocks of data expensive [7].

2.1.1 Sensor Hardware

A sensor node is made up of four basic components as shown in Figure 2.1 a sensing unit, a processing unit, a transceiver unit and a power unit. There are many types of sensors, e.g. thermistors for measuring temperature; accelerometers for measuring acceleration; magnetometer and micropower impulse radar (MIR) sensors for detecting metallic objects; barometric pressure sensors; dielectric-based humidity sensors; acoustic sensors; sensors for various types of chemicals etc. The nature of the sensors may affect the cost and physical size of the sensor nodes, but does not affect the general characteristics of WSNs.

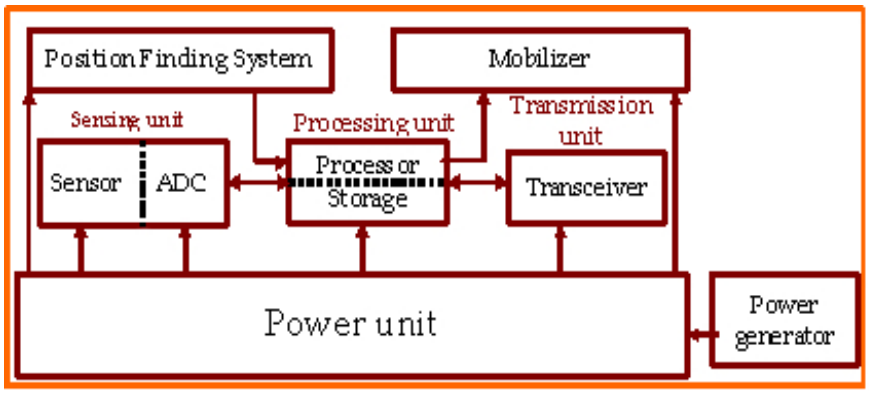


Figure 2.1: Components of a Typical Sensor Node

The sensor nodes have featured commercial off the shelf (COTS) components integrated together on a platform commonly referred to as a "mote" [8, 9] Comparison of family of wireless sensor nodes, or "motes" both available in the market as well as in academia is given in Table 2.1.

Among microcontrollers, 4 MHz Atmels and the 8 MHz MSP430F149 are the more popular choices, whereas Chipcon and RFM are popular among radio transceivers. 4 KB of RAM and 128 KB of program memory should be considered the current upper limits. All in all, motes are

Table 2.1: Comparison of Motes

Mote Type / **Year**	**WeC** / **1998**	**Rene** / **1999**	**Rene 2** / **2000**	**Dot** / **2000**	**Mica** / **2001**	**Mica2Dot** / **2002**	**MicaZ** / **2002**	**Telos** / **2004**
Microcontroller								
Type	AT90LS8535		ATmega163		ATmega128			T1 MSP430
Program memory(KB)	8		16		128			60
RAM(KB)	0.5		1		4			2
Active Power(mW)	15		15		8		33	3
Sleep Power(μW)	45		45		75		75	6
Wakeup Time(μs)	1000		36		180		180	6
Nonvolatile Storage								
Chip	24LC256				AT45DB041B			ST M24M01S
Connection type	I^2C				SPI			I^2C
Size(KB)	32				512			128
Communication								
Radio	TR1000				TR1000	CC1000		CC2420
Date rate(kbps)	10				40	38.4		250
Modulation Type	OOK				ASK	FSK		O-QPSK
Receive Power(mW)	9				12	29		38
Transmit Power at 0dBm(mW)	36				36	42		35
Power Consumption								
Minimum Operation(V)	2.7		2.7		2.7			1.8
Total Active Power(mW)	24				27	44	89	41

notably characterized by their limitations slow processor, little storage, low bandwidth and scarce energy reserve. Among these, energy is the most severe constraint [7]. For example, when used with MICA2 motes, the effective lifetimes of alkaline, NiMH and LiSO2 batteries are 18 hours, 100 hours and 30 days [10] respectively, while these motes are meant to operate, ideally, for years.

Sensor nodes can be used for continuous sensing, event detection, event ID, location sensing, and local control of actuators. The concept of micro-sensing and wireless connection of these nodes promises many new application areas. The applications can be categorized into military, environment, health, home and other commercial areas. It is possible to expand this classification with more categories such as space exploration, chemical processing and disaster relief [3]. Some of the commercial applications are monitoring material fatigue; building virtual keyboards;

managing inventory; monitoring product quality; constructing smart office spaces; environmental control in office buildings; robot control and guidance in automatic manufacturing environments; interactive toys; interactive museums; factory process control and automation; monitoring disaster area; smart structures with sensor nodes embedded inside; machine diagnosis; transportation; factory instrumentation; local control of actuators; detecting and monitoring car thefts; vehicle tracking and detection; and instrumentation of semiconductor processing chambers, rotating machinery, wind tunnels, and anechoic chambers.

Key distribution refers to the problem of establishing shared secrets on sensor nodes such that secret symmetric keys for communication privacy, integrity and authenticity can be generated. In a wireless sensor network, pre-distribution of secret keys is possibly the most practical approach to protect network communications. To meet the stringent resource constraints of the sensor nodes, such as limited storage capability, low computation capability, and limited battery life, key pre-distribution schemes should be highly efficient, namely requiring as little storage space as possible, and at the same time, maintain a strong security strength, i.e., high resilience against node capture. Security of large scale densely deployed and infrastructure-less wireless networks of resource limited sensor nodes requires efficient key distribution and management mechanisms. This chapter gives the overview of wireless sensor network architecture, applications, limitations and security requirements. Further, this chapter reviews several key distribution and key establishment techniques for sensor networks. Brief description of several well known key establishment schemes are given with a more detailed discussion on random key distribution.

Sensor networks have many characteristics that make them more vulnerable to attack than conventional computing equipment. This physical attacks on sensor networks can be prevented if the sensor nodes are made temper-resistant. As sensor nodes are deployed in hostile environment, it is assumed the adversary can mount a physical attack on a sensor node and read secret information

from its memory or can simply destroy it. Therefore the costly solution of making these sensor node temper-resistant in not acceptable/practical. In node replication attack the compromise of even a single node might allow an adversary to populate the network with clones of the captured node to such an extent that legitimate nodes could be outnumbered and the adversary can thus gain full control of the network. In order to develop a key establishment scheme for sensor networks, sensor node constraints include battery life, transmission range, bandwidth, memory, and prior deployment knowledge must be kept in mind.

This chapter gives the overview of wireless sensor network architecture, applications, limitations and security requirements. Further, this chapter reviews several key distribution and key establishment techniques for sensor networks. Brief description of several well known key establishment schemes are given with a more detailed discussion on random key distribution.

2.2 Previous Work : Key Management in Wireless Sensor Networks

Key management schemes [11] in wireless sensor networks are categorized as follows: (1) Single network-wide key [12, 13], (2) Pairwise key establishment, (3) Trusted base station [14], (4) Public key schemes (elliptic curve cryptography) [15, 16, 17, 18] and (5)Key predistribution schemes [**?** 19, 20, 21, 22, 23, 24, 25].

2.2.1 Using a Single Network-Wide Key

The simplest method of key distribution is to pre-load a single network-wide key onto all nodes before deployment. After deployment, nodes establish communications with any neighboring nodes that also possess the shared network key. This can be achieved simply by encrypting all communications in the shared network-wide key and appending a message authentication code (MAC) to ensure integrity. This method requires minimal memory storage because only a single cryptographic key is needed to be stored in memory. No additional protocol steps are necessary

as the protocol works without needing to perform key discovery or key exchange. It is resistant against DoS attack and packet injection. The MACs (message authentication code) guard against arbitrary packet injection by an adversary that does not know the network-wide key. Replay attacks can be prevented by including the source, destination, and a time stamp in the message.

The main drawback of the network-wide key approach is that the compromise of a single node causes the compromise of the entire network, since the network-wide key is now known to the adversary. Basagni use this approach to design a secure routing protocol [12]. No new nodes are ever added to the system after deployment. In this case, the sensor nodes use the network wide key to encrypt unique link keys which are exchanged with each of their neighbors. Zhu follow this approach and set up all keys from a single network-wide key during a short, initial phase after deployment, assuming that no nodes are compromised during this phase, and later all nodes erase the single network key [13]. This approach, however, is vulnerable to compromise of a single node that misses the key setup period, and does not erase its key.

2.2.2 Using Pairwise Key Establishment

In this approach, every node in the sensor network shares a unique symmetric key with every other node in the network. Hence, in a network of n nodes, there are a total of $n * (n - 1)$ unique keys. Every node stores $n - 1$ keys, one for each of the other nodes in the network. After deployment, nodes must perform key discovery to verify the identity of the node that they are communicating with.

This approach is perfectly resilient to node capture. Similar to the asymmetric cryptography scheme, any node that is captured reveals no information about the communications being performed in any other part of the network. Its pair-wise keys could be used to perform a node replication attack throughout the network, but this could be countered using the same method as described for asymmetric cryptography in the previous section.

In this approach compromised keys can be revoked because if a node is detected to be compromised, its entire set of $n-1$ pair-wise keys is simply broadcast to the network. No authentication is necessary. Any node that hears a key in its set of pair-wise keys broadcast in the open immediately stops using it. This effectively cuts off the revoked node from the network. This approach only uses symmetric cryptography. The pair-wise keys scheme achieves many of the benefits of using asymmetric cryptography without needing dedicated hardware or software. This not only makes the nodes cheaper but also makes the network less vulnerable to denial of service attacks.
The main problem with the pair-wise keys scheme is poor scalability. The number of keys that must be stored in each node is proportional to the total number of nodes in the network. If there are total n number of nodes in the network, each node must store $n * (n-1)$ number of keys in its memory in order to securely communicate with other nodes.

2.2.3 Using Trusted Base Station

This method of key distribution uses a trusted, secure base station as an arbiter to provide link keys to sensor nodes, e.g., similar to Kerberos [26, 27]. The sensor nodes authenticate themselves to the base station, after which the base station generates a link key and sends it securely to both parties. An example of such a protocol is part of the SPINS security infrastructure [14].
A sketch of the events of the protocol could be as follows. Prior to deployment, a unique symmetric key is generated for each node in the network. This node key is stored in the node's memory and will serve as the authenticator for the node as well as facilitate encrypted communications between the node and the base station. The base station has access to all the node keys either directly (they are stored in its memory) or indirectly (the base station relays all communications to a secured workstation off site).
This method, unlike the other methods mentioned previously, assumes that some level of reliable transport is available between the node and the base station before any key establishment has

taken place. Now assume that after deployment, node A wants to establish a shared secret session key with node B. Since A and B do not share any secrets, they need to use a trusted third party S, which is the base station in our case. SPINS includes a protocol where A and B can establish through the base station [14]. Small memory is required in this approach because for every node, a single secret symmetric key shared with the base station is needed, as well as one unique link key for each one of its neighbors. Furthermore, the key establishment is efficient, as it only requires symmetric cryptographic primitives. It provides perfect resilience to node capture. Any node that is captured divulges no secret information about the rest of the network.

Revocation of nodes is simple. If a node is to be revoked, the base station securely transmits the revocation message to all the nodes that may be in communication with the revoked node. This revocation message is encrypted with the secret key that is shared only between the recipient node and the base station hence secrecy and authentication are ensured. To prevent any other nodes from establishing links with the revoked node, the base station simply needs to reject requests that involve the revoked node as a principal. Node replication is easily controlled. Since all key establishment activity takes place through the central base station, auditing becomes trivial.

This approach is not scalable and has significant communication overhead. If any two nodes wish to establish a secure communications, they must first communicate directly with the base station. In a large network, the base station may be many hops away, thus incurring a significant cost in communication. The base station can become a target for compromise.

The base station has access to all the secret node keys in the sensor network, compromise of the base station's key store will expose the secrecy of all links that are established after the time of the compromise. This may not be a problem if the communications base station merely acts as a gateway to a workstation at a remote, secured site, since the adversaries would have to successfully attack the secure workstation in order to gain the node keys.

2.2.4 Using Asymmetric/Public Key Cryptography

Traditionally, security is provided through public-key based protocols. However, these protocols require large memory, bandwidth and complex algorithms. The limited resources of WSNs make this type of security schemes unsuitable for implementation. Recently many researchers take this as challenge and use the public key cryptography based schemes in order to provide security in WSN. Two of the major techniques used to implement public-key cryptosystems are RSA and ECC. In this section the review of several public key cryptography based techniques in WSNs is given and come to the conclusion that the public key schemes based on elliptic curve cryptography are more suitable in WSNs [28].

Malan propose that public key infrastructure is viable for TinySec key's distribution [18]. TinySec is the link layer security architecture for wireless sensor networks based on SKIPJACK in cipher-block chaining mode [29]. It is shown that elliptic curves are believed to offer security computationally equivalent to that of Diffie-Hellman based on discrete log problem with remarkably smaller key sizes of 163 bit for the secure distribution of 80-bit TinySec keys. They 1st implement ECC on the MICA2 (ECCM 1.0),a TinyOS module based on code by Rosing [30], but for larger keys (e.g., 63-bit), the module failed to produce results, instead causing the mote to reset as a result of stack overflow. Since optimizations of EccM 1.0 failed to render generation of even 63-bit keys possible, an overhaul of this popular implementation proved necessary for realization of 163-bit keys. A point G is selected from an elliptic curve E, both of which are public. A random integer K_A is selected, which will act as the private key. The public key is then $T_A = K_A \times G$. Bob performs a similar set of operations to compute $T_B = K_B \times G$. Alice and Bob can now easily compute the shared secret using their own private keys and the public keys that have been exchanged. In this case, Alice computes $K_A \times T_B = K_A \times K_B \times G$ while Bob computes $K_B \times T_A = K_B \times K_A \times G$. Because $K_A \times T_B = K_B \times T_A$, Alice and Bob now share

a secret key. The implementation results show that a fixed point multiplication takes 6.74 sec and random point multiplication takes 17.28 sec. Key generation process consumes 6.74 sec. A complete key exchange takes on average 17.28 sec. In Elliptic curve digital signature algorithms the signature generation consists of only one point multiplication of the fixed point P taking only around 6.88sec. A signature verification step takes about 24.17sec.

Blab give implementations of asymmetric encryption and signature generation schemes for the 8Bit ATMEL sensor platform that features acceptable run-time and memory consumption while preserving a level of acceptable security for sensor networks [31]. They implement algorithms i.e. Diffie-Hellman ,El-Gamal, DSA based on ECC that offer the same security than traditional based algorithms but consume a lot less of memory and computing power. In this scheme fast implementation is based on the pre-computation of points and it save main memory (RAM) by moving all larger unchangeable data from RAM to flash-ROM.

Wander give the energy analysis of authentication and key exchange based on RSA and ECC on an 8-bit microcontroller platform [32]. The mutual authentication and key exchange protocols is based on SSL handshake [33]. They give the energy cost of digital signature and key exchange by using RSA and ECDSA (elliptic curve digital signature algorithm). The RSA-based key exchange protocol relies on party A to encrypt a randomly generated secret key with party B's public key, and party B decrypting the key using its private key. With ECC, both parties perform a single ECDH (elliptic curve Diffie-Hellman) operation to derive the secret key. For ECC, key generation only involves generating a random number, which becomes the user's private key, and executing an ECDH operation to compute the corresponding public key. RSA key generation is much more time consuming as it requires the generation of large prime numbers [34]. They analyze energy usage of the simplified SSL handshake based on RSA-1024 and ECC-160. The amount of combined energy spend by both parties is determined by public-key computation, transmitting and

receiving handshake messages, hash computation, and random number generation. The results of [32] indicate that an RSA-1024 computation consumes 4.9 times the energy of ECC-160 computations.

Abraham proposed a set of security protocols for wireless sensor networks based on Tiny Diffusion [35] protocol, for the underlying network communication and ECC as a public key system [36] . This paper discusses use of elliptic curve cryptosystems and how Tiny Diffusion components of TinyOS [8] are to be modified to implement essential security protocols like establishing shared key between base station and any source node using ECDH, transferring data with end-to-end security and authenticated rekeying using ECDSA. Implementation of ECDH and ECDSA in this proposed work has considered Koblitz curve. The curve parameters are selected over F_2^P field having degree $P = 163$ bits and is taken from recommended elliptic curves for Federal Government use [37]. The simulation results of [36] show that the proposed protocols are efficient in respect with memory requirement and communication delay compared to [32].

Watro focuses on providing authentication and key agreement between a sensor networks and a third party as well as between two sensor networks using RSA public key cryptosystems and Deffie-Hellman [38] key agreement techniques [17]. TinyPK is based on the well-known RSA cryptosystem, using $e = 3$ as the public exponent. Any third party that wishes to interact with the motes also requires its own public/private key pair and must have its public key signed by the CA's private key, thus establishing its identity. Finally, as each mote is loaded with CA's public key. TinyPk challenge response protocol for authenticating the third party starts when the third party provides challenge consisting of its own public key, signed by the CA private key; and a compound object consisting of a nonce and a message checksum, signed with the third party's own private key. Upon receipt of the message, a sensor node uses the preloaded CA public key to verify the first part of the challenge and extract the third party's public key. It then uses this public

key to verify the second part of the message and extract the nonce and checksum. If the nonce and checksum pass validation, the third party has successfully authenticated to the sensor network and is considered to be an authorized entity for sensor data. The sensor node now encrypts the session key plus the received nonce using the third party's public key. This combination is sent back to the third party, which decrypts it using its private key, checks that the nonce is the same as the one it sent, and if so, can record the session key for future use. They use Diffie-Hellman to generate a secret suitable for use in creating a new or replacement TinySec key.

Gaubatz proposes a custom hardware assisted approach by implementing public key algorithms [16] i.e. Rabin's Scheme [39] and NtruEncrypt [40] on sensor nodes and analyze their architecture and performance according to various established metrics like power consumption, chip area, delay, throughput, level of security and energy per bit. For Rabin's Scheme they select an operand size of 512 bits, which according to Lenstra and Verheul provides a security level of around 60 bits [41]. In the case of NtruEncrypt they chose the system parameters as $(N, p, q) = (167, 3, 128)$, based on findings in [40], offering a security level of 57 bits. For both algorithms only encryption operation is considered and the public key is either hardwired or realized as look-up table in combinational logic.

Manley present a hybrid scheme that shared secrets, preloaded on nodes with a random key pre-distribution scheme [42], to allow node-to-node authentication so that trust may be verified before setting up cryptographic keys with public-key methods [43]. In this proposed public key extension of the random key pre-distribution scheme, the key pre-distribution and shared key discovery phases are performed as in basic approach. After the shared-key discovery phase, any two nodes that share a key may set up a new key for secure communication using public-key methods. For pairs of nodes in communication range that do not share a key, the public-key methods can also be used, by using the following method. An authenticator set, s is defined when the network

is designed. Two nodes that wish to set up a new key must have s commonly shared trusted neighbors. When a key-establishment message is sent between the two nodes, a MAC must be computed for each of the s nodes in the authenticator set by using pre-distributed shared key. These s codes are then sent to each of the nodes in the authenticator sent. Each node in the authenticator set verifies the integrity of the message and then creates a new MAC using the message and its key shared with the message's receiver. Each authenticator sends the message authentication code to the receiving party in the key setup. The receiver verifies all s MACs before accepting the message and continues with the protocol.

2.2.5 Random Key Predistribution Scheme (Basic scheme)

In this subsection, Random Key Predistribution Scheme proposed by Eschenauer and Gligoris described in detail [42]. This scheme is also referred to as Basic Scheme. In the Basic Scheme, key distribution is divided into three stages: key predistribution, shared-key discovery, and path-key establishment.

2.2.5.1 Key Predistribution Stage

In the key predistribution stage, a large key pool of $\|S\|$ keys and their identifiers are generated. From this key pool, K keys are randomly drawn and pre-distributed into each nodes key ring, including the identifiers of all those keys. At the point that each node has K keys and the identifiers of those keys, trusted nodes in the network are selected as controller nodes, and all the key identifiers of a key ring and the associated sensor identifiers on controller nodes are saved. Following this, the ith controller node is loaded for each node with the key that is shared with that node. This key predistribution process ensures that, though the size of the network is large, only a few keys need to be stored in each nodes memory, thereby saving storage space. These few keys are enough to ensure that two nodes share a common key, based on a selected probability.

2.2.5.2 Shared Key Discovery Stage

Once the nodes are initialized with keys, they are deployed in the respective places where they are needed, such as hospitals, war fields, etc. After deployment, each node tries to discover its neighbors with which it shares common keys. There are many ways for finding out whether two nodes share common keys or not. The simplest way is to make the nodes broadcast their identifier lists to other nodes. If a node finds out that it shares a common key with a particular node, it can use this key for secure communication. This approach does not give the adversary any new attack opportunities and only leaves room for launching a traffic analysis attack in the absence of key identifiers. More secure alternate methods exist for finding out the common keys shared between two nodes though. For example, for every key on a key ring, each node could broadcast a list $\alpha, E_{K_i}(\alpha), i = 1 \ldots k$, where α is a challenge. The decryption of $E_{K_i}(\alpha)$ with the proper key by a recipient would reveal the challenge α and establish a shared key with the broadcasting node.

2.2.5.3 Path Key Establishment Stage

A link exists between two nodes only if they share a key, but the path key establishment stage facilitates provision of the link between two nodes when they do not share a common key. Suppose that node u wants to communicate with node v, but they do not share a common key between them. Node u can send a message to node y saying that it wants to communicate with node v; this message is then encrypted using the common key shared between node u and node y and, if node y has a key in common with node v, it can generate a pairwise key $K_{u,v}$ for nodes u and v, thereby acting like a key distribution center or a mediator between the communication of nodes u and v. As all the communications are encrypted using their respective shared keys, there will not be a security breach in this process. After the shared key discovery stage is finished there will be a number of keys left in each sensors key ring that are unused and can be put to work by each sensor node for path key establishment.

2.2.5.4 Key Revocation

A compromised sensor node can cause a lot of damage to a network and therefore, revocation of a compromised node is very important in any key distribution scheme. In the Basic Scheme, node revocation is conducted by the controller node. When a node is revoked, all the keys in that particular node key ring have to be deleted from the network. Assume that the controller node has knowledge about a compromised node in the network and broadcasts a message to all the nodes in the network, the message includes a list of the key identifiers of the compromised nodes key ring. To sign the list of key identifiers, the controller node uses a signature K_e and then encrypts its message with K_{c_i} , which is the key that the controller node shares with the nodes during the key predistribution stage. Once each node receives the message, it decrypts the message using the key already shared with the controller node. When the signature is verified, the nodes search their key rings for the list of identifiers provided in the message and, if there is any match, the corresponding keys are deleted from the key ring. After the matching keys are completely deleted from all the nodes, there may be links missing between different ones and they then have to reconfigure themselves starting from the shared key discovery stage so that new links can be formed between them. As only few keys are removed from the network, the revocation process only affects a part of it and does not incur much communication overhead.

