2007-04-06

# Communication Security in Wireless Sensor Networks

Kui Ren
*Worcester Polytechnic Institute*

# Communication Security in Wireless Sensor Networks

by
Kui Ren

A Dissertation
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the
Degree of Doctor of Philosophy
in
Electrical and Computer Engineering

---

May, 2007

Approved:

| | |
|---|---|
| Prof. Wenjing Lou<br>ECE Department<br>Dissertation Advisor | Prof. Kaveh Pahlavan<br>ECE Department<br>Dissertation Committee |
| Prof. Berk Sunar<br>ECE Department<br>Dissertation Committee | Dr. Muxiang Zhang<br>Verizon Tech. Organization<br>Dissertation Committee |

Prof. Fred J. Looft
ECE Department Head

# Abstract

A wireless sensor network (WSN) usually consists of a large number of small, low-cost devices that have limited energy supply, computation, memory, and communication capacities. Recently, WSNs have drawn a lot of attention due to their broad applications in both military and civilian domains. Communication security is essential to the success of WSN applications, especially for those mission-critical applications working in unattended and even hostile environments. However, providing satisfactory security protection in WSNs has ever been a challenging task due to various network & resource constraints and malicious attacks. This motivates the research on communication security for WSNs.

This dissertation studies communication security in WSNs with respect to three important aspects. The first study addresses broadcast/multicast security in WSNs. We propose a multi-user broadcast authentication technique, which overcomes the security vulnerability of existing solutions. The proposed scheme guarantees immediate broadcast authentication by employing public key cryptography, and achieves its efficiency through integrating various techniques from different domains. We also address multicast encryption to solve data confidentiality concern for secure multicast. Utilizing the fact that sensors are both routers and end-receivers, we propose a lightweight multicast key management scheme that supports a broad range of multicast semantics.

The second study addresses data report security in WSNs. A location-aware end-to-end security framework for WSNs is proposed, in which secret keys are bound to geographic locations so that the impact of sensor compromise are limited only to their vicinity. The proposed scheme effectively defeats both bogus data injection attacks and various denial of service (DoS) attacks. In addition, we address event boundary detection as a specific case of secure data aggregation in WSNs. We propose a secure and fault-tolerant event boundary detection scheme, which is a localized statistic approach that detects the boundaries of large spatial events.

The third study addresses key management security in WSNs. We propose a keyed-hash-chain-based key pool generation technique for random key pre-distribution, which leads to a higher scheme efficiency and better security resilience against sensor compromise.

# Acknowledgements

There are three people without whom I am certain I never would have reached this far. They are my wife, Jie, my mother, Lan, and my father, Junshu. Pursuing and completing a PhD study in three years is a challenging mission and I cannot imagine I can do this without Jie's support and sacrifice. She always has belief in me and never stops encouraging me to go beyond. My parents are my strongest supporters for all these years. They love me more than themselves and have made enormous sacrifices to support me.

I would like to thank my advisor, Dr. Wenjing Lou. Her guidance, patience, and wisdom show me not only the way of doing research but also the way of being a decent and responsible society member. I would like to thank my committee members, Dr. Kaveh Pahlavan, Dr. Berk Sunar, and Dr. Muxiang Zhang, for their valuable feedbacks and comments on my research, from which I have benefited greatly. I would like to thank Dr. Fred J. Looft, our department head, who has been very supportive on my research and graduation issues. My special thanks also go to Dr. Yuguang Fang at University of Florida for his constant help and guidance. I would like to thank Dr. Yanchao Zhang, Dr. Jens Kaps, Dr. Bo Zhu, Dr. Xiaodong Lin, Dr. Zhiguo Wan, Mr. Kai Zeng, Mr. Chi Zhang, Mr. Yun Zhou, Mr. Shucheng Yu, and Mr. Benjamin Woodacre for their helpful discussions on various research ideas.

In addition, many others have indirectly helped me make it to this point. For brevity, I will mention only a few by name. Dr. Junren Gan, my supervisor at SIMIT, CAS, China, introduced the concept of modern cryptography and network security to me. Dr. Robert H. Deng provided me a precious research training opportunity at I2R, Singapore. Dr. Feng Bao, Dr. Jianying Zhou, and Dr. Yongdong Wu have helped me a lot during this period. Dr. Kwangjo Kim offered me an opportunity to study at ICU, Korea. I thank all of them for their help.

Finally, I also want to thank the National Science Foundation, Airsprite Corp., and WPI Institute Fellowship for funding my research.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

A wireless sensor network (WSN) is a wireless network consisting of a large number of spatially distributed sensor nodes. These sensor nodes can be easily deployed at strategic regions at a low cost. Equipped with various types of sensors, sensor nodes cooperate with each other to monitor physical or environmental conditions, such as temperature, sound, image, vibration, pressure, motion or pollutants. Each sensor node is also equipped with a radio transceiver or other wireless communication device, a microprocessor, and an energy source (e.g., a battery). Due to cost and size constraints, sensor nodes are usually resource limited with respect to their energy, memory, computational, and communication capacities. Table 1.1 shows the system parameters for several typical types of sensors [43].

The development of WSNs was originally motivated by military and homeland security applications such as battlefield surveillance. However, WSNs are now also widely applied in civilian application areas, including industrial sensing, environment and habitat monitoring, health-care applications, home automation, and traffic control. In the context of ubiquitous computing, WSNs can be used to perform ubiquitous information sensing, storing, and provide content delivering services. Due to their broad applications in both military and civilian domains, WSNs have drawn a

| Example | Processor | Memory | | Processor | | Radio | |
|---|---|---|---|---|---|---|---|
| | | ROM [KB] | RAM [KB] | Active [mW] | Sleep [$\mu$W] | Send [mW] | Receive [$\mu$W] |
| Spec | Custom 8-bit | 0 | 3 | 1.5 | | 1 | 900 |
| Mica2 | ATmega128L | 128 | 512 | 24 | <45 | 42 | 29 |
| Telos | TI MSP430 | 48 | 1024 | 6 | <15 | 35 | 38 |
| Imote | ARM7TDMI | 512 | 64 | 195 | 300 | incl. in Power | |
| BTnode | ATmega128L | 128 | 244 | 39.6 | 9.9 | 66 | 50 |
| Stargate | Intel PXA255 | 32,000 | 64,000 | < 2,500 | | | |

Table 1.1: System parameters for different types of sensors

lot of attention recently [12, 107, 20, 6, 13, 29, 63, 33, 5].

Communication security is essential to the success of WSN applications, especially for those mission-critical applications working in unattended and even hostile environments. To ensure that the network functions correctly and safely as purposed, the following are four major security requirements for WSNs [119, 75].

- Authenticity: Authenticity enables a sensor to make sure the identities of its communicating entities so that no adversary could masquerade another entity, and disseminate forged messages.

- Integrity: Integrity ensures that a message being transferred is never corrupted or modified by an adversary without being detected.

- Confidentiality: Confidentiality ensures that the content of the message being transferred is never disclosed to unauthorized entities. Network transmission of sensitive information, such as military information, requires confidentiality.

- Availability: Availability ensures the survivability of network services despite denial of service (DoS) attacks [119]. An DoS attack could be launched at

any layer of sensor networks and could be of various forms. At the physical and MAC layers, an adversary could employ jamming attacks. At the network layer, an adversary could disrupt the routing protocol and disconnect the network. At the application layer, an adversary may bring down high-level services such as network broadcast, multicast, data report, key management services, and so on.

Despite the importance, providing satisfactory security protection in WSNs has never been an easy task. This is because sensor networks not only suffer from various malicious attacks; but also are subject to many resource and network constraints as compared to traditional wireless networks. This motivates the research on communication security for WSNs [59, 12, 75, 17, 55, 56, 26, 25, 123, 35, 111, 120, 109, 11, 53, 94, 103, 81, 118, 117].

## 1.1 Motivation

The following unique features of WSNs make it particularly challenging to protect communication security in WSNs.

- Resource Constraints: As shown in Table 1.1, small sensors only have limited communication and computation capabilities, which makes it difficult to implement expensive security operations for WSNs. This precludes the direct transplantation of the existing security designs aimed for traditional wireless networks, where network nodes are much more powerful devices. Sensor nodes are battery-powered, having only limited energy supply. This again requires the security design to be efficient regarding both communication and computation overheads. In most cases, sensors only have very limited memory spaces, which further narrows down the security design choices. All these resource constraints require that the security design can only be efficient and lightweight; otherwise it will not be practical for WSNs.

- Network Constraints: WSNs use wireless open channel, therefore an adversary can easily eavesdrop all the network communications, as well as arbitrarily injecting messages and launching jamming attacks at different network layers. This means that the security design has to take into account both passive and active attacks. WSNs are distributed in nature, therefore centralized security solutions cannot be an option for WSNs. This also means that WSNs are vulnerable to various DoS attacks. WSNs are often very large in scale, which in turn imposes scalability requirement on the security design.

- Malicious Attacks: Give the large scale of the WSNs, it is impractical to protect or monitor each individual sensor node physically. In addition, sensors are also not tamper resistant[1]. Therefore, the adversary may capture and compromise a certain number of sensor nodes without being noticed and obtain all the secrets stored on these sensors. The adversary is thus able to launch a variety of malicious insider attacks against the network through these compromised nodes in addition to outsider attacks. For example, the compromised nodes may report bogus observations in order to mislead the network owner or users; they may also discard important messages such as data reports in order to hide some critical events from being noticed. All these attacks could cause severe results that may disable network functionality at least temporarily. Hence, it is highly important for the security design to be robust against sensor compromise and against both outsider and insider attacks.

All these malicious attacks, resource and network constraints, interleaved together, impose many challenging requirements for the security design in WSNs. Sophisticated techniques and careful design are demanded to balance among all these competing and sometimes even conflicting requirements as desired by the underlying applications. In

[1]Though using tamper-resistance hardware may help to protect security sensitive data on sensor nodes, this solution generally increases the cost of an individual sensor node dramatically [].

this thesis, we study communication security in WSNs with respect to three important aspects: broadcast/multicast[2] security, data report security, and random key predistribution.

### 1.1.1 Multicast Security

Multicast communication from a sink or network users to sensors is the most common communication paradigm in WSNs and of great importance, as it enables the sink/network users to disseminate query and control messages to the sensors and thus efficiently operate the WSN. Multicast security is hence one of the most important security services in WSNs [6, 75, 92, 15].

Recently, many schemes have been proposed to address the problem of broadcast authentication in WSNs [75, 92]. These schemes aim to provide efficient authentication solutions for the broadcast traffic, and hence ensure the message authenticity and prevent message fabrication and alteration attacks. Broadcast authentication in WSNs was first addressed by $\mu$TESLA [75]. In $\mu$TESLA, users of WSNs are assumed to be one or a few fixed sinks, which are always trustworthy. The scheme adopts a one-way hash function $h()$ and uses the hash preimages as keys in a message authentication code (MAC) algorithm. Initially, sensor nodes are preloaded with $K_0 = h^n(x)$, where $x$ is the secret held by the sink. Then, $K_1 = h^{n-1}(x)$ is used to generate MACs for all the broadcast messages sent within time interval $I_1$. During time interval $I_2$, the sink broadcasts $K_1$, and sensor nodes verify $h(K_1) = K_0$. The authenticity of messages received during time interval $I_1$ are then verified using $K_1$. This delayed disclosure technique is used for the entire hash chain and thus demands loosely synchronized clocks between the sink and sensor nodes. $\mu$TESLA is later enhanced in [57] to overcome the length limit of the hash chain. Most recently, $\mu$TESLA is also

[2]In the following, we do not distinguish broadcast from multicast for the purpose of this thesis unless necessary.

extended in [58] to support the multiuser scenario but the scheme assumes that each sensor node only interacts with a very limited number of users.

It is generally held that $\mu$TESLA-like schemes have the following shortcomings even in the single-user scenario: 1) all the receivers have to buffer all the messages received within one time interval; 2) they are subject to Wormhole attacks [34], where messages could be forged due to the propagation delay of the disclosed keys. However, here we point out a much more serious vulnerability of $\mu$TESLA-like schemes when they are applied in multi-hop WSNs. Since sensor nodes buffer all the messages received within one time interval, an adversary can hence flood the whole network arbitrarily. All he has to do is to claim that the flooding messages belong to the current time interval which should be buffered for authentication until the next time interval. Since wireless transmission is very expensive in WSNs, and WSNs are extremely energy constrained, the ability to flood the network arbitrarily could cause devastating DoS attacks. Moreover, this type of energy-depletion DoS attacks become more devastating in multiuser scenario as the adversary now can have more targets and hence more chances to generate bogus messages without being detected. Obviously, all these attacks are due to delayed authentication of the broadcast messages. In [34], TIK is proposed to achieve immediate key disclosure and hence immediate message authentication based on precise time synchronization between the sink and receiving nodes. However, this technique is not applicable in WSNs as pointed out by the authors. Therefore, multiuser broadcast authentication still remains a wide open problem in WSNs.

While multicast authentication has been extensively studied, there has been very little work addressing the problem of multicast encryption in the context of WSNs. Multicast encryption is orthogonal to multicast authentication; it provides message confidentiality and ensures that the message content can only be recovered by the intended receivers. The demand of multicast encryption is two-fold. First, it en-

sures message confidentiality and privacy. For example, the query message regarding the health status of patients should always be kept confidential from people other than the responsible doctors/nurses in the case of a health-oriented WSN such as CodeBlue [60]. Second, it minimizes the security risk (i.e., information leakage, key compromise) resulted from sensor compromise, which is unavoidable when the WSN is deployed in hostile environments. Hence, the problem of multicast encryption has to be addressed before multicast services can be deployed in practice. Designing an applicable multicast encryption scheme for WSNs is challenging. On the one hand, multicast services in WSNs have various semantics and are inherently multigroup oriented. On the other hand, WSNs usually have a large network size and sensors are resource-constrained and subject to potential compromise when deployed in hostile environments. These factors pose drastic efficiency and security requirements on the design of multicast encryption schemes.

### 1.1.2 Data Report Security

One of the most severe security threats in WSNs is security compromise of sensor nodes due to their lack of tamper resistance [17]. In WSNs, the attacker could compromise multiple nodes to obtain their carried keying materials and control them, and thus is able to intercept data transmitted through these nodes thereafter. As the number of compromised nodes grows, communication links between uncompromised nodes might also be compromised through malicious cryptanalysis. Hence, this type of attacks could lead to severe data confidentiality compromise in WSNs. Furthermore, the attacker may use compromised nodes to inject bogus data traffic in WSNs. In such attacks, compromised nodes pretend to have detected an event of interest within their vicinity, or simply fabricate a bogus event report claiming a non-existing event at an arbitrary location. Such *insider* attacks can severely damage network function and result in the failure of mission-critical applications. Such

attacks also induce network congestion and wireless contention, and waste the scarce network resources such as energy and bandwidth, hence, severely affecting both data authenticity and availability. Lastly, the attacker could also use compromised nodes to launch selective forwarding attack [44], in which case compromised nodes selectively drop the going-through data traffic and thus data availability can be severely damaged. The existence of aforementioned attacks, together with the inherent constraints of sensor nodes, makes it rather challenging to provide satisfying data security in WSNs with respect to all its three aspects, i.e., confidentiality, authenticity and availability [12, 107, 44, 75, 97].

Recent research has seen a growing body of work on security designs for WSNs [28, 17, 55, 56, 26, 25, 123, 35, 16, 120, 111, 109]. Due to the resource constraint, most of the proposals are based on symmetric cryptography and only provide data authenticity and/or confidentiality in a hop-by-hop manner. End-to-end encryption/authentication is considered less feasible, particular in a WSN consisting of a large number of nodes [17]. However, lack of the end-to-end security guarantee could make WSNs particularly vulnerable to the aforementioned attacks in many applications, where node-to-sink communication is the dominant communication pattern. This could give the attacker the advantage to obtain/manupulate its desired data at a much less effort without having to compromise a large number of nodes. To make things worse, existing security designs are highly vulnerable to many types of DoS attacks, such as report disruption attacks and selective forwarding attacks.

The fundamental application of WSNs is to monitor, detect, and report the occurrences of events of interest. In many applications [5, 63, 48, 23], this includes the detection of a large-scale spatial phenomenon such as the transportation front line of a contamination or the diagnosis of network health. Due to the strict resource limitations (e.g., battery power, bandwidth, etc.) of sensor nodes and the nature of some events, it is not feasible to collect all sensor measurements and compute

event boundaries in a centralized manner [23, 62]. A localized approach that allows in-network processing is therefore demanded. Sensor nodes are expected to collaborate with each other based on each own local view and provide a global picture for spatially distributed phenomena with greatly improved efficiency. Recently, several localized boundary detection schemes have been proposed [19, 48, 23, 71, 47]. All these schemes assume a trustworthy environment, and would fail in adversarial environments. Their resilience to node random measurement error is also very limited. However, for WSNs deployed in security-sensitive environments, it is critical for an event boundary detection scheme to be highly resilient against both node compromise and random faults.

### 1.1.3 Random Key Pre-distribution

In many applications, it is important to protect communications among sensor nodes to maintain message confidentiality and integrity. Recent research suggests that symmetric secret key pre-distribution is possibly the only practical approach for establishing secure channels among sensor nodes since the low-power sensor nodes have very limited computational capacity which excludes the applicability of computation-intensive public key cryptographic algorithms.

Recently, many random key pre-distribution schemes have been proposed [28, 17, 55, 56, 26, 25, 123, 35]. Random key pre-distribution was first proposed by Eschenauer et al. [28]. The basic idea behind this scheme is to have a large pool of keys, from which a set of keys is randomly chosen and stored in each sensor node. Any two nodes which are able to find common keys within their key subsets can use those shared keys for secure communication. Chan et al. extended the above scheme to enhance the security and resilience of the network using $q$-compositeness [17]. In the $q$-composite scheme, at least $q$ common keys are required to establish the secure channel between two nodes instead of using only one key. This method achieves

higher security strength when a network is prone to small scale attacks (less than 100 captured nodes) but not large-scale attacks. However a higher value of $q$ makes the network less scalable - it requires a larger number of keys stored at each node in order to maintain the necessary probability of finding $q$ keys. Du et al. [26] and Liu et al. [55] further extended random key pre-distribution approach to pairwise key pre-distribution in which the shared key between any two sensor nodes is uniquely computed so that the resilience against node capture is significantly improved. All above mentioned schemes assume no network pre-deployment knowledge. In the case that certain pre-deployment knowledge is available, the performance of the key pre-distribution can be improved by exploiting such knowledge [25, 56].

The drawback of the above mentioned random key pre-distribution schemes [28, 17] is that they are not suitable for large scale sensor networks as they require each node to load a large number of keys, although they do not rely on any pre-deployment knowledge. For instance, implementation of random key distribution schemes in [28, 17] results in a storage overhead of at least 200 keys at each sensor node for a WSN of size 10,000, which is almost half of the available memory space at a low-end sensor node (assume 64-bit keys and less than 4KB of data memory [75]). The problem becomes even worse when the network size is larger. This fact makes the previously proposed random key distribution schemes less practical for large-scale WSNs.

## 1.2 Main Contributions

The main contributions of this thesis are summarized as follows.

- Multicast Security: We identify the problem of multiuser broadcast authentication in WSNs and point out a serious security vulnerability inherent to the symmetric-key based $\mu$TESLA-like schemes. We then propose several PKC-

based schemes to address the proposed problem with minimized computational and communication costs. We achieve our goal by integrating several cryptographic building blocks, such as the Bloom filter, the partial message recovery signature scheme, and the Merkle hash tree, in an innovative manner. We also analyze both the performance and security resilience of the proposed schemes. A quantitative energy consumption analysis is given in detail and demonstrates the effectiveness and efficiency of the proposed schemes.

To address multicast encryption problem, we first analyze and classify the multicast group semantics that are inherently demanded by WSNs. We then propose the GPLD scheme which, to our best knowledge, is the first multicast encryption scheme of its kind that supports various multicast group semantics and is tailored for WSNs. GPLD advances the current state-of-the-art by enabling dynamic changing and simultaneous formation of multiple multicast groups. We develop a novel multicast encryption technique called *global-partition, local-diffusion*. This technique effectively minimizes global (sink-to-sensor) group key distribution and re-keying traffic, while maintaining its support to various multicast group semantics. The efficiency and security properties of GPLD are justified through both analysis and simulations.

- Data Report Security: To address data report security, we propose a novel location-aware multi-functional key management framework. LEDS efficiently embeds the location (cell) information of each sensor into all types of symmetric secret keys owned by that node, and thus provides end-to-end security guarantee. Each legitimate event report in LEDS is endorsed by multiple sensing nodes and is encrypted with a unique secret key shared between the event sensing nodes and the sink. Furthermore, the authenticity of the corresponding event sensing nodes can be individually verified by the sink. This novel setting successfully eliminates the possibility that the compromise of nodes other

than the sensing nodes of an event report may result in security compromise of that event report. LEDS possesses efficient en-route false data filtering capability to deal with the infamous bogus data injection attack, which at the same time significantly reduces energy cost as unnecessary forwarding is eliminated. LEDS also provides high level assurance on data availability by dealing with both report disruption attack and selective forwarding attack, simultaneously.

For applications related to large-scale spatial phenomena monitoring, we further introduce the problem of securing event boundary detection in WSNs and show how existing boundary detection schemes would fail in adversarial environments. We present a Secure Event Boundary Detection (SEBD) scheme, which is to the best of our knowledge the first protocol of its kind to secure event boundary detection in WSNs. SEBD withstands many types of attacks. We propose an enhanced statistic model for localized event boundary detection with proactive faulty measurements correction. Our model is more accurate and robust against node compromise and random fault as compared to existing schemes [19, 48, 23]. Moreover, it is nonparametric without relying on any prior knowledge of node compromise and fault probability, which, however, is required by existing schemes to achieve optimal results [19, 23]. We use extensive simulations to evaluate SEBD and show a very good performance and security strength.

- Random Key Pre-distribution: We propose a highly efficient random key pre-distribution scheme, which combines the random key pre-distribution technique and the hash chain technique. The novelty of our scheme lies in that, instead of requiring each sensor node to store all the chosen keys, the majority of the keys a node possesses are represented and stored in the form of a small number of key-generation keys by carefully designing the key pool, and therefore, the storage overhead is significantly reduced while the same security strength holds. Compared with the existing schemes, the proposed scheme is more scalable and

more secure in the sense that 1) Under the given resilience requirement against node capture, the proposed scheme requires a much smaller key ring size than the previous schemes; 2) Under the given maximum allowed key ring size, the proposed scheme has a much better resilience property against node capture than the previous schemes.

## 1.3 Dissertation Outline

The organization of this dissertation is as follows.

Chapter 2 presents multi-user broadcast authentication schemes. In Section 2.1, we introduce the background of the cryptographic mechanisms to be used. Section 2.2 presents the system assumption, adversary model, and security objectives. In Section 2.3, we introduce two basic schemes. We further propose two advanced schemes and detail the underlying design logic in Section 2.4. Section 2.5 analyzes the performance of the proposed schemes. In Section 2.6, we summarize the chapter.

In Chapter 3, we propose a secure and efficient multicast encryption scheme which allows Ad-hoc Group Formations. In Section 3.1, we discuss multicast group semantics in WSNs. Section 3.2 is related work. In Sections 3.3 and 3.4, we introduce our proposed scheme in detail. Sections 3.5 and 3.6 are the security and performance analysis of the proposed scheme, respectively. We summarize the chapter in Section 3.7.

Chapter 4 discusses data report security. Section 4.1 articulates the data security goals in WSNs and evaluates related work with respect to these goals. Section 4.2 details the proposed LEDS design. Section 4.3 presents the detailed security analysis of the proposed LEDS, followed by the performance analysis in Section 4.4. In Section 4.5, we summarize the chapter.

Chapter 5 presents secure and fault-tolerant event boundary detection techniques.

Section 5.1 details the proposed event boundary detection scheme, called SEBD. Section 5.2 gives the security analysis of SEBD. Section 5.3 reports the simulation results of SEBD regarding both performance and security strength. We give the summarization in Section 2.6.

Chapter 6 presents a new approach for random key pre-distribution for large scale WSNs. We describe the background and related work in Section 6.1. Then we define the terms and notation and describe the proposed scheme in Section 6.2. We further discuss the performance and security strength of the proposed scheme in Sections 6.3 and 6.4. Finally, the summarization is given in Section 6.5.

Chapter 7 concludes the dissertation and offers some directions for future work.

# Chapter 2

# Multi-user Broadcast Authentication

In this chapter, we address multiuser broadcast authentication problem in WSNs by designing PKC-based solutions with minimized computational and communication costs. We focus on providing multi-user broadcast authentication in WSNs, where the broadcast messages are initiated by a number of network users. Please note that the network users refer to personnel or devices that use the WSN; they are not sensor nodes. On the one hand, we aim to achieve immediate message authentication and resist DoS attacks in the presence of both user revocation and node compromise. On the other hand, we want to optimize both computational and communication costs.

We propose four different public-key-based approaches and provide in-depth analysis of their advantages and disadvantages. In all the four approaches, the users are always authenticated through their public keys. We first propose a straightforward certificate-based approach and point out its high energy inefficiency with respect to both communication and computation costs. We then propose a direct storage based scheme, which has high efficiency but suffers from the scalability problem. A Bloom filter based scheme is further proposed to improve the memory efficiency over the

direct storage based scheme. Further techniques are also developed to increase the security strength of the proposed scheme. Lastly, we propose a hybrid scheme to support a larger number of network users by employing the Merkle hash tree technique. We give an in-depth quantitative analysis of the proposed schemes and demonstrate their effectiveness and efficiency in WSNs in terms of energy consumption.

## 2.1 Preliminaries

### 2.1.1 Digital Signature and Its Application in WSNs

A digital signature algorithm is a cryptographic tool for generating non-repudiation evidence, authenticating the integrity as well as the origin of a signed message. In a digital signature algorithm, a signer keeps a private key secret and publishes the corresponding public key. The private key is used by the signer to generate digital signatures on messages and the public key is used by anyone to verify signatures on messages. The digital signature algorithms mostly used are RSA [93] and DSA [68]. ECDSA is referred to Elliptic Curve Digital Signature Algorithm [32]. While RSA with 1024-bit keys (RSA-1024) provides the currently accepted security level, it is equivalent in security strength to ECC with 160-bit keys (ECC-160). Hence, for the same level of security strength, ECDSA uses a much short key size and hence has a short signature size (320-bit).

When $\mu$TESLA was proposed, sensor nodes were assumed to be extremely resource-constrained, especially with respect to computation capability, bandwidth availability, and energy supply [75]. Therefore, public key cryptography (PKC) was thought to be too computationally expensive for WSNs, though it could provide much simpler solutions with much stronger security resilience. At the same time, the computationally efficient one-time signature schemes are also considered unsuitable for WSNs, as they usually involve intense communications [75]. However, recent studies [104, 27, 92]

showed that, contrary to widely held beliefs, PKC with even software implementations only is very viable on sensor nodes. For example [104], Elliptic Curve Cryptography (ECC) signature verification takes 1.61s with 160-bit keys on ATmega128 8MHz, a processor used in current Crossbow motes platform [1]. Furthermore, the computational cost is expected to fall faster than the cost to transmit and receive. For example, ultra-low-power microcontrollers such as the 16-bit Texas Instruments MSP430 [3] can execute the same number of instructions at less than half the power required by the 8-bit ATmega128L. The benefits of transmitting shorter ECC keys and hence shorter messages/signatures will in turn be more significant. Moreover, next generation sensor nodes are expected to combine ultra-low power circuitry with so-called power scavengers such as Heliomote [41], which allow continuous energy supply to the nodes. At least $8 - 20\mu$W of power can be generated using MEMS-based power scavengers [77]. Other solar-based systems are even able to deliver power up to 100mW for the MICA Motes [41, 42]. These results indicate that, with the advance of fast growing technology, PKC is no longer impractical for WSNs, though still expensive for the current generation sensor nodes, and its wide acceptance is expected in the near future [27].

### 2.1.2 The Bloom Filter and Counting Bloom Filter

A Bloom filter is a simple space-efficient randomized data structure for representing a set in order to support membership queries [65]. A Bloom filter for representing a set $S = s_1, s_2, ..., s_n$ of $n$ elements is described by a vector $\mathcal{V}$ of $m$ bits, initially all set to 0. A Bloom filter uses $k$ independent hash functions $h_1, ..., h_k$ with range $0, ..., m-1$, which map each item in the universe to a random number uniform over $[0, ..., m-1]$. For each element $s \in S$, the bits $h_i(s)$ are set to 1 for $1 \leq i \leq k$. Note that a bit of $\mathcal{V}$ can be set to 1 multiple times. To check if an item $x$ is in $S$, we check whether all bits $h_i(x)$ are set to 1. If not, $x$ is not a member of $S$ for certain,

that is, no false negative error. If yes, $x$ is assumed to be in $S$. A Bloom filter may yield a false positive. It may suggest that an element $x$ is in $S$ even though it is not. The probability of a false positive for an element not in the set can be calculated as follows. After all the elements of $S$ are hashed into the Bloom filter, the probability that a specific bit is still 0 is $(1 - \frac{1}{m})^{kn} \approx e^{-kn/m}$. The probability of a false positive $f$ is then

$$f = (1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - e^{-kn/m})^k.$$

We let $f = (1 - p)^k$. From now on, for convenience, we use the asymptotic approximations $p$ and $f$ to represent, respectively, the probability that a bit in the Bloom filter is 0 and the probability of a false positive. Let $p = e^{-kn/m}$.

The counting Bloom filter is a variation of the Bloom filter, which allows member deletion. In the counting Bloom filter, each entry in the Bloom filter is not a single bit but a small counter that tracks the number of elements that have hashed to that location [49]. When an element is deleted, the corresponding counters are decremented. To avoid overflow, counters must be chosen large enough [49].

### 2.1.3 The Merkle Hash Tree

A Merkle Tree is a construction introduced by Merkle in 1979 to build secure authentication schemes from hash functions [64]. It is a tree of hashes where the leaves in the tree are hashes of the authentic data values $n_1, n_2, ..., n_w$. Nodes further up in the tree are the hashes of their respective children. For instance, assuming that $w = 4$ in Fig. 2.1, the values of the four leaf nodes are the hashes of the data values, $h(n_i), i = 1, 2, 3, 4$, respectively, under a one-way hash function $h()$ (e.g., SHA-1 [69]). The value of an internal node A is $h_a = h(h(n_1)||h(n_2))$, and the value of the root node is $h_r = h(h_a||h_b)$. $h_r$ is used to commit to the entire tree to authenticate any subset of the data values $n_1, n_2, n_3$, and $n_4$ in conjunction with a small amount of auxiliary authentication information `AAI` (i.e., $\log_2 N$ hash values where

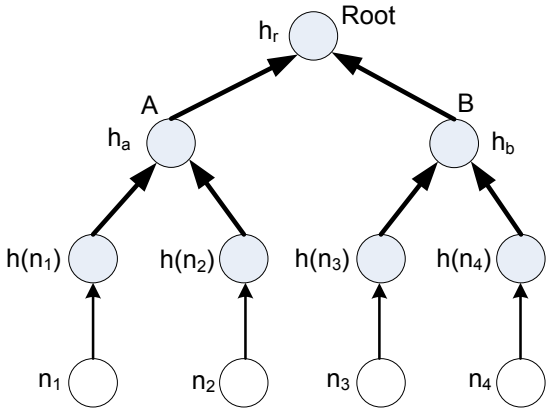


Figure 2.1: An example of Merkle hash tree

$N$ is the number of leaf nodes). For example, a receiver with the authentic $h_r$ requests for $n_3$ and requires the authentication of the received $n_3$. The source sends the `AAI` $:< h_a, h(n_4) >$ to the receiver. The receiver can then verify $n_3$ by first computing $h(n_3)$, $h_b = h(h(n_3)||h(n_4))$ and $h_r = h(h_a||h_b)$, and then checking if the calculated $h_r$ is the same as the authentic root value $h_r$. Only if this check is positive, the user accepts $n_3$. The Merkle hash tree can prevent an adversary from sending bogus data to deceive the client. In the earlier example, an adversary impersonating can not send a bogus $n_3$ to the client without being detected. This is because he can not find $h_a$ and $h(n_4)$ such that $h(h_a||h(h(n_3)||h(n_4))) = h_r$, as $h()$ is one-way.

## 2.2 System Model, Adversary Model, and Design Goals

*System Model*: We consider a large spatially distributed WSN, consisting of a fixed sink(s) and a large number of sensor nodes. The **sensor nodes** are usually resource-constrained with respect to memory space, computation capability, bandwidth, and power supply. The WSN is aimed to offer information services to many network

users that roam in the network, in addition to the fixed sink(s) [9]. The **network users** may include mobile sinks, vehicles, and people with mobile clients, and they are assumed to be more powerful than sensor nodes in terms of computation and communication abilities. For example, the network users could consist of a number of doctors, nurses, medical equipment (acting as actuators) and so on, in the case of CodeBlue [60], where the WSN is used for emergency medical response. These network users broadcast queries/commands through sensor nodes in the vicinity, and expect the replies that reflect the latest network information. The network users can also communicate with the sink or the backend server directly without going through the WSN if necessary. We assume that the sink is always trustworthy but the sensor nodes are subject to compromise. At the same time, the users of the WSN may be dynamically revoked due to either membership changes or compromise, and the revocation pattern is not restricted. We also assume that the WSN is loosely synchronized.

*Adversary Model*: We assume that the adversary's goal is to inject bogus messages into the network, attempt to deceive sensor nodes, and obtain the information of his interest. Additionally, DoS attacks such as bogus message flooding, aiming at exhausting constrained network resources, is another important focus of the paper. We assume that the adversary is able to compromise both network users and the sensor nodes. The adversary hence could exploit the compromised users/nodes for such attacks. However, we do assume that adversary cannot compromise an unlimited number of sensor nodes. Neither can they break any cryptographic primitive, on which we base our design. Otherwise, it is unlikely for any feasible security solution to be designed.

*Design Goals*: Our security goal is straightforward: all messages broadcasted by the network users of the WSN should be authenticated so that the bogus ones inserted by the illegitimate users and/or compromised sensor nodes can be efficiently

rejected/filtered. We also focus on minimizing the overheads of the security design. Especially, energy efficiency (with respect to both communication and computation) and storage overhead are given priority to cope with the resource-constrained nature of WSNs.

## 2.3 The Basic Schemes

We explore the PKC domain for the possible solutions to multiuser broadcast authentication in WSNs. The PKC-based solutions realize immediate message authentication and thus can overcome the delayed message authentication problem present in $\mu$TESLA-like schemes. However, the design of PKC-based solutions is not trivial. Simple solutions such as the certificate-based approach are not very useful in the resource-constrained WSNs due to their high scheme overhead. Sophisticated approaches are required to balance different competing factors and achieve a desirable performance tradeoff.

### 2.3.1 The Certificate-Based Authentication Scheme (CAS)

CAS works as follows. Each user (not a sensor) of the WSN is equipped with a public/private key pair ($\mathsf{PK}/\mathsf{SK}$), and signs every message he broadcasts with his $\mathsf{SK}$ using a digital signature scheme such as ECDSA [32]. Note that in all our designs, we do not require sensors to have public/private key pairs for themselves. To prove the user's ownership over his public key, the sink[1] is also equipped with a public/private key pair and serves as the certification authority (CA). The sink issues each user a public key certificate, which, to its simplest form, consists of the following contents: $\mathtt{Cert}_{U_{ID}} = U_{ID}, \mathsf{PK}_{U_{ID}}, \mathtt{ExpT}, \mathrm{SIG}_{\mathsf{SK}_{Sink}}\{h(U_{ID}||\mathtt{ExpT}||\mathsf{PK}_{U_{ID}})\}$, where $U_{ID}$ denotes the user's ID, $\mathsf{PK}_{U_{ID}}$ denotes its public key, $\mathtt{ExpT}$ denotes certificate expiration

[1]We assume that the sink represents the network planner.

time, and $\mathrm{SIG}_{\mathsf{SK}_{Sink}}\{h(U_{ID}||\mathtt{ExpT}||\mathsf{PK}_{U_{ID}})\}$ is a signature over $h(U_{ID}||\mathtt{ExpT}||\mathsf{PK}_{U_{ID}})$ with $\mathsf{SK}_{Sink}$. Hence, a broadcast message is now of the form as follows:

$$< M, tt, \mathrm{SIG}_{\mathsf{SK}_{U_{ID}}}\{h(U_{ID}||tt||M)\}, \mathtt{Cert}_{U_{ID}} > \qquad (I)$$

Here, $M$ denotes the broadcast message and $tt$ denotes the current time. For the purpose of message authentication, sensor nodes are preloaded with $\mathsf{PK}_{Sink}$ before the network deployment; and message verification contains two steps: the user certificate verification and the message signature verification.

CAS suffers from two main drawbacks. First and foremost, it is not efficient in communication, as the certificate has to be transmitted along with the message across every hop as the message propagates in the WSN. A large per message overhead will result in more energy consumption on every single sensor node. In CAS, the per message overhead is as high as $|tt| + |\mathrm{SIG}_{\mathsf{SK}_{U_{ID}}}\{h(U_{ID}||M)\}| + |\mathtt{Cert}_{U_{ID}}| = 128$ bytes. As in [104], the user certificate is at least 86 bytes, when ECDSA-160 [32] is used. Here, we assume that $tt$ and $U_{ID}$ are both two bytes, in which case the scheme supports up to $65,535$ network users. Moreover, $|\mathrm{SIG}_{\mathsf{SK}_{U_{ID}}}\{h(U_{ID}||M)\}| = 40$ bytes, when ECDSA-160 [32] is assumed. Second, to authenticate each message, it always takes two expensive signature verification operations. This is because the certificate should always be authenticated in the first place.

### 2.3.2 The Direct Storage Based Authentication Scheme (DAS)

One way to reduce the per message overhead and the computational cost is to eliminate the existence of the certificate. A straightforward approach is then to let sensor nodes simply store all the current users' ID information and their corresponding public keys. In this way, a broadcast message now only contains the following contents:

$$< M, tt, \mathrm{SIG}_{\mathsf{SK}_{U_{ID}}}\{h(U_{ID}||tt||M)\}, U_{ID}, \mathsf{PK}_{U_{ID}} > . \quad (II)$$

Verifying the authenticity of a user public key is reduced to finding out whether or not the attached user/public key pair is contained in the local memory. Upon user revocation, the sink simply sends out ID information of the revoked user, and every sensor node deletes the corresponding user/public key pair in its memory.

The drawbacks of DAS are obvious. Given a storage limit of 5 KB, only 232 users can be supported at most; even with a memory space of 19.5 KB, DAS can only support up to 1, 000 users. At the same time, CAS can support up to 2, 560 users given the same storage limit 5 KB. The reason is that in CAS only the ID information of the revoked users are stored by the sensor nodes. Therefore, DAS is neither memory efficient nor scalable. However, the advantage of DAS is also significant as compared to CAS. It successfully reduces the per message overhead down to $|tt| + |\text{SIG}_{\mathsf{SK}_{U_{ID}}}\{h(U_{ID}||M)\}| + |U_{ID}| + |\mathsf{PK}_{U_{ID}}| = 64$ bytes. The above analysis clearly shows that more advanced schemes are needed other than DAS and CAS. And the direction to seek is to improve storage efficiency while retaining or further reducing the per message overhead.

## 2.4 The Advanced Schemes

In this section, advanced schemes are proposed to achieve both storage efficiency and communication efficiency simultaneously. The proposed schemes significantly outperform the previous basic schemes through a novel integration of several cryptographic techniques.

### 2.4.1 The Bloom Filter Based Authentication Scheme (BAS)

**System Preparation:** The sink generates the public keys for all network users, and constructs the set:

$$\mathbf{S} = \{< U_{ID_1}, \mathsf{PK}_{U_{ID_1}} >, < U_{ID_2}, \mathsf{PK}_{U_{ID_2}} >, ...\},$$

where $\#\{\mathbf{S}\} = N$, and $\#\{\}$ denotes the cardinality of the set. Using the Bloom filter, the sink can apply $k$ system-wide hash functions (cf. Section II.B) to map the elements of $\mathbf{S}$ (each with $L+2$ bytes, that is, $|U_{ID}| = 2$ bytes, and $|\mathsf{PK}_{U_{ID}}| = L$ bytes) to an $m$-bit vector $\mathcal{V}$ with $\mathcal{V} = v_0 v_1 ... v_{m-1}$, where we have $m < N(L+2)$ to reduce the filter size and $m > kN$ to retain a small probability of a false positive. These $k$ hash functions are known by every node and the sink. For each $v_i, i \in [0, m-1]$, we have

$$v_i = \begin{cases} 1, & \text{if } \exists\ l \in [1,k], j \in [1,N], \\ & s.t.\ h_l(U_{ID_j}||\mathsf{PK}_{U_{ID_j}}) = i \\ 0, & \text{otherwise} \end{cases}$$

Additionally, the sink constructs a counting Bloom filter $\overline{\mathcal{V}}$ of $m * c$ bits with $\overline{\mathcal{V}} = \overline{v}_0 \overline{v}_1 ... \overline{v}_{m-1}$, where each $\overline{v}_i, i \in [0, m-1]$ is a $c$-bit counter, i.e., $|\overline{v}_i| = c$ bits. The value of $\overline{v}_i$ is determined as follows:

$$\overline{v}_i = \#\{(ID_j, \mathsf{PK}_{U_{ID_j}}) | h_l(U_{ID_j}||\mathsf{PK}_{U_{ID_j}}) = i,$$

$$for\ \exists\ l \in [1,k], j \in [1,N]\}.$$

And $c = \lceil \log_2(\max(\overline{v}_i, i \in [0, m-1])) \rceil$ bits, which is usually of 4 bits for most applications [49]. The above operations are illustrated in Fig. 2.2. The sink finally preloads each sensor node with $\mathcal{V}$ (not including $\overline{\mathcal{V}}$), as well as the sink's public key and the common domain parameters of the ECDSA signature scheme.

**Message Signing and Authentication:** Let $\mathsf{PK}_{U_{ID}} = sG$, be the public key of user $U_{ID}$, where $s$ is the private key of the signer, and $G$ is the generator of a subgroup of an elliptic curve group of order $r$. Let $S_K(\cdot)$ be a symmetric key cipher such as AES. To broadcast a message $M$ ($|M| \geq 10$ bytes), $U_{ID}$ takes the steps below following [66], a variant of ECDSA with the partial message recovery property:

- Concatenate $< M||tt||U_{ID} >$, and break it into two parts, $M_1$ and $M_2$, where $|M_1| \leq 10$ bytes.

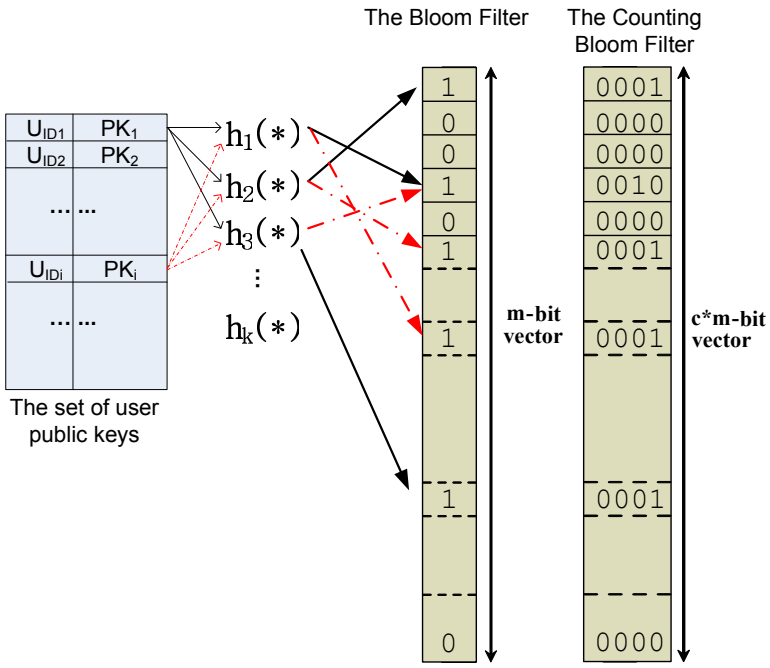


Figure 2.2: An example of the Bloom filter and Counting Bloom Filter

- Generate a random key pair $\{u, V\}$, where $u \in [1, r-1]$, $V = uG = (x_1, y_1)$, and $(x_1 \mod r) \neq 0$.
- Encode-and-hash $V$ into an integer $I$ [66].
- Form $F_1$ from $M_1$ by adding the proper redundancy [99].
- Compute $\mathtt{C} = (I + F_1) \mod r$, and make sure that $\mathtt{C} \neq 0$ or repeat the above steps otherwise.
- Compute $F_2 = h(M_2)$, and $\mathtt{D} = u^{-1}(F_2 + s\mathtt{C}) \mod r$.
- Repeat all the above steps if $\mathtt{D} = 0$; Output the signature as $< \mathtt{C}, \mathtt{D} >$ otherwise.

Then, $U_{ID}$ broadcasts

$$< M_2, \mathtt{C}, \mathtt{D}, \mathsf{PK}_{U_{ID}} >, \quad (III)$$

where $tt$ and $U_{ID}$ are parts of $M_2$. And this is the known simplest message format that can be achieved using PKC[2]. Now, upon receiving a broadcast message (not from the sink), a sensor node checks the authenticity of the message in two steps. First, it checks the authenticity of the corresponding public key by verifying its membership in $\mathbf{S}$. To do so, the sensor node checks whether $\mathcal{V}[h_l(U_{ID}||\mathsf{PK}_{U_{ID}})] \stackrel{?}{=} 1, l \in [1, k]$, and a negative result will lead to the discarding of the message. We note that here a false positive may happen due to the probabilistic nature of the Bloom filter, but only with a very small (negligible) probability when appropriate parameters are chosen as we will analyze later. Second, it verifies the attached signature as follows:

- Discard the message if $\mathtt{C} \notin [1, r-1]$ or $\mathtt{D} \notin [1, r-1]$.
- Compute $F_2 = h(M_2)$, $\mathtt{H} = \mathtt{D}^{-1} \mod r$, and $\mathtt{H}_1 = F_2\mathtt{H} \mod r$.
- Compute $\mathtt{H}_2 = \mathtt{CH} \mod r$, and $P = \mathtt{H}_1 G + \mathtt{H}_2\mathsf{PK}_{U_{ID}}$.
- Discard the message if $P = \mathcal{O}$.
- Encode-and-hash $P$ into an integer $I$ [66] and compute $F_1 = \mathtt{C} - I \mod r$.
- Discard the message if the redundancy of $F_1$ is incorrect.
- Otherwise accept $M_1$ (obtained from $F_1$) and the signature and reconstruct $M||tt||U_{ID} = M_1||M_2$.

**User Revocation/Addition:** To revoke a user, say $U_{ID_j}$, the sink follows the steps below:

[2]The claim is true only when ID-based cryptography [95] is excluded from consideration, in which case the user's ID is also his public key. Furthermore, the shortest signature size possibly obtained from pairing is around 22 bytes [8], which is shorter than 40 bytes obtained from ECDSA. However, to apply a pairing-based scheme (i.e., a ID-based signature or short signature) on sensor nodes, the known reachable signature size has to be 84 bytes, even when a 32-bit microprocessor can be used [118]. And the energy cost is also multiple times higher than that of an ECDSA-160 signature.

- First, it hashes $h_l(U_{ID_j}||\mathsf{PK}_{U_{ID_j}}) = i$ and decreases $\overline{v}_i$ by 1. It repeats this operation for all $h_l, l \in [1, k]$.

- From the updated counting Bloom filter $\overline{\mathcal{V}}$, the sink obtains the corresponding updated Bloom filter $\mathcal{V}'$ with $\mathcal{V}' = v'_0 v'_1 ... v'_{m-1}$. Here, $v'_i = 1$ only when $\overline{v}_i \geq 1$, and $v'_i = 0$ otherwise.

- The sink further calculates $\mathcal{V}_\Delta = \mathcal{V}' \oplus \mathcal{V}$ and deletes $\mathcal{V}$ afterwards. Here $\oplus$ denotes bitwise exclusive OR operation. Obviously, $\mathcal{V}_\Delta$ is an $m$-bit vector with at most $k$ bits set to 1. Hence, $\mathcal{V}_\Delta$ can be simply represented by enumerating its 1-valued bits, requiring $\overline{k}\lceil \log_2 m \rceil$ bits for indexing ($\overline{k} \leq k$). This representation is efficient for a small $\overline{k}$ as will be analyzed in Section VI.B.

- The sink finally broadcasts $\mathcal{V}_\Delta$ after signing it. The message format follows (III) but with the sink's public key omitted, as every sensor already has it.

- Upon receiving and successfully authenticating the broadcast message, every sensor node updates its own Bloom filter accordingly, that is, if $v_{\Delta,i} = 1$, then $v_i = 0$, $i \in [0, m-1]$.

BAS also supports simultaneous multiuser revocation. Suppose that $N_{rev}$ users are revoked simultaneously. The sink follows the same manner to construct $\mathcal{V}_\Delta$ with $\overline{k}$ bits set 1. Now we have $\overline{k} \leq kN_{rev}$. Furthermore, the compressed message for representing $\mathcal{V}_\Delta$ now could achieve $mH(p)$ bits theoretically, where $H(p) = -p\log_2 p - (1-p)\log_2(1-p)$ is the entropy function and $p = (1 - \frac{1}{m})^{\overline{k}}$ is the probability of each bit being 0 in $\mathcal{V}_\Delta$. As pointed out in [65], using arithmetic coding technique can efficiently approach this lower bound.

BAS supports dynamic user addition in two ways. First, it enables a later binding of network users and their (ID, public key) pairs. In this approach, the sink may generate more (ID, public key) pairs than needed during system preparation. When

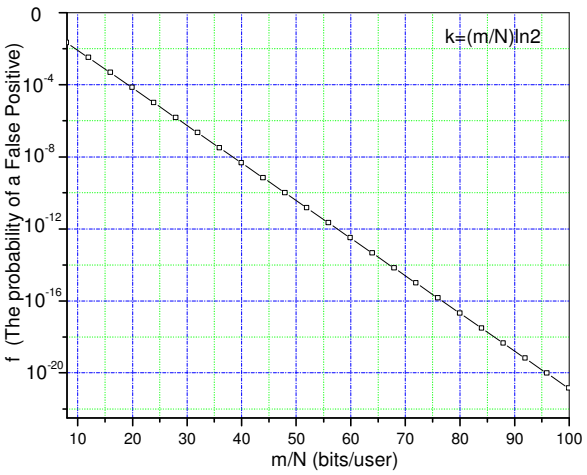


Figure 2.3: The minimum probability of a false positive regarding $\frac{m}{N}$

a new network user joins the WSN, it will be assigned an unused ID and public key pair by the sink. Second, BAS could add new network users after the revocation of old members. This approach, however, could only add the same number of new users as that of the revoked. This requirement ensures that the probability of a false positive never increases in BAS. To do so, the sink updates its counting Bloom filter by hashing the new user's information into the current Bloom filter. The sink then obtains a $\mathcal{V}_\Delta$ in the same way as in the revocation case, and broadcasts it after compression. This time, if $v_{\Delta,i} = 1$, sensor nodes will set $v_i = 1$, $i \in [0, m-1]$ to update their current Bloom filters.

### 2.4.2 Minimize the Probability of a False Positive

Since the Bloom filter provides probabilistic membership verification only, it is important to make sure that the probability of a false positive is as small as possible.

**Theorem 1**: Given the number of network users $N$ and the storage space $m$ bits for a single Bloom filter, the minimum probability of a false positive $f$ that can be

achieved is $2^{-k}$ with $k = \frac{m}{N} \ln 2$, that is,

$$f = (0.6185)^{\frac{m}{N}}.$$

*Proof*: since $f = (1-(1-\frac{1}{m})^{kN})^k \approx (1-e^{-kN/m})^k$, we then have $f = e^{k \ln(1-e^{-kN/m})}$. Let $g = k \ln(1 - e^{-kN/m})$. Hence, minimizing $f$ is equivalent to minimizing $g$ with respect to $k$. We find

$$\frac{dg}{dk} = \ln(1 - e^{-kN/m}) + \frac{kN}{m} \frac{e^{-kN/m}}{1 - e^{-kN/m}}$$

It is easy to check that the derivative is 0 when $k = \frac{m}{N} \ln 2$. And it is not hard to show that this is a global minimum [65]. Note that in practice, $k$ must be an integer. □

Fig. 5.3 shows the probability of a false positive $f$ as a function of $\frac{m}{N}$, i.e., bits per element. We see that $f$ decreases sharply as $\frac{m}{N}$ increases. When $\frac{m}{N}$ increases from 8 to 96 bits, $f$ decreases from $2.1 * 10^{-2}$ to $9.3 * 10^{-21}$. Obviously, $f$ determines the security strength of our design. For example, when $\frac{m}{N} = 92$ bits, the adversary has to generate around $2^{63.8}$ public/private key pairs on average before finding a valid one to pass the Bloom filter. This is almost computationally infeasible, at least within the lifetime of the WSN (usually at most several years). However, when $\frac{m}{N} = 64$ bits, the adversary is now expected to generate around $2^{44.4}$ public/private key pairs before finding a valid pair. The analysis below shows the time and cost of the attack. To generate a public/private key pair in ECDSA-160, a point multiplication operation has to be performed, for which the fastest known implementation speed is 0.21ms through a specialized FPGA design [38]. Suppose the adversary could afford 100,000 such FPGAs, which would cost no less than one million dollars. Then, by executing 100,000 FPGAs simultaneously, to generate one valid key pair still takes 13.2 hours roughly. With the above analysis, we suggest to select the value of $f$ carefully according to the security requirements of the different types of applications. Given a highly security sensitive military application, we suggest that $f$ should be no

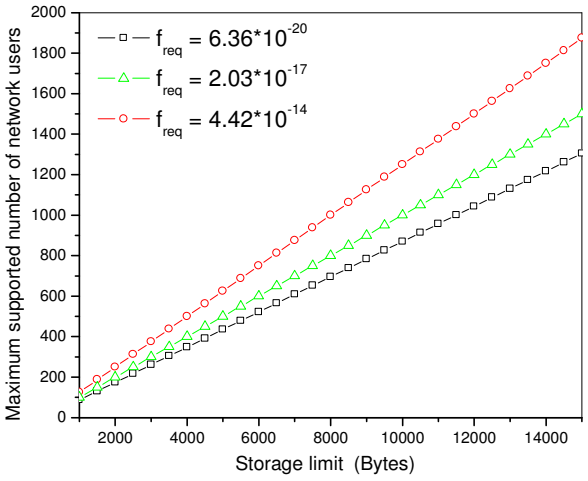


Figure 2.4: Maximum supported number of network users with respect to storage limit

larger than $6.36 * 10^{-20}$, i.e., $m/N \geq 92$ bits. On the other hand, when the targeted applications are less security sensitive as in the civilian scenario, we can tolerate a larger $f$. This is because the adversary is now generally much less resourceful as compared to the former case.

### 2.4.3 Maximum Number of Network Users Supported

It is important to know how many network users can be supported in BAS so that the WSN can be well planned. The following theorem provides the answer.

**Theorem 2**: Given the storage space $m$ bits for a single Bloom filter and the required probability of a false positive $f_{req}$ ($f_{req} \in (0, 1)$), the maximum number of network users that can be supported is $\frac{-m(\ln 2)^2}{\ln f_{req}}$, that is,

$$N \leq \frac{-0.4805m}{\ln f_{req}}.$$

*Proof*: Since the minimal probability of a false positive $f = 2^{-k}$ is achieved with $k = \frac{m}{N} \ln 2$, we have $f_{req} = 2^{-\frac{m}{N} \ln 2}$. Then, we can easily get $N = \frac{-m(\ln 2)^2}{\ln f_{req}}$ in this case; and this is the maximum number of users that can be supported given $f_{req}$ and $m$. □

Fig. 2.4 illustrates the maximum supported number of network users as a function of the storage limit. Fig. 2.4 shows that BAS supports up to 1,250 users when $f_{req} = 4.42 * 10^{-14}$, 1,000 users when $f_{req} = 2.03 * 10^{-17}$, and 869 users when $f_{req} = 6.36 * 10^{-20}$, for a storage space of 9.8 KB. Obviously, BAS also allows tradeoff between the maximum supported number of network users and the probability of a false positive given a fixed storage limit.

*D. Supporting More Users using the Merkle Hash Tree: The Hybrid Authentication Scheme (HAS)*

Through the above analysis, we know that the maximum supported number of network users is usually limited given the storage limit and the probability of a false positive. For example, if $f_{req} = 6.36 * 10^{-20}$ and the storage limit is 4.9 KB, the maximum number of users supported by BAS is 434. Therefore, an additional mechanism has to be employed to support more users when necessary. HAS achieves this goal by employing the Merkle hash tree technique, which trades the message length for the storage space. That is, by increasing the per message overhead, HAS can support more network users. Specifically, HAS works as follows.

The sink first calculates the maximum number of users supported in case of BAS according to the given storage limit and the desired probability of a false positive. It then collects all the public keys of the current network users and constructs a Merkel hash tree. In fact, the sink constructs $N$ leaves with each leaf corresponding to a current user of the WSN. For our problem, each leaf node contains the binding between the corresponding user ID and his public key, that is, $h(U_{ID}, \mathsf{PK}_{U_{ID}})$. The values of the internal nodes are determined by the method introduced in Section II.C. The sink further prunes the Merkle hash tree into a set of equal-sized smaller trees. We denote the value of the root node of a small hash tree as $h_r^i, i = 1, ..., |\mathbf{S}|$, where $|\mathbf{S}|$ equals the maximum number of supported users the sink calculates in BAS.

Next, the sink constructs a Bloom filter $\mathcal{V}$ following the same way as described

in the last section. The difference is that now the member set $\mathbf{S} = \{h_r^1, h_r^2, ..., h_r^{|\mathbf{S}|}\}$. Then, the sink preloads each sensor node with $\mathcal{V}$. At the same time, each user should obtain its $\mathtt{AAI}$ according to his corresponding leaf node's location in the smaller Merkle hash tree. Let T denote all the nodes along the path from a leaf node to the root (not including the root), and A be the set of nodes corresponding to the siblings of the nodes in T. Then, $\mathtt{AAI}$ further corresponds to the values associated with the nodes in A. Obviously, $\mathtt{AAI}$ is of size $(\overline{L} * \log_2 \frac{N}{|\mathbf{S}|})$ bytes, where $\overline{L}$ is the length of the hash values. Upon user revocation, the sink simply updates all the sensor nodes with the ID information of the revoked users. And each node directly stores the revoked IDs as described earlier. Now a message sent by a user $U_{ID}$ is of form

$$< M_2, \mathtt{C}, \mathtt{D}, \mathsf{PK}_{U_{ID}}, \mathtt{AAI}_{U_{ID}} > . \quad (IV)$$

Each node verifies the authenticity of a user public key in two steps. First, it calculates the corresponding root node value $h_r^i$ using $\mathtt{AAI}_{U_{ID}}$ attached in the message. Second, it checks whether or not the calculated $h_r^i$ is a member of $\mathcal{V}$ stored by itself. By checking Message (IV), we can easily find that HAS doubles the maximum supported number of users as compared to BAS at the cost of 20 more bytes per message overhead, assuming SHA-1 is used [69]. And the number can be further doubled with 40 more bytes per message overhead.

## 2.5 Further Enhancements

### 2.5.1 Dealing with Long Messages

The messages broadcast in WSNs are usually short, due to the application specific nature of WSNs. The query or command messages can be less than one hundred bytes. However, there are few cases that long messages may be required to be broadcast in WSNs. For example, the sink may broadcast code images to the sensor nodes for the

purpose of retasking WSNs [36]. The size of such code images can be on the order of KB. In this case, it is not desirable to apply the proposed BAS or HAS scheme directly by signing the whole message (i.e., the message hash) only once or signing on every single packet otherwise. This is because of two reasons. First, if we sign the whole message once, then each sensor node can authenticate a message only after it obtains the entire message. That is, the sensor nodes have to buffer a large number of received packets before it can authenticate them. This obviously introduces a severe vulnerability that could result in message flooding attacks. Second, if we sign every packet belonging to the same message, the scheme overheads will increase significantly with respect to both computation and communication. This is because now every packet is attached with a signature, which is 40 bytes in our setting.

Fortunately, several solutions were proposed to solve this problem in the context of code update in WSNs [50, 51, 22]. The first solution is suitable for lossless network environments, which employs off-line hash chain technique to amortize the cost of a single digital signature over multiple packets and allow for incremental message authentication and packet pipelining [50, 51, 31]. The second solution is aimed at tolerating packet losses. This solution makes use of a signed hash tree technique and trades message overhead for potential packet losses [22]. Both solutions can be directly superimposed with BAS and HAS in dealing with long messages. We omit the details of these solutions for space interest.

### 2.5.2 Reducing the Probability of a False Positive

In [61], a method is introduced to use two families of $k$ hash functions, instead of using one. And an element is in the set if either family gives back all 1s from the filter. The trick is to choose one of the two families of the hash functions adaptively: choose which family of hash functions to use for each element of your set in such a way to keep the number of 1s in the filter as small as possible. In such a way, a smaller false

positive probability in the same space can be achieved at the cost of more hashing. This method can reduce the probability of a false positive to the half under certain conditions using the same storage space. This technique can be exploited by BAS so that we achieve a desirable probability of a false positive with a smaller storage space.

### 2.5.3 Optimization on Constructing the Merkle Hash Tree

Different types of network users may have different broadcast frequencies in practice. This fact can be exploited by HAS, when supporting a vast number of network users is a must. Instead of pruning the user Merkle hash tree into a set of equal-sized smaller trees, now the tree can be trimmed into the same number but different-sized smaller ones based on user broadcast frequency. The higher the frequency is, the smaller hash tree the user is grouped in. In such a way, the energy efficiency can be improved in the overall sense, as more messages being broadcast containing only smaller `AAI` sizes. This is similar to the idea introduced in [27].

## 2.6 Performance Analysis

In this section, we analyze the performance of BAS and HAS with respect to communication and computational overheads (in terms of energy consumption), and security strength. We give a quantitative analysis of the schemes and compare them with the other two basic schemes.

### 2.6.1 Communication Overhead

We study how the message size affects the energy consumption in communication in a WSN. We investigate the energy consumption as the function of the size of the WSN (denoted as $W$). We denote by $E_{tr}$ the hop-wise energy consumption

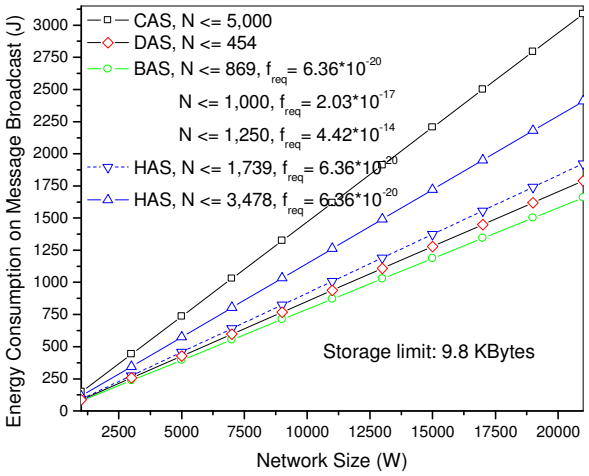


Figure 2.5: Energy consumption in communication regarding different schemes

for transmitting and receiving one byte. As reported in [104], a Chipcon CC1000 radio used in Crossbow MICA2DOT motes consumes 28.6 and 59.2 $\mu$J to receive and transmit one byte, respectively, at an effective data rate of 12.4 Kb/s. Furthermore, we assume a packet size of 41 bytes, 32 bytes for the payload and 9 bytes for the header [104]. The header, ensuing an 8-byte preamble, consists of source, destination, length, packet ID, CRC, and a control byte [104]. We also assume that $|M| = 20$ bytes.