The keys used in a sensor network must be rekeyed to lessen the chance that an adversary may access all of the network keys when a few nodes and their keys are captured. Rekeying effectively increases a networks resilience without incurring much communication and computation overhead.

2.2.5.5 Analysis of the Basic Scheme

Let us assume that the probability of a common key existing between two nodes in the network is p, and the size of the network is n. The degree of a node d is derivable using both p and n since

the degree of any node is simply the average number of edges connecting that node with other nodes in its neighborhood; therefore,$d = p \times (n-1)$. First we have to find the value of d such that a network of n nodes is connected with a given probability P_c. We then must calculate the key ring size k and the size of the key pool $|S|$.

According to Random Graph Theory, a random graph $G(n,p)$ is a graph consisting of n nodes and p representing the probability of establishing a link between two nodes. Erdos and Renyi [12] showed that there exists a probability state p, which moves from state zero to state one for large random graphs. The function that defines p is called the threshold function of a property. If we are given a desired probability (P_c) for graph connectivity, then p is given as $P_c = lim_{n\to\infty}$Pr[G(n,p) is connected]$= e^{e^{-c}}$, $p = \frac{ln(n)}{n} + \frac{c}{n}$, where c is a real constant.

Then, to calculate the key ring size k and the size of the key pool $|S|$, we need to first note that wireless constraints limit the number of nodes in a range to be smaller than n, represented by the value $\acute{n}$. Now the probability of sharing a key between two neighbor nodes varies to $\acute{p} = d/(\acute{n}-1)$, for a given d value. Also, $\acute{p}$ can be denoted as the difference between the total probability and the probability that two nodes do not share a common key; i.e., $\acute{p} = 1-$ Pr[two nodes do not share any key] and, thus, $\acute{p} = 1 - \frac{(1-\frac{k}{p})^{2(p-k+\frac{1}{2})}}{(1-\frac{2k}{p})^{(p-2k+\frac{1}{2})}}$, where $|S|$ is the size of the key pool and k is the key ring size.

Eschenauer and Gligor [42] have shown that for a pool size S = 10,000 keys, only 75 keys need to be stored in a nodes memory to have the probability that they share a key in their key rings to be p = 0.5. If the pool size is ten times larger, i.e., S = 100,000, then the number of keys required is still only 250. Thus, the Basic Scheme is a key management technique that is scalable, flexible and can also be used for large networks.

Trade-offs in the Basic Scheme can be made between sensor memory and connectivity, but it does not provide the node-to-node authentication property that ascertains the identity of a node with

which another node is communicating. This property is very useful when revoking misbehaving nodes from the network and also helps in resisting the node replication attack.

Many key management schemes [19], [20] , [21], [22], [23], [24], [25], etc. are proposed as extensions of the basic scheme to make it even more secure and reliable. Advantages of this scheme include flexible, efficient, and fairly simple to employ, while also offering good scalability. Disadvantages of this scheme include that it cannot be used in circumstances demanding heightened security and node to node authentication.

2.2.6 Other Schemes based on Basic Random Key Predistribution Scheme

Chan et al. [19] propose the q-composite key pre-distribution, which allows two sensors to setup a pairwise key only when they share at least q common keys. The q-composite keys scheme is a modification to the basic scheme [42], where q common keys ($q > 1$) are needed, instead of just one. By increasing the amount of key overlap required for key-setup, the scheme increases the resilience of the network against node capture. The q-composite keys scheme first proceeds in a similar manner to the basic random keys scheme. That is, the deployer picks a set S of random keys out of the total key space and for each node in the network, selects m random keys from S and stores them into the node's key ring. Nodes then perform key discovery with their neighbors. After key discovery, each node can identify every neighbor node with which it shares at least q keys. Let the number of actual keys shared be $\acute{q}$, where $\acute{q} \geq q$. A new communication link key K is generated as the hash of all shared keys, e.g.$K = hash(k_1, k_2, \ldots, k_{\acute{q}})$. The probability of connection keys are hashed in some canonical order, for example, based on the order they occur in the original key pool S. Key-setup is not performed between nodes that share fewer than q keys. The analysis shows that as the amount of required key overlap increases, it becomes exponentially harder for an attacker with a given key set to break a link. However, to preserve the given probability p of two nodes sharing sufficient keys to establish a secure link, it

is necessary to reduce the size of the key pool $|S|$. This allows the attacker to gain a larger sample of S by breaking fewer nodes. The interplay of these two opposing factors results in an optimal amount of key overlap to pose the greatest obstacle to an attacker for some desired probability of eavesdropping on a link.

In multi-path key reinforcement [19] it is assumed that initial key-setup has been completed. There are now many secure links formed through the common keys in the various nodes' key rings. Suppose node A has a secure link to node B after key-setup. This link is secured using a single key k from the key pool S. k may be residing in the key ring memory of some other nodes elsewhere in the network. If any of those nodes are captured, the security of the link between A and B is compromised. To address this, the communication key is updated to a random value after key-setup. The key-update is coordinated over multiple independent paths in this approach. It is assumed that enough routing information can be exchanged such that A knows all disjoint paths to B created during initial key-setup that are h hops or less. Specifically, $A, N_1, N_2, \ldots, N_i, B$ is a path created during the initial key-setup if and only if each link$(A, N_1), (N_1, N_2, \ldots, (N_{i-1}, N_i), (N_i, B))$ has established a link key during the initial key-setup using the common keys in the nodes' key rings. Let j be the number of such paths that are disjoint (do not have any links in common). A then generates j random values $v_1, \ldots, v_j$. A then routes each random value along a different path to B. When B has received all j keys, then the new link key can be computed by both A and B as:

$$\acute{k} = k \oplus v_1 \oplus v_2 \oplus \ldots \oplus v_j \tag{2.1}$$

The secrecy of the link key k is protected by all j random values. Unless the adversary successfully manages to eavesdrop on all j paths, they will not know sufficient parts of the link key to

reconstruct it. However, for any given path, the probability that the adversary can eavesdrop on the path increases with the length of the path. If any one link on the path is insecure then the entire path is made insecure. Further, it is increasingly expensive in terms of communication overhead to find multiple disjoint paths that are very long. This approach has the advantage that path discovery overhead is minimized and the paths are naturally disjoint and no further effort needs to be taken to guarantee this property.

Zhu et al. [22] adopted the similar mechanism which uses threshold secret sharing for key reinforcement. S_A generates a secret key $K^{\gamma}_{A,B}$, $j-1$ random shares sk_i, and $sk_j = K^{\gamma}_{A,B} \oplus sk_1 \oplus \ldots \oplus sk_{j-i}$. S_A sends the shares through j disjoint secure paths. S_B can recover $K^{\gamma}_{A,B}$ upon receiving all shares.

Di Pietro et al. [23] provide further improvements to basic scheme [42]. Keys are assigned to a node according to the output of a pseudorandom generator with a public seed and the node's ID as inputs. In Co-operative pair-wise key establishment protocol, S_A first chooses a set $C = (c_1, c_2, \ldots, c_m)$ of co-operative nodes. A co-operative node provides a hash $HMAC(k_{c_1,B}, ID_A)$. Reinforced key is then $K^{\gamma}_{A,B} = K_{A,B} \oplus HMAC(k_{c,B}, ID_A)$ where $K_{A,B}$ and $K_{C,B}$ are the established link keys. Node S_A shares set C with node S_B; therefore, S_B can generate the same key. This approach requires nodes S_A and S_B to send and receive c more messages. Moreover, cooperative nodes have to send and receive two extra messages. In addition to increased communication cost, each cooperative node has to execute $HMAC$ function twice for S_A and S_B. When a node joins the network, it discovers its neighbors' keys by just knowing its neighbors' IDs.

The key reinforcement solutions in general increase processing and communication complexity, but provide good resilience in the sense that a compromised key-chain does not directly affect security of any links in the WSN. But, it may be possible for an adversary to recover initial link keys. An adversary can then recover reinforced link keys from the recorded multi-path reinforcement

messages when the link keys are compromised.

Zhu Localized Encryption and Authentication Protocol (LEAP) is a complete key management framework for static WSNs [13]. For key deployment, each node has to store 4 kinds of keys: (1)an individual key, (2)a group key, (3)cluster keys, and (4)pairwise shared keys. In addition to these keys: a node also has to store a one-way key chain it creates, the commitments of the key chains its neighbors create, and the commitment of the base station's key chain. In key establishment phase, a node uses its individual key to encrypt messages it sends to the base station. Messages broadcast by the base station are encrypted with the group key but authenticated with μTESLA. The combination of cluster key and one-way key chain is interesting because if only the cluster key is used, a compromised neighbor would disclose the cluster key; if only the key chain is used, the keys in the key chain would have to be broadcast in clear, allowing replay attacks to take place; but if used together, the cluster key can be used to hide the keys in the key chain from cluster-outsiders, so that the keys do not need to be disclosed according to a schedule as in SPINS, and the keys in the key chain can be used for authentication as usual [14]. In order to add a new node to the network, a new node u is imprinted with an individual key for communication with the base station, the group key, the commitment of the base station's key chain, and an initial key . To evict a node, the base station broadcasts an eviction message using μTESLA.

Cheng propose an efficient pair-wise key establishment and management scheme [24]. In this approach, a two-dimensional key matrix is generated to distribute symmetric keys into sensor nodes. Each sensor randomly stores a row and a column from the matrix before the deployment. Since each row has an intersection entry with each column in the matrix, every pair of sensors would share at least two common keys between them. After the deployment, two neighboring nodes combine their shared common keys and their node identities to generate a pairwise key between them. Because all the established pairwise keys are distinct to each other, any sensor's

compromise cannot affect the secure communication between non-compromised nodes. The communication overhead for this scheme is still too high for large scale dense networks, too many keys need to be pre-loaded into sensor nodes, node addition is a complicated and energy consuming procedure.

Liu propose the Polynomial Pool Based Key Predistribution Scheme [44]that offers several efficient features the other key predistribution schemes lack, including:(1) Any two sensors can definitely establish a pairwise key when there are no compromised sensors (2) Even with some nodes compromised, the others in the network can still establish pairwise keys (3) A node can find the common keys to determine whether or not it can establish a pairwise key and thereby help reduce communication overhead. In the initialization stage, the setup server randomly generates a bivariate t degree polynomial $f(x, y)$ over a finite field F_q, where $f(x, y) = \sum_{i,j=0}^{t} a_{ij} x^i y^j$. The setup server then generates a polynomial share of the equation for every node in the sensor network; e.g., node i in the network receives an $f(i, y)$ share and node j receives an $f(j, y)$ share. If both nodes i and j want to establish a common key $f(i, j)$ between them, then node i can compute the common key by computing $f(i, y)$ at node j and then node j can compute $f(j, y)$ at node i for the common key $f(i, j)$. Advantages of this scheme include that it allows the network to grow to a larger size after deployment. Disadvantages of this scheme include t-collision resistance (compromising more than t polynomials leads to network compromise).

Hypercube Key Distribution Scheme guarantees that any two nodes in the network can establish a pairwise key if there are no compromised nodes present as long as the two nodes can communicate [45]. Also, nodes can still communicate with high probability if compromised nodes are present. Nodes can decide whether or not they can directly communicate with other nodes and what polynomial they should use when transmitting messages [45]. If the total the number of nodes in the network to N, then this scheme computes an n-dimensional hypercube with m_{n-1}

polynomials. Before node distribution, a setup server assigns each node an exclusive coordinate in a matrix. Also the setup server assigns each node a set of polynomials in which it can compute a pairwise key with other nodes for communication. To compute the initial polynomials, the setup server makes $n \times m_{n-1}$ number of polynomials over a space of F_q. Each node occupies an empty space in the matrix. If nodes a and b share a common polynomial, they can make a direct connection and compute a pairwise key to communicate. If the two nodes do not share a common polynomial they have to use the path discovery method to compute an indirect key.

Oliveira show how random key pre-distribution, widely studied in the context of flat networks, can be used to secure communication in hierarchical (cluster-based) protocols [20] such as LEACH [46]. They presented SecLEACH, a protocol for securing node-to-node communication in LEACH-based networks. A large pool of S keys and their ids are generated in pre-deployment phase of SecLEACH,. Each node is then assigned a ring of m keys drawn from the pool pseudo-randomly, without replacement. Also prior to deployment, for each node is assigned a pairwise key shared with the BS. The LEACH clustering algorithm can then be run with the given modifications: when a self-elected CH broadcasts its advertise message , it includes the ids of the keys in its key ring; the remaining nodes now cluster around the closest CH with whom they share a key. Their estimates show that the overhead incurred by SecLEACH is manageable; and memory usage, energy efficiency, and security level can be each traded off for another, depending on what is most critical in a system.

Jolly proposed a low-energy key management protocol for hierarchical WSNs [47]. In their scheme, a wireless sensor network is partitioned into several distinct clusters by some gateway nodes. Before deployment, each gateway node stores a set of keys in its memory; each sensor randomly selects a key from a gateway node and stores it with the gateway node's id in its memory. After the deployment, each sensor exchanges its key information with its cluster head, if the clus-

ter head has the key in its memory, they can establish a secure link directly. Otherwise, the cluster head request the intended key from the corresponding gateway node. This scheme provides better network performance than previous key pre-distribution schemes since a hierarchical network model is used, but it does not address the node capture attack problem which is the major threat in WSNs. If a gateway node is compromised, the adversary could track all the communications between gateway nodes. Since all the communications in the hierarchical network are relayed by the gateway nodes, the whole network would be crashed by a single gateway node's failure.

Ren proposed a new approach for random key pre-distribution to achieve both efficiency and security goals [25]. The novelty of this approach lies in that, instead of using a key pool consisting of random keys, a key generation technique is carefully designed such that a large number of random keys can be represented by a small number of key-generation keys. Then, instead of storing a big number of random keys, each sensor node stores a small number of key-generation keys while computing the shared secret keys during the bootstrapping phase on the fly using the computationally efficient hash function. This scheme outperforms the previous random key pre-distribution schemes in that it reduces the storage requirement significantly while holding the comparable security strength.

2.2.7 Key Management for Heterogeneous Sensor Networks

There is also active research in key management for heterogeneous sensor networks. Du proposes the asymmetric pre-distribution (AP) scheme for heterogeneous sensor networks consisting of a small number of high end sensor(H-sensor) and a large number of low end sensors(L-sensor) [48] . Further, they assumed that H-sensors are equipped with temper resistant hardware and are utilized to provide simple, efficient and effective key set up schemes for L-sensors. The AP scheme provides better security with low complexity and significant reduction on storage requirement by storing more number of pre-configured keys on H-sensors and small number of pre-configured

keys on L-sensors.

Bulusu proposes two key predistribution based scheme for heterogeneous networks which consist of nodes which are stationary as well as highly mobile [49] . The first scheme uses a separate key pool for links between mobile and static nodes. From this key pool the mobile and stationary nodes randomly select $m$keys and $e(e << m)$ keys respectively. Having fewer keys in the stationary nodes ensures that the capture of a stationary node compromises a small fraction of the mobile key pool. The second scheme uses a large key pool which is segmented into smaller key pools. All the nodes of a particular stationary network select m keys randomly from one of the small segments whereas the mobile nodes select m nodes from the entire key pool. It is ensured that the probability that a mobile node would have some keys from each of the segments is high. The advantage of using a separate disjoint key pool to establish links between the stationary and mobile nodes of the network is that the compromise of keys in one network would not lead to compromise of keys in all the networks. The main disadvantage of key pre-distribution with separate key pools scheme is that the mobile key pool must be known before the deployment of stationary nodes.The main disadvantage of key predistribution with segmented key pools scheme is that it is not scalable if the number of networks becomes high.

Traynor demonstrate that a probabilistic unbalanced distribution of keys throughout the network that leverages the existence of a small percentage of more capable sensor nodes can not only provide an equal level of security but also reduce the consequences of node compromise [50]. In [51] they characterize the effects of the unbalanced key management system, and design a complementary suite of key establishment protocols known as LIGER, a hybrid key management system for heterogeneous sensor networks, as extension of the work done in [50]. Similar to [48, 50], it is assumed that there are nodes in the network that are more powerful and more secure than others, and these more powerful nodes are also in tamper proof boxes or well guarded. A node that has

limited memory and processing power is identified as $L1$ and a node that have more memory and more processing power is identified as $L2$. L2 nodes act as head nodes for the L1 nodes and have the responsibility of routing packets throughout the network. These L2 nodes have access to gateway servers which are connected to a wired network. All nodes are loaded with a random set of keys drawn from a common pool before being deployed. In addition, the mapping of keys to nodes is stored in a KDC. An architecture is created that allows networks to operate securely in the absence and presence of a key distribution center(KDC). In the absence of a KDC, a stand-alone protocol, referred to as LION is used to implement the theory in many papers based on probabilistic keying, differ from most of the previously defined systems by supporting optimizations that allow keys to be deployed in an unbalanced manner, i.e., more keys are deployed in more capable nodes (L2). The scheme relying upon the presence of the KDC is referred to as TIGER. If the network has access to a KDC, the knowledge of the pre-deployed keys is used to perform probabilistic authentication with a high degree of confidence. In addition, session keys are established with the enforcement of least privilege. Nodes gather information in this mode of operation so that they may continue to perform some level of probabilistic authentication if the KDC becomes unavailable for periods of time. The mode of operation may change between stand-alone and KDC-mode.The combination of slightly modified versions of these two schemes results in LIGER - a more robust method of key management for heterogeneous sensor networks. The combination enables different levels of probabilistic authentication without increasing memory requirements of the L1 sensor nodes. Lu proposes a framework for key management schemes in distributed wireless sensor networks with heterogeneous sensor nodes [52].

2.3 Previous Work: Secure Group Communication in Wireless Sensor Networks

Recently there have been several proposals to address the secure group communication issues. The most known technique is the construction of a logical key tree where group members are

associated with leaves and each member is given all the keys from his leaves to the root, as proposed in [53, 54, 55, 56], where root key is the group key. This approach allows reducing the communication cost for key update, on the event of group membership change, to $O(logM)$ where M is the number of group members.

Several extensions are proposed to deal with reliability [57], node dependent group dynamics [58], and time variant group dynamics [59]. Extensions to wireless networks are discussed in [60] and several secure multicast protocols are proposed in [61] [62].

Park proposes a lightweight security protocol(LiSP) for efficient rekeying in dynamic groups [63]. LiSP utilizes broadcast transmission to distribute the group keys and uses one-way key chains to recover from 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.

Wong proposes three rekeying strategies: user-oriented, key-oriented, and group-oriented and specify join/leave protocols based upon these rekeying strategies [64]. A user u who wants to join (leave) a secure group sends a join (leave) request to the key server, denoted by s. For a join request from user , it is assumed that group access control is performed by server using an access control list provided by the initiator of the secure group.6 A join request initiates an authentication exchange between and . If user u is not authorized to join the group, server s sends a join-denied reply to u . If the join request is granted, it is assumed that the session key distributed as a result of the authentication exchange will be used as the individual key K_u of u. The group rekeying, which relies only on current rekeying message and the node's initial configuration. A non-revoked node can decrypt the new session keys independently from the previous rekeying messages without contacting the GC, even if the node is off-line for a while. They use keys of multiple granularity to reduce the rekeying overhead associated with membership management.

Carman gives a comprehensive analysis of various group key schemes and find that the group size is the primary factor that should be considered when choosing a scheme for generating and distributing group keys in a WSN [65].

Staddon proposes a self-healing group key distribution scheme based on two-dimension t-degree polynomials [66]. Liu further improves the work in [66] by reducing the broadcast message size in situations where there are frequent but short-term disruptions of communication, as well as long-term but infrequent disruptions of communication [44]. Blundo also presents a design of self-healing key distribution schemes which enables a user to recover from a single broadcast message where all keys are associated with sessions where it is a member of the communication group [67].

Jiang proposes a key distribution scheme with time-limited node revocation based on dual directional hash chains for WSNs [68]. Dutta proposes two constructions for self-healing key distribution based on one-way hash key chains with t revocation capability using polynomial based node revocation [69].

2.4 Conclusion

In this chapter the idea of WSNs is being put in the broader perspective. The particular characteristics of sensor node and its hardware constraints are reviewed and discussed. The wireless sensor network model, its communication architecture and number of application scenarios are also given. In addition many Key management protocols for WSNs have been proposed in literature are discussed. It is shown that they suffer from one or more of the problems of weak security guarantees if some nodes are compromised, lack of scalability, high energy overhead for key management, and increased end to end data latency. In general key pre-distribution protocols expose the security of the whole network when a certain fraction of nodes is compromised. Secure group communication in wireless sensor networks is also discussed. Based on the analysis of

above schemes, it is concluded that there are significant tradeoffs and, there is no one-size-fits-all solution for key distribution problems in WSNs.

Chapter 3

SECURE AND EFFICIENT KEY MANAGEMENT SCHEME FOR HETEROGENEOUS SENSOR NETWORKS

3.1 Introduction

Cryptography is the foundational technology used for protecting and securing the communication in sensor networks [70]. This technology relies on keys as the centerpieces, and many attacks focus on disclosing these keys. As a result, the management of the keys (the process by which keys are generated, stored, protected, distributed, used, and destroyed) is a very important and challenging problem in a large-scale network consisting of several hundreds or thousands of sensor nodes.

Conventional security protocols are usually master key based or distributed key based management schemes. In master key based schemes, every node shares a single pre-loaded master key. Further, master key is used to negotiate session keys for securing different wireless links. For example, Menezes uses a simple three way handshaking and authentication protocol based on the master-key for setting up session keys [71]. This type of key management scheme has the underlying assumption that the sensor nodes are tamper proof and the master key that is stored inside each node cannot be retrieved by the adversary. However, the assumption that the nodes are tamper proof cannot be ensured in many sensor network applications because sensor nodes are usually left unattended in a hostile and remote environments. Once the master key is compromised, the adversary can use it to break the security of the entire network.

Other commonly used schemes in WSNs are key pre-distribution schemes [19, 20, 42, 21, 22,

23, 24, 25]. In these approaches, with minimal resources, one can achieve a known probability of connectivity within a network. These efforts assume a deployment of homogeneous nodes, and therefore use a balance distribution of random keys among the nodes. Most existing research mainly considers homogeneous sensor networks, where all sensor nodes have identical capabilities in terms of communication, computation, sensing, and reliability; however, homogeneous WSNs are not scalable.