Then, for BAS, the signature size is still the same as that of ECDSA, but only part of the message now has to be transmitted, with the saving of up to 10 bytes. Therefore, the per message overhead of BAS is 54 bytes, which is 10 bytes less than that of DAS. As Message (III) is 74 bytes, there should be 3 packets in total, among which two of them are 41 bytes, and one is 19 bytes. Therefore, there should be $41 * 2 + 19 * 1 + 8 * 3 = 125$ bytes for transmission (including 8-byte preamble per packet). Hence, the hop-wise energy consumption of message transmission is 125$*$59.2 $\mu$J $= 7.40$ mJ; and the energy consumption of message reception is 125$*$28.6 $\mu$J $= 3.58$ mJ. For each message broadcast, every sensor node should retransmit the message once and receive $w'$ times of the same message assuming the blind flooding is used[3].

[3]In an idealized lossless network, blind flooding, i.e., every node always retransmits exactly once

Here, $w'$ denotes node density in terms of the total number of sensor nodes within one unit disc, where a unit disc is a circle area with radius equal to the transmission range of sensor nodes[4]. Hence, the total energy consumption in communication will be $W * (7.4 + 3.58 * w')$ mJ.

Fig. 4.11 illustrates the energy consumption in communication as a function of $W$ with $w' = 20$. Clearly, BAS consumes a much lower energy as compared to others. For example, when $W = 15,000$, CAS always costs 2.20 KJ, while BAS costs only 1.18 KJ. The energy saving for a single broadcast can be more than 1,000 J between BAS and CAS. Note that although DAS also consumes much less energy than CAS, DAS only supports up to $10000/22 \approx 454$ users. At the same time, BAS can handle 869 users even when $f_{req} = 6.36 * 10^{-20}$. CAS handles more users than BAS and DAS, however, at the cost of much higher energy consumption. Moreover, HAS can handle a large number of users but with a much lower energy consumption when compared to CAS. In summary, BAS demonstrates the highest communication efficiency, as well as a desirable storage efficiency. From Fig. 4.11, we also find that the energy consumption in communication is the critical cost for WSNs, as a single broadcast of a message of only 20 bytes in length could cost energy on the order of KJ. This also exposes the severe vulnerability of the $\mu$TESLA-like schemes, as they allow the adversary to flood the WSN arbitrarily.

### 2.6.2 User Revocation/Addition Traffic Overhead

Another important performance metric for the broadcast authentication schemes is the overhead of the user revocation/addition traffic. As analyzed in Section V.A, BAS requires the sink to broadcast $\mathcal{V}_\Delta$ upon user revocation/addition. We have shown

every unique message it receives, is wasteful, as individual nodes are likely to receive the same broadcast multiple times. In practice, however, blind flooding is a commonly used technique, as its inherent redundancy provides some protection from unreliable (lossy) wireless networks [39].

[4]We assume an uniform transmission range for all sensor nodes.

that in the single user case, $\mathcal{V}_\Delta$ can be efficiently represented by simply enumerating all its 1-valued bits, the length of which is bounded by $\overline{k}\lceil\log_2 m\rceil$ bits. That is, the per user revocation traffic overhead is upper bounded by $\overline{k}\lceil\log_2 m\rceil$ bits. And the theoretical lower bound obtained from the entropy function is $mH(p)$ bits with $H(p) = -p\log_2 p - (1-p)\log_2(1-p)$ and $p = (1-\frac{1}{m})^{\overline{k}}$. It is not hard to see that the expectation value of $\overline{k}$ is around $k/2$, where $k = \frac{m}{N}\ln 2$. Our simulation shows that $\overline{k}$ is always around $k/2$. Hence, for a given $f_{req} = 6.36 * 10^{-20}$, we will have $\overline{k}\lceil\log_2 m\rceil = 68$ bytes, and $mH(p) \approx 52$ bytes, for $N = 1,000$. This implies that the per user revocation traffic $\mathcal{V}_\Delta$ only ranges from 52 to 68 bytes on average for $N = 1,000$, depending on the used coding method[5]. And for $N \leq 11,000$, $\mathcal{V}_\Delta$ is at most 80 bytes on average. This overhead is much lower as comparable to that of the $\mu$TESLA-like scheme proposed for supporting multiple users [58]. In [58], the per user revocation traffic (i.e, a revocation certificate) is no less than $1+\lceil\log_2 N\rceil$ hash values, which is 220 bytes for $N = 1,000$, and 300 bytes for $N = 11,000$, assuming the same hash length of 20 bytes. We further note that in contrast to $\mu$TESLA-like schemes, BAS does not require periodic key chain update (for running out of available keys) among users and sensor nodes. This is the advantage inherent to the PKC-based schemes.

### 2.6.3 Computational Overhead

It was previously widely held that PKC is not suitable in WSNs, as sensor nodes are extremely computation constrained. However, recent studies [104, 27] showed that PKC with only software implementations, is very viable on sensor nodes. For example in [104], an ECC signature verification takes 1.61s with 160-bit keys on ATmega128 8MHz processor used in a Crossbow mote. We analyze the computation cost of the proposed schemes to further justify the suitability of PKC-based schemes in WSNs.

[5]We assume that the number of simultaneous network users are always around $N$.

In all our proposed schemes, the major computational cost is due to the signature verification operation. In the following analysis we omit the cost of other operations such as hash operations and table lookup, as they are negligible as compared to the signature verification operation [104].

In CAS, two ECDSA signature verifications are needed for each broadcast message. In BAS, to verify a message takes $k = \frac{m}{N} \ln 2$ hash operations and one ECDSA signature verification. It was reported in [104] that an ECDSA-160 signature verification operation costs 45.09 mJ on a 8-bit ATmega128L processor running at 4 MHz. If we assume that the sensor CPU is a low-power high-performance 32-bit Intel PXA255 processor, the energy cost can be further minimized. Note that the PXA255 has been widely used in many sensor products such as Sensoria WINS 3.0 and Crossbow Stargate running at 400 MHz. According to [2], the typical power consumption of PXA255 in active and idle modes are 411 and 121 mW, respectively. It was reported in [7] that it takes 92.4 ms to verify an ECDSA-160 signature with the similar parameters on a 32-bit ARM microprocessor at 80 MHz. Therefore, the same computation on PXA255 roughly needs $80/400 \times 92.4 \approx 18.48$ ms, and the energy cost is hence around 7.6 mJ. Therefore, we can obtain the computational costs of the proposed CAS and BAS schemes on different sensor platforms[6]. The results are summarized below.

| Scheme | ATmega128L | PXA255 |
|---|---|---|
| CAS | 90.18 mJ | 15.4 mJ |
| BAS | 45.09 mJ | 7.6 mJ |

BAS is obviously also more computationally efficient than CAS. Furthermore, when we compare the computational cost with the communication cost on hop-wise

[6]DAS and HAS consume similar amount of energy as BAS does, as they both require one signature verification.

message transmission, we can find that both are on the same order, which justifies the suitability of PKC-based schemes in WSNs.

### 2.6.4 Security Strength

The Bloom filter based public key verification ensures the security strength of the proposed scheme by enabling immediate message authentication. That is, there is no authentication delay on messages being broadcast. Therefore, it is very hard for the adversary to perform network wide flooding in the WSN. As we analyzed above, by appropriately choosing a suitable value of $f_{req}$, such as $6.36 * 10^{-20}$ in military applications, it is infeasible to forge a valid public/private key pair during the lifetime of the WSN. Furthermore, by embedding a time stamp into the message, the message replay attack is also effectively prevented, as WSN is assumed to be loosely synchronized [73]. Therefore, the immediate message authentication capability provided by the proposed schemes can effectively protect the WSN from network wide flooding attacks. This is the most significant security strength over the $\mu$TESLA-like schemes, in which network wide flooding attacks are always possible.

Moreover, since the public key operation is expensive, it is also important that sensor nodes can be resistant to the local jamming attacks. Under such attacks, the adversary may simply broadcast random bit strings to the sensor nodes within his transmission range. If these neighbor sensors have to perform the expensive signature verification operation for all received messages, it will be a heavy burden on them. CAS obviously suffers from this type of attacks, as the signature verification operation has to be performed for every received message. However, in both BAS and HAS, such an attack can be effectively mitigated. This is because in both schemes, a sensor node first verifies the authenticity of the attached user public key through hash operations, so it performs signature verification operation for a bogus public key only with a negligible probability (e.g., $6.36 * 10^{-20}$). As reported in [104], the

energy cost of SHA-1 is only 5.9 $\mu$J/byte on a 8-bit ATmega128L processor, while ECDSA-160 could consume 45.09 mJ on signature verification. An adversary may also flood the sensor nodes with forged messages but containing valid user public keys, which can be obtained by eavesdropping the network traffic. In this case, the forged messages can only be discarded after signature verification, and sensor nodes that are physically close to the adversary can thus be abused. We note that this type of attacks is always possible for PKC-based security mechanisms. However, this attack can still be mitigated in BAS by implementing an alert report mechanism. If a sensor node fails to authenticate the received messages multiple times in a row, it will derive that an attack is going on and alert the sink about the attack. The sink further carries out field investigations or other means to detect the adversary and take corresponding remedy actions that are outside the scope of this paper.

## 2.7 Summary

In this chapter, we studied the problem of multiuser broadcast authentication in WSNs. We pointed out that symmetric-key-based solutions such as $\mu$TESLA are insufficient for this problem by identifying a serious security vulnerability inherent to these schemes: the delayed authentication of the messages can easily lead to severe energy-depletion DoS attacks. We then came up with several effective PKC-based schemes to address the problem. Both computational and communication costs of the schemes are minimized through a novel integration of several cryptographic techniques. A quantitative energy consumption analysis, as well security strength analysis were further given in detail, demonstrating the effectiveness and efficiency of the proposed schemes.

# Chapter 3

# Multicast Encryption

In this chapter, the problem of multicast encryption in WSNs is addressed. We aim at providing message confidentiality for the multicast traffic from a sink to the sensors. We approach the problem by first classifying the multicast group semantics in WSNs. We then propose our scheme called GPLD. GPLD focuses on scheme efficiency and its support to various multicast group semantics. GPLD partitions sensors into a series of elementary groups using their location and class information, and accordingly builds a location-class-aware symmetric key management framework. Further leveraging the fact that sensors are both end receivers and routers, GPLD develops a novel multicast encryption technique called *global-partition, local-diffusion*. This technique effectively minimizes global (sink-to-sensor) group key distribution and re-keying traffic, while maintaining its support to various multicast group semantics. The efficiency and security property of GPLD are justified through both analysis and simulations.

## 3.1 Multicast Group Semantics in WSNs

Consider a military application where a large number of sensors with different functionalities are deployed in the strategic field to detect and identify the presence of

critical events of interest as illustrated in Fig. 3.1, where each different shape of symbols denotes a different sensor class with a different functionality, such as image sensors, acoustic sensors, or actuators[1]. Different classes of sensors are used for different purposes. For example, image sensors may be used to identify enemy tanks and soldiers; acoustic sensors may be used to detect other targets based on acoustic signals. Actuators may launch certain actions like activating the preinstalled military devices upon the command from the sink. At the same time, all sensors also collaborate with each other and form a multi-hop wireless network to support network communications.

As WSNs are inherently location-aware and function-specific, multicast group semantics from the sink to the sensors can be classified into four most common categories as shown in Fig. 3.1: *a*) broadcast – all network sensors are the intended recipients of multicast messages, i.e., *recipient sensor*s. *b*) class-based multicast – only the sensors of certain class are the *recipient sensor*s. *c*) location-based multicast – the sink may multicast groups of sensors subject to certain dynamic spatial constraints. Since sensors are always deployed in a discrete manner at certain density, we can easily express the location constraints of sensor groups as a few basic geometric shapes, which can be efficiently described using simple mathematical representations. In Fig. 3.1 (c), the *recipient sensor*s are those sensors located inside the elliptic area. *d*) location-class-based multicast – the sink may also multicast messages to groups of sensors subject to both spatial constraints and class requirements. In Fig. 3.1 (d), the *recipient sensor*s are those sensors of classes $'\star'$ and $'\Box'$ located inside the rectangular area. Depending on different applications, more sophisticated semantics may exist, but these four categories are certainly the most common ones and suffice in most scenarios. Therefore, any multicast encryption scheme designed for WSNs has to support (at least) these multicast group semantics.

[1]For our purpose in this chapter, we do not distinguish sensors from actuators.

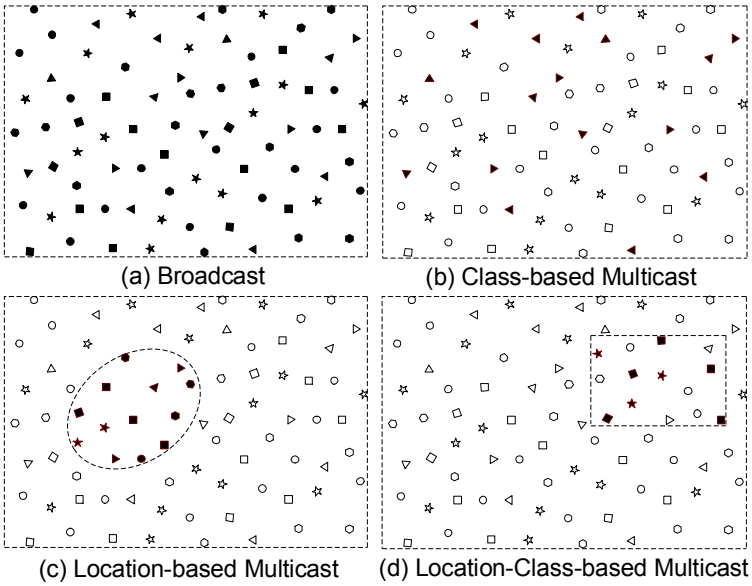


Figure 3.1: Multicast group semantics in WSNs with the solid symbols denote the intended recipients of multicast messages in each case

## 3.2 Related Work

Multicast encryption problem has been extensively addressed in the context of wired networks and ad hoc networks. Below we introduce some typical schemes that are closely related to this work.

*Group Key Distribution Schemes*: The Logical Key Hierarchy (LKH) model was first introduced in [106] to address secure multicast for the Internet. For each group, LKH maintains a key tree which is used for group key update and distribution. The root of the key tree is the group key used for encrypting data in multicast, and is shared by all users. The leaf nodes of the key tree are keys shared only between the individual users and the key distribution center (KDC), whereas the intermediate nodes are auxiliary key encryption keys used to facilitate the distribution of the root key, i.e., the group key. Of all these keys, each user stores the keys from its leaf node all the way up to the root of the key tree. As a result, when a user joins/leaves

the group, all the keys on its path (i.e., from its leaf node to the root node of the key tree) have to be changed and re-distributed to maintain backward/forward data confidentiality. Various schemes such as OFT [10], ELK [74], Seclor [52] are later proposed to further optimize rekeying overhead. Group key distribution schemes are unsuitable for WSNs because they are inherently single group oriented. For a single group, these schemes require a storage overhead of $\mathcal{O}(\log N)$ keys; and to revoke a single user, KDC has to send the rekeying message containing $\mathcal{O}(\log N)$ keys, where $N$ is the group size. However, in WSNs there may exist a large number of ad hoc and dynamic groups due to its abundant multicast group semantics. Thus, it is highly inefficient, if not impossible, for these schemes to support multicast encryption in WSNs.

*Broadcast Encryption Schemes*: First introduced in [67], broadcast encryption schemes enable a centralized server to securely multicast messages to a dynamically changing subset of users of a group. In [67], an efficient broadcast encryption scheme called SD was proposed based on a subset-cover framework. In contrast to group key distribution schemes, SD is stateless. That is, a user receiving only the current rekeying message can recover the group key used for the current session based on his initial configuration, even if he missed previous rekeying operations. Also, unlike group key distribution schemes, SD allows multiuser revocation at a time. SD is by far the most efficient broadcast encryption scheme in terms of rekeying message size, which is $1.25r$ keys on average and bounded by $2r - 1$ keys, and $r$ is the number of group users excluded from the recipients of the current session. SD further requires a storage overhead of $\mathcal{O}(\log^2 N)$ keys at each user. When applied in WSNs, SD is still highly inefficient. For example, consider a multicast session in a WSN that consists of 10,000 sensors. If the sink wants to multicast a subset of sensors, say 8,000 of them, the size of the rekeying message for this session is $2,500$ keys on average; and such rekeying messages are broadcasted to the whole WSN. Obviously, this is impractical

in WSNs. We further note the existence of PKC based broadcast encryption schemes [24], which are efficient in communication. However, because these schemes are highly inefficient in computation as they require a large number of PKC operations, they are also inapplicable in WSNs.

*Other Multicast Encryption Schemes*: In [121], GKMPAN was proposed to address secure multicast in ad hoc networks. GKMPAN assumes that all nodes in an ad hoc network are predistributed certain number $m$ of keys randomly out of a big pool of $l$ keys, which are used to update group keys. If a node is compromised, the key server first determines a non-compromised key that is the most common among the remaining members of the group. Then, the key server broadcasts a new group key encrypted with the chosen non-compromised key. Consequently, nodes that have this key can decrypt the group key independently. These nodes further re-encrypt the new group key with another non-compromised key and forward it to those neighbors yet to obtain it. In this way, the new group key is propagated to all the members in a hop-by-hop fashion. However, GKMPAN is vulnerable to the selective node compromise attack. Compromising as low as 50 out of 1,000,000 nodes could be sufficient to break the whole scheme given $m = 100$ and $l = 5,000$. This attack is possible because the attacker can derive which keys are carried by which nodes simply based on nodes' ids, and hence could selectively compromise those nodes carrying no keys in common. Additionally, GKMPAN only supports single multicast group scenario. Hence, it is inapplicable in WSNs.

In [76], LKHW was proposed, which directly applies the LKH technique into WSNs while using directed diffusion [37] to support membership management. LKHW only considers the single group case, and also suffers from many attacks. There are also two other group key rekeying schemes proposed for WSNs. The scheme proposed in [14] aims to maintain a network wide group key in the presence of node compromise; and the scheme in [114] provides an approach to renew group keys for mulitgroups. None

of the above schemes supports the multicast group semantics discussed in Section 3.1.

## 3.3 GPLD: Setup

### 3.3.1 System Assumptions and Design Goals

*Network Model*: In this work, we consider a large-scale WSN that monitors a vast terrace of interest via a large number of static sensors of different functionalities. We assume that the WSN is densely deployed and always well connected; sensors of each class are also interconnected among themselves. We further assume that the approximate estimation on the size and shape of the terrain of interest is known a priori. Without loss of generality, we assume that the terrain is square in shape. In WSN, there exists a sink which is the data collection center equipped with sufficient computation and storage capabilities. We assume that all sensors can receive the messages from the sink, since the WSN is well connected. We do not address reliability issue of the message delivery [72], since it is orthogonal to this work. In this work, the sink is centralized authority being responsible for the key management tasks to ensure multicast security. We assume that sensors are classified into several different classes based on their functionalities and resource-constrained regarding computation, communication, and storage capabilities. Sensors are also not tamper-resistant.

*Threat Model*: We assume that the WSN is deployed in hostile environments with attackers exist. The attackers not only eavesdrop all the network communications, but also are able to compromise a small number of sensors in order to obtain the contents of the messages multicast by the sink. On the other hand, we also assume that compromised sensors can be detected in a timely manner, and no new sensors are compromised before the current rekeying operation is completed. We do not specify the particular mechanisms that detect compromised sensors, as it is orthogonal to this paper. But schemes like watchdog [116] can be well suited for this purpose.

We note that before compromised sensors are detected, no key management scheme is able to prevent information from being leaked to the adversary through compromised sensors. However, an effective key management scheme can always exclude the detected compromised sensors from the WSN so that no further damage can occur. Furthermore, we assume that the sink is always secure and has a secure mechanism (e.g., $\mu$TESLA [75]) to authenticate its multicast messages to all sensors. In addition, we do not consider Denial of Service (DoS) attacks against multicast messages as it is also out of the scope of this paper.

*Design Goals*: GPLD is designed to achieve the following goals: 1) Support the multicast group semantics discussed in Section 3.1; 2) Provide efficient group key distribution mechanism to support ad-hoc group formations; 3) Provide efficient rekeying mechanism to support group membership dynamics.

### 3.3.2 The *'Global-Partition, Local-Diffusion'* Technique

The performance of secure multicast schemes is determined by its group key distribution and/or rekeying operation overhead, as well as the storage and computation overhead. In most existing schemes, it is always the central authority's sole responsibility to deliver each individual group member the keying materials whenever required; group members are all end hosts, which neither have the responsibility nor are possible for such tasks[2]. However, for secure multicast in WSNs, sensors are both group members and routers; any multicast message sent by the sink has to be relayed by intermediate sensors before reaching all the target recipients. Consequently, it is possible and also convenient for sensors to diffuse the group key obtained to other members of the same group at their vicinities. The sink thus could reduce the length of the keying materials it broadcasts to the whole WSN.

Based on this key observation, we develop a *Global-Partition, Local-Diffusion* tech-

[2]One group member might not even aware the existence of other group members.

nique, which provides highly efficient group key distribution and rekeying operations. On the one hand, the proposed technique partitions sensors into a series of predefined elementary groups based on their location and class information. According to this partition, the proposed technique further assigns each elementary group a common group key encryption key (GKEK), and preloads each sensor with the GKEKs corresponding to all the elementary groups it belongs to. The proposed technique can hence efficiently support dynamic group formation by utilizing elementary groups and the corresponding GKEKs. These GKEKs can be used to efficiently and securely deliver the fresh group keys to the members of the dynamically formed groups. On the other hand, the proposed technique further avoids a large portion of global (sink-to-sensor) keying material traffic by carrying minimum number of GKEKs; But it still guarantees that all the group members obtain the group keys by allowing efficient local (sensor-to-neighbor-sensor) key diffusion.

### 3.3.3 Grid and Elementary Group Setup

*Grid Setup*: Before network deployment, the network planner prepares a *geographic virtual grid system* for the targeted terrain [83], which partitions the terrain into multilevel cells of different sizes following a quad-tree approach. Such a grid is described through three parameters, i.e., $< (x_0, y_0), L, len >$. $(x_0, y_0)$ is a reference point of the grid, which is usually set as the location of the sink for convenience; $L$ is the maximum level of the corresponding quad-tree; and *len* is the side length of the lowest level cells. Note that sensors in the same lowest level cell are always within the direct communication range of each other. Fig. 3.2(a) shows an example of such a grid, where the quad-tree has four levels, i.e., $L = 4$, and level-1 is the lowest level. Each cell in the grid is uniquely indexed based on its position; a level-$i$ cell is uniquely indexed by $L - i$ digits with each digit ranging from one to four. Particularly, the level-$L$ cell refers to the whole WSN and is indexed by 0. In the example, cell 222

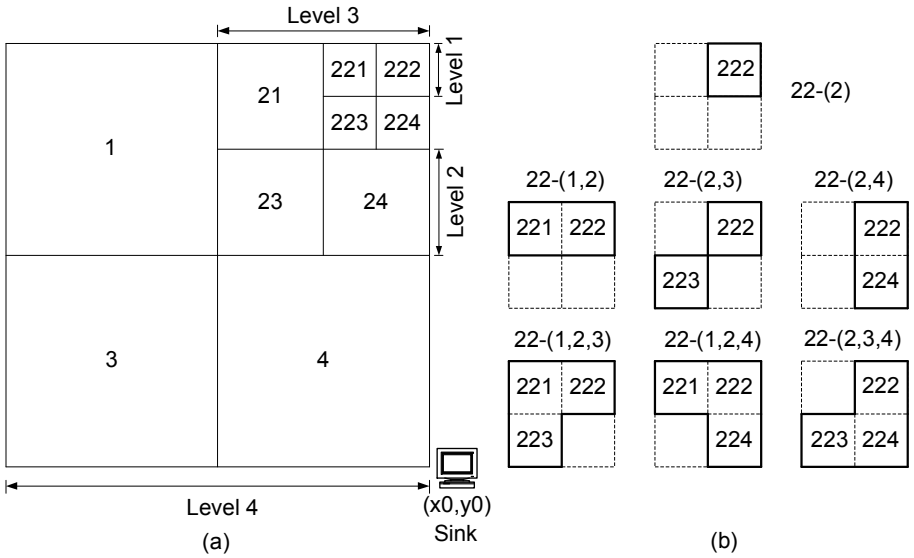


Figure 3.2: a) A virtual grid system that partitions sensor field using a quad-tree approach (L = 4); b) Seven level-1 location-based elementary groups that all sensors located at cell 222 belong to and their group IDs.

denotes a level-1 cell located at the top right corner of its belonging level-2 cell; and this level-2 cell is located at the top right corner of its own belonging level-3 cell, etc. In our definition, if a sensor is located at a certain cell, we call this cell a *home cell* of that sensor. Clearly, every sensor has one *home cell* at each level.

*Elementary Groups*: GPLD further defines the following six kinds of elementary sensor groups based on the grid concept:

1. Network-wide group: Sensors from the level-L cell form a network-wide sensor group.

2. Individual groups: Each sensor itself is an elementary group by definition.

3. Neighbor-pair groups: Each pair of immediate neighbor sensors form such a group.

4. Class-based groups: Sensors of each different class form a class-based group,

respectively.

5. Location-based groups: For every four level-i ($i \in [1, L-1]$) cells constituting a level-(i+1) cell, sensors from each possible combination of these four level-i cells form a location-based group, respectively.

6. Location-class-based groups: Within each location-based group, sensors of each different class form a location-class-based group, respectively.

Here, the network-wide group is the largest group, while an individual group is the smallest, and a neighbor-pair group is the second smallest. Furthermore, we say that one elementary group is larger than another, if the former contains more level-1 cells than the latter; and a location-based group is said larger than a location-class-based group containing the same number of level-1 cells.

*Group ID*: Each of these elementary groups is uniquely indexed in GPLD to facilitate the subsequent scheme operations. For the network-wide group, the group ID is set as (`'all'`). For an individual group corresponding a sensor $S_u$, the group ID is set as (`sink, u`). For a neighbor-pair group between two sensors $S_u$ and $S_v$, the group ID set as (`u,v`), suppose $u < v$ in its binary expression. For a class-based group corresponding to $C_j$, the group ID set is as ($\mathtt{C_j}$). For a location-based group at level-i, the group ID is set as the ID(s) of the corresponding cell(s) at level-i with common prefix suppressed. An example is shown in Fig. 3.2. For a location-based group at level-1 consisting of cells 222 and 223, we have its group ID as (`22-(2,3)`). Last, for a location-class-based group regarding $C_j$, its group ID is composed of $C_j$ and the ID of the corresponding location-based group it derives from. For a location-class-based group regarding $C_j$ at level-1 consisting of cells 222 and 223, we have its group ID as ($\mathtt{22\text{-}(2,3)}, \mathtt{C_j}$). This indexing approach allows to 1) compare the size of different groups directly from their group IDs; 2) support efficient location-based message forwarding as will be shown shortly.

### 3.3.4 Key Setup

GPLD initializes each sensor with the GKEKs corresponding to the elementary groups it belongs to during the bootstrapping phase. GPLD adopts a robot-assisted network bootstrapping technique [118]. We assume that a group of mobile robots are dispatched to sweep across the whole sensor field along pre-planned routes after the deployment of sensors. Mobile robots have GPS capabilities as well as more powerful computation and communication capacities than ordinary sensors. The leading robot is also equipped with the network master secret key $K$. The robots securely localize every sensor using the secure localization protocol given in [11]. For a sensor $S_u$ of class $C_j$ with its level-1 *home cell* as $a_{L-1} \cdots a_i \cdots a_1$ ($a_i \in \{1, 2, 3, 4\}, i = 1, ..., L-1$), the following GKEKs corresponding to the elementary groups it belongs to are loaded:

1) A *broadcast key* (BCK): corresponding to (`'all'`); $\mathsf{BCK} = H(K|0|K)$, where '|' denotes concatenation operation, and $H()$ denotes a cryptographically secure hash function such as SHA-1 [69].

2) An *individual key* (IDK): corresponding to (`sink,u`); $\mathsf{IDK} = H(K|\mathtt{u}|K)$. IDK is known only to $S_u$ and the sink.

3) A set of *pairwise key*s (PWKs): For every pair between $S_u$ and its immediate neighbors, there is a PWK; corresponding to (`u,v`) formed by $S_u$ and a neighbor $S_v$, $\mathsf{PWK}_{u,v} = H(K|\mathtt{u}|\mathtt{v}|K)$, assuming $\mathtt{u} < \mathtt{v}$.

4) A *class key* (CLK): corresponding to ($\mathtt{C_j}$); $\mathsf{CLK} = H(K|C_j|K)$.

5) A set of *location-aware key*s (LAKs): At each level, $S_u$ belongs to all the groups that involve $S_u$'s *home cell* at that level. There are totally seven such groups at each level. The corresponding group IDs and LAKs at level $i$ are:

$\mathtt{a}_{L-1} \cdots \mathtt{a}_{i+1}\text{-}(\mathtt{a}_i)$ :

$\mathsf{LAK}^{a_i}_{a_{L-1} \cdots a_{i+1}} = H(K|a_{L-1} \cdots a_{i+1} a_i|K)$,

$\mathtt{a}_{L-1} \cdots \mathtt{a}_{i+1}\text{-}(\mathtt{a}_i, \mathtt{a}'_i)$ :

$\mathsf{LAK}^{a_i, a'_i}_{a_{L-1} \cdots a_{i+1}} = H(K|a_{L-1} \cdots a_{i+1} a_i|K|a_{L-1} \cdots a_{i+1} a'_i$

$|K)$, for $\forall a'_i \in \{1,2,3,4\} \backslash a_i$.

$\mathtt{a}_{L-1} \cdots \mathtt{a}_{i+1}\text{-}(\mathtt{a}_i, \mathtt{a}'_i, \mathtt{a}''_i)$ :

$\mathsf{LAK}^{a_i,a'_i,a''_i}_{a_{L-1}\cdots a_{i+1}} = H(K|a_{L-1} \cdots a_{i+1}a_i|K|a_{L-1} \cdots a_{i+1}a'_i$

$|K|a_{L-1} \cdots a_{i+1}a''_i|K)$, for $\forall a'_i, a''_i \in \{1,2,3,4\} \backslash a_i$.

Here, the sequence of the concatenation depends on the actual values of $a_i, a'_i$, and $a''_i$ and $a_i \neq a'_i \neq a''_i$. An example is illustrated in Fig. 3.2(b), where seven location-based elementary groups at level-1 that $S_u$ belongs to are shown, assuming that $S_u$'s *home cell* is $a_3a_2a_1 = 222$. The corresponding group IDs and LAKs are:

$\mathtt{22\text{-}(2)} : H(K|222|K)$,

$\mathtt{22\text{-}(1,2)} : H(K|221|K|222|K)$,

$\mathtt{22\text{-}(2,3)} : H(K|222|K|223|K)$,

$\mathtt{22\text{-}(2,4)} : H(K|222|K|224|K)$,

$\mathtt{22\text{-}(1,2,3)} : H(K|221|K|222|K|223|K)$,

$\mathtt{22\text{-}(1,2,4)} : H(K|221|K|222|K|224|K)$,

$\mathtt{22\text{-}(2,3,4)} : H(K|222|K|223|K|224|K)$.

The number of LAKs is $7 * (L-1)$ for every sensor.

6) A set of *location-class key*s (LCKs): For each location-based group $S_u$ belongs to, $S_u$ also belongs to the corresponding location-class-based group defined for class $C_j$ sensors; and a LCK is derived from the corresponding LCK as follows: $\mathtt{C_j}\text{-}\mathsf{LCK} = H(K|\mathsf{LAK}|C_j|K)$. For example, $\mathtt{C_j}\text{-}\mathsf{LCK}^{1,2}_{22} = H(K|\mathsf{LAK}^{1,2}_{22}|C_j|K)$. Clearly, the number of LCK for $S_u$ is also $7 * (L-1)$.

In addition to GKEKs, each sensor is also loaded with $< (x_0, y_0), L, len >$, and the locations of the sensors in its level-1 *home cell* and all eight neighboring level-1 cells. Note that the authentication between the sensors and the leading robot can be easily achieved using the technique introduced in [115]. We omit it here for space limit. By the end of the bootstrapping phase, mobile robots leave the sensor field and the leading robot should securely erase all the keys from its memory but should report

the locations of sensors to the sink. The assumption underlying this approach is that adversaries do not launch active and explicit pinpoint attacks on mobile robots at this stage which usually does not last too long. That is, the robots are not likely subject to compromise. We further note that the above bootstrapping operation can also be realized through key predistribution approach [28, 26], instead of using mobile robots. In this case, sensor nodes utilize secure localization protocols [117, 103, 53] to obtain their locations. The choice of the approaches could depend on their availabilities in practice.

## 3.4 GPLD: Operation

In this section, we illustrate how fresh group keys and key update keys can be efficiently distributed using the '*Global-Partition, Local-Diffusion*' technique.

### 3.4.1 Notation

$\mathbb{W}$: all network sensors except for the revoked ones

$\mathbb{N}$: all the *recipient sensor*s of a multicast/rekeying session

$\mathbb{R}$: all the revoked sensors in a rekeying session

$\mathbb{S}_u$: the set of all immediate (non-revoked) neighbor sensors of a sensor $S_u$

$\mathbb{E}$: an elementary group

$\mathsf{K_g}$: a fresh group key of a multicast session

$\mathsf{K_{upd}}$: a fresh key refresh key of a rekeying session

$\overline{\mathbb{S}}_u$: the (sub)set of $\mathbb{S}_u$ that contains only those *recipient sensor*s yet to obtain $\mathsf{K_g}$ or $\mathsf{K_{upd}}$ in a multicast/rekeying session.

$\mathsf{Msg}$: a to-be-sent message

$\mathsf{Hdr}$: a header attached to a to-be-sent message.

### 3.4.2 Multicast Operation

To ensure the security strength, GPLD requires the sink to generate a fresh group key for encrypting the to-be-sent message in each multicast session. For this purpose, the sink attaches a header to the message, which includes the specifications of the multicast group, and the keying materials that enable the *recipient sensor*s to recover the group key.

*Group Description*: As GPLD allows dynamic formation of multicast groups to support various multicast group semantics discussed in Section 3.1, it is impossible for sensors to know in advance their memberships of a given multicast session. Hence, there has to be a group description mechanism. One way to do so is to list all the IDs of the *recipient sensor*s in the message header. Another way, as in broadcast encryption schemes [67], is to list all the indices of keys that are used to encrypt the group key in the message header; if a sensor possesses one of the corresponding keys, it is a *recipient sensor* for the session. However, both methods are very costly in WSNs because the resulted message header could be very long in both cases. Moreover, both methods implicitly entail the use of network-wide flooding to deliver the multicast messages to the *recipient sensor*s, which is neither necessary nor efficient. Derived from the multicast group semantics discussed in Section 3.1, GPLD, however, efficiently describes multicast groups using the location and/or class information of the *recipient sensor*s. Since sensors are always deployed in a discrete manner at certain density, we can easily express location constraints in terms of basic geometric shapes, which can be efficiently expressed using simple mathematical representations. More importantly, this location-aware group description approach is naturally supported by efficient message delivery approaches like geocast [45, 100] so that network-wide broadcast can be avoided.

*Message Format*: In GPLD, a multicast message contains two parts, the header

and message body:

$$\{\mathsf{Hdr}, E(\mathsf{K_g}, \mathsf{Msg})\},$$

where $E(K, \bullet)$ is a symmetric encryption algorithm such as AES [70] and encrypts $\bullet$ with key $K$. Hdr further contains two fields: $\{\mathsf{Hdr} = \mathsf{Grp\_Spec}, \mathsf{GK\_Info}\}$. Grp_Spec contains the multicast group information so that each sensor can judge whether or not it is a *recipient sensor* of the session. $\mathsf{Grp\_Spec} = (\mathsf{Loc\_Info}, \mathsf{Cla\_Info})$, where Loc_Info is the description of the location constraints of $\mathbb{N}$, and Cla_Info is the class information of $\mathbb{N}$. Recall that $\mathbb{N}$ denotes the *recipient sensor*s of a multicast session. GK_Info contains the encrypted $\mathsf{K_g}$ and the ID of the elementary group corresponding to the GKEK used for encryption.

*Header Generation*: In a multicast session, Hdr is generated as follows, once $\mathbb{N}$ is determined:

1) Generate $\mathsf{Grp\_Spec} = (\mathsf{Loc\_Info}, \mathsf{Cla\_Info})$ according to the location and class constraints of $\mathbb{N}$.

2) Find the largest elementary group $\mathbb{E}$ with $\mathbb{E} \subseteq \mathbb{N}$; if there is a tie, select the one that is the closest to the sink.

3) Generate a fresh $\mathsf{K_g}$ and encrypt it with the GKEK corresponding to $\mathbb{E}$. GK_Info thus contains the encrypted $\mathsf{K_g}$ and the group ID of $\mathbb{E}$.

*Message Delivery*: GPLD employs geocast to deliver multicast messages. By making use of the location-aware nature of WSNs, geocast utilizes a greedy forwarding for the packet delivery toward the target region. In greedy forwarding, a packet is forwarded to only one of the neighbor nodes whose geographical location is closest to the destination. As soon as the message reaches the target region, a restricted flooding or intelligent flooding technique [100] can be used to disseminate the packet inside the target region. Specifically, the multicast message is delivered via a localized and hop-by-hop manner as follows:

1) The sink uses greedy forwarding to deliver Hdr to the region taken up by $\mathbb{E}$.

2) As soon as Hdr reaches the target region, sensors in $\mathbb{E}$ that receive Hdr directly recover $\mathsf{K_g}$ from the attached GK_Info.

3) Once having obtained $\mathsf{K_g}$, each *recipient sensor* $S_u$:

3.1) Determine whether or not it should diffuse the key to its neighbor *recipient sensor*s based on the underlying multicast technique, the preloaded location information of the sensors in its neighboring level-1 cells, and Grp_Spec. If not, proceed to Step 4). If yes, proceed to Step 3.2).

3.2) Find $\overline{\mathbb{S}}_u$ (Ref. Section 3.4.1) out of $\mathbb{S}_u$. For every member of $\overline{\mathbb{S}}_u$, find the largest elementary group $\mathbb{E}_i$ it belongs to (based on $S_u$'s own location knowledge), where $\mathbb{E}_i \subseteq \mathbb{N}$. If there is a tie, select the one that $S_u$ belongs to (if applicable), and otherwise randomly select one.

3.3) For every found $\mathbb{E}_i$, if $S_u \in \mathbb{E}_i$, encrypt $\mathsf{K_g}$ with the GKEK corresponding to $\mathbb{E}_i$; if $S_u \notin \mathbb{E}_i$, pick up one member from $\overline{\mathbb{S}}_u \cap \mathbb{E}_i$ and encrypt $\mathsf{K_g}$ with the PWK shared between $S_u$ and the selected member.

3.4) Replace GK_Info with the encryptions of $\mathsf{K_g}$ obtained in Step 3.3) and the group IDs corresponding to the GKEKs used for encryption. Locally broadcast the updated Hdr.

4) The sink further uses greedy forwarding to deliver $E(\mathsf{K_g}, \mathsf{Msg})$ to the region taken up by $\mathbb{N}$. As soon as $E(\mathsf{K_g}, \mathsf{Msg})$ reaches the target region, a sensor in $\mathbb{N}$ that receives it determines whether or not to diffuse it in the neighborhood based on the underlying routing strategy such as restricted flooding or intelligent flooding [100]. If yes, $S_u$ locally rebroadcast $E(\mathsf{K_g}, \mathsf{Msg})$.

5) Finally, every *recipient sensor* recovers Msg using the obtained $\mathsf{K_g}$, and deletes $\mathsf{K_g}$ in the end.

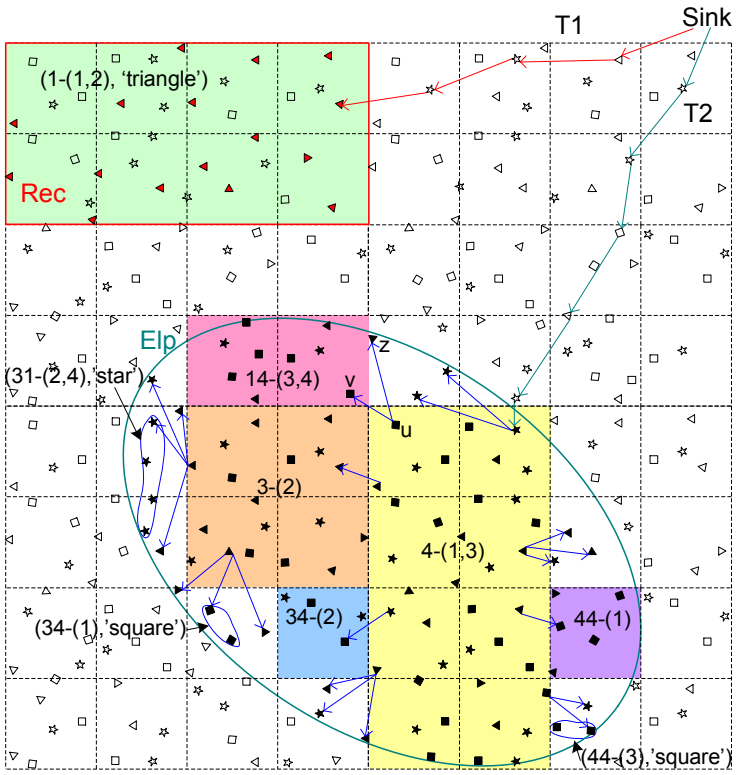


Figure 3.3: Two examplnary multicast sessions, where each solid symbol denotes a *recipient sensor*, each shadowed area denotes one location-based elementary group, and each of the three irregular circled area denotes a location-class-based elementary group.

### 3.4.3 Examples

The two examples shown in Fig. 3.3 illustrate a location-class-based multicast session at time T1 and a location-based multicast session at time T2, respectively. In the former session, the multicast group happens to be an elementary group. That is, $\mathbb{N}$ is the set of class '$\triangle$' sensors located inside `Rec` consisting of cells 11 and 12, i.e., the group $(\texttt{1-(1,2)}, \text{`}\triangle'\text{)}$, and `Rec` is the rectangle function. Hence, $\mathsf{Grp\_Spec} = (\mathsf{Loc\_Info} : \texttt{Rec}, \mathsf{Cla\_Info} : \text{`}\triangle'\text{)}$. According to the header generation algorithm, $\mathsf{GK\_Info} = (E(\text{`}\triangle'\text{-}\mathsf{LCK}_1^{1,2}, \mathsf{K_g}), (\texttt{1-(1,2)}, \text{`}\triangle'\text{))}$. The sink then uses greedy forwarding to send $\mathsf{Hdr}$ to the closest *recipient sensor* in `Rec`. Next, $\mathsf{K_g}$ is securely diffused among $\mathbb{N}$ according to the message delivery step 3). In this example, if a *recipient sensor* determines that it should diffuse $\mathsf{K_g}$, it simply locally rebroadcasts $\mathsf{Hdr}$.

In the latter session, $\mathbb{N}$ is the set of sensors located inside `Elp`, and `Elp` is the corresponding elliptic curve. Here, $\mathsf{Grp\_Spec} = (\mathsf{Loc\_Info} : \texttt{Elp}, \mathsf{Cla\_Info} : \text{`}\texttt{all}'\text{)}$. $\mathsf{GK\_Info} = (E(\mathsf{LAK}_4^{1,3}, \mathsf{K_g}), (\texttt{4-(1,3)}))$. Again, the sink uses greedy forwarding to send $\mathsf{Hdr}$ to the closest *recipient sensor*s in cells 41 and 43. Then $\mathsf{K_g}$ is securely diffused among $\mathbb{N}$. According to the scheme, $\mathsf{K_g}$ is securely diffused inside each shadow area (i.e., each corresponding location-based group) by using the corresponding LAK, respectively. For instance, inside $(\texttt{3-(2)})$, $\mathsf{K_g}$ is encrypted using $\mathsf{LAK}_3^2$. Furthermore, $\mathsf{K_g}$ is securely diffused from one elementary group to another using a GKEK shared between the sender sensor in the former group and the receiver sensor in the latter. For instance, Between $(\texttt{4-(1,3)})$ and $(\texttt{14-(3,4)})$, $\mathsf{K_g}$ is securely diffused from sensor $S_u$ to $S_v$ after being encrypted with $\mathsf{PWK}_{u,v}$; and between $(\texttt{4-(1,3)})$ to individual sensor $S_z$, $\mathsf{K_g}$ is securely diffused from $S_u$ to $S_z$ after being encrypted with $\mathsf{PWK}_{u,z}$.

### 3.4.4 Rekeying Operation

Once compromised sensors are detected, all the GKEKs they possess should be either obsoleted or securely refreshed in such a way that no compromised sensor could do so even by colluding. Thus, all subsequent multicast communications can be kept secret from the revoked sensors. GPLD supports both on demand and batched (periodical) rekeying strategies. Suppose $r$ compromised sensors, i.e., $\#\{\mathbb{R}\} = r$, are to be excluded in a rekeying session, where $r$ is usually a small number. The rekeying operation works as follows:

*The Sink*:

1) Find the largest location-based elementary group $\mathbb{E}$, where $\mathbb{E} \subseteq \mathbb{N} = \mathbb{W}$; if there are multiple sets of the same cardinality, select the one that is the closest to the sink.

2) Generate a fresh $\mathsf{K}_{\mathtt{upd}}$ and encrypt it with the LAK corresponding to $\mathbb{E}$.

3) Generate the rekeying message containing the following information: i) the IDs of revoked sensors; ii) the encrypted $\mathsf{K}_{\mathtt{upd}}$; iii) the group ID of $\mathbb{E}$; iv) $E(\mathsf{K}_{\mathtt{upd}}, \texttt{'Revocation'})$, the encrypted revocation notice.

4) Geocast the rekeying message to $\mathbb{E}$.

*Sensors (except for the revoked ones)*:

1) Diffuse $\mathsf{K}_{\mathtt{upd}}$ according to the same approach described for multicast operation (Ref. Section 3.4.2).

2) Perform key refreshing operation. For every GKEK held by each sensor (except for the IDK and PWKs), $\mathsf{GKEK} = H(\mathsf{K}_{\mathtt{upd}}|\mathsf{GKEK}|\mathsf{K}_{\mathtt{upd}})$.

3) Delete $\mathsf{K}_{\mathtt{upd}}$; delete the *revoked sensor*s from $\mathbb{S}_u$ and the PWKs shared with them, if any.

Hence, after a rekeying operation, all GKEKs held by the *revoked sensor*s are now obsoleted, and they are therefore permanently excluded from the WSN.

## 3.5 Security Analysis

*Correctness*: The correctness of GPLD derives from the following facts. First, no *revoked sensor*s excluded from the WSN can refresh the GKEKs they hold after revocation. This is true because the *revoked sensor*s can never obtain a $\mathsf{K_{upd}}$ using their GKEKs. Since the status of the system is reinstated to its original setting after every rekeying, we only need to consider the possible security issues that arise during a single rekeying operation. There are only two ways for a sensor to obtain a $\mathsf{K_{upd}}$ in a rekeying session. That is, a sensor recovers a $\mathsf{K_{upd}}$ either by directly decrypting the rekeying message sent by the sink or indirectly receiving it from a neighbor *recipient sensor*, which encrypts $\mathsf{K_{upd}}$ with a GKEK shared between the two and known only to the *recipient sensor*s. However, neither way can be exploited by *revoked sensor*s. A *revoked sensor* cannot recover $\mathsf{K_{upd}}$ because it has no corresponding GKEKs; at the same time, its neighbor sensors will not send it the key, as its ID is explicitly listed in the rekeying message. Without $\mathsf{K_{upd}}$, it is computationally infeasible for a *revoked sensor* to refresh its GKEKs due to the underlying cryptographically secure hash function used. Consequently, the *revoked sensor*s can never recover the group keys of the multicast sessions after their revocation, due to the obsoleteness of their GKEKs.

Second, the *recipient sensor*s can always verify the correctness of the update keys and group keys they obtain. The reason is as follows: 1) The authenticity of the rekeying and multicast messages, and hence that of $E(\mathsf{K_{upd}}, \texttt{`Revocation'})$ and $E(\mathsf{K_g}, \mathsf{Msg})$, can always be guaranteed through authentication schemes like $\mu$TESLA [75]. 2) Both `'Revocation'` and $\mathsf{Msg}$ follow certain predefined format and are meaningful. Therefore, by decrypting $E(\mathsf{K_{upd}}, \texttt{`Revocation'})$ and $E(\mathsf{K_g}, \mathsf{Msg})$, and verifying the validity of the recovered `'Revocation'` and $\mathsf{Msg}$, the correctness of the received $\mathsf{K_{upd}}$ and $\mathsf{K_g}$ can further be verified.

Last, GPLD allows that all *recipient sensor*s in a rekeying/multicast session to se-

curely obtain the corresponding $\mathsf{K}_{\mathsf{upd}}$ or $\mathsf{K}_{\mathsf{g}}$. That is, no sensor can be excluded from the session in GPLD as long as it is physically reachable. In the worst case, a sensor can always be updated through the IDK it shares with the sink. Note that the security of GPLD can be formally proved following the notion of '*key-indistinguishability*' [67]; we omit it here for space limit.

*Compromise Resilience*: Since sensor compromise is unavoidable when WSNs are deployed in hostile environments, it is crucial to minimize the resulted security risk. Ideally, after a sensor is compromised and before its revocation, the keying information it possesses should only allow the adversary to compromise those multicast messages, of which it is a legitimate *recipient sensor*; all other messages should still be kept secure against the adversary. That is, the security of a multicast message is broken only if at least one of the corresponding *recipient sensor*s is compromised and yet revoked. GPLD achieves this full security strength for all four multicast group semantics discussed in Section 3.1. The reason is that 1) A fresh key is always generated in each different rekeying/muliticast session; 2) The fresh key is securely diffused among the *recipient sensor*s, always encrypted with the GKEKs that are known only to the *recipient sensor*s.

*Other Attacks*: We assumed that the adversary may eavesdrop on all traffic, inject packets or replay old packets. Because the sink authenticates all the rekeying/multicast messages by $\mu$TESLA [75], no sensors can inject any fake messages into the WSN or modify any messages they forward while impersonating the sink. The adversary also cannot replay old rekeying packets because of time-stamp information is used in $\mu$TESLA. The adversary may also want to launch refusal-of-service attacks, such as dropping the packets and jamming the network[3]. However, *revoked sensor*s normally does not help the adversary drop the packets, because all the *revoked sen-*

[3]Such attacks are always possible and are not specific to multicast encryption schemes. Mechanisms dealing with such attacks can be found in [108].

*sor*s have already been excluded from the WSN, that is, no traffic is going through them. The worst situation caused by such attacks is hence equivalent to that due to packet losses. One salient property of GPLD is that it allows a sensor to miss certain multicast sessions without affecting its ability to participate any future multicast session, as long as it does not miss any rekeying operation. Therefore, GPLD is also resilient to such attacks.

## 3.6 Performance Analysis and Simulation

In this section, the performance of GPLD is analyzed. We mainly focus on the communication cost of GPLD, as it is most significant factor of energy consumption in WSNs. The computation and storage cost of GPLD is discussed as well.

### 3.6.1 Communication Overhead

*Models for Lower Bound and No-design Cases:* The lower bound of the communication overhead happens in the ideal situation where a different elementary group is established for each possible combination of network sensors; each sensor stores all GKEKs for the groups it involves in, which are up to $2^{\#\{\mathbb{W}\}-1}$ keys. In this ideal situation, every multicast group is an elementary group. Hence, to securely diffuse a message among the *recipient sensor*s, single GKEK corresponding to the elementary group is sufficient. On the contrary, when there is no pre-distributed keys for elementary groups except for the pairwise keys existing between neighbor-pairs, the multicast/rekeying message has to be encrypted using various PWKs at each step of the diffusion. This is the typical setting provided by most key management schemes designed for WSNs [28, 16], without involving any designs for the purpose of multicast encryption.

In the simulation, we adopts the above two models as the bases for analyzing

and comparing the communication overhead of GPLD. We denote the two models, in which the diffusion of messages is achieved through single GKEK (i.e. the lower bound case), and through only PWKs, as the LB model and the PWKD model, respectively. Hence, including GPLD, three models are simulated below.

*Simulation Settings and Evaluation Metrics:* The communication overhead of a multicast/rekeying session in GPLD consists of two parts: 1) the cost to unicast the multicast/rekeying message to the largest elementary group of the *recipient sensor*s; 2) the cost to locally diffuse the message among *recipient sensor*s. Since the former is relative small and is the same for all the models, only the latter is considered in the simulation. A multicast/rekeying message in GPLD contains the header and message body. We do not analyze the cost spent on message body since this cost is independent to the multicast encryption scheme. Instead, we focus on the header part, which contains i) the description of multicast group or revoked sensors, and ii) the keying materials. While the size of part i) is usually small and is identical in all the models, part ii) dominates the communication overhead of a multicast/rekeying session and may vary greatly in length.

Consequently, in the simulation, we use the total number of keys sent or forwarded by all the unrevoked sensors as the metric of evaluating the communication cost. Note that, in the case that a sensor sends/forwards the fresh group/update key to its neighbors using pairwise keys, we count the number of keys sent as the number of its neighbors. This sensor may put all the encrypted key materials in one message, but this will increase the length of the key materials transmitted anyway.

In the simulation, there are 10,000 sensors randomly distributed in the network, the size of which is 3000m x 3000m. The transmission range denoted as $tr$ is 100m or 135m, which is corresponding to 36 or 64 neighbors per sensor. For each setting, we run the simulation for 100 times, and calculate the average values. Two routing strategies are simulated. One is *Restricted Flooding* (**RF**), where each sensor broad-

| | RF $tr = 100$ $L = 6$ | RF $tr = 100$ $L = 7$ | RF $tr = 100$ $L = 8$ | OPC $tr = 135$ $L = 5$ | OPC $tr = 135$ $L = 6$ |
|---|---|---|---|---|---|
| LB | 858.86 | 940.50 | 878.92 | 107.64 | 398.38 |
| GPLD | 1844.81 | 1349.26 | 1233.88 | 285.63 | 478.28 |
| PWKD | 27489.39 | 30188.34 | 28107.67 | 830.43 | 869.33 |

Table 3.1: Comparison of Multicast Cost under Different Settings

casts any message received once using the key according to the largest elementary group to which it and all or part of its neighbors belong, and for those neighbors that only share pair-wise keys with this sensor, it will send the message to them individually. In the other strategy called *Once-Per-Cell* (**OPC**), the same message is broadcasted exactly once within any level-1 cell within the target region using a key corresponding to an elementary group covers this level-1 cell, if any. If such a key does not exist, the message is diffused using pairwise keys. Since in GPLD we assume that sensors in the same level-1 cell are always within the direct communication range of each other, the optimization can still ensure the successful transmission of fresh group/update keys.

*Multicast:* Table 3.1 compares the communication cost of a multicast session under all the models. In the simulation, the multicast group consists of all the sensors within a randomly-generated rectangle for simplicity. The lengths of the sides of the rectangle are uniformly chosen between 300m and 1500m. As shown in Table 3.1, not only GPLD is more efficient than the PWKD model under both RF and OPC, but also by appropriately choosing $L$ its communication overhead is only 20.06% and 40.39% more than the LB model under RF and OPC, respectively, with significantly less pre-distributed keys.

We also notice that, in RF the multicast cost of GPLD can be decreased by in-

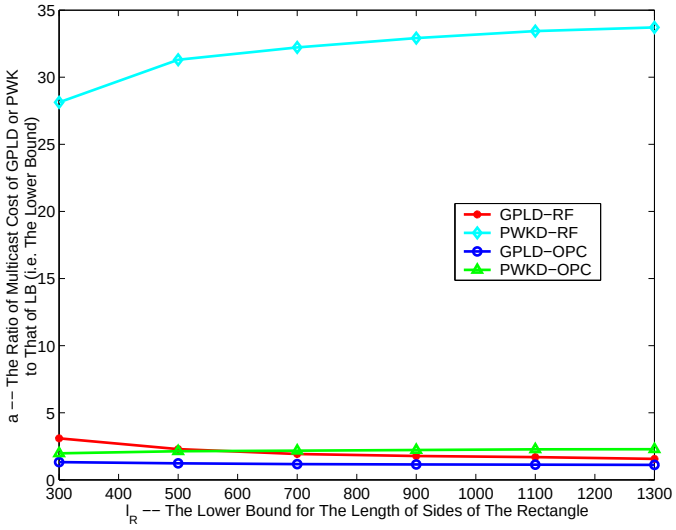


Figure 3.4: Multicast Cost (in terms of Ratio to The LB Model) under Different Multicast Group Sizes and Various Routing Strategies ($tr = 100m$, $L = 6$)

creasing the maximum level of the quad-tree, i.e. $L$. For example, the cost decreases by 26.86% when increasing $L$ from 6 to 7. However, the advantage of further increasing $L$ has recessive effects. When increasing $L$ from 7 to 8, the cost decreases by only 8.55%. Therefore, we need to balance between the storage overhead and the communication overhead while selecting the optimal value of $L$. Table 3.1 also shows that, by employing the optimal routing strategy (i.e. OPC[4]) the communication cost of GPLD can be lowered down to only 286 when $L = 5$. Since OPC helps only when the number of sensors per level-1 cell is more than one, we only simulate the scenarios of $L = 5$ and $L = 6$.

To evaluate the effectiveness of GPLD under different sizes of multicast groups, we uniformly choose the length of the sides of the rectangle between $l_R$ and $l_R + 200$, and increase $l_R$ from 300m to 1300m. Figure 3.4 shows that GPLD is more effective

[4]Since in GPLD we assume that sensors in the same level-1 cell are always within the direct communication range of each other, we cannot set $L = 5$ when $tr = 100m$. Thus, to show the effectiveness of OPC under different $L$'s, we simulate OPC under $tr = 135m$.

| | GPLD-RF | PWKD-RF | GPLD-OPC | PWKD-OPC |
|---|---|---|---|---|
| $r = 10$ | 1.007 | 61.127 | 1.015 | 2.442 |
| $r = 20$ | 1.012 | 61.103 | 1.027 | 2.443 |
| $r = 30$ | 1.017 | 61.017 | 1.038 | 2.444 |
| $r = 40$ | 1.022 | 60.949 | 1.049 | 2.444 |
| $r = 50$ | 1.027 | 60.884 | 1.060 | 2.445 |

Table 3.2: Comparison of Rekeying Cost under Different Number of Revoked Sensors and Various Routing Strategies

when the size of a multicast group is large. It is because, the larger is the area that a multicast group covers, the higher is the percentage that a fresh group/update key is encrypted by LAKs/LCKs instead of PWKs during the diffusion. By employing the method for optimally choosing the LAKs/LCKs (Ref. Section 3.4.2), the diffusion using LAKs/LCKs is more efficient than that using PWKs. As a result, GPLD presents higher efficiency for larger multicast groups.

*Rekeying:* In the simulation, for each rekeying session we randomly choose $r$ revoked sensors from the network. Table 3.2 shows the rekeying cost of GPLD and the PWKD model (in terms of the ratio to that of the LB Model) under different $r$ and various routing strategies, when $tr = 135m$ and $L = 6$. Similar to multicast, GPLD is more efficient than the PWKD model. Moreover, the ratio of the rekeying cost of GPLD to that of the LB model is much smaller than the multicast case. The extra overhead of GPLD over the lower bound is only 2.7% to 6%. It is due to the reason that, given that the number of revoked sensors is small, in a rekeying session the majority of diffusion messages are encrypted using LAKs/LCKs. More importantly, simulation results also show that the performance of rekeying in GPLD is not sensitive to the increase of the number of revoked sensors. For example, when $r$ increases from 10 to 50, the additional keying materials required are only around 160 under both

RF and OPC. It is a significant advantage over other works. Other schemes (like LKH and SD [106, 67]) either can only revoke one member per session, or have the revocation cost (i.e., the number of keys broadcasted to the whole network) at least linear to the number of revoked members.

### 3.6.2 Storage and Computation Overhead

*Storage Overhead*: In GPLD, a sensor stores the GKEKs corresponding to all the elementary groups it belongs to. Specifically, a sensor of class $C_j$ belongs to the network-wide group, the individual group of itself, and the class-based group consisting of all class $C_j$ sensors. Moreover, there are $n'$ neighbor-pair groups defined for each sensor, where $n'$ is the number of immediate neighbors a sensor has. Additionally, each sensor also belongs to 7*(L-1) location-based groups and 7*(L-1) location-class-based groups (Ref. Section 3.3.4). Therefore, there are totally $1+1+1+n'+7(L-1)+7(L-1) = 14L+n'-11$ GKEKs that should be stored by each sensor. In a WSN, $n'$ usually could range from 20 to 60, depending on different applications [28, 16, 90], while $L$ is a system parameter of the grid. Recall that sensors in a level-1 cell are within each others' direct communication range as required in GPLD. Then, the number of sensors in a level-1 cell is around 4 to 10, given $n'$ ranging from 20 to 60. Hence, for a WSN, whose size is no more than 100,000, $L = 9$ will be more than enough to support GPLD as there will be up to $4^{L-1} = 65,536$ level-1 cells. Thus, each sensor stores at most 161 GKEKs. Suppose each GKEK is 8 bytes, then 161 GKEKs require a storage space of 1.26 KB only. Also note that although the sink is required to know all the GKEKs, it does not have to directly store all of them. Instead, the sink could store only the master key and the locations of each sensor, and compute the GKEK on-the-fly.

*Computation Overhead*: The computation overhead introduced by GPLD is lightweight, as each sensor are only required to perform several times of encryption and decryption

operations over a very short message (i.e., one key). GPLD does not require sensors to perform any kind of expensive public-key or polynomial based operations.

## 3.7 Summary

In this chapter, we analyzed and classified the multicast group semantics for WSNs that are inherently demanded by most applications. We then proposed GPLD to address multicast encryption problem in WSNs, which, to our best knowledge, is the first scheme of its kind that supports various multicast group semantics and is tailored for WSNs. Our proposed scheme advances the current state-of-the-art by enabling not only the dynamic changing but also dynamic formation of multicast groups. We developed a novel multicast encryption technique called *global-partition, local-diffusion* to achieve scheme efficiency and meet the resource-constrained nature of WSNs. The security and performance of the proposed scheme are justified through both analysis and simulations.

# Chapter 4

# Data Report Security

In this chapter, we propose an integrated security design providing comprehensive protection over data confidentiality, authenticity, and availability. Our design establishes a location-aware end-to-end data security (LEDS) framework in WSNs.

In WSNs, data of interest, which may vary depending on different applications, usually appear as event reports sent by the sensing nodes from event happening area via multihop paths to the sink. As the communication range of sensor nodes are limited, the reports will be relayed by the intermediate nodes before finally reaching the sink. Hence, the requirement on data confidentiality in WSNs is naturally as follows: as long as the event sensing nodes are not compromised, the confidentiality of the corresponding data report should not be compromised due to any other nodes' compromise including the intermediate nodes along the report forwarding route.

Data reports collected by WSNs is usually sensitive and even critical such as in military applications, and hence, it is important to assure data authenticity in addition to confidentiality. Since the undetected compromised node(s) can always send false reports, cryptography can not fully prevent such attacks. However, if we require a valid report be collectively endorsed by a number, say $T$ ($T > 1$), of sensor nodes who sense the event at the same time, we can protect data authenticity to the

extent that no less than $T$ compromised nodes can forge a valid report. Furthermore, by exploiting static and location aware nature of WSNs, we can Furthermore require that a legitimate event report corresponding to certain area can only be generated by the collaborative endorsement of no less than $T$ nodes of that area. That is, to generate a valid report on a non-existing event happening at a certain area, the only way is to compromise $T$ nodes at that area, and otherwise impossible.

As compromised nodes are assumed existing in WSNs, it is important to prevent or be tolerant to their interference as much as possible to protect data availability. In this regard, security designs should be as robust as possible in the presence of compromised nodes. In-network processing such as false data filtering is important to save scarce network resources and to prolong network lifetime. To this end, any security design in WSNs should be highly resilient against two types of DoS attacks: report disruption attack [109] and selective forwarding attack [44], in which compromised nodes purposefully drop legitimate packets to disrupt the event report service by taking advantage of the en-route filtering policy.