Several recent works, on the other hand, investigate heterogeneous sensor networks (HSNs). Girod develops tools to support heterogeneous systems as well as the measurement and visualization of operational systems [72]. Lazos studies the coverage problem in planar heterogeneous sensor networks and formulate the coverage problem as a set intersection problem [73]. They formulate expressions in order to determine the required number of sensors for a field of interest. Ma proposes a resource oriented protocol for heterogeneous sensor networks to build the network model that adapts according to the members' resources [74]. Du proposes a differentiated coverage algorithm which can provide different coverage degrees for different areas; the algorithm is energy efficient since it only keeps minimum number of sensors in active state [75]. Melo analyze the energy consumption and lifetime of HSN by providing periodic data from a sensing field to a remote receiver [76].

In this chapter, a scalable and efficient protocol for key management is proposed that is sensitive to the sensor nodes resource constraints, including storage, computation and communication. The proposed key management scheme is based on random key pre-distribution for HSNs; As large-scale homogeneous networks suffer from high costs of communication, computation, and storage requirements, the HSNs are preferred because they provide better performance and security solutions for scalable applications in dynamic environments. The propose scheme consider heterogeneous sensor network consisting of two types of sensors: high-end ($\mathcal{H}$-node) and low-end

($\mathcal{L}$-node). Further, for scalable solutions, the proposed scheme uses hierarchical structure, where $\mathcal{H}$-node act as CH and $\mathcal{L}$-nodes as cluster members.

To address storage overhead problem in the constraint sensor nodes, a key generation process is incorporated, where instead of generating a large pool of random keys, a key pool is represented by a small number of generation keys. For a given generation key and a publicly known seed value, a keyed-hash function generates a key chain; these key chains collectively make a key pool. Further, each sensor node is assigned a small number of randomly selected generation keys. As a result, by using generation keys, the proposed scheme significantly reduces the storage requirements.

Dynamic network topology is native to WSNs because nodes can fail or be added. As result, the proposed scheme allows dynamic node addition and removal. In the case of node addition, the proposed scheme is able to distinguish between legitimate and malicious nodes. Further, as adversaries can compromise sensors and acquire all security information, a re-keying scheme is incorporated to update all types of keys periodically.

This chapter also reports the implementation and the performance of the proposed scheme on Crossbow's MicaZ motes running TinyOS where it consider the computation and storage costs of two keyed-hash algorithms for key chain generation, HMAC-SHA1 and HMAC-MD5.The results indicate that the proposed scheme can be applied efficiently in resource-constrained sensor networks.

3.2 Network Model

The heterogeneous sensor network is considered, consisting of three types of nodes: base station, $\mathcal{H}$-node, and $\mathcal{L}$-node.

Base Station: The BS is assumed to be secure, not prone to failures, and does not have any resource constraints such as bandwidth, energy, memory, and processing.

$\mathcal{H}$-Nodes: $\mathcal{H}$-nodes have more memory and processing capability. These nodes are equipped with tamper resistant hardware and communicate directly with the BS. Although $\mathcal{H}$-nodes have rich resources, but these are still limited as compared to the BS. For instance, Crossbow's stargate nodes can be used as $\mathcal{H}$-nodes.

$\mathcal{L}$-Nodes: $\mathcal{L}$-nodes are ordinary sensor nodes that are limited in terms of memory and processing capability. The $\mathcal{L}$-nodes acquire data from the surrounding environment and forwards the collected data to the $\mathcal{H}$-node. Further each $\mathcal{L}$-node (and $\mathcal{H}$-node) is aware of its own location.

In a typical initial HSN deployment, there would be a small number of H sensors and a large number of $\mathcal{L}$-nodes. Further, for scalability and easy maintenance, both $\mathcal{H}$-nodes and $\mathcal{L}$-nodes can be added as needed. $\mathcal{H}$-nodes and $\mathcal{L}$-nodes are assumed to be uniformly and randomly distributed in the field. Clustering of sensors enable local data processing, which reduces communication load in the network in order to provide scalable solutions.

HSN consists of a hierarchical structure where sensors are divided into clusters and each cluster is managed by a CH, as shown in Figure 3.2. All $\mathcal{H}$-nodes act as CHs; whereas each $\mathcal{L}$-node is a cluster member and cannot act as a CH.

Most traffic in HSN can be classified as many-to-one, one-to-one and local communication. In many-to-one communication, multiple $\mathcal{H}$-nodes and $\mathcal{L}$-nodes send sensor readings to a BS or aggregation point in the network. In one-to-many communication, a single node (either a BS or $\mathcal{H}$-nodes) multicasts or floods a query or control information to several $\mathcal{L}$-nodes. In local communication the neighboring $\mathcal{L}$-nodes and $\mathcal{H}$-nodes send localized messages to discover and coordinate with each other. A node may broadcast messages intended to be received by all neighboring nodes or unicast messages intended for a only single neighbor.

Each $\mathcal{L}$-node is able to securely communicate with all other $\mathcal{L}$-nodes in its neighborhood and its CH (H-senor). Moreover, $\mathcal{H}$-nodes maintain secure communication with following entities: BS,

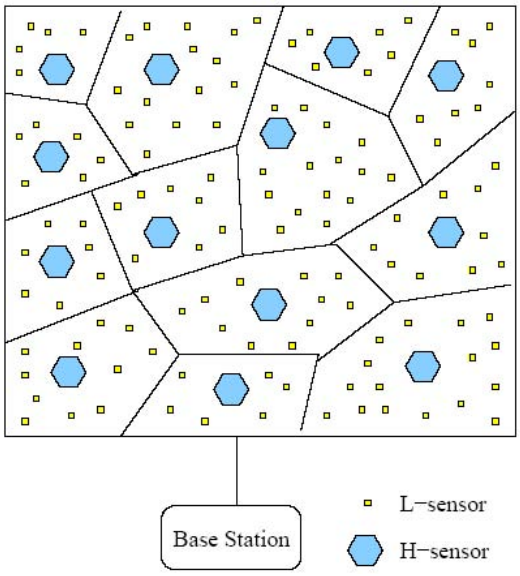


Figure 3.1: Network Model

cluster member ($\mathcal{L}$-node) and other CHs ($\mathcal{H}$-nodes).

3.2.1 Threat Model

A malicious node can be either an external node that does not know the cryptographic keys, or an internal node ($\mathcal{L}$-node), that possesses the keys. An adversary can create an internal compromised node by capturing a legitimate $\mathcal{L}$-node. All these malicious nodes can exhibit Byzantine behavior which can be described as a behavior when one or more sensors or devices work in collusion to disrupt the network. It could include several security challenges such as denial of service attack, dropping or altering packets, topology distortion, impersonation, and wormholes.

3.3 Preliminaries

A few terms and definitions used in the rest of the thesis are given in appendix 1.

3.4 Proposed Scheme

This section presents an efficient key management scheme designed for heterogeneous sensor networks. The proposed scheme uses a symmetric-key mechanism to distribute, revoke, and renew keys during the lifetime of HSN.

3.4.1 Key Pre-Distribution Phase

The key pre-distribution phase includes key pool generation and key ring assignment.

3.4.1.1 Key Pool Generation

This section describes the process of key pool generation. Generate a large pool of random symmetric keys as follows. First, the cardinality (size) of key pool $|K|$ is selected. Then, the number of key chains M is chosen accordingly. A key pool K consists of M different key chains, as given in Equation 1. Further, there are no common keys between any two key chains, which is formally given as follows:

$$C_i \cap C_j = \phi, \forall i \neq j \tag{3.1}$$

As a key chain C_i is generated independently via a unique generation key gk_i and publicly known seed S by applying a keyed hash algorithm repeatedly [25], the j-th key of the key chain C_i is computed as:

$$k_{C_{i,j}} = \begin{cases} HMAC(gk_i, S) & : \quad j = 0 \\ HMAC(gk_i, k_{C_{i,j-1}}) & : \quad 1 \leq j \leq N-1 \end{cases} \tag{3.2}$$

Figure 3.2 shows a block diagram to illustrate the process of key generation. The first key is generated by using seed S and gk_i as inputs to keyed hash function (HMAC); however, the remaining keys are generated by applying HMAC over gk_i and the previous key. The total number of keys in a key chain is N, where

$$N = \frac{K}{M} \tag{3.3}$$

Further, BS generates a special key K_M known as master key, which is used for authentication.

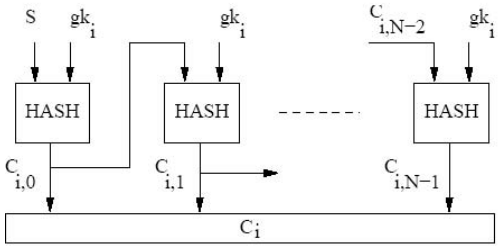


Figure 3.2: Key Chain Generation Process

3.4.1.2 Key Ring Assignment

For each node ($\mathcal{L}$-node or $\mathcal{H}$-node), a unique identity (id) is generated using a pseudorandom function (PRF). Before deploying the nodes, each node is loaded with its assigned key ring R, where R consists of the number of generation keys used to generate corresponding key chains. According to the assigning rules, each $\mathcal{L}$-node is assigned r number of randomly selected generation keys of corresponding key chains. First, input the $\mathcal{L}$-node id as seed to pseudorandom number generator (PRNG) of a large enough period to produce a sequence of r numbers, as given in Equation 3.4.

$$PRNG(id_i) = n_1, n_2, \ldots n_r \tag{3.4}$$

Second, the set of key ids assigned to $\mathcal{L}$-node, can then be obtained by mapping each number in the sequence to its correspondent value modulus M, as given in Equation 3.5.

$$id_{gk_i} = n_i \; mod \; M \tag{3.5}$$

where $0 \leq id_{gk_i} \leq M - 1$. From these r generation keys, $r \times N$ random keys can be calculated effectively. In addition, each $\mathcal{L}$-node is preloaded with an authentication key K_{M,L_i}, which is generated by applying one way hash function on the id of $\mathcal{L}$-node and master key i.e. $K_{M,L_i} =$

$H(K_M, id_{L_i})$.

Each $\mathcal{H}$-node is preloaded with S randomly selected generation keys of corresponding key chains as described above. However, it should be noted that $S >> r$. Each $\mathcal{H}$-node is also preloaded with a master key K_M.

3.4.2 Cluster Formation Phase

During the cluster formation phase, all $\mathcal{H}$-nodes broadcast Hello messages to nearby $\mathcal{L}$-nodes with some random delay, in order to avoid collisions of Hello messages from neighboring $\mathcal{H}$-nodes.The probability of collision is quite small when a non-persistent CSMA protocol is used for medium access control [77]. Moreover, an $\mathcal{H}$-node can broadcast its ID multiple times to increase the probability so that is received by all its neighbors. The Hello message includes the ID of the $\mathcal{H}$-node. The transmission range of the broadcast is large enough so that all $\mathcal{L}$-nodes can receive Hello messages from several $\mathcal{H}$-nodes. Then, each $\mathcal{L}$-node selects the $\mathcal{H}$-node as the CH whose Hello message has the best received signal strength indicator(RSSI) value. Each $\mathcal{L}$-node also records the ids of other $\mathcal{H}$-nodes from which it receives the Hello messages, and these $\mathcal{H}$-nodes are listed as backup CHs in case the primary CH fails. Only $\mathcal{H}$-node can act as a CH; whereas the $\mathcal{L}$-nodes act as cluster members; the details of clustering scheme can be found in [78].

3.4.3 Cluster Head based Shared Key Discovery Phase

The shared key discovery phase begins after cluster formation phase. First, each cluster member sends to its CH a message, which includes its ID, nonce, its neighboring nodes information, and MAC which is calculated on all these values using a key K_{M,L_i}.

Second, this phase also includes a neighborhood discovery, as shown in Figure 3.3. In message 1, $\mathcal{L}$-node (L_i) broadcasts hello messages for a short range in order to discover neighbors. In

message 2, one of L_i's neighbor, say L_j acknowledges with $HelloReply$ message. Then, L_i adds L_j's id in its neighbors list. The neighborhood discovery phase ends when all the $\mathcal{L}$-nodes have obtained neighborhood information.

1	$L_i \Rightarrow * : Hello(id_{L_i})$
2	$L_j \Rightarrow L_i : HelloReply(id_{L_j})$
3	$L_i : adds\ the\ id_{L_j}\ into\ List$
4	$L_j : Repeat\ 2\ and\ 3\ for\ every\ HelloReply$

Figure 3.3: Neighboring Node Discovery

Third, CH discovers the shared generation keys between neighboring $\mathcal{L}$-nodes in its cluster, as shown in Figure 3.4.

1	$L_i \Rightarrow H_a : id_{L_i}, nonce_{L_i}, List, MAC_{K_{M,L_i}}(id_{L_i} \| nonce_{L_i} \| List)$
2	$L_j \Rightarrow H_a : id_{L_j}, nonce_{L_j}, List, MAC_{K_{M,L_j}}(id_{L_j} \| nonce_{L_j} \| List)$
3	$H_a \Rightarrow L_i : n, id_{gk_m}, id_{L_i}, id_{L_j}, MAC_{K_{M,L_i}}(n \| id_{gk_m} \| id_{L_i} \| id_{L_j} \| nonce_{L_i})$
4	$H_a \Rightarrow L_j : n, id_{gk_m}, id_{L_i}, id_{L_j}, MAC_{K_{M,L_j}}(n \| id_{gk_m} \| id_{L_i} \| id_{L_j} \| nonce_{L_j})$

Figure 3.4: Neighboring $\mathcal{L}$-nodes with Common Preloaded Generation Key

In messages 1 and 2, $\mathcal{L}$-nodes L_i and L_j send messages to CH H_a, where messages contain their ids, nonce, the list of their neighboring $\mathcal{L}$-nodes ids, and MAC on all these values. H_a determines the generation keys in L_i and L_j's key rings (R_{L_i} and R_{L_j}) by using the pseudorandom scheme described above in (Section 3.4.1). The CH chooses the common generation key gk_m, where $gk_m \in R_{L_i} \cap R_{L_j}$; a generation key with minimum index is selected in case of multiple common keys. Then, H_a determines the shared pairwise key between L_i and L_j by generating a random number n, where $[0 \leq n \leq N-1]$, which is used as an index in the key chain C_m for selecting the pairwise shared key, i.e. $K_{C_{m,n}}$. Then, H_a disseminates the shared-key information to L_i and L_j sensors, as shown in messages 3 and 4. The shared-key information consists of the following: a) ids of neighboring $\mathcal{L}$-nodes (L_i and L_j), b) id of the common generation key, i.e. id_{gk_m}, c) n that represents the index of shared pairwise key of C_m key chain, d) nonce, and e) MAC that

is calculated on all these values using corresponding authentication keys K_{M,L_i} and K_{M,L_j}. In other words, $\mathcal{L}$-nodes L_i and L_j share n-th key of C_m key chain by applying the common key generation algorithm shown in Figure 3.5 on n and gk_m. Further, $\mathcal{L}$-nodes L_i and L_j also share the same n-th key of C_m key chain with their CH.

1: function GetKey (n, gk_m)
   /* param n: random number */
   /* param gk_m: common generation key */
2: $K_{C_{m,0}} = HMAC(gk_m, seed)$
3: for i=1 to n-1
4: $\quad K_{C_{m,i}} = HMAC(gk_m, K_{C_{m,i-1}})$
5: end for
6: return $K_{C_{m,n-1}}$

Figure 3.5: Session Key Generation

3.4.3.1 No Common Preloaded Generation Key between $\mathcal{L}$-nodes

Some $\mathcal{L}$-nodes may not share any preloaded generation key with their neighbors. For each pair of $\mathcal{L}$-nodes (say X and Y) that do not share any generation key, $R_x \cap R_y = \phi$, CH H_a obtains a shared-key between H_a and L_x and a shared-key between H_a and L_y. Then, H_a generates a pair-wise key for each pair (L_x and L_y), and securely sends the key to them.

Figure 3.6 shows an example for neighboring $\mathcal{L}$-nodes that do not share common preloaded generation key. H_a first checks if it has a preloaded generation key shared with the $\mathcal{L}$-nodes (e.g., L_x and L_y), $R_{L_x} \cap R_{H_a} \neq \phi$ and $R_{L_y} \cap R_{H_a} \neq \phi$. As H_a is preloaded with a large number of generation keys, there is a high probability that H_a can find at least one shared generation key with L_x and L_y, i.e. $gk_i \in R_{L_x} \cap R_{H_a}$ and $gk_j \in R_{L_y} \cap R_{H_a}$. H_a generates random numbers p, where $[0 \leq p \leq N-1]$ and q where $[0 \leq q \leq N-1]$. H_a sends messages 3 and 4, which means that H_a shares the p-th key ($K_{C_{i,p}}$) of C_i key chain with node L_x and q-th key ($K_{C_{j,q}}$) of C_j key chain with node L_y. Then, H_a generates a new shared key between L_x and L_y and sends this key to both L_x and L_y, encrypting with shared key between nodes (L_x, L_y) and H_a, as shown

in messages 5 and 6.

1	$L_x \Rightarrow H_a : id_{L_x}, nonce_{L_x}, List, MAC_{K_{M,L_x}}(id_{L_x} \| nonce_{L_x} \| List)$
2	$L_y \Rightarrow H_a : id_{L_y}, nonce_{L_y}, List, MAC_{K_{M,L_y}}(id_{L_y} \| nonce_{L_y} \| List)$
3	$H_a \Rightarrow L_x : id_{L_x}, id_{g_i}, p, MAC_{K_{M,L_x}}(id_{L_x} \| id_{g_i} \| p \| nonce_{L_x})$
4	$H_a \Rightarrow L_y : id_{L_y}, id_{g_j}, q, MAC_{K_{M,L_y}}(id_{L_y} \| id_{g_j} \| q \| nonce_{L_y})$
5	$H_a \Rightarrow L_x : \left\langle K_{L_x,L_y} \right\rangle_{K_{H_a,L_x}}$
6	$H_a \Rightarrow L_y : \left\langle K_{L_x,L_y} \right\rangle_{K_{H_a,L_y}}$

Figure 3.6: Neighboring $\mathcal{L}$-nodes without Common Preloaded Generation Key

3.4.3.2 No Common Preloaded Generation Key Between CH and $\mathcal{L}$-node

In case that an $\mathcal{L}$-node does not share any preloaded generation key with its CH, then CH finds a key to communicate securely with that $\mathcal{L}$-node. CH H_a generates a key K_{H_a,L_i} and sends the key to $\mathcal{L}$-node encrypted with that $\mathcal{L}$-node's authentication key K_{M,L_i}, as given in Equation 3.6

$$H_a \rightarrow L_i : \langle K_{H_a,L_i} \rangle_{K_{M,L_i}} \tag{3.6}$$

3.4.4 Inter-cluster Communication

An inter-cluster communication between CHs (say H_i and H_j) is achieved through a key K_{H_i,H_j} generated by applying hash function on id_{H_i} and id_{H_j} using key K_M, as given in Equation 3.7.

$$K_{H_i,H_j} = HMAC(K_M, id_{H_i} \| id_{H_j}) \tag{3.7}$$

If node L_a wishes to communicate with a node that lies in a different cluster, then two CHs are involved in order to setup a session key between $\mathcal{L}$-nodes. Say L_a lies in Cluster i and L_b in j and the respective CHs are H_i and H_j. As shown in Figure 3.7, in line 1, L_a sends request to H_i consisting of its id (id_{L_a}), the id of $\mathcal{L}$-node with which it wants to communicate (id_{L_b}), a nonce, and MAC that is calculated on all these values using K_{M,L_a}. H_i determines the common

generation key (gk_m) in the key rings of $\mathcal{L}$-nodes (L_a and L_b). Then H_i generates a random number n and sends the shared key message to L_a consisting of both $\mathcal{L}$-nodes ids (id_{L_a}, id_{L_b}), id of the common generation key (id_{gk_m}),n, $nonce_{L_a}$ and MAC that is calculated by using K_{M,L_a}, as shown in line 2 of Figure 3.7. Similarly, in line 3, H_i sends a message to H_j containing the shared key information between L_a and L_b. After receiving this message, H_j forwards the shared key message to L_b consisting of ids of both L_a and L_b, id of the common generation key, n, and MAC that is calculated on all these values using K_{M,L_b}, as shown in line 4. Now $\mathcal{L}$-nodes L_a and L_b use n-th key of C_i key chain to communicate securely.

1:	$L_a \Rightarrow H_i : id_{L_a}, id_{L_b}, nonce_{L_a}, MAC_{K_{M,L_a}}(id_{L_a} \| id_{L_b} \| nonce_{L_a})$
2:	$H_i \Rightarrow L_a : id_{L_a}, id_{L_b}, id_{gk_i}, n, MAC_{K_{M,L_a}}(id_{L_a} \| id_{L_b} \| id_{gk_i} \| n \| nonce_{L_a})$
3:	$H_i \Rightarrow H_j : id_{L_a}, id_{L_b}, id_{gk_i}, n, MAC_{K_M}(id_{L_a} \| id_{L_b} \| id_{gk_i} \| n)$
4:	$H_j \Rightarrow L_b : id_{L_a}, id_{L_b}, id_{gk_i}, n, MAC_{K_{M,L_b}}(id_{L_a} \| id_{L_b} \| id_{gk_i} \| n)$

Figure 3.7: $\mathcal{L}$-node to $\mathcal{L}$-node Inter-cluster Communication

3.4.5 Addition of New Nodes

A desirable property in a scalable key management scheme is the ability of adding new sensors to the network. These newly deployed sensor nodes need to establish secret key with existing nodes. However, before adding new nodes into network, it should be ensured that the newly deployed sensor node is not an adversary node. The proposed scheme is robust for adding new legitimate $\mathcal{L}$-nodes in the network. After an $\mathcal{L}$-node L_x is deployed in the network, L_x determines its neighbors using neighboring node discovery as described in Figure 3.3. L_x sends join request to the CH (say H_a), for which it has the best RSSI and LQI values, as shown in Figure 3.8. H_a authenticates the node L_x by verifying the MAC. If authenticated, H_a determines the shared key for each of L_x's neighbors and unicasts the shared key message to L_x and its neighbors.

$$\boxed{L_x \rightarrow H_a : \quad id_{L_x}, nonce_{L_x}, List, MAC_{K_{M,L_x}}(id_{L_x} \| nonce_{L_x} \| List)}$$

Figure 3.8: New Node Addition

3.4.6 Setting up Cluster Key

Cluster key is used by both CH and cluster members to securely broadcast messages within a cluster. After setting up shared pairwise key between cluster members, CH (say H_a) generates a cluster key CK_a, which is sent to each cluster member, where CK_a is encrypted with the corresponding shared key between CH and the cluster member. For example, CH H_a sends to L_u (cluster member) the message shown in Equation 3.8,where K_{H_a,L_u} is the shared key between $\mathcal{L}$-node L_u and CH H_a.

$$H_a \rightarrow L_u : \langle CK_a \rangle_{K_{H_a,L_u}} \tag{3.8}$$

3.4.7 Key Revocation

Revocation procedures are involved after detecting compromised or faulty nodes. The BS is responsible for monitoring sensor behavior and detecting a sensor failure or compromise. For a compromised node, the BS sends this information to the corresponding CH. The CH (say H_a) broadcasts to its member the $Revocation$ message containing the list of key ids to be revoked, where the message is signed with CK_a. The $Revocation$ message is formed as shown in Equation 3.9

$$list(id_{gk_1}, id_{gk_2}, \ldots id_{gk_r}), MAC_{CK_a}(list) \tag{3.9}$$

Each $\mathcal{L}$-node when receives a $Revocation$ message, it verifies the MAC to check the integrity of message and to locate those key ids in its key ring, and remove the keys (if any). After key revo-

cation, some links may disappear and affected nodes need to reconfigure those links by restarting the shared key discovery phase.

3.4.8 Re-keying

Using the same encryption key for extended duration may result in a cryptanalytic attack. A remedy could be to ignore this threat because it is anticipated that in most cases the lifetime of nodes would be less than the lifetime of the shared key between two nodes [14]. However, in some cases, since it is possible that lifetime of keys expires, it is necessary to renew the encryption keys (session keys). In order to accomplish the renewal of the session keys, the affected nodes remove the expired keys and restart the shared key discovery phase with CHs.

3.5 The Routing Structure in HSN

From the routing point of view, hierarchy can significantly increase the routing efficiency by reducing the number of nodes involving in the routing. Thus the network throughput can be increased (for given radio link capabilities), data packet delay can be improved, and routing overhead can be reduced.

Routing in HSN consists of two phases: 1) Intra-cluster routing: Each cluster member ($\mathcal{L}$-node) sends data to its CH (a $\mathcal{H}$-node); and 2) Inter-cluster routing: Each CH receives messages from the cluster members($\mathcal{L}$-nodes) and transmits the aggregated messages to a distant BS via the $\mathcal{H}$-node backbone.

The intra-cluster routing scheme in [79] determines how to route packets from a $\mathcal{L}$-node to its CH. When a $\mathcal{L}$-node sends a packet to its CH (say H_a), the packet is forwarded by other $\mathcal{L}$-nodes in the cluster. The basic idea is to let all $\mathcal{L}$-nodes (in a cluster) form a tree rooted at the CH H_a. It has been shown in [75] that: (1) If complete data fusion is conducted at intermediate nodes, (i.e., two k-bit packets come in, and one k-bit packet goes out after data fusion) then a minimum

spanning tree (MST) consumes the least total energy in the cluster. (2) If there is no data fusion within the cluster, then a shortest-path tree (SPT) consumes the least total energy. (3) For partial fusion, it is a NP complete problem of finding the tree that consumes the least total energy.

For sensor networks where data generated by neighbor sensors are highly correlated (e.g., two k-bit packets are aggregated to one m-bit packet, where m is close to k), a MST may be used to approximate the least energy consumption case. To construct a MST, each $\mathcal{L}$-node sends its location information to the CH H_a, and then H_a can run a centralized MST algorithm to construct the tree. After constructing the MST, H_a can disseminate the tree structure (parent-child relationships) to all $\mathcal{L}$-nodes using one or more broadcasts.

For example, a pair (L_u, L_v) can be used to denote that $\mathcal{L}$-node L_u is L_vs parent node. If the cluster is small, one broadcast message can include all the pairs. If the cluster is large, then it can be divided into several sections and $\mathcal{H}$-nodes can notify $\mathcal{L}$-nodes in each section by one broadcast. Note that the broadcast from a CH needs to be authenticated. Otherwise, an adversary may broadcast malicious messages and disrupt the dissemination of routing information. The cluster key is used for authenticating broadcasts from $\mathcal{H}$-nodes. When $\mathcal{H}$-nodes broadcasts the routing structure information (e.g., the MST) to $\mathcal{L}$-nodes, a MAC is calculated over the message using cluster key CK_i of that cluster. Each $\mathcal{L}$-node can verify the MAC by using cluster key, and thus authenticate the broadcast.

For sensor networks where the data from neighbor sensors have little correlation, a SPT can be constructed; using either centralized or distributed algorithms.

Since $\mathcal{L}$-nodes are small, unreliable devices and may fail overtime, robust and self-healing routing protocols are critical to ensure reliable communications among $\mathcal{L}$-nodes. During the tree setup, the MST or SPT algorithm can find more than one parent nodes for each $\mathcal{L}$-node. One parent node serves as the primary parent, and other parent nodes serve as backup parents. In case the primary

parent node fails, a $\mathcal{L}$-node uses a backup parent for routing.

After the routing tree (an MST or an SPT) is constructed, the following secure data forwarding scheme is used by $\mathcal{L}$-nodes. Assume that $\mathcal{L}$-node L_u sends data packets to its parent L_v as shown in Equation 3.10, where the data is encrypted with the shared-key K_{L_u,L_v} and $packet_{ID}$ (not encrypted) is a local ID assigned by the sender L_u.

$$L_u \rightarrow L_v : packet_{ID} + \langle Data \rangle_{K_{L_u,L_v}} + MAC_{K_{L_u,L_v}}(v, ...) \tag{3.10}$$

The $packet_{ID}$ is used by L_u to monitor packet transmission from L_v to next node. A MAC is appended at the end of the packet to detect any modification. The input to the MAC is everything before the MAC. Node L_v sends the packet to its parent node in the tree. To guarantee the delivery, each $\mathcal{L}$-node is responsible for confirming that its successor has successfully forwarded the packet. This may be implemented by the transmitter monitoring the packet just sent out to the next node and overhearing if that node has passed it on within a time period using the $packet_{ID}$ field. The acknowledgment scheme reduces the impact of channel or node error and can detect selective forwarding attack. If L_u does not get an acknowledgment within a certain time period, L_u will re-transmit the packet to L_v. If the transmission to L_v fails again, L_u will send the packet to a backup parent node. The process continues until the data packet reaches the CH H_A.

3.5.1 Fault Tolerance

Our approach should support the ability to allow $\mathcal{L}$-nodes to change the cluster even after the initial deployment. This research develops a new solution capable of handling the change in network topology based on node mobility. Suppose an $\mathcal{L}$-node L_x moves from cluster a to cluster b. So it needs to find a cluster key CK_b, shared pairwise key with CH H_b and also the shared pairwise keys with its new neighboring $\mathcal{L}$-nodes.

So the L_x sends the join request to H_b consisting of its id, id of the previous CH (id_{H_a}),a nonce, list of its new neighbors and MAC is calculated on all these values using its authentication key as shown in message 1 of Figure 3.9 . The CH (H_b) authenticates the node L_x by verifying the MAC. If authenticated, CH determines the shared key for each of L_x's neighbors and unicasts the shared key message to L_x and its neighbors. H_b will also send the message 2 shown in Figure 3.9 to H_a reporting the change in the topology of cluster a because of the mobility of L_x. H_a verifies the message by recalculating the MAC and then unicasts the topology change information to $\mathcal{L}$-nodes which were either parent or children of L_x.

1: $L_x \rightarrow H_b : id_{L_x}, id_{H_a}, nonce_{L_x}, List, MAC_{K_{M,L_x}}(id_{L_x} \| nonce_{L_x} \| List)$
2: $H_b \rightarrow H_a : id_{L_x}, nonce_{H_a}, MAC_{K_{H_a,H_b}}(id_{L_x} \| nonce_{H_a})$

Figure 3.9: Message Transfer between H-node and L-node

3.6 Performance Evaluation

In this section, the proposed key distribution scheme is compared with other commonly used key distribution techniques. The results show that the proposed scheme can significantly reduce the storage requirements, while providing similar probability of key sharing among nodes.

The key pool size $\|K\|$ is a critical parameter because in random key distribution schemes the amount of storage reserved for keys in each node is likely to be a preset constraint, which makes the size of the key ring $\|R\|$ a fixed parameter. Once R is set then for larger values of $\|K\|$ the probability that two L-nodes will share a key is small. Further, the probability that a randomly chosen link is compromised when a node that is at neither end of the compromised link decreases by increasing the value of $\|K\|$. There is a need to find the largest key pool size $\|K\|$, such that the probability of key sharing between two $\mathcal{L}$-nodes, as well as $\mathcal{L}$-node and $\mathcal{H}$-node is not less then the threshold p.

Let p be the probability that an $\mathcal{L}$-node and $\mathcal{H}$-node share at least one common key in their key

ring. The number of possible key ring assignments for an $\mathcal{L}$-node is

$$\frac{M!}{r!(M-r)!} \tag{3.11}$$

The number of possible key ring assignment for an $\mathcal{H}$-node is

$$\frac{M!}{S!(M-S)!} \tag{3.12}$$

The total number of possible key ring assignment for an $\mathcal{L}$-node and $\mathcal{H}$-node is

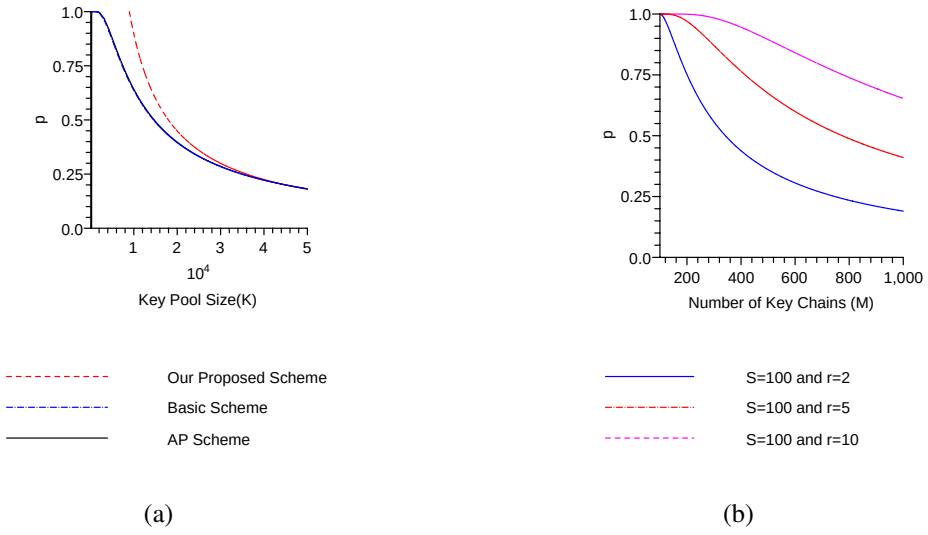
$$\frac{M!}{r!(M-r)!} \times \frac{M!}{S!(M-S)!} \tag{3.13}$$

The probability that an $\mathcal{L}$-node and $\mathcal{H}$-node share a common key can be given as

$$p = 1 - \frac{(M-r)!(M-S)!}{M!(M-r-S)!} \tag{3.14}$$

Figure 3.10 shows probability of key sharing for different schemes. For different values of K, M, S and r the probability of sharing at least one key is plotted, under our proposed scheme, the key pre-distribution scheme [42] which will refer as basic scheme, and Asymmetric Pre-distribution scheme [48] which will refer as AP scheme. In Figure 3.10(a), the key pool size ranges from 1,000 to 50,000 and key ring size is fixed to 100 for basic scheme. For AP scheme, $\mathcal{H}$-node keys are 500 and $\mathcal{L}$-node keys are 20. For our proposed scheme, the number of key chains (M) varies from 100 to 1000, S=90, and r=2. In other words, the number of key chains (M) is 0.02 times of the corresponding key pool size.

Figure 3.10(a) shows that for the proposed scheme, the same probability of key sharing among



(a) (b)

Figure 3.10: The Probability of Key Sharing

nodes can be achieved by just loading 2 generation keys in sensor node as compared to 100 keys in basic scheme [42] and 20 keys in AP scheme [48]. For instance, if there are 1000 $\mathcal{L}$-nodes and 10 $\mathcal{H}$-nodes in an HSN, where each $\mathcal{L}$-node is pre-loaded with 2 generation keys and each $\mathcal{H}$-node is pre-loaded with 100 generation keys, the total memory requirement for our proposed scheme in the unit of key length is $2\times1000+100\times10 = 3000$. However, in AP scheme [48], if each $\mathcal{H}$-node is loaded with 500 keys and each $\mathcal{L}$-node is loaded with 10 keys, the total memory requirement for storing these keys will be $500\times10+1000\times20=25{,}000$, which is 8 times larger than our proposed scheme. Further, for a homogeneous sensor network with 1000 $\mathcal{L}$-nodes, where each $\mathcal{L}$-node is preloaded with 100 keys, the memory requirements will be $100\times1000=100{,}000$, which is 33 times larger than our proposed scheme.

Figure 3.7 shows that the probability of key sharing among nodes and CH increases by a very little increase in the number of preloaded generation keys in $\mathcal{L}$-nodes. For instance, if preloaded keys are increased from 2 to 5, the key sharing probability increases from 0.5 to 0.8 approximately, for 400 key chains.

3.6.1 Security Evaluation

This section investigates the security resilience of our proposed scheme against node compromise attack. Further, the expected number of compromised links due to key revealing of captured nodes are calculated.

Each $\mathcal{L}$-node has a knowledge of $r \times N$ keys. The probability that a given key does not belong to an $\mathcal{L}$-node is $1 - \frac{r}{M}$. If there are n compromised nodes, the probability that a given key is not compromised is $(1 - \frac{r}{M})^n$. The probability of total number of compromised keys, where n number of $\mathcal{L}$-nodes are captured, is given in Equation 3.15.

$$\overline{p} = 1 - \left(1 - \frac{r}{M}\right)^n \tag{3.15}$$

Figure 3.11 shows the compromising probability with respect to the number of compromised nodes. In this Figure, the proposed scheme (PS) is compared to EG [42] and q-composite [19] schemes. For the given parameters: M=1000, K=50,000, r=5, and m=100, the results show that PS is more resilient against node capture as compared to EG and q-composite schemes.

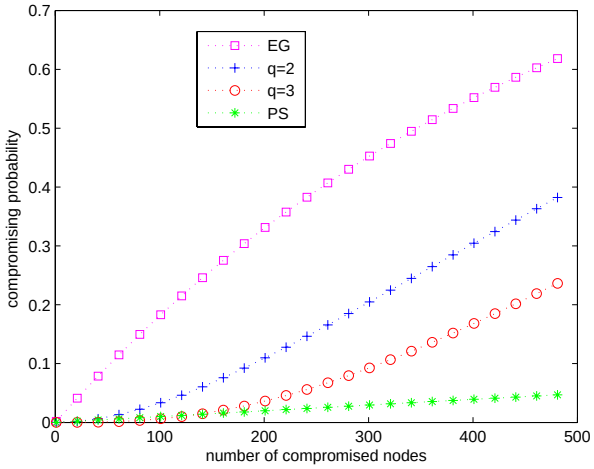


Figure 3.11: The Compromising Probability

Network layer attacks against sensor networks fall into one of the following categories: manip-

ulating routing information [80], selective forwarding [80], Sybil [81], Sink-hole [82], wormhole [83], and Hello flooding (unidirectional) attacks [82]. Brief descriptions of these attacks can be found in [84]. In the following, it is discussed that how routing structure described above can defend against various attacks on sensor network routing.

In Sybil attack [81], a single node presents multiple identities to other nodes in the network. Authentication is used to ensure one node cannot pretend to be other nodes, i.e., when a sensor node L_u sends a packet to another node L_v, L_u must present a MAC computed using the shared pairwise key K_{L_u,L_v} between L_u and L_v. Since the pairwise key K_{L_u,L_v} is only known by L_u and L_v, no adversary node can pretend to be node L_u (unless L_u is captured and the keys in L_u are obtained by the adversary). Thus, the Sybil attack does not work.

The proposed routing structure includes two parts - intra-cluster routing and inter-cluster routing. For intra-cluster routing, an $\mathcal{L}$-node only sends the data to its parent node of the (MST or SPT) tree, and the parent-child relationship is determined by the CH. For inter-cluster routing, given the locations of the $\mathcal{H}$-node and the BS, a serial of cells is determined as Relay Cells, and the packet is forwarded only by $\mathcal{H}$-nodes in the Relay Cells. Other nodes should not participate in routing. An adversary is not able to route in proposed routing structure, and therefore it is resistant to wormhole attack and sink-hole attack.

In proposed routing structure, the routing information is distributed by the CH. Since a CH is an $\mathcal{H}$-node with tamper-resistant hardware, it is well protected and can not be compromised by the adversary. A CH appends keyed MAC to each routing control message. Only the $\mathcal{L}$-node and the CH know the key used to generate the MAC, and thus an adversary is not able to send false routing information.

$\mathcal{H}$-nodes are protected by the tamper-resistant hardware, hence $\mathcal{H}$-nodes can not be compromised, and the selective forwarding attack can not be launched on $\mathcal{H}$-nodes. However, a selective for-

warding attack may happen on an $\mathcal{L}$-node. For example, a powerful adversary always serves as a relay node in a cluster, and she can selectively forward some packets while dropping other packets. The $packet_{ID}$ field is used to defend this attack. Recall that each relay $\mathcal{L}$-node is responsible for confirming that its successor has successfully forwarded the packet by overhearing the transmission. The $packet_{ID}$ field is used to identify the particular packet. If a node selectively drops a packet, this will be detected by the up-stream sender.

3.7 Implementation in Real Sensor Network

In this section, the implementation issues are investigated to show that the proposed scheme can be efficiently implemented on resource-constrained sensor nodes.

In this thesis a representative sensor structure for $\mathcal{L}$-nodes used is called MICAz as shown in Figure 3.12(a). It is from the Crossbow Company [85]. The MICAz is a 2.4GHz, IEEE 802.15.4 compliant, Mote module used for enabling low-power, wireless, sensor networks. The operating system is TinyOS and environment is Cygwin, which are discussed in detail in the next section.

Crossbow's Stargate nodes are used as $\mathcal{H}$-nodes as shown in Figure 3.12(b). The Stargate is a high-performance processing platform designed for sensor, signal processing, control and wireless sensor network applications [86]. It is a gateway node with the following specifications: 400 MHz Intel PXA255 Xscale processor, 64 MB of SDRAM and 32 MB of flash memory. Further, on another set of tests, Stargates were replaced by desktops – desktop specifications: 1.8 GHz AMD Turion(tm) 64X2 mobile TL-56 processor and 2GB of RAM running windows XP.

The proposed protocols are implemented for TinyOS using nesC [87] programming language. Our assessment includes how to discover the neighbors, the delay overhead of generating the shared key by applying keyed hash algorithm on generation keys and seed, and an evaluation of the overall key setup time for proposed scheme.

First, two one-way hash algorithms, SHA-1 and MD5, are implemented. A data stream of 64

(a) (b)

Figure 3.12: MicaZ and Stargate Sensor

bytes is taken. As shown in Table 3.1, for SHA-1 the code consumes 128 bytes of RAM, 4048 bytes of ROM, and takes approximately 10.5 ms to produce a 160-bit hash of a 64-byte message. MD5 produces a 128-bit message digest for a given data stream. The code consumes 176 bytes of RAM, 12.5 KB of ROM, and takes approximately 5.75 ms to hash a message of 64 bytes using 64-byte blocks.

Table 3.1: Time and Memory Requirements for MicaZ

Primitive	Time (msec)	RAM (Bytes)	ROM (Bytes)
SHA-1	10.545	128	4,048
MD5	5.757	176	12,500
HMAC-SHA1	21.959	30	4,424
HMAC-MD5	12.217	90	12,986

A keyed hash message authentication code (HMAC) is a MAC calculated using a cryptographic hash function in combination with a secret key. As with any MAC, it can be used to verify the data integrity and the authenticity of a message. However, in our proposed scheme, HMAC is used to generate key chains from generation keys. Any iterative cryptographic hash function, such as MD5 or SHA-1, may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-MD5 or HMAC-SHA-1 accordingly, where size of the output is same as the underlying hash function. Further, both HMAC-SHA1 and HMAC-MD5 implementations were validated using the test cases given in [88].

This research implements both HMAC-SHA1 and HMAC-MD5 algorithms. The HMAC-SHA1 code consumes 30 bytes of RAM, 4424 bytes of ROM, and takes approximately 21.9 ms to produce of MAC of 64 bytes of data stream, as shown in Table 3.1. Whereas the HMAC-MD5 code consumes 90 bytes of RAM, 12986 bytes of ROM, and takes approximately 12.2 ms to produce a MAC of 64 bytes of data.

The proposed scheme is implemented with both algorithms (HMAC-SHA1 and HMAC-MD5) for key generation. The memory consumption for HMAC-SHA1 (780 bytes RAM and 22.2 KB ROM) is less than HMAC-MD5 (840 bytes RAM and 30.7 KB ROM), as shown in Table 3.2.

Table 3.2: Memory Usage for Proposed Scheme

Proposed Scheme using	RAM (Bytes)	ROM (Bytes)
HMAC-SHA1	780 bytes	22206 bytes
HMAC-MD5	840 bytes	30728 bytes

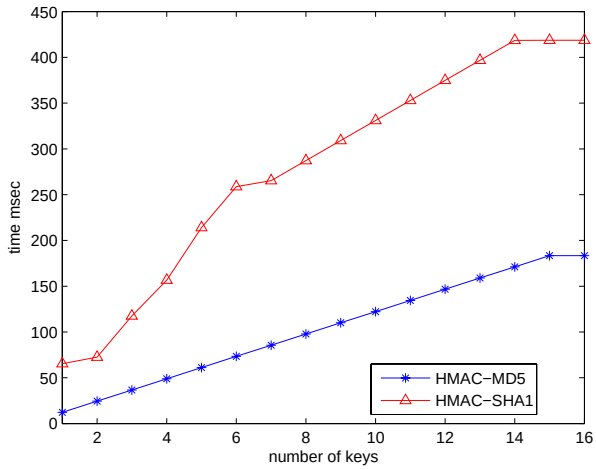


Figure 3.13: Comparison of Key Generation Process

However, the time required to generate MAC using HMAC-SHA1 is greater than HMAC-MD5, as shown in Figure 3.13. Keys are generated from generation keys (8 bytes) and seed (28 Bytes) using both HMAC-MD5 and HMAC-SHA1. Figure 3.13 shows that as the number of keys are increased, the processing time for key-chain generation increases accordingly. However, the in-

crease in HMAC-SHA1 is significantly greater than HMAC-MD5.