## 4.1 Related Work

### 4.1.1 End-to-end vs. Hop-by-hop Design

In the past few years, many secret key pre-distribution schemes have been proposed [28, 17, 55, 56, 26, 25, 122, 16, 114]. By leveraging preloaded keying materials on each sensor node, these schemes establish pairwise keys between a node and its neighbors after network deployment for every network node, respectively, and thus form a hop-by-hop security paradigm. The security strength of these schemes is analyzed in term of the ratio of compromised communication links over total network communication links due to node compromise. Two types of node compromise are considered: random node capture and selective node capture, according to key distribution information

available to the attacker. Then to compromise the whole network communication, the attacker is forced to capture at least several hundreds of sensor nodes even under selective node capture attack. Hop-by-hop security design works fine when assuming an uniform wireless communication pattern in WSNs. However, in many applications node-to-sink communication is the dominant communication pattern in WSNs, that is, data of interest are usually generated from the event happening area and transmitted all the way to the sink. In this case, hop-by-hop security design is not sufficient any more as it is vulnerable to communication pattern oriented node capture attacks. Data confidentiality can be easily compromised due to lack of end-to-end security guarantee, since compromising any intermediate node will lead to exposure of the transmitted data. At the mean time, as the attacker could decrypt the intercepted data, it could therefore, freely manipulate them to deceive the sink and hence, severely affects data availability. The lack of end-to-end security association also makes it hard, if not impossible at all, to enforce data authenticity. We therefore conclude that end-to-end security design is much more desirable for WSNs as compared to hop-by-hop design when node-to-sink communication is the dominant communication pattern as it can offer a much higher security resilience.

### 4.1.2 Existing Data Report Security Designs in WSNs

The general approach adopted to protect data authenticity in WSNs is as follows: to generate a valid report, $T$ ($T > 1$) nodes that sense the event should first agree on the content of the event report, and in order to be forwarded by intermediate nodes and accepted by the sink, a valid report should be collaboratively endorsed (usually through Message Authentication Codes (MACs)) by these $T$ nodes. Reports that are not properly endorsed will be filtered out by the intermediate nodes or the sink. Here the assumption is that every event of interest can be detected by at least $T$ nodes simultaneously and the value of $T$ is a system parameter. In the past two years, a few

schemes have been proposed to design suitable key management schemes based on this approach, including Statistical En-route Filtering (SEF) [111], Interleaved Hop-by-hop Authentication (IHA) [120] and Location-Based Resilient Secrecy (LBRS) [109]. LBRS is the most recently proposed scheme, which aims to solve the problems identified in the two previous schemes (SEF and IHA), and is a major improvement over these two schemes. In both SEF and IHA, compromising $T$ nodes could break down the whole scheme. That is to say, after compromising $T$ nodes, the attacker can then freely forge events "appearing" at arbitrary locations without being detected. In LBRS, the damage caused by node compromise is reduced due to the adopted location-key binding mechanism. Compromising $T$ nodes now enables the attacker to fabricate events "appearing" at certain areas without being detected. However, it is still far from achieving the data authenticity requirement as stated above: to generate a valid report on a non-existing event happening at a certain area, the only way is to compromise $T$ nodes at that area, and otherwise impossible. Therefore, there is still a big gap between the protection that existing schemes can offer and the requirement of data authenticity.

In addition, all three schemes mentioned above are highly vulnerable to report disruption attack and selective forwarding attack. A single compromised node may disrupt the event report service originating in its vicinity or passing through it. Once a node in a certain area is compromised, the attacker can disrupt any event report from that area from being forwarded to the sink thereafter by simply contributing a wrong MAC to the final report. Since the en-route filtering allow intermediate node to drop packets with false MACs, such reports will be rejected on its way to the sink because of the presence of the wrong MAC(s). On the other hand, with the common one-to-one forwarding approach, a compromised node can also drop any data report sent by its downstream nodes. Since the received report can only be verified by the compromised node at that point, there is no way for other nodes in its

vicinity to distinguish such malicious dropping from legal dropping due to failing to pass the endorsement verification. As the number of compromised nodes increases, the resulted damage will increase drastically as discussed later in Section V. Hence, data availability in these schemes is poorly assured. The scheme presented in [114] is a group key pre-distribution method which can serve as a base for designing secure event report delivery approaches.

## 4.2 LEDS: Location-aware End-to-end Data Security Mechanism

### 4.2.1 Assumptions, Threat Model and Design Goals

**System Assumptions**: In LEDS, we consider a large-scale uniformly distributed WSN that monitors a vast terrain of interest via a large number of static sensor nodes, which can be deployed via approaches such as aerial scattering. We assume that an approximate estimation on the size and shape of the terrain of interest is known a priori. Once deployed, each node is assumed to be static and can obtain its geographic location via a secure and suitable localization scheme such as [11, 53, 103, 118, 117]. The network deployment guarantees that the established WSN is well connected and dense enough to support fine-grained collaborative sensing and be robust against node lost and failure. We assume that each event of interest can be detected by multiple sensor nodes [111, 120, 109]. Once an event happens, the sensing nodes agree on a synthesized report, which is then forwarded toward the sink, typically traversing a large number of hops. The sink is a data collection center equipped with sufficient computation and storage capabilities. We assume every sensor node has an unique *id* and is similar to the current generation of sensor nodes (e.g., the Berkeley MICA motes [33]) in its computation and communication capability and power resource. We

also assume sensor nodes are not tamper-resistant.

**Threat Model**: We assume that the attacker could compromise multiple nodes chosen arbitrarily and Furthermore assume that if the node is compromised, all the information it holds will also be compromised. However, the sink is assumed to be secure as it is usually well protected and under the direct control of the network owner [111]. We also assume that the attacker can eavesdrop on all traffic, inject packets, and replay older packets. The attacker can take full control of compromised nodes and thus can manipulate compromised nodes to drop or alter messages going through them. On the other hand, we assume there is a short bootstrapping phase right after network deployment during which no sensor node are compromised.

**Design Goals**: LEDS seeks to provide end-to-end data security, as well as en-route bogus data filtering in WSNs. In particular, we focus on the data such as event reports that are generated by the sensing nodes and transmitted from the sensing area to the sink. More specifically, the design of LEDS aims to achieve the following goals:

- Provide end-to-end data confidentiality and authenticity: Both confidentiality and authenticity of data reports should be guaranteed as long as the sending nodes themselves are not compromised. Moreover, the impact of compromised nodes (if any) should be confined to their vicinity. In other words, the attacker cannot utilize the cryptographic materials obtained from compromised nodes to launch attacks at places other than the locations of compromised nodes.

- Achieve high-level of assurance on data availability: 1) Be resilient against report disruption attack and selective forwarding attack; 2) Be able to early detect and drop bogus reports in an effective and deterministic manner, that is, having en-route filtering capability.

- Realize all the security goals in a single integrated design without relying on any

other security infrastructures; Be simple and efficient while providing end-to-end security guarantee; And have low computation and communication overheads for it to be suitable in WSNs.

### 4.2.2 Notation and Terms

For the convenience of description, we use the following notation and terms. Notation are give in Tab 4.1.

*geographic virtual grid*: A geographic virtual grid is a virtual geographic partition of the target terrain, which divides the terrain into multiple square cells. The parameters of a geographic virtual grid consists of a reference point and the cell size. For convenience, the reference point, referred to as $(x_0, y_0)$, is set to be the location of the sink, which is known before network deployment. For simplicity, we assume there is only one static sink in the WSN. The size of a cell is defined by $l$, which is the side length of the cell. A cell is uniquely indexed by its center's location. Thereafter, when we refer to the location of a cell, we use its center's location for convenience.

*home cell, event cell*: The cell that a node, say $u$, locates in after network deployment, is called *home cell* of $u$, denoted as $I_u$, and $I_u = (x_1, y_1)$ when its location is $(x_1, y_1)$. We call a cell an *event cell*, when a certain event of interest happens in that cell. Each report is therefore corresponding to one particular *event cell*.

*report-forward route*: In LEDS, an event report is relayed from the event cell to the sink in a cell-by-cell basis along its *report-forward route*. A report is always relayed between adjacent cells[1] towards the sink. More specifically, a report is always sent from one cell to one of its four adjacent cells that is closest to the sink[2]. The

[1]Two cells are adjacent if they share a common side.

[2]In the case that two adjacent cells have the same distance to the sink, an agreement to solve the tie needs to be pre-defined. For example, one may pick the cell that has smaller x coordinate. The purpose is to guarantee that the route pre-computed at the node would be the same as the actually route a report travels in a distributed cell-by-cell manner.

| | |
|---|---|
| $N$ | network size |
| $n'$ | number of nodes within one cell |
| $u, v, z, m$ | unique *id*s of sensor nodes |
| $I_u$ | index of node $u$'s home cell |
| $l$ | side length of a cell |
| $K_M^I$, $K_M^{II}$ | two master secret keys |
| $K_u$ | the unique secret key shared between $u$ and sink |
| $K_{I_u}$ | the *cell key* shared among the nodes in the same cell $I_u$ |
| $K_{I_u,I_v}$ | the *authentication key* shared between nodes in cell $I_u$ and nodes in cell $I_v$ |
| $H$ | pseudo-random functions |
| $M$ | the event report to be protected. |
| $C$ | encrypted report |
| $C_u$ | a share of $C$ computed through a LSSS, contributed by node $u$ |
| $C_{share}$ | a set of shares with $\|C_{share}\| = T$ |
| $E_\bullet(M)$ | encryption of $M$ using key "$\bullet$" |
| $Mac_\bullet(M)$ | the message authentication code (MAC) computed over $M$ using key "$\bullet$" |
| $T$ | the number of endorsements included when generating a valid report |
| $t$ | the minimum number of endorsements to validate a report |
| $r$ $(r > l)$ | communication radius of sensor nodes |
| $p$ | a large prime number |

Table 4.1: Notation

*report-forward route* of node $u$ therefore consists of all the cells that are intersected by the line segment that connects the center of $I_u$ and the sink (as shown in Fig. 1

(a)). These cells are sequenced according to their distances to the sink. The cell that a report travels first ranks first and so on.

*report-auth area*: The *report-auth area* of a node $u$ consists of two parts, the *downstream report-auth area* and the *upstream report-auth area*. They are both defined with regard to a sector area that is bound by two rays. Each of these two rays starts from the sink $(x_0, y_0)$ and goes through one vertex of cell $I_u$; and the two rays form the smallest angle which contains $I_u$ (as shown in Fig. 1 (b)). Then the *downstream report-auth area* of $u$ is defined to be all the cells that are farther to the sink than $I_u$ and each has at least half part located inside the sector-area, while the *upstream report-auth area* consists of all the cells that are closer to the sink than $I_u$ and have any part of them falls into the sector-area. Obviously, *report-forward route* of node $u$ is always a part of its *upstream report-auth area*.

*report-auth cell*: A cell is called a *report-auth cell* of node $u$, if it belongs to $u$'s *report-auth area* and every node in this cell shares an *authentication key* with $u$. Furthermore, if a *report-auth cell* of $u$ locates in the *upstream report-auth area* of $u$, it is a *upstream report-auth cell* of $u$. Otherwise, it is a *downstream report-auth cell* of $u$.

These terms are graphically illustrated in Figs. 4.1 and 4.2.

### 4.2.3 Scheme Overview

The proposed LEDS scheme consists of two major components: one is the underlying key management framework and the other is the corresponding end-to-end data security mechanism.

**Location-aware key management framework**: In LEDS, each node stores three different types of location-aware keys: 1) a *unique secret key* shared between the node and the sink which is used to provide node-to-sink authentication; 2) a *cell key* shared with other nodes in the same cell which is used to provide data

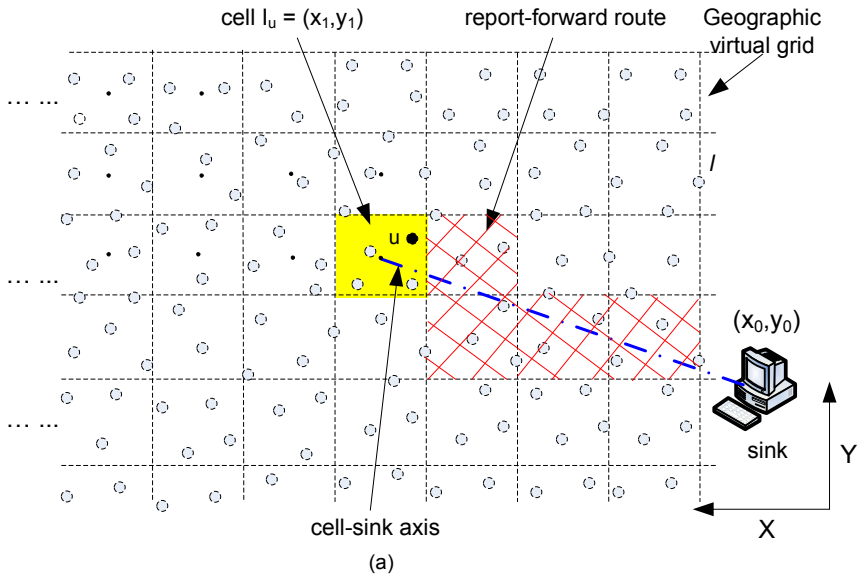


Figure 4.1: Term illustration I: defined for node $u$

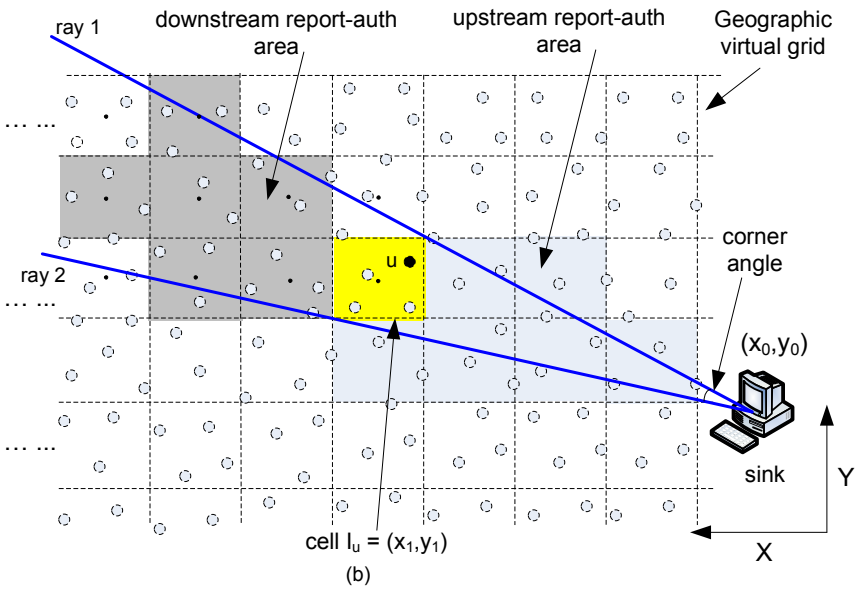


Figure 4.2: Term illustration II: defined for node $u$

confidentiality; 3) a set of *authentication key*s shared with the nodes in its *report-auth cell*s which are used to provide both cell-to-cell authentication and en-route bogus data filtering. Together with a predefined threshold secret sharing scheme, the key management framework serves as the basis for the upper layer end-to-end data security mechanism.

**End-to-end data security mechanism**: LEDS seeks to protect data reports in a comprehensive and end-to-end manner. *Data confidentiality*: In LEDS, every event report is encrypted by the corresponding *cell key* of the *event cell*. As the *cell key* is solely shared among nodes of the *event cell* and the sink, the confidentiality of the report is guaranteed as long as no node in the *event cell* is compromised. *Data authenticity*: 1) Each report is endorsed by multiple sensing nodes and the endorsements can be individually authenticated by the sink and 2) Each report is also authenticated in an interleaved cell-by-cell manner along the report-forwarding route. *Data availability*: 1) Be robust against *report disruption attack*s: the encrypted report is divided into a number of unique shares through a pre-defined linear secret sharing scheme (LSSS). Each share is independently generated by a participating node using its *unique secret key* shared with sink. A pre-defined number of MACs are then computed over all the shares using cell-to-cell *authentication key*s as another layer of endorsements, which enables the intermediate nodes to perform en-route filtering. 2) Be robust against *selective forwarding attacks*: Using cell-to-cell *authentication key*s guarantees that each report can be verified simultaneously by multiple next-hop nodes at any point in the route. This unique feature of LEDS makes it possible for the one-to-many data forwarding approach to be used in LEDS in stead of vulnerable one-to-one approach adopted by most existing security schemes. Sink finally verifies whether the report is indeed sent by the nodes from the *event cell* as claimed through examining both the authenticity of the MACs and the uniqueness of the shares. The sink can always recover the report from a subset of the shares even if a small number

of wrong shares exist due to threshold property of the underlying LSSS.

### 4.2.4 Protocol Detail

#### 4.2.4.1 Location-aware key management framework

Before network deployment, the network planner prepares a *geographic virtual grid* of the targeted terrain with reference point $(x_0, y_0)$ and cell size $l$. Based on total number of nodes in the network $N$, cell size $l$, and average number of nodes in each cell $n'$, the network planner further decides the values of $T$ and $t$: the former is the number of endorsements included when generating a valid report and the latter defines the minimum number of correct endorsements to validate a report. The impact of different values of these parameters will be discussed in Sections IV and V when we analyze security strength and performance of LEDS. The network planner also prepares two master secret keys, $K_M^I$ and $K_M^{II}$. In addition, a large prime number $p$ is prepared, which together with $t$ and $T$ defines a $(t, T)$ LSSS over finite field $GF(p)$.

LEDS adopts a robot-assisted network bootstrapping technique [117]. We assume that a group of mobile robots are dispatched to sweep across the whole sensor field along pre-planned routes after the deployment of sensors. Mobile robots have GPS capabilities as well as more powerful computation and communication capacities than ordinary sensors. The leading robot is also equipped with the following bootstrapping parameters

$$\{K_M^I, K_M^{II}, l, (x_0, y_0), (t, T), p\}.$$

The robots securely localize every sensor using the secure localization protocol given in [11], and load each of them the corresponding location-aware keys in a cell by cell manner.

Specifically, the robots first determine a node $u$'s *home cell* $I_u = (x_1, y_1)$, then

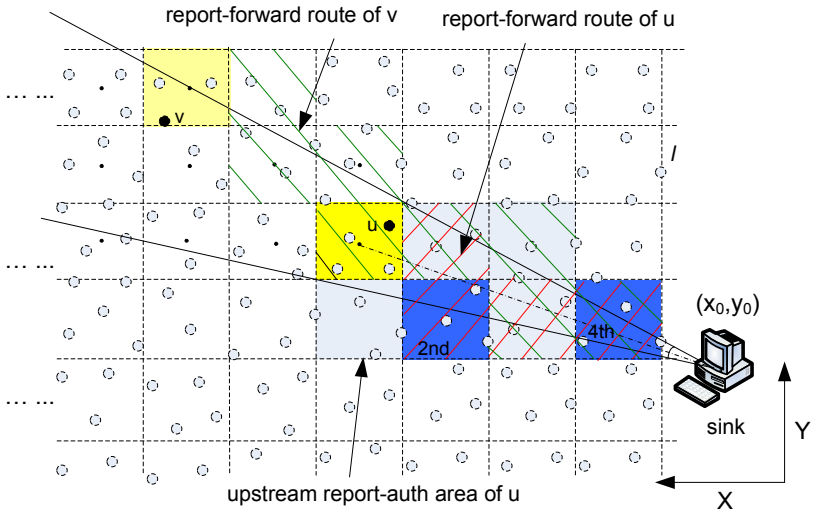


Figure 4.3: Illustration of *report-auth cell*s of node $u$

compute a *unique secret key* $K_u$ which $u$ shares with the sink as

$$K_u = H(K_M^I|u|I_u),$$

where | denotes concatenation operation. A *cell key* $K_{I_u}$ is further calculated which is shared among $u$ and other nodes in $I_u(x_1, y_1)$, and

$$K_{I_u} = H(K_M^I|I_u).$$

The robots load $u$ with $K_{I_u}$ as well as the ID list of all the nodes in $I_u$.

The robots next compute a set of *authentication key*s for all the sensors in the same cell. An *authentication key* is shared among all the sensors in a given cell and its corresponding *report-auth cell*s. Suppose a *report-auth cell* of $I_u$ has its location as $(x_c, y_c)$, then the *authentication key* between the two cells is

$$H(K_M^{II}|(x_1, y_1)|(x_c, y_c)).$$

The *report-auth cell*s of $I_u$ are determined according to $I_u$'s relative location with respect to the sink. In specific, a member of *downstream report-auth cell*s of $u$ is any cell in its *downstream report-auth area* that is no more than $T+1$ cells away from $I_u$[3].

[3]Adjacent cells are considered one cell away.

For example, all the grey cells shown in Fig. 4.2 are $u$'s *downstream report-auth cell*s with $T = 3$. On the other hand, cell $I_v$ is not such a member because only horizontal or vertical cell transverse is allowed in LEDS, that is, no diagonal cell transverse is allowed and hence, $I_v$ is five cells away from $I_u$. The quantitative analysis on the number of *downstream report-auth cell*s of a node will be discussed in Section V in the context of key storage overhead analysis.

Furthermore, the *upstream report-auth cell*s of $u$ comprise of the following ones: the robots first randomly rank all the sensors in $I_u$, assigning each of them a rank between 1 and $T$. Suppose $u$ is assigned a rank as $rank_u$, then the ($rank_u$ mod ($T$+1))-th cell in the *report-forward route* of $u$ is the first such a cell. The remaining ones for $u$ are those cells within its *upstream report-auth area* that are exactly $T + 1$ cells closer to the sink as compared to $I_u$. In case that $I_u$ is less than $T+1$ cells away from the sink, the sink itself is chosen. An example is shown in Fig. 4.3. Suppose $T = 3$, then the 2nd and 4th cells denoted in the figure are $u$'s *upstream report-auth cell*s.

In fact, for any two nodes $u$ and $v$, if $I_v$ is a member of *downstream report-auth cell*s of $u$, then

- every node in $I_u$ shares the *authentication key* $K_{I_u,I_v} = H(K_M^{II}|I_u|I_v)$ with at least one node in $I_v$. Furthermore, if two cells are exactly $T + 1$ cells away from each other in the *report-forward route* of $v$, then every node in $I_u$ shares $K_{I_u,I_v}$ with every node in $I_v$.

- the *upstream report-auth area* of $u$ is a part of that of $v$, that is, the *report-forward route* of $v$ falls into the *upstream report-auth area* of $u$ after the route reaches $I_u$ as shown in Fig. 4.3.

The robots also load every sensor with $\{(t, T), p\}$. The same bootstrapping procedure is repeated for all nodes in every cell. Note that the robots may also need to relocate a small number of sensors to ensure that each cell contains no less than $T$

nodes. The communication between the sensors and the leading robot can be easily secured using the technique introduced in [115]. We omit it here for space limit. By the end of the bootstrapping phase, mobile robots leave the sensor field and the leading robot should securely erase all the keys from its memory but should report the locations of sensors to the sink. The assumption underlying this approach is that adversaries do not launch active and explicit pinpoint attacks on mobile robots at this stage which usually does not last too long. That is, the robots are not likely subject to compromise. We further note that the above bootstrapping operation can also be realized through key pre-distribution approach [28, 26], instead of using mobile robots. In this case, sensor nodes utilize secure localization protocols [117, 103, 53] to obtain their locations. The choice of the approaches could depend on the security conditions and their availabilities in practice. We further point out that some sensors may be dislocated during the network lifetime in many scenarios. In this case, once they are dislocated, their possessed keys should also be updated according to their new location. Such dislocation and update operations can also be fulfilled by using the mobile robots.

#### 4.2.4.2 End-to-end data security mechanism

LEDS requires each valid event report to be encrypted and, at the same time, attached with $T$ endorsements from $T$ different nodes when generated from the *event cell*. While an event report is relayed to the sink, the intermediate nodes will drop any invalid endorsements to the report. Moveover, the report itself will be dropped when the number of valid endorsements becomes less than $t$. This is in contrast to the existing designs in which a report is dropped as soon as an invalid endorsement is found. The proposed design is important as it makes the system more robust in that it tolerates up to $T - t$ compromised nodes in an *event cell* colluding to launch *report disruption attack* by contributing invalid endorsements to the legal event

reports. Meanwhile, the requirement of multiple endorsements makes the system more reliable by disabling the possibility that up to $t-1$ compromised nodes of an *event cell* or unlimited number of compromised nodes from any other cell(s) collude to forge a report of event "appearing" at that *event cell*. The encryption prevents unlimited number of compromised nodes not in the *event cell* from colluding to obtain the content of the reports. LEDS further adopts a one-to-many report forwarding paradigm, which ensures the system being highly resilient to selective message forward attacks [44]. The detailed security mechanism is described as follows.

*Report generation*: Each of $T$ participating nodes first agree on an event report $M$ using the technique introduced in [112] based on signal strength strategy. $M$ usually contains information such as event type, sensing location (i.e., *id* of *event cell*'s), and a timestamp, etc. Note that all the related communications are protected by the *cell key* so that $M$ is confidential against any outside node. Next, each participating node, say $u$, encrypts $M$ using the cell key $K_{I_u}$ and obtains $C = E_{K_{I_u}}(M)$. $u$ further computes an unique share $C_u$ of $C$ through the predefined $(t, T)$ LSSS. In specific, $C_u$ is obtained by evaluating the following univariate polynomial of degree $t-1$ over finite field $GF(p)$ using $K_u$:

$$C_u = \mathcal{F}(K_u) = \sum_{0 \le i < t} a_i K_u^i \bmod p, \qquad (1)$$

where $a_i$ $(i = [0, t-1])$ are a full partition of $C$, and both $p$ and $t$ are the two preloaded parameters. Note that $C_u$ is uniquely generated by $u$ and therefore can be viewed as an endorsement to be verified by the sink. This is because the polynomial is evaluated using $u$'s unique secret key $K_u$ which is only known to $u$ and the sink. Node $u$ then broadcasts tuple $\{u, C_u\}$ and also collects the corresponding $T-1$ shares from other nodes. $u$ then computes two MACs over all the $T$ shares of $C$, i.e., $C_{share}$, as another layer of endorsement to the report, which enables the intermediate nodes to perform en-route filtering. The two MACs are computed using the authentication keys that $u$ shares with two of its *upstream report-auth cell*s. Suppose $I_v$ and $I_o$ are $u$'s

two *upstream report-auth cell*s, and both of them belong to $u$'s *report-forward route*, in which $I_o$ ranks $(T+1)$-th. Then the obtained MACs are $Mac_{K_{I_u,I_v}}(C_{share})$ and $Mac_{K_{I_u,I_o}}(C_{share})$. The tuple $\{u, Mac_{K_{I_u,I_v}}(C_{share}), Mac_{K_{I_u,I_o}}(C_{share})\}$ is then broadcast to complete the synthesization of the final report. Node $u$ constructs and sends out the final report after it collects $T+1$ different MACs and $2T$ MACs in total. The final report contains: 1) *event cell id*, 2) *id*s of $T$ participating nodes, 3) $C_{share}$, and 4) $T+1$ MACs. Note that both the *id*s of the participating nodes and the $T+1$ MACs are listed in the final report in order based on the node ranks (The common MAC is listed lastly). The report is sent by the node who completes the synthesis of the report and seizes the channel first. To avoid sending duplicate reports, each node overhears the channel and uses exactly the same random timer technique described in [109, 40].

*Interleaved cell-by-cell en-route filtering*: In LEDS, data reports are relayed cell by cell and delivered following a robust one-to-many, instead of existing failure-prone one-to-one, forwarding paradigm. A sending/intermediate node locally broadcasts a data report to the next cell in its route-forward route. As we mentioned before, it is easy to determine the next cell on the report-forward route, which is the one that is adjacent to the sending cell and is closer to the sink. Nodes in the receiving cell verify the report and, upon successful verification and processing, one of them rebroadcasts the report further to the next cell. Again, duplicate reports are suppressed by using the techniques like back-off before sending [109, 40].

In LEDS, an appropriate intermediate node authenticates a received report by checking 1) the validity of the first MAC attached in the report and 2) the number of non-zero MACs. The node verifies the first MAC attached in the report through using the corresponding *authentication key*:

- If the 1st MAC is zero, deletes it and attaches another zero to the next to the

end of the report[4];

- If the 1st MAC is valid, deletes it and attaches a new MAC to the next to the end of the report;
- If the 1st MAC is invalid, deletes it and attaches a zero to the next to the end of the report.

Here, the newly attached MAC is computed over $C_{share}$ using the corresponding *authentication key* shared between the node and one of its *upstream report-auth cell*s which is exactly $T+1$ cells closer to the sink with respect to its *report-forward route*.

The node also checks whether or not the number of non-zero MACs is enough and discards the report if the number is not enough. The number of non-zero MACs is considered not enough by an intermediate node if 1) it contains less than $t+1$ different non-zero MACs or 2) it contains less than $T-j+2$ different non-zero MACs, when *event cell* is $j$ cells ($j \in [1, T-t]$) away from its own. If there are enough number of non-zero MACs, the node now forwards the processed report to the next cell. Note that there is no way for a single node to launch selective forwarding attack, since each report can be verified by multiple nodes simultaneously. Every node in the same cell can be the one to forward a legal report. The pseudo-code of the above authentication procedure is shown in Table I.

*Sink verification*: A report is verified at the sink in two aspects to ensure its authenticity: 1) it verifies whether the report contains no less than $t+1$ valid non-zero MACs; 2) it checks whether the report is indeed endorsed by the $T$ nodes as claimed. Sink verifies 1) using the *authentication key*s it shares with the intermediate cells, and checks 2) by recovering the report $C$ from $C_u$. To do this, it tries to recover $C$ from any $t$ correct shares, and then decrypts the recovered $C$ using the corresponding *cell key* of *event cell*[5]. More specifically, the recovery operation of $M$

[4]That is, always keeps the common MAC the last.

[5]Based on the cell *id* contained in the report.

```
1   verify the 1st MAC contained in the report;
2   if (the 1st MAC is zero or invalid)
3     newMAC = 0;
4   if (the 1st MAC is valid)
5     newMAC = createMAC(C_share, key);
6     delete the 1st MAC;
7     attach newMAC to the next to the end of the report;
8   get number of different non-zero MACs;
9   if (((j ≤ T − t)&&(Num_of_MAC < T − j + 2)) || (Num_of_MAC < t + 1))
10   // the event cell is j cells away from its own
11     discard report; // not enough
12  else
13     forward report to the next cell; // enough
```

Table 4.2: Pseudo-code for authenticating a received event report

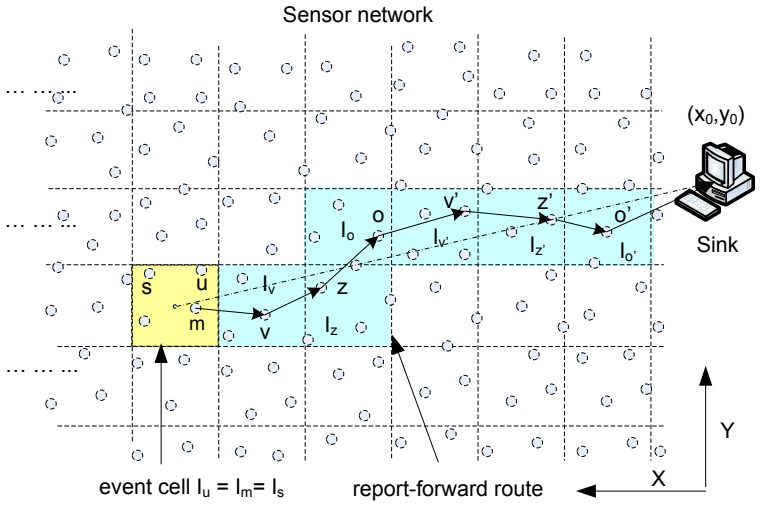


Figure 4.4: An example of the proposed end-to-end data security mechanism

goes as follows: sink picks $t$ out of $T$ shares, using their corresponding secret keys[6], sink solves a $t$-variable linear equation system to get $a_i, i = [0, t-1]$ in Equ. (1) and thus obtains $C$. Sink further decrypts $C$ and gets $M$. At this point, if $M$ is meaningful (i.e., conforming to the pre-defined report format), recovery operation succeeds. Otherwise, sink tries another combination of $t$ shares. Note that as long as there are no more than $T - t$ invalid shares, sink is always able to recover the original report due to the nice threshold property of the adopted $(t, T)$ LSSS. And as long as the sink can recover the original report $M$, it may ascertain that all the corresponding shares are indeed generated by the nodes as claimed.