One of the major constraints on implementing any scheme on a sensor platform is the small available payload size of packets. Specifically, under TinyOS, this limitation is commonly set to 29 bytes. As a result, all of the wireless interactions between nodes must adhere to this restriction. For instance, each of the generation keys deployed in $\mathcal{L}$-nodes or $\mathcal{H}$-nodes are of 8 bytes, which matches the key size used for the TinySec implementation of the RC5 block cipher [29]. Two bytes are allocated for node identifier as well as 2 bytes for nonce value. It is believed that 2 bytes (65,536 or 2^{16}) would be sufficient for a large sensor network. Further, for current battery lifetimes, 2 bytes for nonce would provide sufficient protection against nonce reuse. Authentication is provided by TinySec's CBCMAC and it occupies a total of 4 bytes. While not appropriate for other environments, an online attack of this authentication mechanism would require an average attack span of 20 months because of the limited bandwidth in this setting [29]. Further, the $\mathcal{L}$-nodes life time in the network is also calculated in term of its communication energy as shown in Figure 3.7 and Figure 3.7. As it can be observed that energy consumed in proposed key management scheme is negligible as compared to the energy consumed in sensing.

Figure 3.16 gives the details of the interfaces that are provided and used by the implementation of our proposed scheme in TinyOS. The component provides only one interface `Sensor`. The component uses either `SHA1` or `MD5` interface with `HMAC` interface. Further, also use the interface `BlockCipher` provided by component `RC5` of TinySec. Moreover, there are several standard TinyOS interfaces used in the implementation, such as `Boot, Leds, Packet`, and `SplitControl`.

3.8 Conclusion

In this chapter, a key management scheme is proposed for heterogeneous sensor networks based on random key pre-distribution. In our scheme, instead of storing all the assigned keys in a

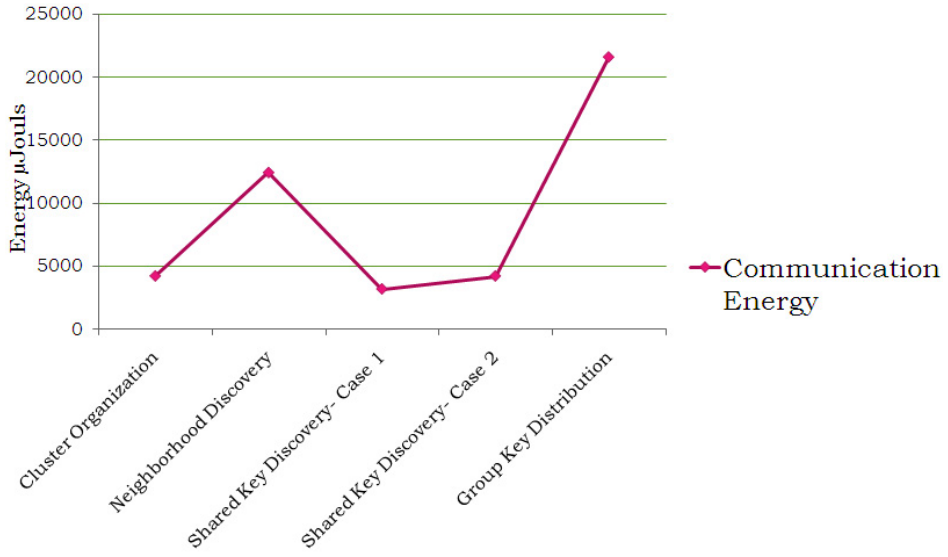


Figure 3.14: Energy Consumption in Key Management Phases

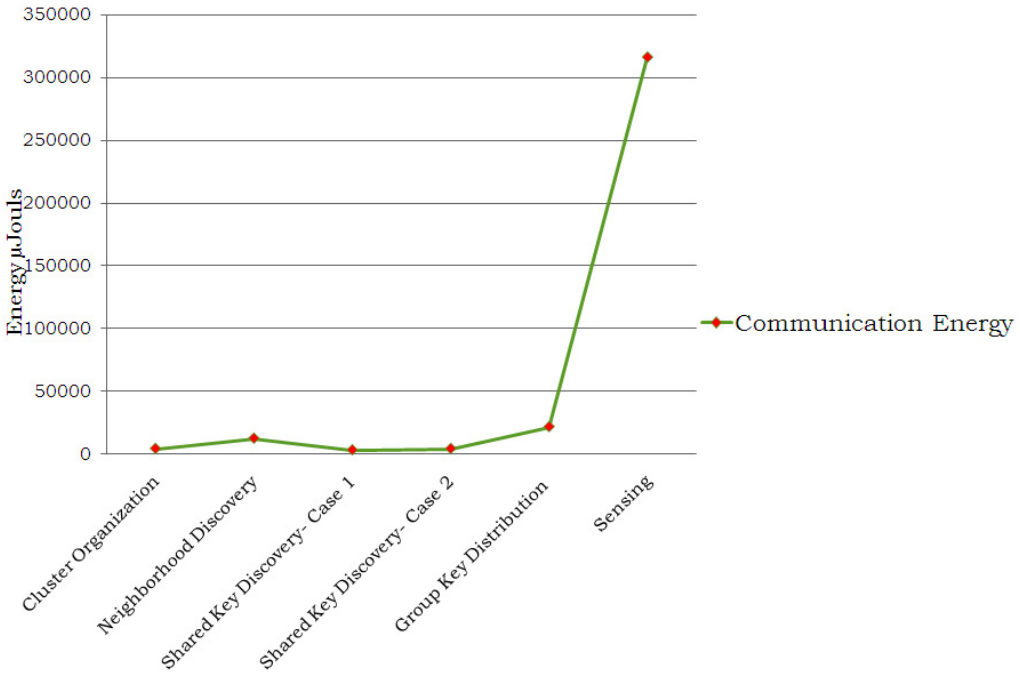


Figure 3.15: Energy Consumption in Proposed Key Management vs. Sensing

sensor node, a small number of generation keys are stored. Adversary or malicious nodes are precluded to join the cluster, as each $\mathcal{L}$-node is authenticated by CH using $\mathcal{L}$-node's authentication key. In pre-deployment phase each $\mathcal{H}$-node is preloaded with the master key and $\mathcal{L}$-node with authentication key (which is generated using master key). This research also provides secure

```
module SensorP {
  /* Provides Interfaces */
  provides interface Sensor;
  /* Uses Interfaces */
  /* Interfaces defined in Proposed Scheme */
  uses interface SHA1;
  uses interface MD5;
  uses interface HMAC;
  /* TinySec Interface */
  uses interface BlockCipher;
  /* Standard TinyOS interfaces */
  uses interface Boot;
  uses interface Leds;
  uses interface Packet;
  uses interface AMPacket;
  uses interface AMSend as RadioSend[am_id_t id];
  uses interface Receive as RadioReceive[am_id_t id];
  uses interface SplitControl as AMControl;
  uses interface Timer<TMilli> as Timer0;
  ...
  }
```

Figure 3.16: The *provides* and *uses* Interfaces for the Proposed Scheme Implementation in TinyOS.

inter-cluster communication. Further, for scalable solution and easy maintenance, dynamic nodes' addition and keys' revocation, in case of node compromise is provided. The results show that our scheme can significantly reduce the storage requirements as compared to other random key pre-distribution schemes. For instance, storage requirements can be reduced by 8 times as compared to AP [48], and 33 times as compared to basic scheme [42]. Also, the resiliency against node capture is better than previous key pre-distribution schemes. The TinyOS implementation shows that the proposed scheme can be efficiently implemented in real sensor networks. This research compares both HMAC-SHA1 and HMAC-MD5 to generate key chains. The results show that although HMAC-SHA1 consumes less memory resources than HMAC-MD5, it is more computationally intensive.

Chapter 4

AN EFFICIENT COLLUSION RESISTANT SECURITY MECHANISMS FOR HETEROGENEOUS SENSOR NETWORKS

4.1 Introduction

An important area of research interest is a general architecture for wide-area sensor networks that seamlessly integrates homogeneous and heterogeneous sensor networks. HSNs have different types of sensors, with a large number of ordinary sensors in addition to a few powerful sensors. Further, as sensor devices are typically vulnerable to physical compromise and they have very limited power and processing resources, it is unacceptable to completely trust the results reported from sensor networks, which are deployed outside of controlled environments without proper security.

In random key pre-distribution (RKP) schemes, a large key pool of random symmetric keys is generated along with the key identifiers. All nodes are given a fixed number of keys randomly selected from a key pool. In order to determine whether or not a key is shared, each node broadcasts its keys' identifiers. The, neighbors sharing a key associated with one of those identifiers, issue a challenge/response to the source. If two nodes do not share keys directly, they can establish a session key with the help of neighbors with which a key is already shared. It is highly likely that all nodes in the network will share at least one key if the following are carefully considered: a) the network density, b) the size of the key pool, and c) the number of keys pre-configured in each sensor node.

While pre-distributing pairwise keys does protect confidentiality, it still loads nodes with a large

number of globally-applicable secrets. By eliminating the eavesdropping attack, the pairwise scheme makes another type of malicious behavior more attractive. As several nodes possess the same keys, any node can make use of them by simply combining the keys obtained from a few nodes, which greatly increases the attacker's chances of sharing keys with other nodes. A collusive attacker can share its pairwise keys between compromised nodes by enabling each node to present multiple 'authenticated' identities to neighboring nodes while escaping detection [89]. Colluding nodes can grow their knowledge about the network security measures. Therefore, it is conceivable that few compromised nodes can collude and reveal all the keys employed in the network to an adversary. Such scenario is considered as capturing the entire network since the adversary would be capable of revealing all encrypted communications in the network.

An adversary who obtains compromised nodes' keys can inject malicious sensor nodes elsewhere in the network since the pool keys that were obtained are always valid and are used to authenticate each node. As a result, RKP is unable to protect the sensor network against collusion attack. In order to counter the collusion attacks, nodes should discard unused keys from the node's memory after the initialization phase; however, it means that new nodes can no longer join the system after the initial network deployment. The other possible way to prevent collusion attacks is updating the preloaded keys in order to prevent the compromised and revoked nodes from launching a collusive attack in which they pool together their keys with the goal of jeopardizing the secure channels between other nodes. Without key updating, both the performance and security of the system will degrade greatly with the number of compromised nodes.

In the previous chapter a scalable and efficient protocol for key management is proposed that is sensitive to the sensor nodes resource constraints, including storage, computation and communication. However this scheme is vulnerable to collusion attacks. In this chapter, an efficient collusion resistant security mechanisms [90, 91] is proposed for sensor networks including both

heterogeneous and homogeneous network. HSN consist of a small number of powerful high-end $\mathcal{H}$-nodes and a large number of ordinary low-end $\mathcal{L}$-nodes as discussed in the Section 3.2 of Chapter 3. However, homogeneous sensor networks (MSN) consists of only $\mathcal{L}$-nodes. Since the collusion attack on key pre-distribution scheme mainly takes advantage of the globally applicable keys, which are selected from the same key pool, update the key ring after initial deployment and generate new key rings by using one-way hash function on nodes' IDs and initial key rings. Further, in the proposed scheme, every node is authenticated by the BS in order to join the network. A good security practice is to use different keys for different cryptographic operations; this prevents potential interactions between the operations that might introduce weaknesses in a security protocol. Therefore propose scheme is using different keys for encryption and authentication. The analysis of the proposed scheme shows that even if a large number of nodes are compromised, an adversary can only exploit a small number of keys near the compromised nodes, while other keys in the network remain safe. It outperforms the previous random key pre-distribution schemes by: a) considerably reducing the storage requirement, b) providing more resiliency against node capture and collusion attacks.

4.1.1 Collusion Attack

In collusion attacks two or more nodes cooperate with each other by sharing their knowledge of pre-deployed secrets and thus increasing their capabilities in overcoming the network security measures. In RKP a collusion attack can be possible in the scenarios: a)when compromised nodes are in the transmission range of one another b) when compromised nodes are not in the transmission range of one another.

In the latter case, for example, consider two compromised nodes n_1 and n_2 which are not in the transmission range of each other. Suppose n_1's neighbors are n_3, n_5 and n_6 and it shares key with n_5 and n_9. Similarly n_2's neighbors are n_4,n_7 and n_9 and it shares keys with n_3, n_6, n_7 as

Table 4.1: Collusion Attack

Nodes	Neighbors	Key Share	Without Collusion	With Collusion
n_1	n_3, n_5, n_6	n_5, n_9	(n_1, n_5)	(n_1, n_5),(n_2, n_3),(n_2, n_6)
n_2	n_4, n_7, n_9	n_3, n_6, n_7	(n_2, n_7)	(n_2, n_7),(n_1, n_9)

shown in table 4.1. Accordingly, n_1 can communicate securely with n_5 and n_2 can communicate securely with n_7. If n_1 colludes with n_2 the resultant keys known to both of them would be $Keys(n_1) \cup Keys(n_2)$. As a result, n_1 can communicate with n_6 and n_3 masquerading as n_2 and similarly n_2 can communicate with n_9 masquerading as n_1. It can be seen that compromised nodes not in the communication range of each other can collude to launch an attack to uncover a large number of employed keys.

Sensor networks are often deployed in hostile environments, yet nodes cannot afford expensive tamper-resistant hardware. The threat model is assumed to be an adversary that tries to capture and compromise a number of nodes in the network. Also, there is no unconditional trust on any sensor node. An adversary may try to eavesdrop on the messages exchanged in the system, intercept these messages as well as inject false messages. If an adversary compromises a node, the memory of that node is known to the adversary; CHs can also be compromised. The goal of the adversary is to uncover the keys used in the network for secure communication. The nodes can collude with each other by sharing their keys with other attacker nodes. The main objective of node collusion is to incrementally aggregate the uncovered keys of individual nodes to a level that allows revealing all encrypting traffic in the network.

4.2 Protocol

In this section a key management protocol is presented that increases the network resilience to collusion attacks.

for every key $k_i \in P$, where $P = (k_1, k_2, ..., k_P)$
compute $z = f_{k_i}(id_{L_x})$
if $z \equiv 0 \; mod \; (\frac{P}{r})$ then
put k_i into R_{L_x}, the key ring of $\mathcal{L}$-node.

Figure 4.1: Key Ring Assignment ($\mathcal{L}$ Node)

for every key $k_i \in P$, where $P = (k_1, k_2, ..., k_P)$
compute $z = f_{k_i}(id_{H_x})$
if $z \equiv 0 \; mod \; (\frac{P}{S})$ then
put k_i into R_{H_x}, the key ring of $\mathcal{H}$-node.

Figure 4.2: Key Ring Assignment ($\mathcal{H}$ Node)

4.2.1 Initial Deployment

Generate a large key pool K consisting of a P number of random symmetric keys and their ids prior to network deployment. Before deploying the nodes, each node is loaded with its assigned key ring R as follows: each $\mathcal{L}$-node is pre-loaded with r number of keys and each $\mathcal{H}$-node is pre-loaded with S number of keys, randomly selected from the key pool, where $S >> r$. As given in [92], the assigning rules for $\mathcal{L}$-nodes are shown in Figure 4.1. In addition to that every $\mathcal{L}$-node is pre-loaded with an authentication key AK_{L_x} shared with BS and public key of BS. Further the key ring assignment to $\mathcal{H}$-nodes is shown in Figure 4.2. In addition to that every $\mathcal{H}$-node is pre-loaded with an authentication key AK_{H_x} shared with BS.

4.2.2 Cluster Heads Authentication

Before entering into the cluster organization phase each $\mathcal{H}$-node need to be authenticated by BS. Let $\mathcal{H}$-node H_a sends a request to BS consisting of its id, a random number $nonce$, and MAC is calculated on all these values using it authentication key AK_{H_a} as shown in message 1 of Figure 4.3. BS authenticate H_a by verifying the MAC. If authentication is successful, BS randomly selects a key, suppose k_m from the key ring of H_a. BS then sends message 2 shown in Figure 4.3 consisting of the id of k_m, nonce, and id_{H_a} encrypting with AK_{H_a}. H_a gets all these

values by decrypting the message 2 and then generates the shared secret key K_{BS,H_a} between BS and H_a by applying one-way hash function on id_{BS}, id_{H_a}, and 0 using k_m as shown in message 3. Propose scheme uses different keys for date encryption and message authentication, therefore H_a generate the MAC key $\acute{K}_{BS,H_a}$ by applying one-way hash function on id_{BS}, id_{H_a}, and 1 using k_m as shown in message 4. After joining the network, H_a deletes AK_{H_a} from its memory.

1	$H_a \rightarrow BS : id_{H_a}, nonce, MAC_{AK_{H_a}}(id_{H_a}\|nonce)$
2	$BS \rightarrow H_a : \langle id_{k_m}, nonce, id_{H_a} \rangle_{AK_{H_a}}$
3	$K_{BS,H_a} = H(k_m, id_{BS}\|id_{H_a}\|0)$
4	$\acute{K}_{BS,H_a} = H(k_m, id_{BS}\|id_{H_a}\|1)$

Figure 4.3: CH Authentication by BS

4.2.3 Cluster Organization Phase

After authentication by BS, $\mathcal{H}$-nodes enter into the cluster organization phase. Let $\mathcal{H}$-node, H_a broadcasts an advertisement message adv, consisting of its id (id_{H_a}) and $nonce$ as shown in message 1 of Figure 4.4. The nearby $\mathcal{L}$-nodes, suppose L_b upon receiving the adv message, determines whether it shares a common key with H_a as follows: for every key $k_j \in R_{L_b}$, L_b computes $z = f_{k_j}(id_{H_a})$. If $z \equiv 0 \; mod(\frac{P}{S})$, it means that H_a also has a key k_j in its key ring i.e. $R_{H_a} \cap R_{L_b} = k_j$.

As L_b could receive adv broadcast messages from several $\mathcal{H}$-nodes, it would be possible that L_b shares a common key with more than one $\mathcal{H}$-node. From these $\mathcal{H}$-nodes, it will choose the $\mathcal{H}$-node as its CH with whom it has the best received signal strength and link quality.

L_b sends the join request to the selected CH (say H_a) protected by MAC, using k_j and include the $nonce$ from CH broadcast (to prevent replay attack), as well as the id of shared key (id_{k_j}) chosen to protect this link (so that the receiving CH knows which key to use to verify the MAC) as shown in message 2 of Figure 4.4.

H_a upon receiving the join request from L_b, authenticates the L_b by verifying the MAC using

k_j. H_a generate the shared pairwise key (K_{H_a,L_b}) with L_b by applying one-way hash function on id_{L_b}, id_{H_a}, and 0 using k_j as shown in message 3. H_a generate the MAC key ($\acute{K}_{H_a,L_b}$) with L_b by applying one-way hash function on id_{L_b}, id_{H_a}, and 1 using k_j as shown in message 4 and sends message 5 to L_b consisting of cluster key CK_a encrypted with K_{H_a,L_b} along with MAC on id_{H_a}, id_{L_b}, nonce, and CK_a calculated using $\acute{K}_{H_a,L_b}$.

L_b gets the cluster key CK_a by decrypting the message using K_{H_a,L_b}, verifies the MAC by using $\acute{K}_{H_a,L_b}$ to ensure the message authenticity and integrity and hence join the cluster.

Each $\mathcal{L}$-node also records other $\mathcal{H}$-nodes from which it receives the adv messages and it has common key with them, as these $\mathcal{H}$-nodes will serve as backup cluster heads in case the CH (H_a) fails.

1	$H_a \rightarrow * : id_{H_a}, nonce$
2	$L_b \rightarrow H_a : id_{L_b}, id_{H_a}, id_{k_j}, nonce, MAC_{k_j}(id_{L_b}\|id_{H_a}\|id_{k_j}\|nonce)$
3	$K_{H_a,L_b} = H(k_j, id_{H_a}\|id_{L_b}\|0)$
4	$\acute{K}_{H_a,L_b} = H(k_j, id_{H_a}\|id_{L_b}\|1)$
5	$H_a \rightarrow L_b : \langle CK_a \rangle_{K_{H_a,L_b}}, MAC_{\acute{K}_{H_a,L_b}}(id_{L_b}\|id_{H_a}\|nonce\|CK_a)$

Figure 4.4: Messages Transferred between Sensor Nodes and CHs

4.2.3.1 Unsupervised Nodes

At the end of cluster organization phase, it is expected that a fraction of the $\mathcal{L}$-nodes will not be matched with a CH because of key sharing constraints; these nodes are called unsupervised nodes. Suppose the unsupervised node L_x have best RSSI with H_a. L_x sends request to H_a consisting of its id, id of H_a, nonce and MAC is calculated on all these values using AK_{L_x} shown in message 1 of Figure 4.5. H_a forwards this message to BS. BS authenticate the L_x by verifying the MAC and select key k_j from the key ring of L_x. BS sends k_j and id_{k_j} to H_a encrypting with the key K_{BS,H_a} along with MAC on k_j, id_{k_j}, and nonce using key $\acute{K}_{BS,H_a}$ as shown in message 3. H_a generates the shared pairwise key with L_x by applying one-way hash function on id_{H_a}, id_{L_x}, and

0 by using k_j as shown in message 4. H_a generates the MAC key with L_x by applying one-way hash function on id_{H_a}, id_{L_x}, and 1 by using k_j as shown in message 5. H_a sends message 6 to L_x consisting of its id_{H_a}, id_{L_x}, id of the key k_j to be used as common shared key, and cluster key encrypted with K_{H_a,L_x} and MAC on all these values using $\acute{K}_{H_a,L_x}$. L_x receives this message and calculate the K_{H_a,L_x} and $\acute{K}_{H_a,L_x}$ by using k_j and use it to get cluster key and hence join the network.

1	$L_x \rightarrow H_a : id_{L_x}, id_{H_a}, nonce, MAC_{AK_{L_x}}(id_{L_x}\|id_{H_a}\|nonce)$
2	$H_a \rightarrow BS$: Forward Message 1 to BS
3	$BS \rightarrow H_a : id_{k_j}, \langle k_j \rangle_{SK_{BS,H_a}}$
4	$K_{H_a,L_x} = H(k_j, id_{H_a}\|id_{L_x}\|0)$
5	$\acute{K}_{H_a,L_x} = H(k_j, id_{H_a}\|id_{L_x}\|1)$
6	$H_a \rightarrow L_x : id_{L_x}, id_{H_a}, id_{k_j}, nonce, \langle CK_a \rangle_{K_{H_a,L_x}}, MAC_{\acute{K}_{H_a,L_x}}(id_{L_x}\|id_{H_a}\|id_{k_j}\|nonce\|CK_a)$

Figure 4.5: Unsupervised Nodes Key Establishment

There may be some $\mathcal{L}$-nodes, suppose L_y in the network that may have common key shared with a CH H_a but have better RSSI with H_b then H_a. However, L_y do not have common key shared with H_b. In that case L_y can contact with H_b as unsupervised node to request key as explained above.

4.2.3.2 Direct Key Discovery Phase

After cluster organization phase, $\mathcal{L}$-nodes learn their neighbors through the exchange of $hello$ messages, and then attempt to establish keys with them. To accomplish this, $\mathcal{L}$-nodes broadcasts $hello$ messages.

Consider an $\mathcal{L}$-node, L_a, it broadcasts a $hello$ message consisting of its id id_{L_a}. Then, it waits for $hello$ messages from its neighboring $\mathcal{L}$-nodes. Suppose, it receive $hello$ message from one of its neighbor L_b, it extracts the node id from message i.e. id_{L_b}. For every key $k_j \in R_{L_a}$, L_a computes $z = f_{k_j}(id_{L_b})$. If $z \equiv 0 \; mod(\frac{P}{r})$, it means that node L_b also has a key k_j in its key ring i.e. $R_{L_a} \cap R_{L_b} = k_j$. After discovering the common key in their key rings, they will generate the

shared pairwise key by applying one-way hash function on id_{L_a} and id_{L_b} by using k_j as given in Equation 4.1.

$$K_{L_a,L_b} = H(k_j, id_{L_a}|id_{L_b}|0) \tag{4.1}$$

If L_a and L_b share more than one common keys in their key rings, the key with the least id would be used to generate the shared pairwise key.

4.2.3.3 Indirect Key Discovery Phase

$\mathcal{L}$-nodes gather information about both types of neighbors: 1) nodes with which they share a key, and 2) nodes with which they do not share keys. When the direct key discovery phase ends, the nodes would have discovered the common keys, if any, with their neighbors. $\mathcal{L}$-nodes use the CH with which keys are already shared to assist it in establishing secure connections with the neighboring $\mathcal{L}$-nodes with which common keys are not found.

Let $\mathcal{L}$-nodes L_x and L_y are neighboring nodes in the same cluster; however, they do not share a common key in their key rings, $R_{L_a} \cap R_{L_b} = \phi$.

The $\mathcal{L}$-node L_x, having already established a link with its CH (H_a), transmits a message to H_a, as shown in Figure 4.6, requesting to transmit a key with $\mathcal{L}$-node L_y encrypted with key K_{H_a,L_x}.

H_a generates a key k_i and unicasts the message 2 to L_x and message 3 to L_y shown in Figure 4.6. When L_x (or L_j) receives its message from H_a, it decrypts the message using key K_{H_a,L_x} to get key k_i. Similarly, L_y uses key K_{H_a,L_y} for decrypting the message. Now, L_x and L_y generate the shared pairwise by applying one-way hash function on id_{L_x} and id_{L_y} by using k_i, as shown in message 4.

1	$L_x \rightarrow H_a : id_{L_x}, id_{L_y}, nonce, MAC_{\acute{K}_{H_a,L_x}}(id_{L_x}\|id_{L_y}\|nonce)$
2	$H_a \rightarrow L_x : id_{L_x}, id_{L_y}, nonce, \langle k_i \rangle_{K_{H_a,L_x}}$
3	$H_a \rightarrow L_y : id_{L_x}, id_{L_y}, nonce, \langle k_i \rangle_{K_{H_a,L_y}}$
4	$K_{L_x,L_y} = H(k_i, id_{L_x}\|id_{L_y})$

Figure 4.6: Indirect Key Discovery Phase

4.2.4 Key Ring Update

After indirect key-discovery phase, all $\mathcal{L}$-nodes and $\mathcal{H}$-nodes destroy their initial key rings. Because these key rings have globally applicable secrets which can be used by adversary to launch a collusion attack, these initial key rings are deleted from node's memory.

First, before a node (say L_x) destroys its initial key ring, it generates a new key ring as shown in Figure 4.7. For every key $k_i \in R_{L_x}$, it generates a new key $\acute{k_i}$ by applying one-way hash function on its id (id_{L_x}) and k_i. In this way, it generates a set of new keys from keys in its initial key ring. Further, in order to keep record of the keys in its initial key ring, these newly generated keys are assigned the same ids as those of the original keys. Then, L_x deletes k_i from its key ring R_{L_x}.Further, the above procedure is also applied for $\mathcal{H}$-nodes to update their key rings.

procedure keyRingUpdate()
  for $\forall k_i \in R_{L_x}$
    $\acute{k_i} = H(k_i, id_{L_x})$
    $id_{\acute{k_i}} = id_{k_i}$
    delete(k_i)
  endfor

Figure 4.7: Key Ring Update

4.3 Other Security Issues in HSN

In this Section, other security issues in HSN are discussed, including setting up keys for newly deployed sensor nodes, node revocation, and periodic re-keying.

1	$L_x \rightarrow * : id_{L_x}, nonce$
2	$H_a \rightarrow L_x : id_{H_a}, nonce$
3	$L_x \rightarrow H_a : id_{L_x}, MAC_{k_j^{'}}(id_{L_x}\|id_{k_j}\|nonce)$
4	$K_{L_x,H_a} = H(k_j^{'}, id_{H_a}\|id_{L_x}\|0)$
5	$K_{L_x,H_a}^{'} = H(k_j^{'}, id_{H_a}\|id_{L_x}\|1)$

Figure 4.8: New Node Addition

4.3.1 Addition of a New Node

The common key pre-disribution schemes are unable to add new nodes in the network if the initial key rings are deleted from node's memory. As a result, this reserach develop a new solution capable of handling addition of new legitimate $\mathcal{L}$-nodes beyond the initial deployment, even after the deletion of initial key rings from node's memory.

Suppose new $\mathcal{L}$-node L_x wants to join a network, it broadcasts a join request consisting of its id (id_{L_x}) and a random number $nonce$, as shown in message 1 of Figure 4.8. Then, it waits for reply from nearby CHs. Let L_x receives a reply message from CH (say H_a). For every key $k_j \in R_{L_x}$, L_x computes $z = f_{k_j}(id_{H_a})$. If for any k_j, $z \equiv 0\ mod(\frac{P}{S})$, it means that $k_j \in R_{H_a}$, but it is no longer available now because R_{H_a} has been deleted. So, L_x computes the corresponding key i.e. $k_j^{'}$ of H_a's new key ring $R_{H_a}^{'}$ by applying one-way hash function on id_{H_a} and k_j i.e. $k_j^{'} = H(k_j, id_{H_a})$. Then, L_x sends a message to H_a consisting of its id id_{L_x}, id of k_j ($id_{k_j} = id_{k_j^{'}}$), $nonce$ and MAC is calculated on all these values using $k_j^{'}$ as shown in message 3 of Figure 4.8. Now, L_x and H_a generate the shared pairwise key by applying one-way hash function on id_{H_a}, id_{L_x} and 0 by using $k_j^{'}$, as shown in message 4. Both L_x and H_a generate MAC key by applying one-way hash function on id_{H_a}, id_{L_x} and 1 by using $k_j^{'}$, as shown in message 5.

Then, L_x discovers the shared key with its neighboring $\mathcal{L}$-nodes by using either direct or indirect key discovery phase, as given above.

4.3.2 Node Revocation

In the proposed scheme there is no need to revoke the key rings of compromised nodes because the initial key rings of all the nodes in the network has been updated and no two nodes in the network has any key common in their key rings after initial deployments. If a node is compromised only the links that are directly associated with that node will be compromise. Therefore, our scheme does not need the revocation of key rings of compromised nodes.

4.3.3 Fault Tolerance

Our approach should support the ability to allow $\mathcal{L}$-nodes to change the cluster even after the initial key rings has been updated. As the above described scheme will not allow the nodes to change cluster once their initial key ring has been updated. As a result, it is imperative that a new solution is developed capable of handling the change in network topology beyond the initial deployment. Suppose an $\mathcal{L}$-node L_x moves from cluster a to cluster b. So it needs to find a cluster key CK_b, shared pairwise key with CH H_b and also the shared pairwise keys with its new neighboring $\mathcal{L}$-nodes. As L_x has updated its initial key ring, it will not have any common key with H_b or any of neighboring $\mathcal{L}$-nodes.

There are three problems that need to be solved, How can H_b who no longer has the initial key rings authenticate L_x?; How can H_b and L_x setup a pairwise key between each other? and How can L_x setup a pairwise key with its neighboring $\mathcal{L}$-nodes? In that case L_x contacts the BS for joining the H_b as CH, as described above in section 4.2.3.1. L_x will setup the pairwise key with neighboring $\mathcal{L}$-nodes by using the indirect key discovery phase described in section 4.2.3.3.

4.3.4 Periodic Re-keying

Periodic re-keying has to be performed if any node finishes $2^{2k/3}$ number of encryptions using the same key, where k is the number of bits in the key. The cluster key re-keying is initiated by CH

by generating the new cluster key, encrypting it with the old cluster key and distributing to the cluster members.

Re-keying of cluster key is also necessary, when a cluster member leaves the cluster because of either its battery power gets exhausted or when it is being compromised by an adversary. In that case CH needs to distribute the new cluster key by unicast it to $\mathcal{L}$-nodes encrypting with the shared pair-wise keys so the nodes that have leave cluster do not receive new cluster key.

4.4 Performance Analysis

This section analyzes the proposed scheme and explains its features that make this scheme feasible to implement and a better alternative option as compared to the other key pre-distribution schemes.

For any pair of nodes to find a secret key between them, the key sharing graph $G(V, E)$ needs to be connected. Given the size and the density of a network, the objective is to determine the key pool size P, the number of keys assigned to $\mathcal{L}$-nodes r, and the number of keys assigned to $\mathcal{H}$-nodes S such that, the graph G is connected with high probability. [19] propose the q-composite keys scheme that allows two sensors to setup a pairwise key only when they share at least q common keys.. The q-composite keys scheme provides better security for sensor networks. The key management schemes proposed in this paper can be easily extended to require at least q shared keys. There is a need to find the largest key pool size such that the probability of an $\mathcal{L}$-node and an $\mathcal{H}$-node sharing at least q keys is no less than a threshold p.

Let $p(j)$ be the probability that an $\mathcal{L}$-node and an $\mathcal{H}$-node have exactly j keys in common. Recall that an $\mathcal{L}$-node and an $\mathcal{H}$-node are pre-loaded with r and S keys, respectively.

An $\mathcal{L}$-node has $\binom{P}{r}$ different ways of picking r keys from a key pool with the size P, and an $\mathcal{H}$-node has $\binom{P}{S}$ different ways of picking S keys from the key pool.

Thus, the total number of ways for an $\mathcal{L}$-node and an $\mathcal{H}$-node to pick r and S keys, respectively,

is $\binom{P}{r}\binom{P}{S}$. Suppose that the two nodes have j keys in common. There are $\binom{P}{j}$ ways to pick j common keys.

After the j common keys are picked, there remain $S + r - 2j$ distinct keys in the two key rings which are to be picked from the remaining pool of $P - j$ keys.

The number of ways to do so is $\binom{P-j}{S + r - 2j}$. The $S + r - 2j$ distinct keys must then be partitioned between the $\mathcal{L}$-node and the $\mathcal{H}$-node. The number of such partitions is $\binom{S+r-2j}{r-j}$.

Hence the total number of ways to choose two key rings with j keys in common is the product of the three terms, i.e.,$\binom{P}{j}\binom{P-j}{S + r - 2j}\binom{S+r-2j}{r-j}$

Thus the probability of sharing at least j keys in common is given in Equation 4.2

$$p(j) = \frac{\binom{P}{j}\binom{P-j}{S + r - 2j}\binom{S+r-2j}{r-j}}{\left[\binom{P}{r}\binom{P}{S}\right]} \tag{4.2}$$

Let p_c be the probability that an $\mathcal{L}$-node and an $\mathcal{H}$-node share sufficient keys to form a secure connection. If q shared-keys are required, then: $p_c = 1 - (p(0) + p(1) + \cdots + p(q-1))$. For given key ring size r and S, key overlap q, and minimum connection probability p, the largest key pool size P can be computed such that $p_c \geq p$.

The probability of an $\mathcal{L}$-node and $\mathcal{H}$-node with key rings sizes r and S sharing at least one key with each other is given in Equation 4.3:

$$p_{sk} = 1 - \frac{\binom{P}{S}\binom{P-S}{r}}{\binom{P}{S}\binom{P}{r}} = 1 - \frac{(P-r)!(P-S)!}{P!(P-r-S)!} \tag{4.3}$$

Similarly the probability of sharing at least one key between two $\mathcal{L}$-node is given in Equation 4.4

$$p_{sk} = 1 - \frac{\binom{P}{r}\binom{P-r}{r}}{\binom{P}{r}^2} = 1 - \frac{(P-r)!^2}{P!(P-2\gamma)!} \tag{4.4}$$

Figure 4.9 shows the probability of key sharing among $\mathcal{H}$-node and $\mathcal{L}$-node with respect to key pool size. Further, a fixed number of pre-loaded keys are used in $\mathcal{H}$-nodes, $S = 500$; whereas pre-loaded keys for $\mathcal{L}$-nodes vary as $r = 10, 20, 30$. The graphs show that the pre-loaded keys in $\mathcal{L}$-nodes can be significantly reduced with acceptable probability of key sharing.

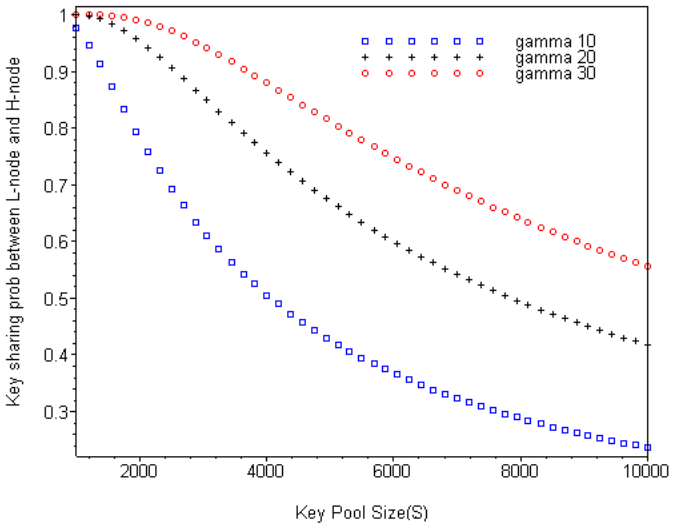


Figure 4.9: The Key Sharing Probability

In our scheme, only a fraction of CHs is probabilistically accessible by an ordinary node. Probability of key sharing between $\mathcal{H}$-node and $\mathcal{L}$-node and the number of CHs β in the network can also determine the expected number of unsupervised nodes, i.e. the probability that an ordinary node will be unsupervised. Given p and β, the probability of the number of unsupervised nodes is given in Equation 4.5:

$$p_{us} = \left(1 - (1 - \frac{(P-r)!(P-S)!}{P!(P-r-S)!})\right)^{\beta} \tag{4.5}$$

In a network with N number $\mathcal{L}$-nodes, it is then expected that $N \times p_{us}$ nodes will be unsupervised. Figure 4.10 shows fraction of unsupervised nodes as a function of β under different values of p. As β increases, the number of unsupervised nodes decrease rapidly. Further, as p increases, the number of unsupervised nodes also increase.

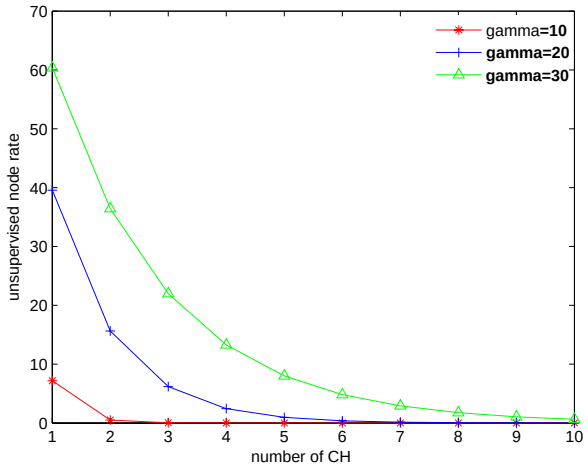


Figure 4.10: Unsupervised Nodes

4.4.1 Security Analysis

Proposed key pre-distribution scheme is evaluated in terms of its resilience against node capture and collusion attack. There is a need investigate when α number of nodes are captured, what fraction of the additional communication (i.e. communication among uncaptured nodes) would be compromised?

To compute this fraction, first compute the probability that any one of the additional communication links is compromised after α nodes are captured. In this analysis, those links are considered which are secured using a pairwise key computed from the common key shared by the two nodes of this link. It should also be noticed that during shared key discovery process, two neighboring nodes find the common key in their key rings and use this key to agree upon another random key to secure their communication. Because this new key is generated by applying one-way hash

function on common shared key and node ids, the security of this new random key does not directly depend on whether the key rings are broken. Further, the nodes' initial key rings are also deleted from their memory, after setting up shared pairwise keys with neighbors. As a result, the fraction of communications compromised when α number of nodes being compromised can be given as

$$\frac{number\ of\ links\ in\ \alpha\ compromised\ nodes}{Total\ number\ of\ links}$$

which means that only those links will be affected which are directly connected with α compromised nodes, while the other links in the network will remain safe. Figure 4.11 shows the graphs

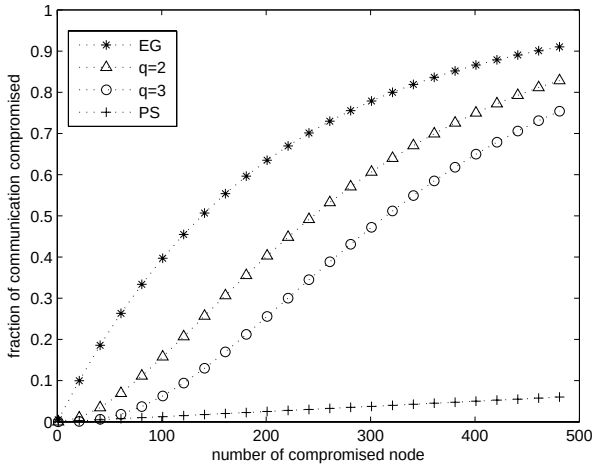


Figure 4.11: The Compromising Probability

of number of compromised communication links with respect to the number of compromised nodes. The proposed scheme (PS) is compared with basic scheme (EG) [42] and q-composite scheme [19]. The graphs show that as the number of compromised nodes increases, the traditional schemes are severely affected as compared to PS.

Further, in collusion attacks, the adversary takes advantage of the pairwise secret keys stored by each sensor node as these keys are globally applicable secrets and can be used throughout the

network, yet ordinary sensors can only communicate with the small fraction of nodes within radio range. So, the adversary can launch a collusion attack by exploiting this lack of communication between nodes and can now share its pairwise keys between compromised nodes, enabling each node to present multiple 'authenticated' identities to neighboring nodes, while escaping detection. In proposed scheme, the initial key rings are deleted from nodes memory after setting up shared pairwise keys with neighbors. However, nodes generate new key rings from initial key rings by applying one-way hash function on node ids and keys in their initial key rings.

Consider two arbitrary $\mathcal{L}$-nodes, L_a and L_b, where $R_{L_a} = \{k_1, k_2, \ldots, k_r\}$, $R_{L_b} = \{k_1, k_2, \ldots, k_r\}$, and $R_{L_a} \cap R_{L_b} = k_i$. As L_a and L_b are not within the communication range of each other, they do not use k_i. After setting up shared pairwise keys with neighbors, both L_a and L_b delete the initial key rings (R_{L_a} and R_{L_b}) and generate the new key rings (say $R^{'}_{L_a}$ and $R^{'}_{L_b}$) by applying one-way hash function on all the keys in their initial key rings and node ids. As a result, $R^{'}_{L_a} \cap R^{'}_{L_b} = \phi$. Similarly, in α number of compromised nodes, there will be no common key in their new key rings i.e $R^{'}_{L_1} \cap R^{'}_{L_2} \cap \ldots \cap R^{'}_{L_\alpha} = \phi$. As no more globally applicable secrets remain in the node's memory, it is not possible by adversary to launch a collusion attack.

4.5 Conclusion

Key establishment is a fundamental prerequisite for secure communication in wireless sensor networks. A key pre-distribution scheme is one of the common solutions for establishing secure communication in sensor networks. Random key pre-distribution schemes are vulnerable to collusion attacks because pre-loading global secrets onto exposed devices can be used in these attacks. This work present a new efficient key distribution scheme for heterogeneous sensor networks which is secure against collusion attack. The analysis shows that the proposed scheme provide more resiliency against node capture and collusion attack by deleting the initial key rings from their memory after generating the shared pairwise key with neighbors. It also allow new nodes

to join the system once initialization is completed and initial key ring has been destroyed from node's memory.

Chapter 5

SECURE GROUP COMMUNICATION WITH SELF HEALING AND REKEYING IN WIRELESS SENSOR NETWORKS

5.1 Introduction

Secure group communication needs a secret shared by all the group members for group oriented applications in wireless sensor networks (WSNs). The shared key provides group secrecy and source authentication. A single symmetric key known only to the group members can effectively protect a multicast group. However, only legitimate users should have access to the group communication in order to achieve privacy [93]. In rekeying, the session keys are updated periodically, when new users join or old users leave the group. The keys are securely redistributed to the existing members of the group in order to provide forward secrecy (FS) as well as backward secrecy (BS). The newly joined users should not be able to derive the previous group keys, even if they are able to derive future group keys with subsequently distributed keying information. Similarly, the revoked users should not be able to derive the future session keys, even if they are able to compute the previous session keys with previously distributed keying information.

The rekeying is performed by the group controller (GC). The most important parameters when performing group rekeying are as follows: the number of keys stored by the GC, the number of keys stored by each group member, the number of keys delivered in the initialization stage, bandwidth required for updating the keys, and latency for updating the session key [94].

As the size of the group grows and/or the rate of membership change increases, the frequency of rekeying becomes the primary bottleneck for rekeying on each membership change. Therefore,

scalable group rekeying is an important and challenging problem to be addressed in order to support secure multicast communication.