### 4.2.5 An example

In Fig. 4.4, we show how the proposed data security framework works through a simple example. For brevity, we show the corresponding security operations only. Suppose $T = 3$, $t = 2$ and nodes $m, s$ and $u$ $(m < s < u)$ are three nodes from *event*

[6]Based on the node *id* contained in the report.

*cell*. Hence, a report can be:

$$\{I_u, m, s, u, C_m, C_s, C_u, Mac_{K_{I_u,I_v}}(C_m|C_s|C_u),$$

$$Mac_{K_{I_u,I_z}}(C_m|C_s|C_u), Mac_{K_{I_u,I_o}}(C_m|C_s|C_u),$$

$$Mac_{K_{I_u,I_{v'}}}(C_m|C_s|C_u)\}.$$

Then a successful protocol run goes as follows: when node $v$ receives the report, it checks that the report contains 4 non-zero MACs. Next, $v$ verifies the 1st MAC in the report using $K_{I_u,I_v}$. Then $v$ removes this MAC and attaches a new one to the end, which is also computed over $C_{share}$ but with $K_{I_v,I_{z'}}$, because $I_{z'}$ is 4 cells closer to the sink with respect to the *report forwarding route* of $I_u$. Lastly, node $v$ forwards the processed report:

$$\{I_u, m, s, u, C_m, C_s, C_u, Mac_{K_{I_u,I_z}}(C_m|C_s|C_u),$$

$$Mac_{K_{I_u,I_o}}(C_m|C_s|C_u), Mac_{K_{I_u,I_{v'}}}(C_m|C_s|C_u),$$

$$Mac_{K_{I_v,I_{z'}}}(C_m|C_s|C_u)\}.$$

As the report is forwarded along the route, it is Furthermore verified and processed by the intermediate nodes accordingly. Therefore, node $z'$ receives the report as

$$\{I_u, m, s, u, C_m, C_s, C_u, Mac_{K_{I_v,I_{z'}}}(C_m|C_s|C_u),$$

$$Mac_{K_{I_z,I_{o'}}}(C_m|C_s|C_u), Mac_{K_{I_o,sink}}(C_m|C_s|C_u),$$

$$Mac_{K_{I_{v'},sink}}(C_m|C_s|C_u)\}.$$

And sink receives the report as

$$\{I_u, m, s, u, C_m, C_s, C_u, Mac_{K_{I_z,I_{o'}}}(C_m|C_s|C_u),$$

$$Mac_{K_{I_o,sink}}(C_m|C_s|C_u), Mac_{K_{I_{v'},sink}}(C_m|C_s|C_u),$$

$$Mac_{K_{I_{z'},sink}}(C_m|C_s|C_u)\}.$$

Sink first verifies all the 4 MACs and then recovers the original $C$ from any two of $C_m, C_u$ and $C_s$. From the *id* information in the report and Equ. 1, sink solves a 2-variable linear equation system and thus obtains $C$. Sink Furthermore decrypts $C$ using $K_{I_u}$, and therefore obtains $M$. If $M$ is meaningful, the recovery operation succeeds. Sink won't be able to recover $M$ if there are more than $T - t = 1$ invalid shares. Hence, as long as sink could recover the report, it accepts the report.

## 4.3 Security Analysis of LEDS

In this section, security strength of the proposed LEDS is analyzed with respect to the three aspects as mentioned in design goals, i.e., data confidential, authenticity and availability.

### 4.3.1 Security Strength of LEDS Regarding Data Confidentiality

In LEDS, every report is encrypted by the corresponding *cell key* and therefore, no nodes out of the *event cell* could obtain its content. Compromising any number of intermediate nodes won't break the confidentiality of the report. Only when a node from the *event cell* is compromised could the attacker obtain the contents of the corresponding reports. We say a cell is compromised with regard to data confidentiality in this case. Our concern here is how compromised nodes under both random and selective node capture attacks affect the confidentiality of the communications from different cells? That is, given the number of compromised nodes, what is the fraction of the compromised cells with respect to total network cells?

*Random node capture attack*: Given network size $N$ and the average number of nodes in each cell $n'$, there are altogether $\frac{N}{n'}$ cells in a *geographic virtual grid*, assuming $n'$ divides $N$. Therefore, if $x$ nodes are compromised under random node

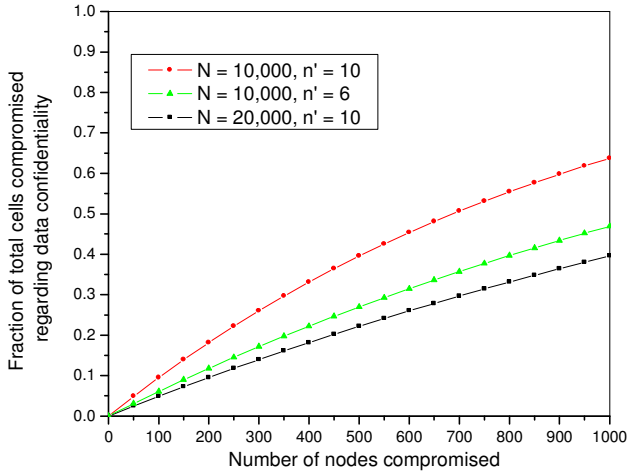


Figure 4.5: Data confidentiality in LEDS under random node capture attack

capture attack, the probability that a given cell is compromised is

$$1 - \frac{\binom{N-n'}{x}}{\binom{N}{x}} \quad (2)$$

On the other hand, Equ. (2) also represents the fraction of total cells that are compromised given $x$ nodes are compromised. In Fig. 4.5, we show how the number of compromised nodes affects data confidentiality in LEDS. It is clear that to compromise 40% of the total cells, at least 5% of the total nodes have to be compromised. This means at least 500 nodes, given $N = 10,000$ and $n' = 10$. Furthermore, the security resilience increases as $n'$ decreases as shown in Fig. 4.5. Therefore, LEDS compares favorably with respect to security resilience against random node capture attacks to existing security designs [17, 26, 25], in which compromising a few hundred nodes usually compromises all the network communications, given the same network size.

*Selective node capture attack*: In this case, to compromise the whole network, the attacker has to selectively capture at least one node from each cell. This implies at least $\frac{N}{n'}$ nodes are required, that is, around 1000 nodes, given $N = 10,000$ and $n' = 10$. Note that this is 10% of the total network nodes. In LEDS the damages caused by the compromised nodes are confined due to location-aware nature of the *cell*

*key*s. Compromised nodes in one area cannot be used to compromise communications originated from other areas, since they do not have any information on *cell key*s of other cells.

### 4.3.2 Security Strength of LEDS Regarding Data Authenticity

In addition to obtaining the content of legitimate reports, the attacker may also want to insert bogus reports to fool the sink with non-existing events. In LEDS, in order for a bogus report to successfully pass both en-route filtering and sink verification, the attacker has to compromise at least $t$ nodes in the corresponding *event cell*. We say a cell is compromised with regard to data authenticity in this case. Notice that under this worst case scenario, namely, $t$ or more nodes in a single cell have been compromised, only events "appearing" in that cell can be forged, due to the location-aware property of the underlying endorsement keys that provides both node-to-sink and cell-to-cell authentications. Therefore, LEDS presents an improvement over existing security designs such as SEF, IHA, and LBRS [111, 120, 109], in which compromising any single node would result in multiple gains, i.e., helping the attacker compromise the authenticity of both its own home cell/cluster and any of its downstream cells/clusters.

Therefore, our first concern is that given the number of compromised nodes, what fraction of the total cells are affected with respect to data authenticity? Under random node capture attack, if the number of compromised nodes is $x$, then the probability that a cell is not affected, i.e., no node in a cell is compromised, is given by

$$P_{\{0\}} = \frac{\binom{N-n'}{x}}{\binom{N}{x}} \quad (3)$$

This also represents the percentage of cells that are *secure*. Accordingly, the percentage of cells that have at least one node compromised, respectively, is given by $1-P_{\{0\}}$.

Furthermore, let $P_{\{i\}}$ represent the probability that exactly $i$ nodes are compromised in a cell, we have

$$P_{\{i\}} = \frac{\binom{n'}{i}\binom{N-n'}{x-i}}{\binom{N}{x}}$$

Then the probability that the authenticity of a cell is compromised, i.e., having at least $T$ compromised nodes is

$$P_{\{\geq t\}} = \sum_{i=t}^{n'} P_{\{i\}} = \sum_{i=t}^{n'} \frac{\binom{n'}{i}\binom{N-n'}{x-i}}{\binom{N}{x}} \quad (4)$$

This also represents the percentage of authenticity *compromised* cells. Then the percentage of *affected* cells, i.e., each of which has at least 1 and at most $t-1$ compromised nodes, can be expressed as $1-P_{\{0\}}-P_{\{\geq t\}}$. Fig. 4.6 illustrates how data authenticity is affected as the number of compromised nodes increases. It is observed that the percentage of compromised cells increases very slowly with the increase of number of compromised nodes. And it is kept very low: even if the compromised nodes reach 1750, only 10% of cells are compromised. This indicates that under random node capture attacks, it is very hard for the attacker to compromise a cell and thus fool the sink with the undetectable bogus reports. On the other hand, it is observed that the percentage of secure cells in the network deceases slowly while the percentage of affected cells increases quickly as the number of compromised nodes increases. This observation tells us that it is relatively easier for the attacker to insert the bogus reports into the network; however, these bogus reports can be deterministically filtered by the intermediate nodes or the sink.

Hence, our next concern is that given the number of compromised nodes, what's the expected filtering position of a bogus report sent from an affected cell? In LEDS, in order for a bogus report from an affected cell to reach the sink (but be rejected by the sink), there should be at least $t-x_2$ of the first $T$ cells in its *report-forward route* being affected simultaneously, assuming the number of compromised nodes in this affected cell is $x_2$ ($1 \leq x_2 \leq T-1$). This is because, to insert a bogus report, the

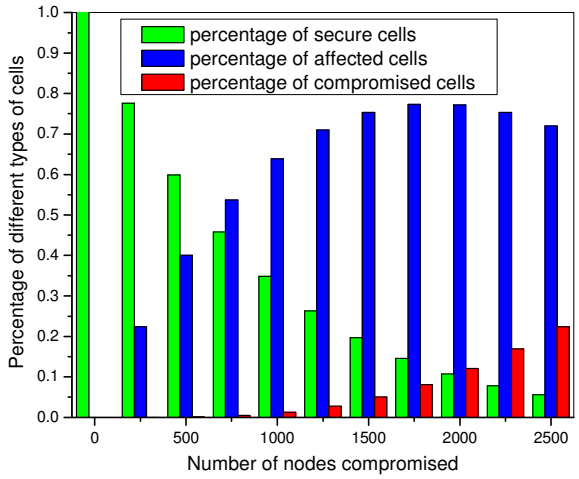


Figure 4.6: Data authenticity in LEDS under random node capture attack, where $N = 10,000$, $n' = 10$ and $(t, T) = (4, 5)$.

compromised nodes in this affected cell have to forge at least $t - x_2$ MACs to have enough number of them. And to let pass these $t-x_2$ invalid MACs, there should be at least $t - x_2$ affected cells of the first $T$ cells in its *report-forward route*: compromised node(s) from each affected cell could therefore let pass one corresponding invalid MAC and attach a new one as defined in LEDS. Therefore, there is no way for the intermediate nodes to check the authenticity of the received report after $T$ cells, since now all the contained MACs in the report are indeed valid ones. In this case, the filtering position of the bogus reports from this affected cell should be its distance to the sink. Otherwise, any bogus report from this cell will be filtered at most at $T$-th cell and $\frac{T}{2}$-th cell on average. Assuming there are less than $t - x_2$ affected cells of the first $T$ cells in its *report-forward route*, then at least one invalid MAC will be detected by nodes from the remaining secure cells. Now the bogus report originated from this cell will be filtered out at most at the $T$-th cell along the route. Under random node capture attack, the average filtering position will be bounded by $\frac{T}{2}$, since the invalid MAC can be detected at any position between the 1st and T-th cell. Therefore, given the number of compromised nodes as $x$, the expected filtering position of the bogus

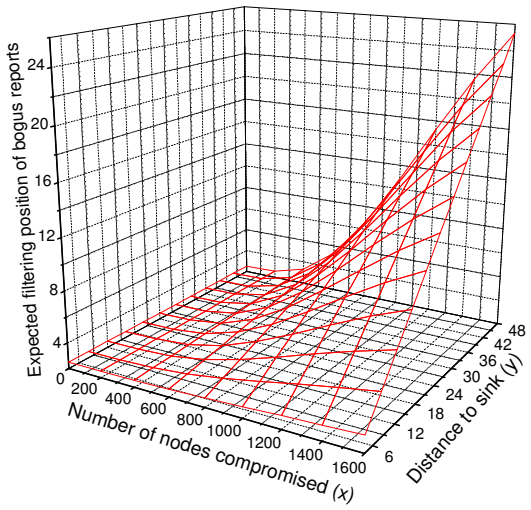


Figure 4.7: Expected filtering position vs. number of compromised nodes with respect to different distance to the sink

reports from an affected cell is bounded by

$$y\sum_{i=1}^{t-1}P_{\{i\}}(1-P_{\{0\}})^{t-i}+\frac{T}{2}(1-\sum_{i=1}^{t-1}P_{\{i\}}(1-P_{\{0\}})^{t-i}), \quad (5)$$

suppose this affected cell is $y$ cells away from the sink with respect to its *report-forward route*. Fig. 4.7 illustrates how the filtering position varies as the number of compromised nodes increases, when $N = 10,000$, $n' = 10$ and $(t, T) = (4, 5)$. It is clearly shown in Fig. 4.7 that the bogus reports sent from most affected cells can be efficiently filtered under random node capture attack. For example, the bogus reports from an affected cell that is 30 cells away from the sink will be filtered at less than the 10-th cell in the route on average, where the number of compromised nodes is 1000.

On the other hand, under selective node capture attack, the attacker can choose as low as $t$ nodes from one particular cell to compromise data authenticity of that cell. As discussed above, unlike existing security designs [109, 120, 111], compromised nodes from one cell in LEDS can not be used to compromise data authenticity of other

cells. Note that in existing security designs, data authenticity of one cell can always be compromised because of the compromise of nodes from other cells. Hence, this feature of LEDS greatly increases the attacker's cost to launch such attacks.

### 4.3.3 Security Strength of LEDS Regarding Data Availability

As discussed before, there are two possible attacks that could severely affect data availability in WSN, namely, report disruption attack and selective forwarding attack. Existing security designs are highly vulnerable to these attacks [111, 120, 109]. In contrast, LEDS makes significant improvement in terms of data availability by being more resilient to such attacks. The strength of LEDS comes from both its report endorsement mechanism and its forwarding mechanism.

On the one hand, in LEDS, each node only contributes one share of the report following a $(t, T)$ threshold LSSS. Therefore, the sink can always recover the original report even if there are up to $T - t$ compromised nodes from the corresponding *event cell* that contribute wrong shares to prevent the sink from obtaining the report. At the mean time, the intermediate nodes only discard a report which contains less than $t$ valid MACs. That is, if there are up to $T - t$ compromised nodes that contributes invalid MACs, the report can still be relayed to the sink. While in existing security designs, a single compromised node could prevent the sink from obtaining any report from that cell. Simply by contributing an invalid MAC to any report sent from that cell, the compromised node can always make the report to be discarded by the intermediate nodes. Under random node capture attack, given the number of compromised nodes $x$, the percentage of cells that have at least one node compromised, respectively, is given by $1 - P_{\{0\}}$; Furthermore the percentage of cells that have at

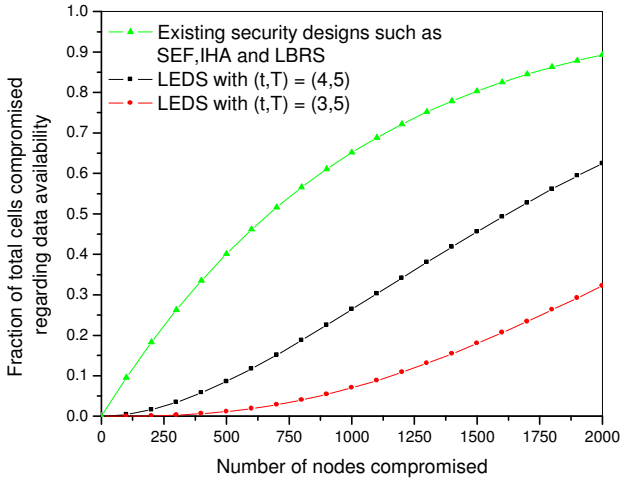


Figure 4.8: Data availability in LEDS under report disruption attack

least $T - t + 1$ nodes compromised, respectively, is given by

$$1 - \sum_{i=0}^{T-t} P_{\{i\}} \quad (6).$$

Fig. 4.8 compares the data availability protection of LEDS with other existing security designs. It clearly shows that LEDS is much more resilient to the report disrupt attacks. In other words, an attacher needs to compromise a lot more nodes to successfully launch report disrupt attacks in LEDS. Given $N = 10,000$, $n' = 10$ and $(t, T) = (4, 5)$, to successfully launch report disrupt attack in 10% of total cells, around 100 nodes have to be compromised in existing security designs, while this number has to be no less than 600 in LEDS. Furthermore, by increasing $T - t$, LEDS can increase the resilience even more, or in other words, making the attack even harder, as shown in Fig. 4.8. Lastly, even under selective node capture attacks, the cost to successfully launch report disrupt attack in the same number of cells in existing security designs, will still be $T - t$ times higher than in LEDS.

On the other hand, a compromised node can always drop all the reports going through itself in existing security designs due to the failure-prone nature of one-to-one forwarding paradigm. Compromising any intermediate node from the report-forward

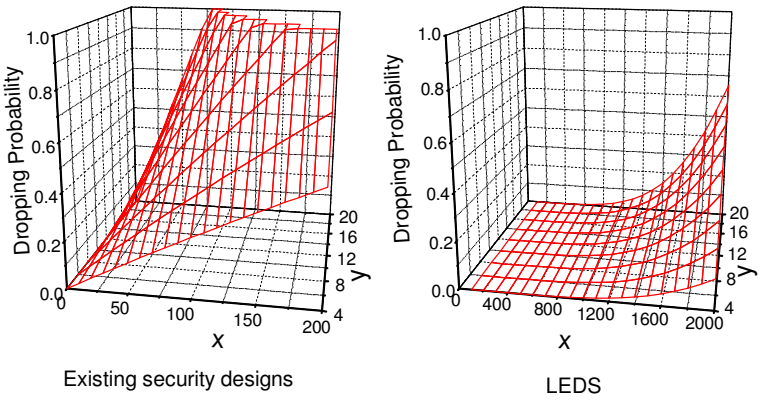


Figure 4.9: Data availability in LEDS under selective forwarding attack

route would be sufficient enough for the attacker to successfully drop the message without being detected, since other nodes have no appropriate keys to verify the authenticity of the report. However, in LEDS it is impossible for a compromised node to prevent the report from being forwarded. This is because every report in LEDS is forwarded to all nodes in the next cell and each of them function the same way. Therefore, as long as not all the nodes that hear the report are compromised, the report can always be forwarded to the next cell. Hence, the proposed one-to-many forwarding approach in LEDS greatly enhances data availability in WSNs.

More precisely, suppose a cell is $y$ cells away from the sink. Then applying one-to-one forwarding approach as in existing security designs, the probability that the corresponding report sent from this cell is dropped by a compromised intermediate node can be estimated by

$$\frac{yl}{r}(1 - P_{\{0\}}), \qquad (7)$$

under random node capture attack, while in LEDS this probability is bounded by

$$y(1 - \sum_{i=0}^{\lfloor \frac{n'(r-l)}{l} \rfloor} P_{\{i\}}), \quad (8)$$

assuming $l \leq r \leq 2l$. Fig. 4.9 clearly illustrates the huge improvement on data availability provided by LEDS.

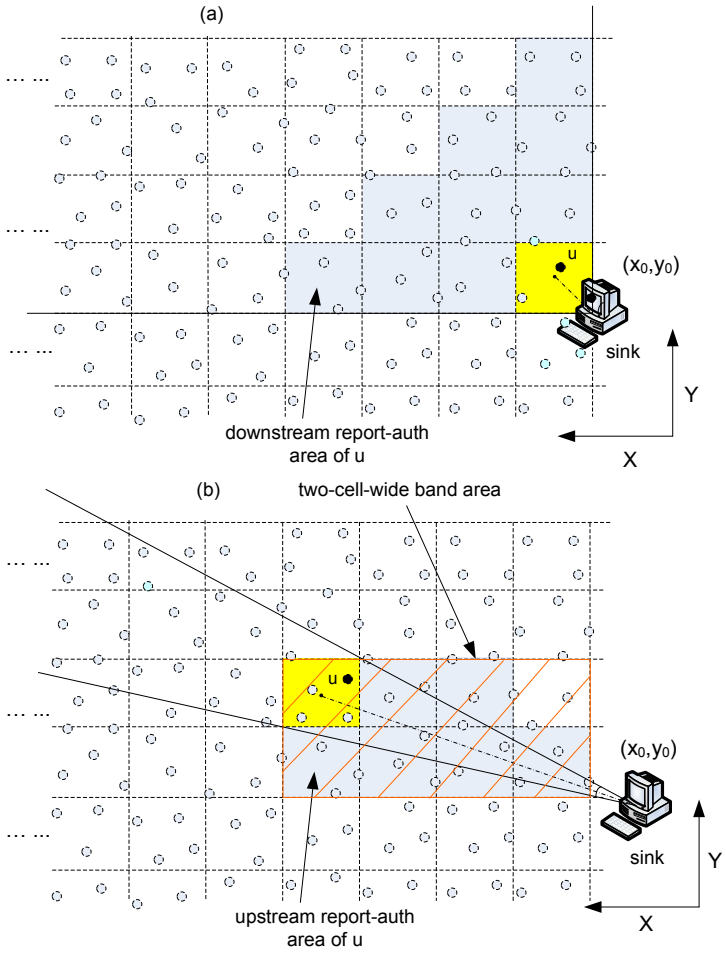


Figure 4.10: The upper bound of the number of *report-auth cell*s

## 4.4 Performance analysis of LEDS

In this section, we evaluate the performance of the proposed LEDS in terms of storage overhead, computation and communication overheads and energy savings.

### 4.4.1 Key storage overhead

In LEDS, each node stores an *unique secret key* which is only known to itself, and one *cell key* shared with all other nodes in its *home cell*. Of course, both keys are also known by the sink in addition. Furthermore, each node also stores one *authentication key* for each of its *report-auth cell*s. For a particular node, say $u$, the number of its *report-auth cell*s is decided by $u$'s relative position with respect to the sink.

More specifically, the number of *downstream report-auth cell*s of $u$ is bounded by $\frac{(T+1)(T+2)}{2}$, when *home cell* $I_u$ is right next to the sink as shown in Fig. 4.10(a). On the other hand, from its definition, we know that any node's *upstream report-auth area* is a subset of the two-cell-wide band area as shown in Fig. 4.10(b). Obviously, in a two-cell-wide band area, all the possible routes monotonically toward the sink[7] have at most two different choices at each step. Therefore, the cells that are exactly $T+1$ cells closer to the sink as compared to $I_u$ also have at most 2 different choices. Hence, the number of *upstream report-auth cell*s of any node is bounded by 3, and the total number of keys stored by each node in LEDS is bounded by

$$\frac{(T+1)(T+2)}{2} + 5. \quad (9)$$

Therefore, LEDS only requires the nodes to storage a small number of keys, which can be as low as 20, given $T = 5$. Moreover, the number of keys is independent to the network size, which makes LEDS highly suitable in large scale WSNs. Furthermore, the sink also stores very few keys in LEDS, i.e., two master keys $K_M^I$ and $K_M^{II}$ only.

[7] horizontal and vertical cell transverse only

All the other keys can be derived on-the-fly from the *id* and location information (i.e., *cell id*) contained in the received data reports.

### 4.4.2 Computation and communication overheads

In LEDS, key establishment only involves efficient hash operations during the bootstrapping period. And since the authentication keys are shared in a cell-to-cell manner, they can be reused for en-route filtering during whole network life. This feature saves a lot of unnecessary computation due to key reestablishment. In contrary, whenever forward route changes, all the authentication keys should be reestablished to enable en-route filtering as in IHA [120] due to the weakness of one-to-one forwarding approach. On the other hand, to generate an authentic report, each node needs to compute two MACs and execute one LSSS operation, which can be performed using efficient $\mathcal{O}(|p| \log^2 |p|)$ algorithms [30]. Furthermore, to forward a report, each node needs to verify one MAC and compute another MAC. Since the energy for computing a MAC is about the same as that for transmitting one byte, the computation cost involved by LEDS is highly efficient. In addition, to judge whether a node belongs to a particular *report-forward route*, only simple geometry computation is involved based on *geographic virtual grid*.

The communication overhead of our scheme arises from two sources as compared to the original report. First, every authentic report contains $T + 1$ MACs. Since the size of these MACs only impacts the capability of en-route filtering, in practice it can be made smaller as a tradeoff between performance and security. For example, if we use 6 bytes for all the MACs, and $T = 5$, the size of a MAC will be 1 byte. Therefore, the introduced additional message overhead is only 6 bytes in this example. Second, since the encrypted report is divided into a set of unique shares as node-to-sink endorsements, this would result in possible message size enlargement. For example, assuming $M$ is 36-byte (288-bit) long as in TinyOS [4] and $(t, T) = (4, 5)$, then each

share will be 9 bytes in length and there will be 5 shares in total according to the underlying LSSS. Hence, the size of additional message overhead is only one-fourth of the original message length, i.e., 9 bytes. Note that these additional message overheads provide much stronger security strength and resilience. Also note that the choice of $T$ should be based on both security and node density. A large $T$ makes it more difficult for the adversary to launch a false data injection attack, but it also requires more nodes to form a cell. Moreover, report delivery in LEDS follows a pre-defined route in a cell-by-cell manner. Hence, it is highly robust and resilient against node failures and other possible routing changes as compared to one-to-one forwarding paradigm in existing security designs [111, 120, 109]. The elimination of unnecessary routing overheads also help LEDS achieve communication efficient.

### 4.4.3 Energy savings

Existing security designs only aim to save the energy of intermediate nodes along the forwarding path to the sink through early detection and dropping of bogus data reports inserted by compromised nodes. However, compromised nodes may also intentionally drop legitimate reports and thus cause futile energy consumption, which implies extra energy waste. To address this problem, LEDS aims to reduce the energy waste resulting from both bogus data report insertion and legitimate report dropping. On the other hand, in doing so, the introduced message overhead and enroute filtering operations inevitably incur extra energy consumption in both communication and computation.

In the following, we employ a similar model to that of [111] to analyze the energy savings caused by the proposed LEDS. We denote by $E_{tr}$ the energy consumption for transmitting and receiving one bit, by $L_n$ the bit-length of an original report without using LEDS, by $L_a$ the bit-length of a report with LEDS, and by $h$ the average number of cells a report travels. We Furthermore assume that the ratio of legitimate data

traffic to bogus data traffic is $1 : \rho$ and an uniform traffic pattern (i.e., nodes from each cell generates the same amount of data reports.). Then the normalized energy waste in delivering all the traffic, denoted by $E_w$ without LEDS and $E'_w$ with LEDS will be:

$$\begin{aligned} E_w &= L_n \rho E_{tr} \frac{hl}{r} + L_n \frac{hl}{r}(1 - P_{\{0\}}) E_{tr} \frac{hl}{2r} \\ &= L_n E_{tr} \frac{hl}{r}(\rho + (1 - P_{\{0\}}) \frac{hl}{2r}) \end{aligned} \tag{10}$$

$$\begin{aligned} E'_w = (L_n + L_a)\rho E_{tr}(h \sum_{i=1}^{t-1} P_{\{i\}}(1 - P_{\{0\}})^{t-i} \\ + \frac{T}{2}(1 - \sum_{i=1}^{t-1} P_{\{i\}}(1 - P_{\{0\}})^{t-i})) \\ + (L_n + L_a)h(1 - \sum_{i=0}^{\lfloor \frac{n'(r-l)}{l} \rfloor} P_{\{i\}}) E_{tr} \frac{h}{2} \end{aligned} \tag{11}$$

It was reported in [30] that Rockwell Science Center's WINS sensor node consumes about $E_{tr} = 10\mu J$ for the hop-wise transmission and reception of one bit. Using this exemplary value, Fig. 4.11 plots the comparison of $E_w$ and $E'_w$ as a function of the bogus traffic ratio $\rho$ and the number of compromised nodes $x$, when $L_n = 288$ bits, $L_a = 112$ bits, $(t, T) = (4, 5)$, $(N, n') = (10,000, 10)$, $l = \frac{2r}{3}$ and $h = 30$.

We can see that $E_w$ increases dramatically with the increase of injected bogus data reports, while $E'_w$ always maintains a rather stable level because 1) most bogus reports can be detected and dropped during their early transmission stages with LEDS in place; 2) it is much harder to drop the legitimate reports with LEDS in place. Furthermore, LEDS shows remarkable energy savings in contrast to the case without using LEDS. For example, when $x = 300$ and $\rho = 5$, LEDS saves more than 85% energy, i.e., $385mJ$.

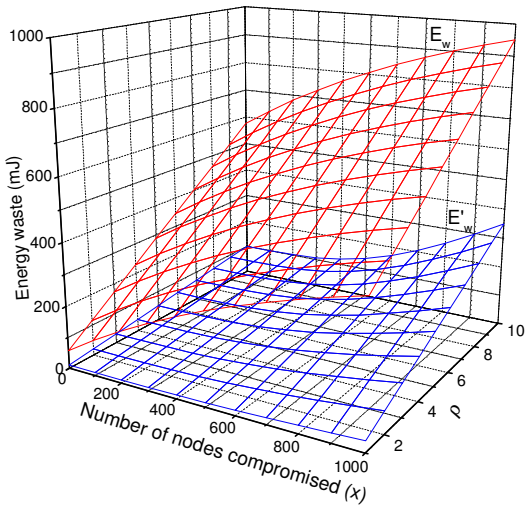


Figure 4.11: Energy waste due to node compromise under different bogus traffic ratio

## 4.5 Summary

In this chapter, through exploiting the static and location-aware nature of WSNs we came up with a location-aware end-to-end security framework to address the vulnerabilities in existing security designs. In our design, the secret keys are bound to geographic locations, and each node stores a few keys based on its own location. This location-aware property successfully limits the impact of compromised nodes only to their vicinity without affecting end-to-end data security. Furthermore, the proposed multifunctional key management framework assures both node-to-sink and node-to-node authentication along report forwarding routes. Moreover, our data delivery approach guarantees efficient en-route bogus data filtering, and is highly robust against DoS attacks. We evaluate our design through extensive analysis, which demonstrates its high resilience against an increasing number of compromised nodes and effectiveness in energy savings, that is, achieving 85% or more energy savings in contrast to the case without using our design when appropriate parameters are chosen.

# Chapter 5

# Event Boundary Detection Security

In this chapter, we study how to securely detect event boundary in WSNs under adversarial environments with enhanced fault-tolerance. In a trustworthy environment, each node reports its measurements honestly and a node with erroneous measurements will suppress/abort its own observation based on the information collected from other nodes in its neighborhood. However, this is not true in an adversarial environment where malicious compromised nodes exist. Compromised nodes can always lie about its measurements, claim to be a boundary node when it is not, or refuse to report itself as a boundary node when it is. Moreover, compromised nodes may also collude to fabricate non-existing event boundary to deceive the sink and cause erroneous actions taken. Sensor random measurement error further complicates the problem. Hence, to fully address these problems, the interferences from both node compromise and random measurement fault should be taken into consideration.