Another important problem in multicast communication is reliability. Since multicasting is an unreliable mode of communication, packets may be lost during the communication. If a packet containing key updating information is lost, authorized receivers may not be able to calculate the session key. This may influence rekeying and so the rekeying system must be self-healing if packet loss occurs. In a large and dynamic group communication over an unreliable network, the main concept of self-healing in key distribution schemes is that users can recover lost session keys on their own, without requesting additional transmissions from the group manager, even if some previous key distribution messages are lost. This reduces network traffic, the risk of user exposure through traffic analysis, and the work load on the group manager.

The key idea of self-healing key distribution schemes is to broadcast information that is useful only for trusted members. Combined with its pre-distributed secrets, this broadcast information enables a trusted member to reconstruct a shared key. On the contrary, a revoked member is unable to infer useful information from the broadcast. The only requirement that a user must satisfy to recover the lost keys through self-healing is its membership in the group both before and after the sessions in which the broadcast packet containing the key is sent. A user who has been off-line for some period is able to recover the lost session keys immediately after coming back on-line. Thus self-healing approach of key distribution is stateless.

The need to form a group might be driven by the query being propagated through a node. As a result, a node may need to define a multicast group to make the query initiated in those nodes and then collect the result efficiently and securely. Further, a node may also modify such queries effectively over the time. For instance, a multicast group could be a region defined with a geometric shape.

As mentioned in Chapter 3, Section 3.4.6 (Setting up Cluster Key), the establishment of cluster/group key is too trivial. The CH/GC sends cluster key to each individual L-Sensor in its cluster. This procedure for group key distribution is computationally not efficient along with no guarantee of forward and backward secrecy and any provision of group based rekeying.

In this chapter a self-healing group key distribution scheme [95] is developed for secure multicast group communications for hierarchical HSN environment. It present a strategy for securely distributing rekeying messages and specify techniques for joining and leaving a group/cluster. Access control in multicast system is usually achieved by encrypting the content using an encryption key, known as the group key that is only known by the GC and all legitimate group members. In our scheme, all rekeying messages, except for unicast of an individual key, are transmitted without any encryption using one-way hash function and XOR operation. In proposed scheme, nodes are capable of recovering lost session keys on their own, without requesting additional transmission from the GC. The proposed scheme provides both backward and forward secrecy. The proposed scheme is analyzed to verify that it satisfies the security and performance requirements for secure group communication.

5.1.1 Node Revocation

The concept of node revocation can be described as follows. Let G be the set of all possible group nodes, and Rev be the set of revoked nodes, where $Rev \subseteq G$. The group node revocation is required to offer a secure way for GC to transmit rekeying messages over a broadcast channel shared by all nodes so that any node $n_i \in \{G - Rev\}$ can decrypt the rekeying messages, whereas any node in Rev, $n_i \in Rev$, cannot decrypt rekeying messages.

5.1.2 Session Key Distribution with Confidentiality

The confidentiality in the session key distribution requires that for any node n_i, the session key is efficiently determined from the personal secret of n_i and the broadcasted rekeying message from GC. However, for any node in Rev it is computationally infeasible to determine the session key. What any node n_i learns from broadcast rekeying message, it cannot be determined from broadcasts or personal keys alone. Let a group of k nodes is defined as $n_1, n_2, \ldots, n_k$. If either the set of m broadcasts $\{B_1, \ldots, B_m\}$ or the set of k personal keys $\{S_1, \ldots, S_k\}$ are considered separately, it is computationally infeasible to compute session key SK_j (or other useful information) from either set.

Let m denote the total number of sessions in the life cycle of the group communication. Each node is assigned a pre-arranged life cycle (t_1, t_2), which depends on the time of joining. In other words, it can be said that each node will participate in the group communication for $k = t_2 - t_1$ number of sessions. Due to which, once a node's life cycle is expired, it is automatically detached from the group session without requiring the direct intervention of the GC.

For a group with life cycle $(0, m)$, the group key for session j is as follows:

$$SK_j = K_j^F + K_{m-j+1}^B \tag{5.1}$$

where K_j^F is the forward key and K_{m-j+1}^B is the backward key for session j.

5.2 Security Properties

A rekeying scheme should provide the following types of security. A rekeying protocol provides forward secrecy if for any set $Rev \subseteq G$, where all $n_l \in Rev$ are revoked before session j, it is computationally infeasible for the members in Rev to get any information about SK_i for all $i \geq j$, even with the knowledge of session keys $\{SK_1, \ldots, SK_{j-1}\}$ before session j.

A rekeying protocol provides backward secrecy if for any set $J \subseteq G$, where all $n_l \in J$ are newly joined nodes after session j, it is computationally infeasible for the members in J to get any information about SK_i for all $i \leq j$, even with the knowledge of group keys $\{SK_{j+1}, \ldots, SK_m\}$ after session j.

A rekeying protocol is key-independent, if it is both forward-secret and backward-secret.

5.3 Proposed Scheme

This section presents the details of proposed scheme of self-healing key distribution with time limited node revocation capability. First, nodes are divided into clusters by using the cluster organization phase discussed already in Section 3.4.2 of Chapter 3, where each cluster is managed by the CH. However, the clusters are dynamic and regrouping is done after specific duration. The details of cluster formation is not discussed in this chapter. In the rest of chapter the cluster will be referred as group and cluster head(CH) will be referred as group controller(GC).

The group life cycle is given by m, which determines the total number of sessions for a group. The GC uses the pseudorandom number generator (PRNG) of a large enough period to produce a sequence of m random numbers $(r_1, r_2, \ldots, r_m)$. The GC randomly picks two initial key seeds, the forward key seed S^F and the backward key seed S^B. In the pre-processing time, it computes two hash chains of equal length m by repeatedly applying the same one-way hash function on each seed. For $K_0^F = S^F$ and $K_0^B = S^B$, the hash sequences are generated as given in Equation 5.2 and Equation 5.3.

$$\{K_0^F, H(K_0^F), \ldots, H^i(K_0^F), \ldots, H^{m-1}(K_0^F), H^m(K_0^F)\} \tag{5.2}$$

$$\{K_0^B, H(K_0^B), \ldots, H^i(K_0^B), \ldots, H^{m-1}(K_0^B), H^m(K_0^B)\} \tag{5.3}$$

During the initial configuration setup, each node n_i is first assigned a prearranged life cycle (t_1, t_2) where $t_1 \geq 1$ and $t_2 \leq m$. n_i will participate in the group communication $k = t_2 - t_1 + 1$ number of sessions. The node n_i joins the group at time t_1 in session p and will have to leave the group at time t_2 in session q, where $q > p$.

The node n_i receives its personal secret from GC consisting of: 1) a forward key in session p i.e. K_p^F, and 2) k number of random numbers corresponding to the sessions in which node n_i will participate in the group communication. Further, GC securely sends the personal secret to n_i using key encryption key KEK_i shared between n_i and GC (using the method described in Section 3.4.3 of Chapter 3) , as given in Equation 5.4:

$$GC \rightarrow n_i : E_{KEK_i}(K_p^F, (r_p, r_{p+1}, \ldots, r_q)), MAC(K_p^F \| (r_p, r_{p+1}, \ldots, r_q)) \tag{5.4}$$

The node n_i decrypts the message by its corresponding KEK_i to retrieve its secret. In the j-th session the GC locates the backward key K_{m-j+1}^B in the backward key chain and computes the broadcast message as given in Equation 5.5.

$$B_j = G(K_{m-j}^B, r_j) \tag{5.5}$$

When the nodes receive the broadcast message B_j, the session key is generated. First, when any node n_i in the group receives the broadcast message, it recovers the backward key K_{m-j+1}^B for session j from B_j, by applying XOR on both B_j and r_j, as given in Equation 5.6.

$$K_{m-j+1}^B = B_j \oplus r_j \tag{5.6}$$

From Equation 5.5 and Equation 5.6:

$$K^B_{m-j+1} = G(K^B_{m-j}, r_j) \oplus r_j \tag{5.7}$$

By substituing the value of G i.e. $G(x, y) = H(x) \oplus y$, backward key is given as given in Equation 5.8

$$K^B_{m-j+1} = H(K^B_{m-j}) \oplus r_j \oplus r_j \tag{5.8}$$

The backward key is obtained:

$$K^B_{m-j+1} = H(K^B_{m-j}) \tag{5.9}$$

Second, the node n_i computes the $j - th$ forward key by applying one-way hash function on its forward key K^F_p as given in Equation 5.10.

$$K^F_j = H^{j-p}(K^F_p) \tag{5.10}$$

Finally, the node n_i computes the current session key SK_j as given in Equation 5.11.

$$SK_j = K^F_j + K^B_{m-j+1} \tag{5.11}$$

Table 5.1 illustrates the example for nodes joining and leaving the group. There are 6 number of nodes $n_1, n_2, \cdots n_6$, where each node has a life cycle(1,3) of 3 sessions. In other words, the node will enter the group in session 1 and will leave the group after session 3. Further, the group life cycle m is 7, which means that the total number of sessions for group life time is 7.

Initially, nodes n_1 and n_3 request GC to join the group. The GC transmits n_1 and n_3 their personal secrets consisting of forward key K^F_1 and 3 random numbers for node's life cycle (r_1, r_2, r_3),

Table 5.1: Example of Nodes Join/Leave Group: m=7, Node Life Cycle = 3

Session	n_1	n_2	n_3	n_4	n_5	n_6
1	$K_1^F, (r_1, r_2, r_3)$ $\surd$	–	$K_1^F, (r_1, r_2, r_3)$ $\surd$	–	–	–
2	$\surd$	–	$\surd$	$K_2^F, (r_2, r_3, r_4)$ $\surd$	–	–
3	$\surd$	$K_3^F, (r_3, r_4, r_5)$ $\surd$	$\surd$	$\surd$	$K_3^F, (r_3, r_4, r_5)$ $\surd$	–
4	$\times$	$\surd$	$\times$	$\surd$	$\surd$	$K_4^F, (r_4, r_5, r_6)$ $\surd$
5	$\times$	$\surd$	$\times$	$\times$	$\surd$	$\surd$
6	$\times$	$\times$	$\times$	$\times$	$\times$	$\surd$
7	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

where secrets are encrypted with their corresponding key-encryption keys. The nodes participation in the current session is represented by a check mark ($\surd$) symbol and the nodes that have not yet joined the network are represented by a dash (–) line. The first row shows that nodes n_1 and n_3 have joined the group whereas n_2, n_4, n_5, n_6 have not yet joined. Then, n_4 joins in session 2. Similarly, n_2 and n_5 join in session 3.

In session 4, n_1 and n_3 detach from the group, which is shown as a cross ($\times$) symbol in the table; however, n_2, n_4, and n_5 remain active in the group communication. Further, n_6 also joins the group. In session 5, n_4's life time expires. In session 6, only n_6 is active. Finally, by the end of session 7, all nodes are detached from the group.

5.3.1 Adding a Group Member

When a node n_i wants to join an active group, first it obtains the permission to attach to the group communication from the GC. If it is successful, n_i establishes a common secret key KEK_i shared with the GC. GC assigns a life cycle to n_i i.e. t_1, t_2. Then, the GC sends the current system configuration to n_i using an InitGroupKey message, as given in Equation 5.12.

$$GC \rightarrow n_i : E_{KEK_i}(K_p^F, (r_p, r_{p+1}, \ldots, r_q)), MAC(K_p^F \| (r_p, r_{p+1}, \ldots, r_q)) \tag{5.12}$$

where K_p^F and $(r_p, r_{p+1}, \ldots, r_q)$ are shared personal secret for n_i with life cycle (t_1, t_2). Upon receiving the broadcast message from GC, n_i computes the current session key and participates in the network communication.

5.3.2 Node Revocation

A node with a life cycle (t_1, t_2) detaches from the group at time t_2. Figure 5.1 shows a time line to illustrate node life cycle and revocation. The node participates only between t_1 and t_2, where the interval is divided into a l number of sessions. Further, as each node is assigned with a $l = t_2 - t_1 + 1$ number of random numbers, it cannot derive the session keys $SK_t = K_t^F + K_{m-t+1}^B$ for $t < t_1$ and $t > t_2$. These l random numbers correspond to the sessions in which the node participate in the group. So, these random numbers can be used for specified sessions only and cannot be used for the remaining sessions. In order to recover K_{m-t+1}^B at time t from the B_t, it requires r_t, which is not available. Thus, a time limited node revocation is achieved implicitly without any intervention from the GC. As a result, the communication and the computation overhead on the GC and group nodes are remarkably reduced.

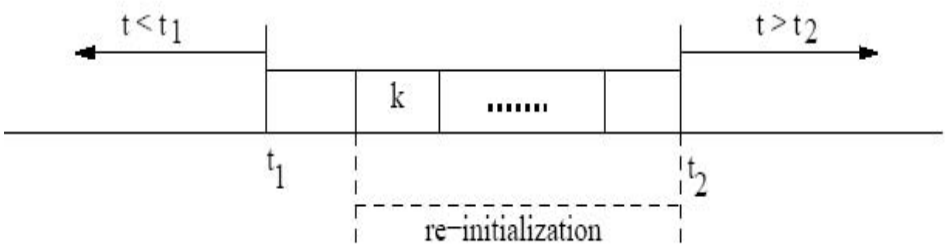


Figure 5.1: Node Revocation

5.3.2.1 Compromised Node

If a compromised node is detected, all nodes are forced to be re-initialized. Let n_i with life cycle (t_1, t_2) is compromised in session k, where $t_1 < k < t_2$, as shown in Figure 5.1. The GC re-initializes the group communication system by re-computing a new random number sequence of length $t_2 - k + 1$ and then unicast it to all group nodes securely.

5.4 Analysis

This section shows that the proposed scheme realizes self-healing key distribution scheme with time limited revocation capability. Further, the forward and backward secrecy is assured with the time-limited node revocation.

A group of sensor nodes is considered as G, $G = \{n_1, n_2, \ldots, n_N\}$, Rev represents a set of revoked nodes $Rev \subseteq G$, J represents a set of newly joining nodes $J \subseteq G$, and m is the total number of sessions in the group life cycle.

5.4.1 Self-healing Property

Consider a node $n_k \in G$ with life cycle (t_1, t_2), which means that n_k joins the group at t_1 (session p) and leaves the group at time t_2 (session q), where $1 \leq p \leq q$, as shown in Figure 5.2.

Suppose node n_k goes offline in session $p + 1$ and comes online again in session $p + j$ where $(p + j) < q$, as shown in Figure 5.2. As a result, the node n_k will miss the broadcast messages $B_{p+1} \cdots B_{p+j-1}$ from GC; hence, the session keys $SK_{p+1} \cdots SK_{p+j-1}$ will not be available. When node n_k comes online in session $p + j$, it receives the broadcast message B_{p+j} from GC and recovers the backward key $K^B_{m-(p+j)+1}$ for session $p+j$. So, it can obtain the sequence of backward keys $\{K^B_{m-(p+j-1)+1} \ldots K^B_{m-(p+1)+1}\}$ by repeatedly applying H on $K^B_{m-(p+j)+1}$. The node n_k also holds the forward key $K^F_p = H^p(K^F_0)$ of the session p, and hence can obtain the sequence of forward keys $\{K^F_{p+1}, \ldots, K^F_{p+j}\}$ by repeatedly applying H on K^F_p. Now n_k can find

Session #	Forward key	Backward Key	
1	$H(K_0^F)$	$H^m(K_0^B)$	
2	$H^2(K_0^F)$	$H^{m-1}(K_0^B)$	
$\vdots$	$\vdots$	$\vdots$	
p	$H^p(K_0^F)$	$H^{m-(p)+1}(K_0^B)$	$\longleftarrow t_1$
$p+1$	$H^{p+1}(K_0^F)$	$H^{m-(p+1)+1}(K_0^B)$	offline
$\vdots$	$\vdots$	$\vdots$	
$p+j-1$	$H^{p+j-1}(K_0^F)$	$H^{m-(p+j-1)+1}(K_0^B)$	
$p+j$	$H^{p+j}(K_0^F)$	$H^{m-(p+j)+1}(K_0^B)$	online
$\vdots$	$\vdots$	$\vdots$	
q	$H^q(K_0^F)$	$H^{m-(q)+1}(K_0^B)$	$\longleftarrow t_2$
$\vdots$	$\vdots$	$\vdots$	
$m-1$	$H^{m-1}(K_0^F)$	$H^2(K_0^B)$	
m	$H^m(K_0^F)$	$H(K_0^B)$	

Figure 5.2: Self-healing in Node Life Cycle

all the session keys from session $p+1$ to session $p+j$ without requiring any extra information from GC.

5.4.2 Key Independence

The proposed scheme also meets the security requirement for forward and backward secrecy, which gives key independence. Informally, forward-secrecy means that the compromise of one or more secret keys does not compromise previous secret keys. Likewise, backward-secrecy refers to that the compromise of one or more secret keys does not compromise future secret keys. Key-independence means that the secret keys used in different sessions are basically independent. Thus, even if the attacker finds out the secret key of a certain session, it does not give any advantage in finding the secret keys of other sessions.

5.4.2.1 Forward Secrecy

It is shown that a single revoked node or a collusion of revoked nodes cannot learn anything about the future group keys since the secrets they knew while they were authorized member of the group

will no longer be used in any future rekeying message.

Let Rev be the set of revoked nodes and all nodes $n_k \in Rev$ are revoked before the current session j. The node n_k cannot get any information about the current session key SK_j even with the knowledge of $\{SK_i, SK_{i+1}, \ldots, SK_{j-1}\}$ before session j, where i is the earliest session of all the nodes in Rev, or in other words, i is the minimum for all t_1's of nodes in Rev. In order to find SK_j, node n_k needs the random number r_j of that session and that r_j will not be available to n_k. Also, because of the one-way property of H, it is computationally infeasible to compute $K^B_{j_1}$ from $K^B_{j_2}$ for $j_1 < j_2$. The nodes in Rev may know the sequence of backward keys $K^B_m, \ldots, K^B_{m-j+2}$; however, they cannot compute K^B_{m-j+1} in order to find current session key SK_j.

5.4.2.2 Backward Secrecy

Let J is the set of nodes that join the group in session j. The collusion of newly joining nodes cannot get any information about any previous session keys before session j even with the knowledge of group keys after session j. Each $n_k \in J$ when joins the group, GC gives it $j-th$ forward key i.e. K^F_j, instead of initial forward seed K^F_0. As $K^F_j = H(K^F_{j-1})$, it is computationally infeasible for n_k to compute the previous forward keys which are required to compute session keys before current session j. Hence, the proposed scheme is backward secure.

5.4.3 Storage Requirements

In our scheme the GC and all nodes do not need any encryption/decryption process of a re-keying message to update session keys. All computation needed for re-keying is one-way hash function and XOR operation, and all information needed for re-keying is in the current transmission and the initial information.

This research implement two one-way hash algorithms, SHA-1 and MD5 using nesC [87] programming language in TinyOS for Moteiv's Tmote Sky sensors. This implementation have con-

sidered voltage level of 3 volts and nominal current (with Radio off) as 1.8×10^{-3} amps, as given in Tmote Sky's data sheet [96]. It takes data stream of 64 bytes. As shown in Table 5.2, for SHA-1 the code consumes 128 bytes of RAM, 4048 bytes of ROM, takes approximately 10.5 ms to produce a 160-bit hash of a 64-byte message, and the energy consumption is 56.94 μJoules. MD5 produces a 128-bit message digest for a given data stream. The code consumes 176 bytes of RAM, 12.5 KB of ROM, takes approximately 5.75 ms to hash a message of 64 bytes using 64-byte blocks, and the energy consumption is 31.09 μJoules. The above implementation shows that SHA-1 consumes less memory than MD5; however, it's processing overhead is almost double than MD5.

Table 5.2: Time and Memory Requirements for Tmote Sky

Algorithm	Time (seconds)	RAM (bytes)	ROM (bytes)	Energy (Joules)
SHA-1	10.545×10^{-3}	128	$4,048$	56.94×10^{-6}
MD5	5.757×10^{-3}	176	$12,500$	31.09×10^{-6}

5.5 Conclusion

Efficient solutions for the problem of key distribution are essential for the feasibility of secure group communication in sensor networks. In this chapter, a key distribution scheme for secure group communication is developed in HSNs. The scheme provides a self-healing mechanism for session key-recovery on possible packet loss in the lossy environment using one-way key chain. Other features include periodic re-keying of group key and time-limited group node revocation. The session keys are updated periodically, where the update is performed regardless of changes in network (group) topology. Periodic rekeying significantly reduces both the computation and communication overhead at the GC and the nodes, and thus improves the scalability and perfor-

mance of the proposed scheme. Further, the time-limited node revocation is achieved without any intervention from the GC.

The analysis shows that the proposed scheme is computationally secure and meets the security requirements for forward and backward secrecy. The implementation of two one-way hash algorithms SHA-1 and MD5 on resource constraint sensor nodes (Tmote Sky) shows the feasibility of the proposed scheme for current wireless sensor network technology. Hence, the scheme results scalable, and particularly attractive for large dynamic groups.

Chapter 6

CONCLUSION

Wireless sensor network (WSN) is an emerging research field with several interesting application domains ranging from battlefield monitoring to environmental observation. Secure communication in sensor networks is the biggest constraint in the successful deployment of sensors. Security is not, however, the only hurdle in applicability of WSN. Based on resource limited nature of sensors, efficient energy and memory consumption and speed efficiency are also some of the issues that need to be catered for. The most important requirement in WSN scenario is the adjustment of security level according to the availability of resources i.e. achieving a level where system security and system efficiency both are acceptable.

This research presents a key management architecture based on random key pre-distribution for hierarchical HSN. The novelty of this work lies in the introduction of a specialized key generation process which significantly reduces the storage requirements in resource constrained sensor networks and also provide better resiliency against node capture as compared to other random key pre-distribution schemes.

This research also proposes an efficient collusion resistant security mechanisms for sensor networks. This is the first work that aims to provide a countermeasure to collusion attacks on random key pre-distribution schemes.

Further a group key distribution scheme for secure multicast communications is presented that is optimized for self-healing , and designed to facilitate secure data aggregation, an important operation primitive in HSNs. It present a strategy for securely distributing rekeying messages and specify techniques for joining and leaving a group. Nodes are capable of recovering lost session

keys on their own, without requesting additional transmission from the group controller. The proposed scheme provides both backward and forward secrecy.

Based on these contributions, now review what have achieved in broader terms.