We propose a Secure Event Boundary Detection (SEBD) scheme, which allows secure detection of event boundaries in a localized manner, and is highly resilient against both node compromise and random measurement fault. In SEBD, with an

efficient key establishment protocol, each sensor node establishes a unique secret key shared only with the sink, and several pairwise secret keys each shared with one of its neighbors. Those keys are bound to a node's physical location so even if the node is compromised, the impact is effectively confined to that particular node and at its particular location only. In SEBD, each node senses its local environment independently. Once an event of interest is detected, senor nodes first exchange their measurements among neighbors and benign nodes suppress possible faulty measurements following a majority rule. To enhance fault tolerance and prevent fabrication, once a node is detected as a boundary node, a number of its neighbors will collaboratively endorse the corresponding boundary claim message. A neighbor node endorses a boundary claim message only if the contained information is consistent with its own knowledge. The sink accepts a claim only when it contains a required number of valid endorsements. A nonparametric statistical boundary detection model is also developed, which is seamlessly integrated with the proposed security mechanism. It facilitates localized boundary node determination, and helps to suppress random measurement fault and malicious false readings. It shows a much higher accuracy and better fault-tolerance and compromise-resilience as compared to previous schemes [19, 48, 23]. The performance and security strength of the proposed SEBD are examined by analysis and extensive simulation study.

## 5.1 Related Work

Several localized event boundary detection schemes have been proposed recently [21, 19, 48, 23, 71, 47]. Among them, Clouqueur, et. al. [21] sought algorithms to collaboratively detect the presence of a target in a region. Each sensor obtains the target energy (or local decision) from all other sensors in the region, drops extreme values if faulty sensors exist, computes the average, and then compares it with a

pre-determined threshold for final decision. Krishnamachari et. al. [48] proposed several localized threshold based decision schemes to detect both faulty sensors and event regions. The 0/1 decision predicates from the neighborhood are collected and the number of neighbors with the same predicates are calculated. This number is used for the final decision based on a majority vote. Another work that targets localized boundary detection in sensor networks was proposed by Chintalapudi et. al. in [19]. All of the three schemes in [19] take as inputs the 0/1 decision predicates from neighboring sensors. The statistical approach computes the number of 0's and 1's in the neighborhood and a boundary sensor is detected if its neighbors contain a "similar" number of 0's and 1's. Here the "similarity" is defined based on a threshold value that can be obtained based on a lookup table. Ding et. al. further proposed a similar approach [19] that takes as input not only binary 0/1 decision predicates but also real values that abstract sensor readings or sensor behaviors [23]. Note that all these schemes work in trustworthy environments.

Meanwhile, several schemes have been proposed to provide secure discrete event detection under adversarial environments [111, 120, 109, 83, 118]. In these schemes, every single event of interest is assumed to be detectable by at least $T$ nodes, where $T$ is a predefined threshold value and usually very small ($< 10$). The approach adopted in these schemes is to let every valid event detection report be collaboratively generated and independently endorsed by $T$ nodes that have detected the event simultaneously. Cryptographic techniques are then used to generate such endorsements to allow both en-route and sink verification, while keeping the event report as short as possible. However, this approach can not be applied to a large-scale event directly, since it is neither feasible nor necessary for all the nodes in the event region to report its detection back to the sink due to the stringent resource constraints in WSNs. So far, there is no published work on secure protocols that aim to correctly identify and communicate event boundaries, rather than the event itself, in the presence of both

node compromise and random node measurement faults.

## 5.2 SEBD: The Scheme

### 5.2.1 Problem Identification

In this part, we describe how the event boundary detection schemes proposed under trustworthy environments would fail in adversarial environments. In adversarial environments, sensor nodes could be compromised and controlled by the attacker [90]. These compromised nodes will lie about their measurements and result in severe security threat, which greatly jeopardizes boundary detection functionality of a WSN. Both faulty nodes and compromised nodes may inappropriately cause non-boundary nodes (including themselves) to be recognized as boundary nodes due to the nature of statistical method used by most of exiting schemes. However, the damage caused by the compromised nodes is much worse than that of faulty nodes. This is because a *faulty* node is still a benign node, and would suppress its own measurements after referring to other measurements in its neighborhood. However, a *compromised* node will always lie about its measurements, report itself as a boundary node when it is not, and suppress such claims when it is[1]. A collection of compromised nodes could prevent the event boundary from being correctly detected by presenting false measurement information. Moreover, compromised nodes may collude to fabricate non-existing events and event boundaries. They may claim such boundaries appearing at any location of the network as desired by the attacker, not necessarily at their own actual locations.

[1]When it does not lie, it does not need to be treated.

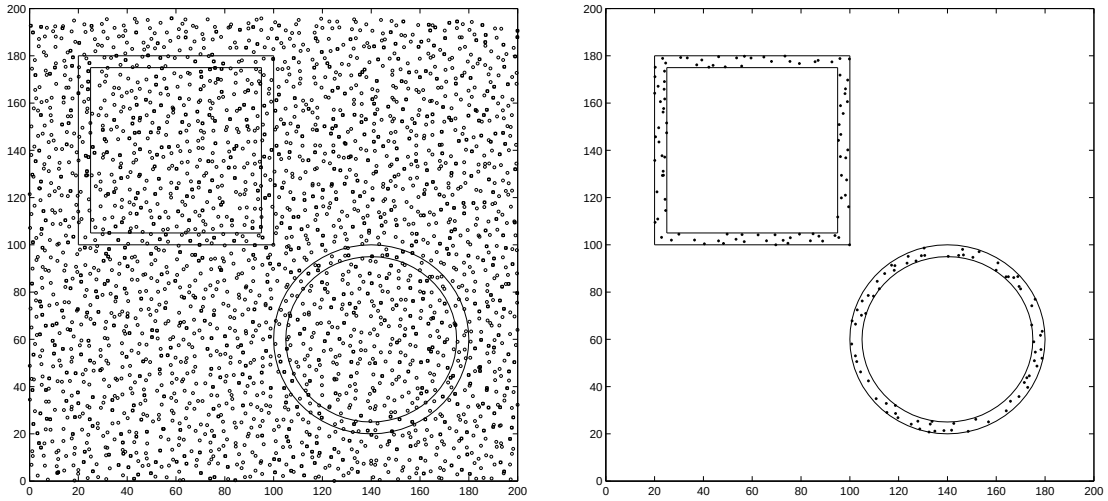


Figure 5.1: The left figure denotes a sensor field, where the event area is located inside the outer square and circle. In the figure, normal nodes are denoted as '$\circ$', compromised and faulty nodes are denoted as '$\Box$'. The right figure is an illustration of boundary $\mathbb{B}$ with $r = \frac{R}{2}$. By definition, the nodes denoted as '*' are the boundary nodes.

### 5.2.2 Assumptions, network model and design goal

We assume that sensor nodes are uniformly deployed in a two-dimensional territory, i.e., a *sensor field*, and they are dense enough to support fine-grained collaborative sensing. Topology control mechanisms for such purpose have been studied in [105]. We assume that sensor nodes are similar to the current generation of sensor nodes (e.g., the Berkeley MICA motes [33]) in their computation and communication capability and power resource and they are loosely synchronized. We assume that even if sensor nodes may execute certain sleeping strategy for energy conservation, they can still wake up periodically or be woken up by ceratin events to work collaboratively, according to certain wakeup mechanism [102]. The term *sensor field*, denoted as $\mathbb{S}$, is

referred to both the geographical region covered by the WSN, and the set of sensors within the region. A sensor $S_i$ and its location will be used interchangeably, that is, $S_i = (x_i, y_i)$. We assume that sensor nodes can make random measurement errors, and such nodes are called *faulty* nodes. (However, we do not address location information measurement errors, since our focus in this chapter is how to securely detect event boundary given correct location information). Furthermore, we assume that sensor nodes can be compromised and controlled by the adversary, whose purposes are to 1) prevent event boundary from being correctly detected; 2) collude to fabricate non-existing events and event boundaries.

We use a refined event boundary model as compared to the ones in [19, 23]. Consider a phenomenon (i.e., event $\mathcal{E}$) that spans some arbitrarily shaped sub-region of $\mathbb{S}$. Each sensor can, based on locally collected measurements, determine whether it belongs to the sub-region covered by the phenomenon or not. Ideally, a boundary node, say $S_i$, is such a node that every closed disc centered at $S_i$ contains both points in $\mathcal{E}$ and $\overline{\mathcal{E}}$ (that is, the boundary node should be right on the *real event boundary*, denoted as $\mathcal{B}_R$.), where $\mathcal{E}$ is the ground truth of the event covering sub-region in $\mathbb{S}$, and $\overline{\mathcal{E}}$ represents the remaining region, i.e., $\overline{\mathcal{E}} = \mathbb{S} - \mathcal{E}$. Hence, an event boundary, denoted as $\mathbb{B}$, when represented by sensor nodes, is simply a collection of such boundary nodes. However, due to the actual node density in practice, an event boundary found in this case constitutes only a very restrictive node set, which is far from enough to approximate/reveal $\mathcal{B}_R$ [19]. For this reason, the notion of *boundary width* is introduced with its value $0 < r < R$ in SEBD, where $R$ is the communication radius of sensor nodes. In SEBD, we define a sensor node, $S_i$, as a boundary node, $\mathbb{B} = \bigcup_i S_i,\ \forall\ i : |S_i \perp \mathcal{B}_R| \leq r, and\ S_i \in \mathcal{E}$, where $|S_i \perp \mathcal{B}_R|$ denotes the distance between $S_i$ and $\mathcal{B}_R$. The definition is illustrated in Fig.5.1.

Naturally, the design goal of SEBD is then to securely identify as many nodes as possible in $\mathbb{B}$ (bounded by the underlying distributed statistical model) under adver-

sarial environments. In other words, SEBD should have a strong security strength to prevent compromised nodes from successfully claiming themselves as boundary nodes when they are actually not. Furthermore, even if a compromised non-boundary node succeeds in claiming itself as a boundary node, it should not be able to claim the boundary at locations other than where it is. That is, the damage caused by compromised nodes should be limited to their vicinity only.

### 5.2.3 Overview of SEBD

The proposed SEBD is designed to be robust against node compromise and random fault. SEBD consists of two key components: the underlying location-aware key management framework, and the corresponding distributed statistic boundary detection model that is seamlessly built upon the former.

Key management framework in SEBD exploits the static and location-aware nature of WSNs. By leveraging robot-assisted secure bootstrapping technique, a secure location-aware key management is efficiently realized through embedding location information into the keys. In SEBD, each node possesses two different types of location-aware symmetric keys: 1) a *unique secret key* shared between the node and the sink that is used to provide node-to-sink authentication and data confidentiality; 2) a set of *neighbor pairwise key*s shared with each of the neighbor nodes respectively for node-to-node authentication and data confidentiality.

In our design, a sensor, after having detected an event of interest, proceeds to find out whether or not it is a boundary sensor. To do so, it first shares its sensing result within the neighborhood, and then makes use of the collective sensing result information for 1) suppressing its own potential sensing error; 2) judging whether or not a neighbor sensor/itself is a boundary node. If a sensor recognizes itself as boundary node, it further proceeds to request endorsements from its neighbors. Every neighbor sensor chooses to or not to endorse such requests independently. The judgement is

based on 1) the collective sensing results in the neighborhood; 2) the behavior of the sensor that seeks endorsements. In this way, the boundary sensors are detected statistically, and, at the same time, the illegal attempts of claiming a non-boundary node as a boundary node are effectively suppressed. More specifically, SEBD detects an event boundary in three essential stages: In 1) *local sensing and measurement adjusting* stage, each node exchanges its event measurement in the neighborhood. Then, every node adjusts its own measurement result according to *the majority rule*. Next, in 2) *distributed boundary detection* stage, each node independently determines whether or not it is a boundary node according to the updated measurements distribution in its neighborhood and the predefined statistic model. Once a node judges itself as a boundary node, it makes a boundary claim and seeks endorsements from its neighbors. Then, a neighbor node that receives a boundary claim will carry out a *consistency check* based on knowledge it learned directly from its neighbors and will follow the same statistical model to judge whether or not the sender is a boundary node or not. Upon getting a positive result, the receiving node endorses the boundary claim using the *unique secret key* it has. Lastly, in 3) *final message composition* stage, a boundary node constructs an overall synthesized endorsement from the individual ones it collected from its neighbors. The sink only accepts a boundary claim with a valid overall synthesized endorsement.

### 5.2.4 The Enhanced Statistical Boundary Detection Model

The observation behind our statistical boundary detection model is two-fold: 1) The original event measurements collected from neighbor nodes usually contain faulty measurements due to node measurement error and compromise; if faulty measurements can be corrected, boundary detection can certainly be more accurate and thus more tolerant against node fault and compromise. 2) Statistically, a boundary node can be determined by comparing the event measurements among its neighbor nodes

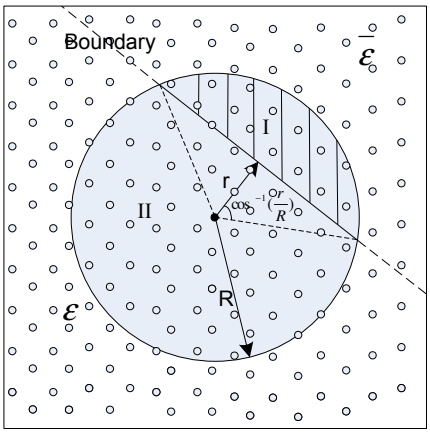


Figure 5.2: An illustration of areas $I$ and $II$

by assuming that the neighbor area of a sensor node is so "small" in comparison to the area covered by the entire event that the ground truth boundary can be approximated by a straight line in this area. In Particular, a boundary node (i.e., belonging to $\mathbb{B}$) will always have the difference between the numbers of '0' and '1' measurements in its neighborhood limited by a certain threshold and its value is determined by *boundary width* $r$ given a uniform node distribution.

In SEBD, the following specific rules are designed to reflect the above observation:

*Majority rule*: A node maintains its own measurement only when this result is the majority result within its neighborhood. Statistically, this rule could lead to error correction, as long as sensor fault probability is less than 50%. Note that the *majority rule* has been proved to be optimal based on an Bayesian fault recognition model in detecting node random measurement errors [47]. We refer the interested reader to [47] for the detailed proof.

*Consistency rule*: Since compromised nodes may lie about its measurements, we further require that, if a node does not follow the *majority rule*, its result will be ignored by its neighbors. This *consistency rule* is enforced in SEBD by consistency verification.

*Determination rule*: A node recognizes itself as a boundary node only when

$$1 - \frac{n_+ - n_-}{n_u} \geq \gamma, \quad (1)$$

where $n_+$ is the number of '1' measurements in a node $u$'s neighborhood, $n_-$ is the number of '0's, and $n_u = n_+ + n_-$ is the actual neighborhood size. Furthermore, $\gamma$ is a preset system parameter, called *normalized acceptance threshold*. In contrast to the previous schemes, the optimal choice of $\gamma$ in SEBD does not rely on the sensor fault probability. In fact, we set $\gamma = 1 - \frac{II(r)-I(r)}{\pi R^2}$, where the areas of $II$ and $I$ are illustrated in Fig.5.2. This selection of $\gamma$ is based on uniform node distribution, since the area size is proportional to the number of nodes located inside the area.

### 5.2.5 Scheme Details

#### 5.2.5.1 Network initialization phase

SEBD adopts a robot-assisted bootstrapping technique, which securely initializes each sensor node with the required scheme parameters and the secret symmetric keys. Specifically, we assume that a group of mobile robots are dispatched to sweep across the whole sensor field along pre-planned routes. Mobile robots have GPS capabilities as well as more powerful computation and communication capacities than ordinary nodes. The leading robot is also equipped with the network master secret keys $K_M^I$ and $K_M^{II}$, and $\gamma$. To localize a node, say $S_{i_u}$, mobile robots run the secure range-based localization protocol given in [103, 117] to first measure their respective absolute distance to node $S_{i_u}$ and then co-determine its location $(x_{i_u}, y_{i_u})$. Subsequently, the leading robot computes the *unique secret key* that is only shared between the sink and $S_{i_u}$ after bootstrapping:

$$K_{S_{i_u}} = H(K_M^I | S_{i_u}), \quad (2)$$

where '|' denotes concatenation operation. It also generates a complete list of neighbor nodes, denoted as $\mathbb{N}_{S_{i_u}}$ for a node $S_{i_u}$[2], and computes a set of *neighbor pairwise key*s: suppose $S_{i_v} = (x_{i_v}, y_{i_v})$ is a neighbor of $S_{i_u}$, then the *neighbor pairwise key* between the two is

$$K_{S_{i_u},S_{i_v}} = H(K_M^{II}|S_{i_u}|S_{i_v}), \quad (3)$$

where $S_{i_u} < S_{i_v}$ in their binary representations. The leading robot repeats the calculation for all nodes in $\mathbb{N}_{S_{i_u}}$ and sends all the generated keys plus $\gamma$ to $S_{i_u}$. Note that the authentication between the sensor nodes and the leading robot can be easily achieved using the technique introduced in [115]. We omit it here for the space limit. Following this process, all the nodes can be furnished with their respective location and the required keys. After that, mobile robots leave the sensor field and the leading robot should securely erase all the keys from its memory. The assumption underlying this approach is that adversaries do not launch active and explicit pinpoint attacks on mobile robots at this stage which usually does not last too long. That is, the robots are not likely subject to compromise. However, the adversaries may still perform relatively passive attacks such as message eavesdropping or strategic channel inference to disturb the localization process [103]. This assumption is reasonable in that mobile robots are much fewer than ordinary sensor nodes and hence we can spend more on them by enclosing them in high-quality tamper-proof hardware and putting them under super monitoring.

#### 5.2.5.2 Boundary detection phase

For each sensor node $S_{i_u}$, the following data structures are defined: $list_{S_{i_u}}$ is a table used to store the measurements from nodes in $\mathbb{N}_{S_{i_u}}$. $n_u$ is the actual number of the one-hop neighbors of node $u$, and $n'$ is the expected node degree. $tt_{S_{i_u}}$ is used to store current operation time. $n_+$ is the number of '1' measurements in $\mathbb{N}_{S_{i_u}}$, while

[2] $\#\{\mathbb{N}_{S_{i_u}}\} = n'$ on average, where $n'$ is the expected number of sensor nodes in $\pi R^2$ area.

$n_-$ is the number of '0's. $\mathbb{EN}$ is a list used to store the *id*s of nodes having endorsed on $S_{i_u}$'s *boundary claim* message if any. Lastly, $Mac$ is for final overall endorsement constructed. Note that all of them are initialized to zero or $\emptyset$.

Upon receiving a boundary extraction request on event type $e_{id}$, from the sink or a pre-defined periodical time-out, a node $S_{i_u}$ performs the measurement, notifies its neighbors of the result, and adjusts the result according to others' notifications if necessary:

---

– $S_{i_u}$ prepares $\mathsf{MR}_{S_{i_u}} := \{e_{id}, S_{i_u}, m_0, 0\}$, and broadcasts $\mathsf{MR}_{S_{i_u}}$ to its neighborhood.

– $S_{i_u}$ collects $\mathsf{MR}_{S_{i_j}}$ for all $S_{i_j} \in \mathbb{N}_{S_{i_u}}$ and updates $list_{S_{i_u}}$, i.e., upon receiving message $\mathsf{MR}_{S_{i_j}} = \{e_{id}, S_{i_j}, m_0, 0\}$, $S_{i_u}$ updates $list_{S_{i_u}}$ by including an entry $(S_{i_j}, m_0)$.

– Once having received $\mathsf{MR}_{S_{i_j}}$ from all $S_{i_j} \in \mathbb{N}_{S_{i_u}}$, $S_{i_u}$ calculates the number of '1' measurements $n_+$ and the number of '0' measurements $n_-$ from $list_{S_{i_u}}$; $S_{i_u}$ adjusts its own measurement $m$ as follows: if $m_0 == 1$ and $n_+ < \lfloor \frac{n_u - 1}{2} \rfloor$, $S_{i_u}$ reverses its measurement to $m_1 := 0$; if $m_0 == 0$ and $n_- < \lfloor \frac{n_u - 1}{2} \rfloor$, $S_{i_u}$ reverses its measurement to $m_1 := 1$, otherwise, the original measurement is retained, $m_1 := m_0$.

– $S_{i_u}$ then broadcasts the updated $\mathsf{MR}_{S_{i_u}} := \{e_{id}, S_{i_u}, m_1, 1\}$ together with $list_{S_{i_u}}$ in its neighborhood.

---

Here, a *measurement report* message $\mathsf{MR}_{S_{i_u}}$ consists of four fields: i) an event *id*, $e_{id}$, ii) a node *id*, iii) $m$, a logic value '0/1', representing whether event $e_{id}$ is detected or not, and iv) a '0/1' valued indicator, indicating the message is either an original measurement report or an updated report after local adjustment for random error correction. Next, if a node $S_{i_u}$'s updated measurement is '1', it proceeds to check whether or not it is a boundary node based on the information it received. Note

that, if the measurement of $S_{i_u}$ is now a '0', then no further operation is needed. The following operations are sequentially executed before reaching the decision:

---

– For every node $S_{i_j}$ in $\mathbb{N}_{S_{i_u}}$, $S_{i_u}$ does the following consistency check and updates $list_{S_{i_u}}$ accordingly: 1) it verifies that the measurements in the common entries in $list_{S_{i_j}}$ and $list_{S_{i_u}}$ are consistent; and 2) it verifies that node $S_{i_j}$'s self-adjusted value, i.e., $m_1$ in $\mathsf{MR}_{S_{i_j}} = \{e_{id}, S_{i_u}, m_1, 1\}$, conforms to the majority measurements in $list_{S_{i_j}}$.

– $S_{i_u}$ then calculates $n_+$ and $n_-$ for the updated $list_{S_{i_u}}$ and further calculates $1 - \frac{|n_+ - n_-|}{n_u}$. If $1 - \frac{|n_+ - n_-|}{n_u} \geq \gamma$, $S_{i_u}$ considers itself a boundary node and prepares a *boundary claim* message, $\mathsf{BC}_{S_{i_u}} := \{e_{id}, S_{i_u}, 1, tt_{S_{i_u}}\}$, where $tt_{S_{i_u}}$ is a time stamp. $S_{i_u}$ then broadcasts $\{list_{S_{i_u}}, \mathsf{BC}_{S_{i_u}}\}$ to the neighbors to seek their endorsements. $list_{S_{i_u}}$ is attached for consistency verification.

---

Now assume that a neighbor node, say $S_{i_j}$, receives $\{list_{S_{i_u}}, \mathsf{BC}_{S_{i_u}}\}$. $S_{i_j}$ proceeds as follows to endorse the BC [3].

Here, the endorsement to $\mathsf{BC}_{S_{i_u}}$ from $S_{i_j}$, i.e., $\overline{MAC}_{S_{i_j}}$, is a unique MAC generated over message $\{e_{id}|S_{i_u}|1|tt_{S_{i_u}}\}$ using the *unique secret key* $K_{S_{i_j}}$ shared between $S_{i_j}$ and the sink. Hence, no node could forge such a MAC on behalf of others. SEBD also ensures that only the claimed sender can get the endorsement from the receiver/endorser: $\overline{MAC}_{S_{i_j}}$ is sent after encryption using the *neighbor pairwise key* shared between the sender and receiver. Meanwhile, $\overline{MAC}_{S_{i_j},S_{i_u}}$ is computed over $\mathsf{ER}_{S_{i_j}}$ using the same *neighbor pairwise key* shared between the sender and receiver, which authenticates the message sender to the receiver. The intended receiver could therefore be assured that the endorsement is indeed from the claimed endorser. Note

[3]Below $MAC(M, K)$ denotes the message authentication code generated over message $M$ using symmetric key $K$ and $E(M, K)$ denotes an encryption operation over message $M$ using $K$.

– Consistency check: 1) it verifies if the time stamp is fresh, i.e., within the allowed delay interval; 2) it checks the measurement consistency of the common entries contained in both $list_{S_{i_u}}$ and its own $list_{S_{i_j}}$; 3) it carries out the same procedure to determine if node $S_{i_u}$ is a boundary node and verifies that the result conforms to $S_{i_u}$'s claim.
– Upon successful checking, $S_{i_j}$ endorses $\mathsf{BC}_{S_{i_u}} := \{e_{id}, S_{i_u}, 1, tt_{S_{i_u}}\}$ by calculating $\overline{MAC}_{S_{i_j}} := MAC(\mathsf{BC}_{S_{i_u}}, K_{S_{i_j}})$, generating $\mathsf{ER}_{S_{i_j}} := \{S_{i_j}, S_{i_u}, e_{id}, tt_{S_{i_j}}, E(\overline{MAC}_{S_{i_j}}, K_{S_{i_j},S_{i_u}})\}$. It further calculates $\overline{MAC}_{S_{i_j},S_{i_u}} := MAC(\mathsf{ER}_{S_{i_j}}, K_{S_{i_j},S_{i_u}})$, and sends $\{\mathsf{ER}_{S_{i_j}}, \overline{MAC}_{S_{i_j},S_{i_u}}\}$ back to $S_{i_u}$.

that $\overline{MAC}_{S_{i_u},S_{i_j}} \neq \overline{MAC}_{S_{i_j},S_{i_u}}$.

Lastly, node $S_{i_u}$ collects all the $\mathsf{ER}$s replied by its neighbors after sending $\mathsf{BC}_{S_{i_u}}$. It then constructs a final synthesized boundary report with appropriate endorsements from its neighbors, and sends it to the sink.

A $\mathsf{BR}_{S_{i_u}}$ is accepted by the sink if and only if i) $\overline{MAC}_{S_{i_u},sink}$ is authentic; and ii) $t \geq \lfloor \frac{n'-1}{2} \rfloor$, where $t$ is the number of members in $\mathbb{EN}$; and iii) all nodes in $\mathbb{EN}$ are indeed the neighbors of $S_{i_u}$[4]; and iv) $Mac$ is authentic, which, in other words, means that all the $t$ individual $\overline{MAC}_{S_{i_j}}$s are authentic; Note that, in this chapter, we focus on the compromise-tolerant event boundary detection mechanism, thus we simply assume all $\mathsf{BR}$s are directly forwarded to the sink [5].

[4]This is achieved by extracting node's location information from its $id$, and ensuring that the distance between two nodes is no farther than $R$.

[5]The sink is always assumed trustworthy and well protected.

– Upon receiving $\{\mathsf{ER}_{S_{i_j}}, \overline{MAC}_{S_{i_j},S_{i_u}}\}$ from its neighbor $S_{i_j}$, $S_{i_u}$ first checks the time stamp included in $\mathsf{ER}$ to make sure the freshness of the message. It further verifies $\overline{MAC}_{S_{i_j},S_{i_u}}$.

– Upon successful verification, $S_{i_u}$ then includes $S_{i_j}$ into $\mathbb{EN}$, i.e., $\mathbb{EN} := \mathbb{EN} \cup S_{i_j}$, and recovers the unique MAC generated by $S_{i_j}$, which is further combined to the synthesized MAC, i.e., $Mac := Mac \oplus D(E(\overline{MAC}_{S_{i_j}}, K_{S_{i_j},S_{i_u}}), K_{S_{i_u},S_{i_j}})$, where $D(M, K)$ denotes a decryption operation over message $M$ using symmetric key $K$ and '$\oplus$' denotes exclusive or operation.

– Upon $\#\{\mathbb{EN}\} \geq \lfloor \frac{n'-1}{2} \rfloor$, $S_{i_u}$ forms a *boundary report* message $\mathsf{BR}_{S_{i_u}} := \{\mathsf{BC}_{S_{i_u}}, Mac, \mathbb{EN}\}$, and forwards $\{\mathsf{BR}_{S_{i_u}}, \overline{MAC}_{S_{i_u},sink}\}$ to the sink, where $\overline{MAC}_{S_{i_u},sink} := MAC(\mathsf{BR}_{S_{i_u}}, K_{S_{i_u}})$.

## 5.3 Security Analysis

### 5.3.1 Qualitative Analysis

The proposed SEBD presents several nice security features as below, which greatly mitigate the security threat caused by compromised nodes. Firstly, in SEBD, to successfully claim itself as a boundary node, a node has to collect enough endorsements from its neighbors so that its claim can be accepted by the sink. Meanwhile, each sensor node independently makes endorsement decisions by itself based on the information it collected directly from its neighborhood. Therefore, in contrast to existing boundary detection schemes, before compromised nodes are able to make such claims, a required number of endorsements have to be collected. Secondly, even if a compromised node has collected enough endorsements, it can only claim the boundary at its own location. That is, the damage caused by compromised nodes is limited to their vicinity only. This is because the location information has been embedded into the

*unique secret key* shared between each node and the sink, and any claim other than a node's actual location will be rejected by the sink due to lack of the corresponding *unique secret key*.

SEBD also withstands the following attacks:

Cheating attack: Under this attack, a compromised node lies about some of its neighbors' measurements in a *boundary claim* message in order to deceive its neighbors and obtain the endorsements. However, this attack will not likely succeed in SEBD. Although a broadcast BC in plaintext cannot be verified through cryptographic means like MAC, it is indeed verified through consistency check, which relies on the local and direct knowledge each node learned from its neighborhood. This is so designed because, in order to provide a cryptographically authentic BC, the sender has to attach up to $n_u$ different MACs from its neighbors. This, however, will unnecessarily waste large amounts of the precious energy and bandwidth, and greatly decrease the protocol efficiency.

Consistency check, on the other hand, is more efficient, and effectively prevents cheating attack. This is because every node in SEBD is required to maintain a table storing the measurement information of its neighbors, which serves as the information source for boundary nodes self-determination. This table, at the same time, is also conveniently used for consistency check: each node $u$ independently verifies the authenticity of a BC using its own local knowledge stored in $list_{S_{i_u}}$. Although each node may not be able to verify the whole information contained in a received BC, a lie about one node's measurement will always be detected by some corresponding neighbor nodes, since the messages are always broadcast. It is very hard for a compromised node to lie about a number of nodes' measurements simultaneously in order to be falsely recognized as a boundary node (or reverse) and still get enough endorsements without being detected. If a complete consistency check is necessary, we may allow each node to increase its transmission range to $2R$ only when broadcasting its

event measurements. Then, a $list_{S_{i_u}}$ can contain the whole measurement information of all two-hop neighbors. And, in this case, any individual node can detect a lie in $list_{S_{i_j}}$ from its immediate neighbors. Thus, the cheating attack can be completely prevented.

Impersonating and colluding attack: Under this attack, a compromised node may try to impersonate another node at a different location. And compromised nodes at different locations may collude and endorse each others' *boundary claim* message. However, this attack is not possible in SEBD because the sink only accepts those endorsements obtained from neighbor nodes. Since the location information is embedded into the *unique secret key* shared between a node and the sink, and the communication of any two neighbor nodes is protected by the corresponding *neighborhood pairwise key*, there's no way for a node to impersonate another node or generate a valid endorsement for a colluding node which is not in its neighborhood.

Replay attack: Under this attack, a compromised node may replay old messages in response to a new boundary query. This attack is prevented in SEBD through embedding time information in the BR messages. Any boundary report message that is out of the pre-specified delay tolerance will be automatically rejected.

Node Relocation and Replication Attack: Under this type of attacks, the adversary may 1) compromise and relocate some sensor nodes to other positions in the sensor field; 2) replicate compromised sensors and place them to the positions of the adversary's interest. The goal of the adversary is to have the compromised nodes outnumber the normal nodes at certain areas, and hence try to cheat the sink with bogus boundary claims. However, this type of attacks are also not possible in SEBD because the location information is always embedded into all the keys that are used to endorse the boundary claims. Similar to the analysis for impersonating and colluding attacks, we can easily find that the relocated sensors can never obtain the legitimate *unique secret key* and the *neighborhood pairwise key*s.

### 5.3.2 Quantitative Analysis

In SEBD, compromised and faulty nodes could still possibly result in false recognition of non-boundary nodes (including compromised and faulty nodes themselves) as boundary nodes. This is because the sink accepts a boundary claim as long as such a message is endorsed by $\lfloor \frac{n'-1}{2} \rfloor$ nodes, and $n'$ is the expected neighborhood size. That is, a compromised/faulty node could pass its boundary claim, as long as there are no fewer than $\lfloor \frac{n'-1}{2} \rfloor$ faulty/compromised nodes in its neighborhood. However, the attack is actually very hard to succeed as the following analysis shows.