To start with, a key management scheme is proposed based on random key pre-distribution in respect to all the three phases of pre-deployment, at deployment and post-deployment. In a key generation process, instead of generating a large pool of random keys, a key pool is represented by a small number of generation keys. For a given generation key and publicly known seed value, a one-way hash function generates a key chain, and these key chains collectively make a key pool. Each sensor node is assigned a small number of randomly selected generation keys. The results show that our scheme can significantly reduce the storage requirements as compared to other random key pre-distribution schemes. For instance, storage requirements can be reduced by 8 times as compared to AP Du [48], and 33 times as compared to basic scheme Eschenauer and Gligor [42]. Also, the resiliency against node capture is better than previous key pre-distribution schemes. The TinyOS implementation shows that the proposed scheme can be efficiently implemented in real sensor networks. HMAC-SHA1 and HMAC-MD5 are compared to generate key chains. The results show that although HMAC-SHA1 consumes less memory resources than HMAC-MD5, it is more computationally intensive. The ease of implementation, the low resource requirement, combined with the level of security it provides, should be appealing to any WSN developer that is concerned by the complexity and resource requirement of many current schemes.

A special type of collusion attacks is identified on random key pre-distribution schemes and propose a mechanism which shows that collusion attacks on random key pre-distribution are avoided by deleting the initial key rings from nodes' memory after generating the shared pairwise key with neighbors. Our propose mechanism also allow new nodes to join the system once initialization is complete and initial key ring has been destroyed from nodes memory. The analysis shows that

even if a large number of nodes are compromised, an adversary can only exploit a small number of keys near the compromised nodes, while other keys in the network remain safe.

A computationally secure and efficient group key distribution scheme is provided with self-healing property and time-limited node revocation capability for large and dynamic groups over insecure HSNs. Other features include periodic re-keying of group key and time-limited group node revocation. The group keys are updated periodically, where the update is performed regardless of changes in network (group) topology. Periodic rekeying significantly reduces both the computation and communication overhead at the GC and the nodes, and thus improves the scalability and performance of the proposed scheme. Further, the time-limited node revocation is achieved without any intervention from the GC. The results show that the proposed scheme can tolerate high channel loss rate, and hence make a good balance between performance and security, which is suitable for HSN applications.

6.1 Future Work

This thesis provides a suite of effective key management techniques that will be of significant value to designers of future HSNs and help to ensure that their deployment provides a safe and reliable environment for their many useful and important applications. As future work, it is proposed that current research work in collision resistant key management methodologies can be extended to deal with dynamic topologies. Furthermore, our proposed secure group communication scheme can be extended to overcome the constraint of time limited node revocation. A secure routing structure can also be devised as an application of our proposed key management scheme so that the network throughput can be increased (for given radio link capabilities), data packet delay can be improved, and routing overhead can be reduced. As a final thought, there is a plan to carry out large scale deployment of the proposed schemes for critical applications to unveil other deficiencies which could not be dealt with under current limitations.

APPENDIX 1

Definition 1 *A one-way hash function H can map an input M of the arbitrary length to an output of the fixed length, which is called hash value h : $h = H(M)$, where the length of M is m-bits. One-way hash function H has the following properties [97]:*

- *Given a hash value h, it is computationally infeasible to find the input M such that $H(M) = h$*
- *Given an input M, it is computationally infeasible to find a second input $\acute{M}$ such that $H(\acute{M}) = h$, where $\acute{M} \neq M$*

Definition 2 *A cryptographically secure one-way keyed hash function $HMAC$ has the following property: for $y = HMAC(k, x)$, 1) given x, it is computationally infeasible to find y without knowing the value of k; 2) given y and k, it is computationally infeasible to find x.*

Definition 3 *(Key Graph) Let V represent all the nodes in the sensor network. A Key-Sharing graph G(V,E) is constructed in the following manner: For any two nodes i and j in V , there exists an edge between them if and only if nodes i and j have at least one common key in their key ring. Note that $|V|$ = n for a WSN of size n, the key graph G(V;E) is connected if and only if any two nodes i and j belonging to V can reach each other via edge set E only.*

Definition 4 *A pseudo-random function is an efficient (deterministic) algorithm which given an h-bit seed,y, and an h-bit argument, x, returns an h-bit string, denoted $f_y(x)$, so that it is infeasible to distinguish the responses of f_y , for a uniformly chosen y, from the responses of a truly random function.*

Definition 5 *Let H be a one-way hash function and s be a random seed. Then, a hash chain can be deduced by iteratively hashing s, which can be written as: $H^i(s) = H(H^{(i-1)}(s)), i = 1, 2, \ldots$*

where, s is regarded as "trust anchor" of the one-way hash chain. The hash chain includes a sequence of hash values, which can be denoted by $h_1 = H(s), h_2 = H(h_1), \ldots, h_i = H(h_{i-1}), i = (1, 2, \ldots)$

Definition 6 *Let $G(x, y) = H(x) \oplus y$, where H is a one-way hash function and $\oplus$ denotes the bitwise XOR. Given x and $G(x, y)$, without the knowledge of y, it is computationally infeasible to find $\acute{y}$ such that $G(x, \acute{y}) = G(x, y)$*

Terms

- *Key Pool*: A key pool K is a large pool of random symmetric keys.
- *Key Chain*: A key chain C is a subset of K, $C \subseteq K$. Each key chain is generated independently via a unique generation key and a publicly known seed S by applying a keyed hash algorithm repeatedly. A publicly known seed value is same for every key chain. Each key chain can be uniquely identified as C_i, where $i = 0 \ldots M - 1$. The key pool K consists of M equal sized key chains, as given in Equation 1.

$$K = C_0 \cup C_1 \ldots \cup C_{M-1} \tag{1}$$

- *Key Ring*: A key ring R consists of randomly selected generation keys of corresponding key chains. Each sensor node is assigned a ring of R keys.

BIBLIOGRAPHY

[1] T. Gao, L. K. Hauenstein, A. Alm, D. Crawford, C. K. Sims, A. Husain, and D. M. White, "Vital signs monitoring and patient tracking over a wireless network.," Conf Proc IEEE Eng Med Biol Soc 1 (2005).

[2] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, T. H. Qing Cao, J. A. Stankovic, T. Abdelzaher, and B. H. Krogh, "Lightweight detection and classification for wireless sensor networks in realistic environments," In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pp. 205–217 (ACM, New York, NY, USA, 2005).

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," IEEE Communications Magazine (2002).

[4] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: mobile networking for Smart Dust," In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 271–278 (ACM, New York, NY, USA, 1999).

[5] in *ESAS*, Vol. 3313 of *Lecture Notes in Computer Science*, C. Castelluccia, H. Hartenstein, C. Paar, and D. Westhoff, eds., (Springer, 2005).

[6] F. Zhao and L. J. Guibas, *Wireless sensor networks : an information processing approach*, *The Morgan Kaufmann series in networking.* (Morgan Kaufmann, Amsterdam ; San Francisco, 2004).

[7] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless

sensor networks," In *International Conference on Embedded Networked Sensor Systems (SenSys 2003)*, pp. 1–13 (2003).

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," SIGPLAN Not. **35,** 93–104 (2000).

[9] J. Hill, *System Architecture for Wireless Sensor Networks, PhD Thesis* (University of California, Berkeley, 2003).

[10] R. Cardell-Oliver, K. Smettem, M. Kranz, and K. Mayer, "Field testing a wireless sensor network for reactive environmental monitoring," In *International Conference on Intelligent Sensors, Sensor Networks & Information*, (IEEE Computer Society, 2004).

[11] X. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway, "A survey of key management schemes in wireless sensor networks," Computer communications (2007).

[12] E. R. S. Basagni, K. Herrin and D. Bruschi, "Secure Pebblenets," In *International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 156–163 (ACM, 2001).

[13] S. S. S. Zhu and S. Jajodia, "LEAP: Efficient security mechanisms for large-scale distributed sensor networks," In *CCS*, pp. 62–72 (ACM, 2003).

[14] A. Perrig, R. Szewczyk, J. Tygar, Victorwen, and D. E. Culler, "Spins: Security protocols for sensor networks," In *Seventh Annual Int'l Conf. on Mobile Computing and Networks*, (2001).

[15] A. W. H. E. N. Gura, A. Patel and S. Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," In *Workshop on Cryptographic Hardware and Embedded Systems*, (2004).

[16] G. Gaubatz, J.-P. Kaps, and B. Sunar, "Public Key Cryptography in Sensor Networks - Revisited," In *ESAS*, pp. 2–18 (2004).

[17] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: securing sensor networks with public key technology," In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pp. 59–64 (ACM, New York, NY, USA, 2004).

[18] M. D. Malan and M.Smith, "A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography," In *First IEEE International Conference on Sensor and Ad Hoc Communications and Networks*, (2004).

[19] H. Chan, A. Perrig, and D. Song, "Random Key Pre-Distribution Schemes For Sensor Networks," In *IEEE Symposium on Security and Privacy*, pp. 197–213 (2003).

[20] L. B. Oliveira, H. C. Wong, M. Bern, R. Dahab, and A. A. F. Loureiro, "Sec LEACH: A Random Key Distribution Solution for Securing Clustered Sensor Networks," In *5th IEEE international symposium on network computing and applications*, pp. 145–154 (2006).

[21] H. Chan, A. Perrig, and D. Song, "Random Key Pre-Distribution Schemes For Sensor Networks," In *IEEE Symposium on Research in Security and Privacy*, (2003).

[22] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing Pairwise Keys For Secure Communication In Ad Hoc Networks: A Probabilistic Approach," In *11th IEEE International Conference on Network Protocols (ICNP'03)*, (2003).

[23] R. D. Pietro, L. V. Mancini, and A. Mei, "Random Key Assignment Secure Wireless Sensor Networks," In *1st ACM workshop on Security of Ad Hoc and Sensor Networks*, (2003).

[24] Y. Cheng and D. P. Agrawal, "Efficient pairwise key establishment and management in static wireless sensor networks," In *Second IEEE International Conference on Mobile ad hoc and Sensor Systems*, (2005).

[25] K. Ren, K. Zeng, and W. Lou, "A New Approach For Random Key Pre-Distribution In Large-Scale Wireless Sensor Networks," Wireless communication and mobile computing **6,** 307–318 (2006).

[26] S. P. Miller, C. Neuman, J. I. Schiller, and J. H. Saltzer, "Kerberos authentication and authorization system," In *Project Athena Technical Plan, Section E.2.1*, (1987).

[27] J. Kohl and B. C. Neuman, "The Kerberos Network Authentication Service (V5)," In *RFC 1510*, (1993).

[28] F. Kausar and A. Masood, "An Analysis of Public Key Cryptography based Techniques in Wireless Sensor Networks," In *International Multi-Conference of Engineers and Computer Scientists*, (Hong Kong, 2007).

[29] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 162–175 (ACM Press, New York, NY, USA, 2004).

[30] M. Rosing, "Implementing Elliptic Curve Cryptography," In *Ed. Manning Publications Co*, (1999).

[31] E.-O. Blab and M. Zitterbart, "Towards Acceptable Public-Key Encryption in Sensor Networks," In ,

[32] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks," In *PERCOM '05: Proceedings of the*

Third IEEE International Conference on Pervasive Computing and Communications, pp. 324–328 (IEEE Computer Society, Washington, DC, USA, 2005).

[33] P. K. A. Freier and P. Kocher, "The SSL Protocol Version 3.0," In ,

[34] A. Juels and J. Guajardo, "RSA Key Generation with Verifiable Randomness," In *PKC '02: Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems*, pp. 357–374 (Springer-Verlag, London, UK, 2002).

[35] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," SIGOPS Oper. Syst. Rev. **35,** 146–159 (2001).

[36] J. Abraham and K. S. Ramanatha, "Security Protocols for Wireless Sensor Networks based on Tiny Diffusion and Elliptic Curves," In *IASTED international conference Network and Communication Systems*, (2006).

[37] "Recommended elliptic curves for Federal Government Use," In *National Institute of Standards and Technology*, (1999).

[38] W. Diffie and M. E. Hellman, "New Directions in Cryptography," IEEE Transactions on Information Theory **IT-22,** 644–654 (1976).

[39] R. M.O, "Digitalized signatures and public key functions as intractable as factorization," Mit/lcs/tr-212,Massachusetts Institute of Technology (1979).

[40] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A Ring-Based Public Key Cryptosystem," In *ANTS-III: Proceedings of the Third International Symposium on Algorithmic Number Theory*, pp. 267–288 (Springer-Verlag, London, UK, 1998).

[41] A. K. Lenstra and E. R. Verheul, "Selecting Cryptographic Key Sizes," In *PKC '00: Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography*, pp. 446–465 (Springer-Verlag, London, UK, 2000).

[42] L. Eschenauer and V. D. Gligor, "A Key Management Scheme For Distributed Sensor Networks," In *ACM CCS*, (2002).

[43] E. D. Manley, J. Deogun, and H. A. Nahas, "Public-Key Cryptography in Sensor Networks," In *Fifth IASTED international Multi-Confernce Wireless Networks and Emerging Technologies*, (2005).

[44] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," In *10th ACM Conference on Computer and Communications Security (CCS 03)*, pp. 52–61 (2003).

[45] D. Liu, P. Ning, and R. Li, "Establishing pairwise keys in distributed sensor networks," ACM Trans. Inf. Syst. Secur. **8,** 41–77 (2005).

[46] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," In *IEEE Hawaii Int. Conf. on System Sciences*, pp. 4–7 (2000).

[47] G. Jolly, M. C. Kusçu, P. Kokate, and M. Younis, "A Low-Energy Key Management Protocol for Wireless Sensor Networks," In *ISCC '03: Proceedings of the Eighth IEEE International Symposium on Computers and Communications*, p. 335 (IEEE Computer Society, Washington, DC, USA, 2003).

[48] X. Du, Y. Xiao, M. Guizani, and H.-H. Chen, "An Effective Key Management Scheme for Heterogeneous Sensor Networks," Ad Hoc Networks **5,** 24–34 (2007).

[49] V. Bulusu, A. Durresi, V. Paruchuri, M. Durresi, and R. Jain, "Key Distribution in Mobile Heterogeneous Sensor Networks," In *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, pp. 1–5 (2006).

[50] P. Traynor, R. Kumar, H. B. Saad, G. Cao, and T. L. Porta, "Establishing Pair-Wise Keys in Heterogeneous Sensor Networks," In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1–12 (2006).

[51] P. Traynor, R. Kumar, H. B. Saad, G. Cao, and T. L. Porta, "Efficient Hybrid Security Mechanisms for Heterogeneous Sensor Networks," IEEE Transactions on Mobile Computing **6,** 663–677 (2007).

[52] K. Lu, Y. Qian, and J. Hu, "A framework for distributed key management schemes in heterogeneous wireless sensor networks," In *IEEE International Performance Computing and Communications Conference*, pp. 513–519 (2006).

[53] H. Kurnio, R. Safavi-Naini, and H. Wang, "A Secure Re-keying Scheme with Key Recovery Property," In *Proceedings of the 7th Australian Conference on Information Security and Privacy*, pp. 40–55 (Springer-Verlag, London, UK, 2002).

[54] L. Wang and C.-K. Wu, "Authenticated Group Key Agreement for Multicast," In *The 5th International Conference on Cryptology and Network Security*, (Springer-Verlag, 2006).

[55] J. H. Ki, H. J. Kim, D. H. Lee, and C. S. Park, "Efficient Multicast Key Management for Stateless Receivers," Information Security and Cryptology - ICISC 2002 **2587/2003,** 497–509 (2002).

[56] J. Pegueroles, W. Bin, M. Soriano, and F. Rico-Novella1, "Group Rekeying Algorithm Us-

ing Pseudo-random Functions and Modular Reduction," Grid and Cooperative Computing **3032/2004,** 875–882 (2004).

[57] Y. R. Yang, X. S. Li, X. B. Zhang, and S. S. Lam, "Reliable group rekeying: a performance analysis," In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 27–38 (ACM Press, New York, NY, USA, 2001).

[58] R. Poovendran and J. S. Baras, "An Information Theoretic Analysis of Rooted-Tree Based Secure Multicast Key Distribution Schemes," In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pp. 624–638 (Springer-Verlag, London, UK, 1999).

[59] G. Noubir, F. Zhu, and A. H. Chan, "Key Management for Simultaneous Join/Leave in Secure Multicast," In *Proceedings of MILCOM*, (2003).

[60] L. Gong and N. Shacham, "Multicast security and its extension to a mobile environment," Wirel. Netw. **1,** 281–295 (1995).

[61] D. Bruschi and E. Rosti, "Secure multicast in wireless networks of mobile hosts: protocols and issues," Mob. Netw. Appl. **7,** 503–511 (2002).

[62] T. Kostas, D. Kiwior, G. Rajappan, and M. Dalal, "Key Management for Secure Multicast Group Communication in Mobile Networks," In *Proceedings of DARPA Information Survivability Conference and Exposition*, (2003).

[63] T. Park and K. G. Shin, "LiSP: A lightweight security protocol for wireless sensor networks," Trans. on Embedded Computing Sys. **3,** 634–660 (2004).

[64] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," IEEE/ACM Trans. Netw. **8,** 16–30 (2000).

[65] G. C. D. Carman, B. Matt, "Energy-efficient and low-latency key management for MSN networks," In *Proceedings of 23rd Army Science Conference, Orlando FL*, (2002).

[66] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, and D. Dean, "Self-healing key distribution with revocation," In *Proceedings of IEEE Symposium on Security and Privacy*, pp. 241 – 257 (2002).

[67] C. Blundo, P. Darco, A. D. Santis, and M. Listo, "Design of Self-Healing Key Distribution Schemes," Des. Codes Cryptography **32,** 15–44 (2004).

[68] Y. Jiang, C. Lin, M. Shi, and X. Shen, "Self-healing group key distribution with time-limited node revocation for wireless sensor networks.," Ad Hoc Networks **5,** 14–23 (2007).

[69] R. Dutta, E.-C. Chang, and S. Mukhopadhyay, "Efficient Self-healing Key Distribution with Revocation for Wireless Sensor Networks Using One Way Key Chains," In *Proceedings of 5 th International Conference on Applied Cryptography and Network Security (ACNS'07)*, (2007).

[70] O. Goldreich, *Foundations of Cryptography: Basic Tools* (Cambridge University Press, New York, NY, USA, 2000).

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

[72] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer, "A System for Simulation, Emulation, and Deployment of Heterogeneous Sensor Networks," In *2nd international conference on Embedded networked sensor systems*, (2004).

[73] L. Lazos and R. Poovendran, "Stochastic Coverage in Heterogeneous Sensor Networks," ACM Transactions on Sensor Networks (TOSN) **2,** 325–358 (2006).

[74] Y. Ma, S. Dala1, M. Alwan, and J. Aylor, "ROP: A Resource Oriented Protocol for Heterogeneous Sensor Networks," In *Virginia Tech Symposium on Wireless Personal Communications*, (2003).

[75] X. Du and F. Lin, "Maintaining Differentiated Coverage in Heterogeneous Sensor Networks," EURASIP Journal on Wireless Communications and Networking pp. 565–572 (2005).

[76] E. J. Duarte-Melo and M. Liu, "Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks," In *Proceedings of IEEE Globecom*, (2002).

[77] A. Woo and D. E. Culler, "A transmission control scheme for media access in sensor networks," In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pp. 221–235 (ACM Press, New York, NY, USA, 2001).

[78] X. Du and Y. Xiao, "Energy Efficient Chessboard Clustering and Routing in Heterogeneous Sensor Network," International Journal of Wireless and Mobile Computing **1,** 121–130 (2006).

[79] X. Du, M. Guizani, Y. Xiao, and H.-H. Chen, "Two Tier Secure Routing Protocol for Heterogeneous Sensor Networks.," IEEE Transactions on Wireless Communications **6,** 3395–3401 (2007).

[80] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," Computer **35,** 5462 (2002).

[81] J. R. Douceur, "The Sybil attack," In *IPTPS*, p. 251260 (2002).

[82] C. Karlof and D. Wagner, "Secure routing in sensor networks: Attacks and countermeasures," In *IEEE 1st Int. Workshop Sensor Network Protocols Applications*, p. 113127 (2003).

[83] Y. Hu, A. Perrig, and D. B. Johnson, "Wormhole detection in wireless ad hoc networks," Technical Report TR01-384, Department of Computer Science, Rice University (2002).

[84] X. Du, Y. Xiao, H.-H. Chen, and Q. Wu, "Secure cell relay routing protocol for sensor networks," Mobile Computing **6,** 375391 (2006).

[85] http://www.xbow.com, "Crossbow Technology Inc.," Processor/Radio Modules .

[86] www.xbow.com/products/Product_pdf_files/Wireless_pdf/Stargate_Datasheet.pdf, .

[87] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," SIGPLAN Not. **38,** 1–11 (2003).

[88] P. Cheng and R. Glenn, "Test Cases for HMAC-MD5 and HMAC-SHA-1," In *RFC 2202*, (1997).

[89] T. Moore, "A Collusion Attack on Pairwise Key Predistribution Schemes for Distributed Sensor Networks," In *PERCOMW '06: Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops*, p. 251 (IEEE Computer Society, Washington, DC, USA, 2006).

[90] F. Kausar, S. Hussain, J. H. Park, and A. Masood, "A Key Distribution Scheme Preventing Collusion Attacks in Ubiquitous Heterogeneous Sensor Networks," In *SecUbiq-07*, pp. 745–757 (LNCS, Springer, Taiwan, 2007).

[91] F. Kausar, S. Hussain, T. hoon Kim, and A. Masood, "Attack Resilient Key Distribution

Scheme for Distributed Sensor Networks," In *TRUST-07*, pp. 1–11 (LNCS, Springer, Taiwan, 2007).

[92] R. D. Pietro, L. V. Mancini, and A. Mei, "Energy efficient node-to-node authentication and communication confidentiality in wireless sensor networks," Wirel. Netw. **12,** 709–721 (2006).

[93] D. Wallner, E. Harder, and R. Agee, "Key Management for Multicast: Issues and Architectures," (1999).

[94] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast Security: A Taxonomy and Some Efficient Constructions," In *INFOCOMM'99*, (1999).

[95] F. Kausar, S. Hussain, J. H. Park, and A. Masood, "Secure Group Communication with Self-Healing and Rekeying in Wireless Sensor Networks," In *3rd International Conference on Mobile Ad Hoc and Sensor Networks (MSN 07),*, pp. 737–748 (LNCS, Springer, Beijing, China, 2007).

[96] http://www.moteiv.com/products/docs/tmote-sky datasheet.pdf, .

[97] NIST, "Secure Hash Standard," In *National Institute for Standards and Technology, Gaithersburg, MD, USA*, (1995).