Assume a WSN consisting of $N$ nodes, node compromise and fault probability $p_c$ and $p_f$ respectively, the expected number of compromised nodes is then $N(p_c + p_f)$. Therefore, the probability that exactly $i$ nodes are compromised/faulty in a neighborhood is $P_{\{i\}} = \frac{\binom{n'}{i}\binom{N-n'}{N(p_c+p_f)-i}}{\binom{N}{N(p_c+p_f)}}$, assuming compromised and faulty nodes are uniformly distributed in the sensor field. Hence, the expected probability that a non-boundary node is falsely detected as a boundary node, is

$$P_{\{\geq \lfloor \frac{n'-1}{2} \rfloor\}} = \sum_{i=\lfloor \frac{n'-1}{2} \rfloor}^{n'} P_{\{i\}} \quad (4)$$

Obviously, $P_{\{\geq \lfloor \frac{n'-1}{2} \rfloor\}}$ also represents the fraction of nodes that are falsely detected as boundary nodes when there are no events going on in a WSN.

Fig. 5.3 shows how the value of $P_{\{\geq \lfloor \frac{n'-1}{2} \rfloor\}}$ changes as node compromise probability changes under different neighborhood size $n'$. It is clearly shown that as long as $n'$ is reasonably large ($\geq 15$), the value of $P_{\{\geq \lfloor \frac{n'-1}{2} \rfloor\}}$ keeps below 0.8%, given $p_c \leq 15\%$. Even when $p_c$ reaches as high as 20%, we still have $P_{\{\geq \lfloor \frac{n'-1}{2} \rfloor\}} < 3\%$ with $n' = 30$. Furthermore, the number of nodes in a single area can be modelled as poisson random variable as we assume a uniform node distribution in WSN [19]. This implies that, given the expected neighborhood size (i.e., node degree) $n' \leq 50$, the probability that the number of nodes in a neighborhood is less than $n' - 4$ is very small. That is, given an expected $n'$, the size of a neighborhood in WSN has an overwhelming probability

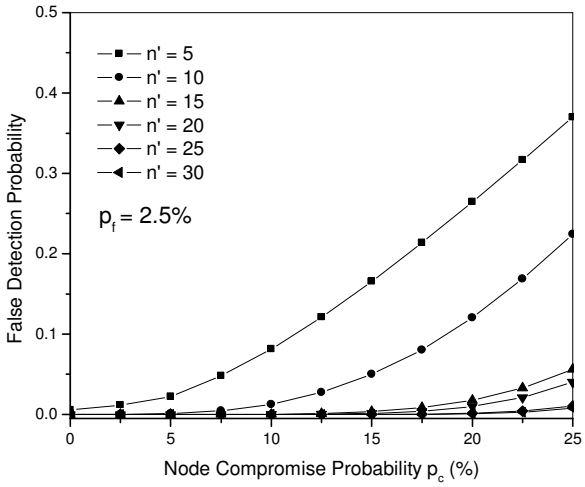


Figure 5.3: False detection probability vs. node compromise probability

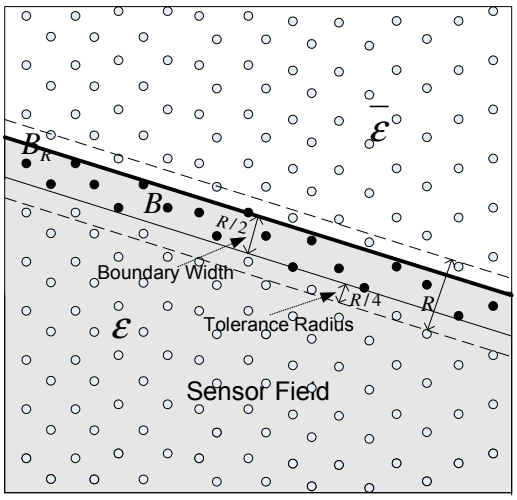


Figure 5.4: An illustration of boundary model with $r = R/4$

to be larger than $n' - 4$. Therefore, in SEBD, it is very hard for compromised nodes to fabricate non-existing events and event boundaries.

## 5.4 Simulation Studies

### 5.4.1 Metrics for Performance and Security Strength

The following three metrics are used to evaluate the performance of SEBD. Let $\mathbb{B}'$ be the set of boundary nodes detected by SEBD. Let $\mathbb{B}$ be the set of actual boundary nodes as defined in Section III.A.

*Hit Rate* $e_f$: $e_f$ represents the fraction of sensors in $\mathbb{B}$ that are detected by SEBD, with respect to the size of $\mathbb{B}$:

$$e_f = \frac{\#\{\mathbb{B} \cap \mathbb{B}'\}}{\#\{\mathbb{B}\}} \quad (5)$$

On top of *boundary width* $r$, we futher introduce the notion of *tolerance radius* to characterize the distribution of the boundary nodes detected by SEBD. In particular, any falsely detected boundary node that has its distance to real boundary $\mathbb{B}$ no more than $\frac{R-r}{2}$ is said to be within *tolerance radius*. We are more interested in the fraction of falsely detected boundary nodes that are far from the event boundary $\mathbb{B}$. An illustration of this definition is shown in Fig. 5.4. Based on the definition of *tolerance radius*, *False Detection Rate* is defined.

*False Detection Rate* $e_d$: $e_d$ represents the ratio of falsely detected sensors with respect to the size of $\mathbb{B}$. Here, only those falsely detected sensors whose distance to the boundary are at least $\frac{R-r}{2}$ are counted. Let $\mathbb{A}$ denote the set of falsely detected nodes whose distance to the boundary is larger than $\frac{R-r}{2}$.

$$e_d = \frac{\#\{\mathbb{A}\}}{\#\{\mathbb{B}\}} \quad (6)$$

Furthermore, we denote the mean distance of the nodes in $\mathbb{B}'$ to $\mathbb{B}_R$ as $d_{\mathbb{B}'}$.

*Normalized Mean Distance* $e_w$: $e_w$ represents the normalized mean distance of $\mathbb{B}'$ regarding boundary width:

$$e_w = \frac{d_{\mathbb{B}'}}{r} \quad (7)$$

### 5.4.2 Simulation setup

In all simulations, sensor nodes are located in a 200m by 200m area, their locations drawn from a uniform distribution over the area. The radio range of all the sensors is 10m and assumed omni-directional. In all simulations, we arbitrarily choose the *boundary width* $r = R/2$. And $\gamma = 1 - \frac{II(r=\frac{R}{2}) - I(r=\frac{R}{2})}{\pi R^2} = \frac{2\pi - 3\sqrt{3}}{3\pi} \approx 0.12$; the areas of $II$ and $I$ are illustrated in Fig. 5.2. Note that our $\gamma$ value is independent of node compromise probability. We report the results for event regions with ellipses or straight lines as the boundaries. And our simulation produces similar results for event regions with other boundary shapes. The simulation runs under different densities and node compromise probabilities. For each given density and node compromise probability, the results for the three security and performance metrics are averaged over 50 simulation runs.

### 5.4.3 Simulation results

In this subsection, the simulation results are reported in details. The three performance and security strength evaluation metrics defined in section V.A with regard to network density and node compromise probability are reported in Fig. 5.5 (a), (b) and (c), respectively. In contrast to the previous schemes, we did not change any setting on parameters as node compromise probability increases from 0% to 25%. That is, our simulation results do not rely on the pre-knowledge of node compromise/fault probability, which, in fact, may not be available as a priori in many practical applications.

Firstly, we observe that the proposed SEBD performs very well, when node compromise probability equals to zero. In this case, the hit rate $e_f$ is always as high as 93%, no matter what the network density $n'$ is. Note that we still have $p_f = 2.5\%$ in this case. In fact, when both $p_c$ and $p_f$ equal to zero, $e_f$ is always around 95% in SEBD. In the previous schemes [19, 23], $e_f$ is generally no more than 85%, even if all

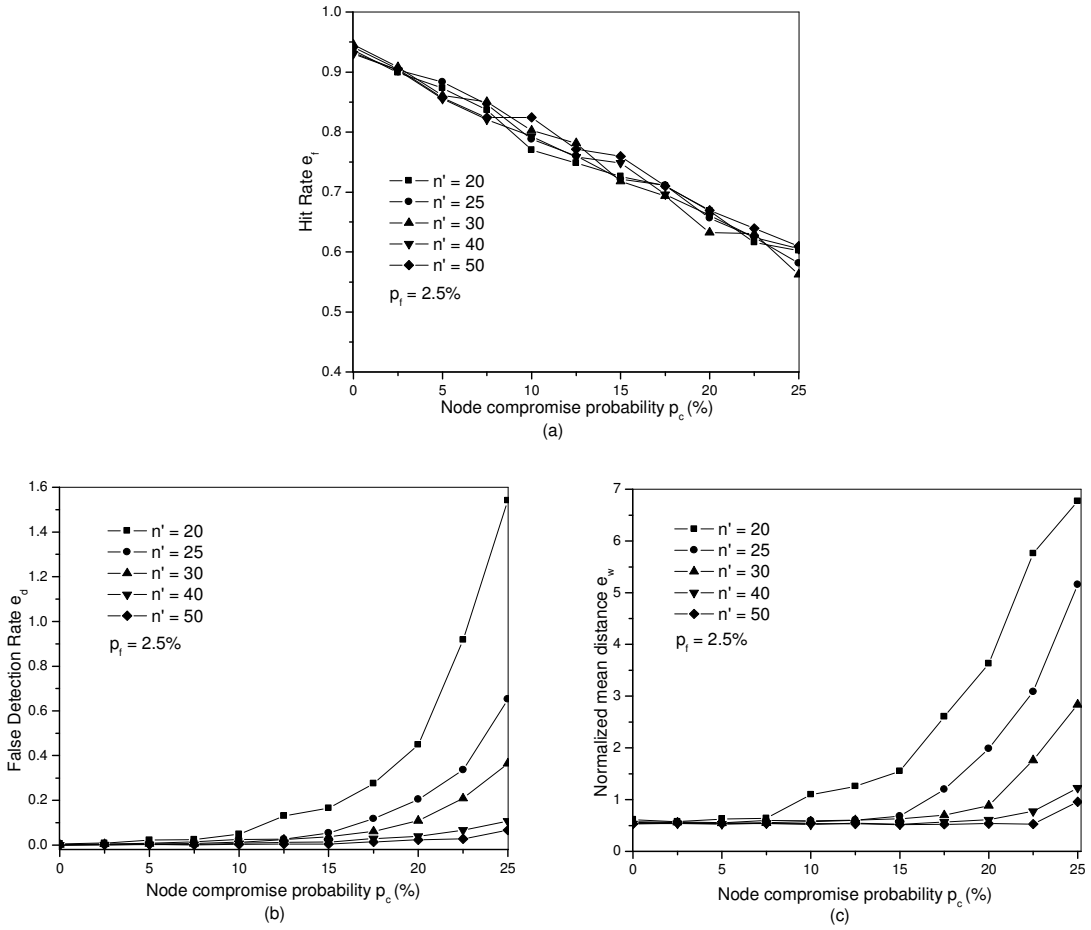


Figure 5.5: Simulation results with respect to three evaluation metrics

the compromised nodes can be assumed as random faulty ones. Hence, SEBD has the highest hit rate in the ideal situation as compared to the previous schemes.

Secondly, Fig. 5.5 (a) shows that 1) the hit rate $e_f$ in SEBD does not rely on network density; this is because we intentionally used normalized threshold value in boundary node detection process. 2) SEBD is very good at detecting boundary nodes: $e_f$ remains to be larger than 55%, when $p_c$ reaches as high as 25% plus 2.5% node fault probability. This result significantly outperforms any of the previous schemes [19, 23, 48].

Thirdly, SEBD presents a high security strength as shown in Fig. 5.5 (b). When

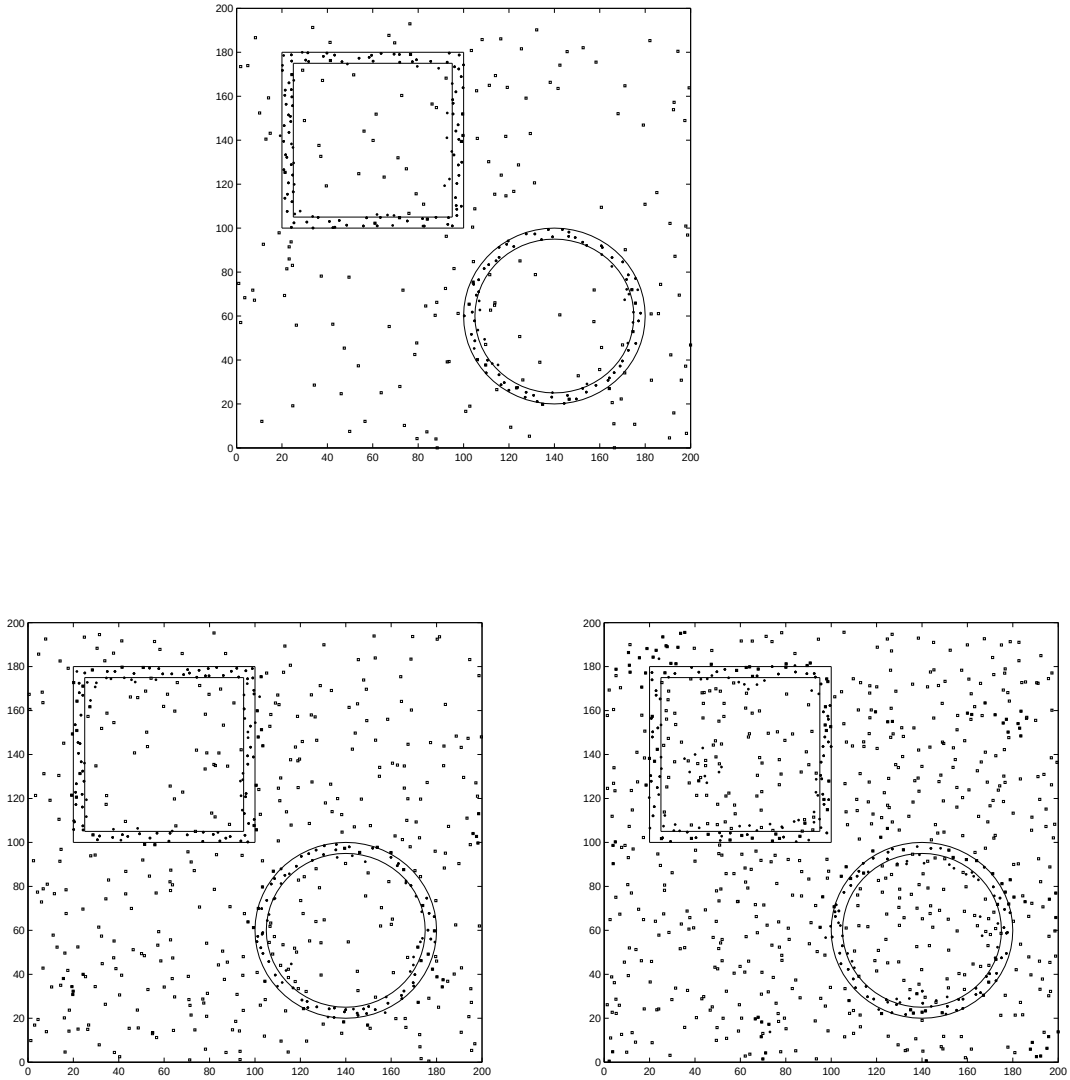

Figure 5.6: Simulation results: top): $e_f = 85\%$ with $p_c = 5\%$; left) $e_f = 79\%$ with $p_c = 12.5\%$; right) $e_f = 60\%$ with $p_c = 25\%$. And $p_f = 2.5\%$ in all three cases. '*' denotes the detected boundary nodes, and '□' denotes compromised and faulty nodes.

$n'$ is as low as 20, the false detection rate $e_d$ is still less than 5%, given $p_c = 10\%$. And for the same $p_c$, $e_d$ can be kept as low as 30% when $n' = 50$. Furthermore, given $n' = 50$, $e_d$ increases very slowly as $p_c$ increases; $e_d$ equals to only 67%, when $p_c$ reaches to 25% plus 2.5% node fault probability.

Fourthly, Fig. 5.5 (c) shows that the detected boundary nodes by SEBD are very close to the defined boundary $\mathbb{B}$. It is shown that as long as $n' \geq 25$, the normalized mean distances of the detected boundary nodes are always kept to be around the ideal value 0.5, given $p_c \leq 15\%$.

In summary, the simulation results shown in Fig. 5.5 indicate that 1) SEBD performs well until $p_c$ is up to 10%, even when $n'$ is as low as 20; 2) SEBD keeps presenting a very good performance and security strength even when $p_c$ goes up as high as 20%, given a reasonable high $n'$; 3) SEBD significantly outperforms the previous schemes in all the three metrics [19, 23].

Fig. 5.6 gives several visualized results to illustrate the performance of SEBD. The top figure gives the performance of SEBD at low node compromise probability. Clearly, when $p_c = 5\%$ and $p_f = 2.5\%$, SEBD has a very high hit rate: $e_f = 85\%$. The left figure gives the performance of SEBD at medium node compromise probability: $e_f = 79\%$, when $p_c = 12.5\%$ and $p_f = 2.5\%$. Obviously, the detected event boundaries in both top and left figures are very good approximations of the real event boundaries as defined in Fig. 5.1. In the right figure, we can find that as node compromise probability continues to be higher, the detected boundary presents a larger false detection rate as compared to the previous ones. But still, we have $e_f = 60\%$, given $p_c$ as high as 25% and $p_f = 2.5\%$.

## 5.5 Efficiency Evaluation of SEBD

**Communication Overhead**: One can easily see that the performance of SEBD improves as we require each node to collect more event measurements from more sensor nodes in its neighborhood. This is because each node can get more samples from both the interior and exterior of the event, and makes more accurate estimate in the presence of node compromise and fault. However, collecting more measurements from more nodes other than the immediate neighbors incurs much higher communication overhead. As mentioned in [19], communication overhead increases roughly quadratically as the neighbor range increases. This will result in a much higher energy consumption. In SEBD, we assume that the underlying network is well connected, that is, neighborhood size is reasonably large to support fine grained collaborative event detection. Hence, a good energy-accuracy tradeoff is achieved by letting each node collect the measurements from their immediate neighbors only. As we have shown in Fig. 5.5 (a), $e_f$ is larger than 80% with $n'$ as low as 20, given $p_c$ up to 10%.

**Computation Overhead**: SEBD uses very simple arithmetic computations to obtain the measurement statistics. At the same time, SEBD also involves some security related computations: endorsement operation, message authentication operation, and overall endorsement synthesization operation. SEBD exploits highly efficient security primitives to construct these operations: the first two are both realized through highly efficient MAC algorithm, while the last requires "exclusive or" operation only. More specifically, to accomplish a boundary node detection and authentication process, there are up to $2n'$ MAC operations required in total. Hence, the computation costs incurred by the security related operations in SEBD is light-weight.

## 5.6 Summary

In this chapter, we have studied a special instance of fault-tolerant collaborative in-network processing tasks in WSNs, i.e., distributed event boundary detection. We first introduced the problem of securing event boundary detection in WSNs for the applications related to large-scale phenomena monitoring, and showed how existing boundary detection schemes fail in adversarial environments. Then, we presented the SEBD scheme, which withstands many different types of attacks. To the best knowledge of the authors, SEBD is the first protocol of its kind to secure event boundary detection in WSNs. Along with SEBD, we also proposed an enhanced nonparametric statistic model for localized event boundary detection, which allows faulty measurement correction and thus achieves higher performance. The security strength and performance of SEBD are justified by our extensive analysis and simulations.

# Chapter 6

# Random Key Pre-distribution

In this chapter, we focus on the random key pre-distribution scheme without network pre-deployment knowledge. The drawback of the above mentioned random key pre-distribution schemes [28, 17] is that they are not suitable for large scale sensor networks as they require each node to load a large number of keys. For instance, implementation of random key distribution schemes in [28, 17] results in a storage overhead of at least 200 keys at each sensor node for a WSN of size 10,000, which is almost half of the available memory (assume 64-bit keys and less than 4KB of data memory [75]). The problem becomes even worse when the network size is larger. This fact makes the previously proposed random key distribution schemes less practical for large-scale WSNs.

We propose a highly efficient random key pre-distribution scheme in this chapter, which combines the random key pre-distribution technique and the hash chain technique. The novelty of our scheme lies in that, instead of requiring each sensor node to store all the chosen keys, the majority of the keys a node possesses are represented and stored in the form of a small number of key-generation keys by carefully designing the key pool, and therefore, the storage overhead is significantly reduced while the same security strength holds. Compared with the existing schemes, the proposed

scheme is more scalable and more secure in the sense that 1) Under the given resilience requirement against node capture, the proposed scheme requires a much smaller key ring size than the previous schemes; 2) Under the given maximum allowed key ring size, the proposed scheme has a much better resilience property against node capture than the previous schemes. The performance of the proposed scheme is justified by our thorough analysis and simulation.

The rest of the paper is organized as follows. We describe the background and related work in Section 6.1. Then we define the terms and notation and describe our new scheme in Section 6.2. Next we discuss the performance and security strength of the proposed scheme in Sections 6.3 and 6.4. Finally, the conclusion is drawn in Section 6.5.

## 6.1 Background on Key Management in WSN

In a WSN without pre-deployment knowledge, sensor nodes can be viewed as random points which are uniformly distributed (i.e., with equal probability). Thus, the sufficiency problem of the secure links resided in a WSN can be reduced to the connectivity problem of the generalized random graph, which, hence, can be mathematically treated using the well known connectivity theory for random graph by Erdós and Rényi [98]. The connectivity of a *key graph* $G(V, E)$ is then given as: for monotone properties, there exists a value of $p$ such that the property moves from "nonexistent" to "certainly true" in a very large random graph. The function defining $p$ is called the threshold function of a property. If $p = \frac{ln(n)}{n} + \frac{c}{n}$, with $c$ being any real constant then

$$P_c = \lim_{n \to \infty} Pr([G(n,p)\ connected]) = e^{-e^{-c}} \tag{6.1}$$

where $P_c$ denotes the desired probability that the key graph is connected. In addition, $n$ denotes the network size and $d$ denotes the node degree (i.e., the average

number of edges connected to each node) necessary to assure that the key graph is connected with probability $P_c$; $p$ is the probability that an edge between any two nodes exists in $G(V, E)$:

$$p = \frac{d}{n} \tag{6.2}$$

Due to the inherent communication constraints in WSNs, a sensor node can only communicate directly with its $n'$ neighboring nodes. Since the expected node degree must be at least $d$ as calculated, the required probability of successfully performing key-setup with some neighboring node is now:

$$p_{required} = \frac{d}{n' - 1} \tag{6.3}$$

This implies that any two nodes in the WSN should share at least one secret key with probability no less than $p_{required}$. Further, the probability of two nodes $i$ and $j$ sharing at least one secret key can be computed as follows:

$$p = P(\mathcal{R}_i \cap \mathcal{R}_j \neq \emptyset) = 1 - P(|\mathcal{R}_i \cap \mathcal{R}_j| = 0) \tag{6.4}$$

For the key pre-distribution scheme in [28], $p$ is computed as

$$p = 1 - \frac{\binom{K-R}{R}}{\binom{K}{R}} \tag{6.5}$$

where $K$ is the size of the *key pool*, and $R$ is the size of the *key ring*. In $q$-composite scheme proposed in [17], the above calculation is now

$$p = P(|\mathcal{R}_i \cap \mathcal{R}_j| \geq q) = 1 - \sum_{s=0}^{q-1} P(|\mathcal{R}_i \cap \mathcal{R}_j| = s) \tag{6.6}$$

Note that in [28, 17]

$$P(|\mathcal{R}_i \cap \mathcal{R}_j| = s) = \frac{\binom{K}{s}\binom{K-s}{2(R-s)}\binom{2(R-s)}{m-s}}{\binom{K}{R}^2} \tag{6.7}$$

Therefore, key pool size $K$ and key ring size $R$ can be calculated by relating Eq. (6.3) with Eq. (6.5) or (6.6).

## 6.2 The Proposed Random Key Pre-Distribution Scheme

### 6.2.1 Terms and Notation

In this chapter we use the following notation and terms for the convenience of description.

- *Key Pool*: A *key pool* $\mathcal{K}$ with $|\mathcal{K}| = K$ is a pool of random symmetric keys, from which each sensor node is independently assigned a subset, namely, a *key ring* in the key pre-distribution scheme for a WSN. The cardinality of $\mathcal{K}$ equals to $K$.

- *Key Chain*: A *key chain* $\mathcal{C}$ with $|\mathcal{C}| = C$ is a subset of $\mathcal{K}$, and $L$ equal-sized *key chains* in total form a complete *key pool*. Therefore, we have $C = K/L$. Each *key chain* is independently generated via a unique generation key, namely, $g_i$ and a publicly known seed, namely, *seed*, by applying a keyed hash algorithm repeatedly. The value of the publicly known seed is the same for every key chain. Each key chain is uniquely indexed by its ID, namely, $\mathcal{C}_i$ and $\mathcal{C}_i \in [0, L-1]$.

- *Key Ring*: A *key ring* $\mathcal{R}_i$ with $|\mathcal{R}_i| = R$ is a subset of *Key Pool* with the cardinality of $R$ ($R \leq K$), which is independently assigned to a sensor node $i$ following the assignment rules defined by the key pre-distribution scheme. Note that $R$ is the same for every sensor node.

- *Key Graph*: Let $V$ represent all sensor nodes in a WSN. A *key graph* $\mathcal{G}(V, E)$ is constructed in the following manner: for any two nodes $i$ and $j$ in $V$, there exists an edge $e_{ij} \in E$ between them if and only if $\mathcal{R}_i \cap \mathcal{R}_j \neq \emptyset$. Note that $|V| = n$ for a WSN of size $n$. We say that a *key graph* $\mathcal{G}(V, E)$ is *connected* if

and only of any two nodes $i$ and $j$ belonging to $V$ can reach each other via edge set $E$ only.

- In a WSN of size $n$, each network node is uniquely identified through its ID, which ranges from 0 to $n-1$. The length of a node ID is therefore up to $\log_2 n$ bits.

We say that a *key graph* $\mathcal{G}(V, E)$ is *connected* if and only if any two nodes $i$ and $j$ belonging to $V$ can reach each other via edge set $E$ only. In $q$-composite scheme [17], a *key graph* $\mathcal{G}(V, E)$ is *connected* if and only if any two nodes $i$ and $j$ belonging to $V$ can reach each other through no less than two independent paths via edge set $E$ only.

A cryptographically secure one-way hash function $\mathcal{H}$ has the following property: for $y = \mathcal{H}(x, k)$, 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 $x$. A keyed hash algorithm like HMAC is provably secure and can be easily constructed on top of any secure one-way hash algorithms like SHA-1 [46]. However, a general purpose hash algorithm like SHA-1 is not suitable for sensor nodes, because 1) it is too complicated for an 8-bit micro-processor; 2) its message block length is at least 512-bit, which might be too large for sensor nodes and thus is not energy efficient. In [113], a class of universal hash functions WH is proposed for sensor nodes, whose message block is $w$-bit with a $2^{-w}$ collision probability. This hash function is highly power efficient. The implementation of WH shows that it consumes only $11.6\mu W$ at 500 kHz. In the proposed scheme, we use WH in our key chain generation. The input and output length will be both 64-bit and no padding operation is needed at all. By applying the keyed hash function $\mathcal{H}$ repeatedly on an initial value $m$, one can obtain a chain of outputs. Based on the properties described above, we know that these outputs are independent with each other and without knowing the secret key used by $\mathcal{H}$, one can not deduce any value on the chain even from other values of the same hash chain.

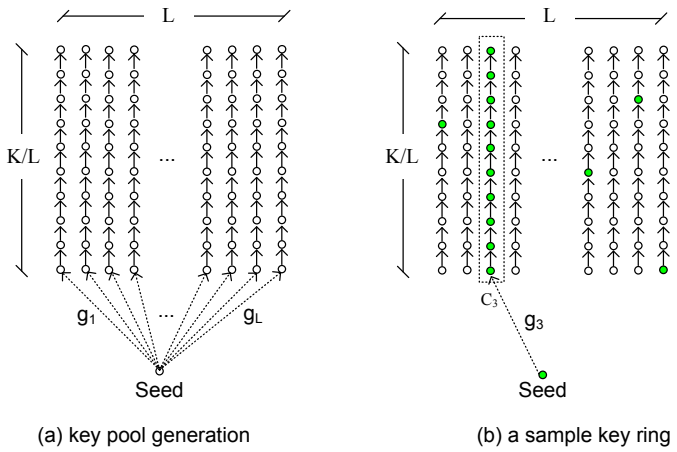


Figure 6.1: A sample key pool and key ring

### 6.2.2 Random Key Pre-distribution Scheme

The proposed key pre-distribution scheme consists of two phases: key assignment phase and shared-key discovery & path-key establishment phase. Although the way to find shared keys is different, the shared-key discovery and path-key establishment phase is more or less the same as in the previous schemes. In our scheme, the most significant difference lies in the key assignment phase. We propose two different schemes: the basic scheme and the $q$-composite scheme for key assignment phase. The details of the proposed schemes are described below.

**Key Assignment Phase**:

- *Key pool generation*: Key pool $\mathcal{K}$ is determined by the following two parameters: key pool size $K$ and the number of key chains $L$. Therefore, a key pool $\mathcal{K}$ consists of $L$ different key chains: $\mathcal{K} = \bigcup_i \mathcal{C}_i$ $(i = 0, .., L-1)$ and $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ $(i \neq j)$. Each key chain $\mathcal{C}_i$ is generated via a unique generation key $g_i$ and the publicly known seed *seed* by applying a keyed hash algorithm repeatedly. Thereby, the $l$-th key of key chain $\mathcal{C}_i$ is conceptually computed as

$$k_{c_i,l} = \mathcal{H}^l(seed, g_i) \tag{6.8}$$

where $\mathcal{H}^l(seed, g_i) = \mathcal{H}(\mathcal{H}^{l-1}(seed, g_i), g_i)$ and so on. Note that $g_i$ is only known to its assigned sensor nodes and should be strictly kept secret from other nodes in the WSN. At the same time, we use the pair $(\mathcal{C}_i, l)$ to index the corresponding key. Hence,

$$\mathcal{C}_i = \bigcup_{l=1}^{K/L} k_{c_i,l} \tag{6.9}$$

A graphical illustration of the concepts of key pool and key chains is shown in Fig. 6.1(a).

- *Key ring loading*: In this step, each node is loaded with its assigned key ring $\mathcal{R}$, which contains two parts, $\mathcal{R}_1$ and $\mathcal{R}_2$, where $\mathcal{R}_1$ is the generation knowledge of a number of key chains and $\mathcal{R}_2$ is a set of individual random keys from different key chains. To be more specific, for node $i$, $\mathcal{R}_i = \mathcal{R}_{i,1} \cup \mathcal{R}_{i,2}$. The assignment rules are as follows. First, node $i$ is assigned with $r_0$ randomly selected key chains. However, instead of storing all the $K/L$ keys in each key chain, node $i$ only stores the corresponding key chain generation keys (one key per key chain). Therefore, it stores $r_0$ keys for this part, i.e., $|\mathcal{R}_{i,1}| = r_0$. From these $r_0$ key-generation keys, $r_0 \times (K/L)$ random keys can be calculated effectively. Second, node $i$ is additionally assigned with $r_1$ randomly selected keys each from a different key chain. Hence, we have $|\mathcal{R}_{i,2}| = r_1$. An example is shown in Fig. 6.1(b), where the key chain and keys in green (filled with color) can be a sample key ring, where $r_0 = 1$. For the proposed $q$-composite scheme, the assigning rules are the same but with larger $r_0, r_1$ values in general.

**Shared-key Discovery** & **Path-key Establishment Phase**:

During the network bootstrapping phase, each sensor node is required to broadcast the key index information of its key ring, i.e., $\mathcal{R}_i$, to expose its key information to the neighbor nodes. Hence, each node will know which keys its neighbors have. Each node then examines the key index information of its own key ring to find or calculate

the keys it shares with the neighbor nodes. For node $i$ to find the shared key(s) with node $j$, it matches the key indexes of $\mathcal{R}_i$ and $\mathcal{R}_{j,2}$. If $\mathcal{R}_{i,2} \cap \mathcal{R}_{j,2} \neq \emptyset$, those are the keys node $i$ shared with node $j$. If $\mathcal{R}_{i,1} \cap \mathcal{R}_{j,2} \neq \emptyset$, node $i$ needs to calculate the key(s) in common. For example, if node $x$ contains a key indexed as $k_{c_i,l}$ and node $y$ contains key chain $\mathcal{C}_i$, node $y$ immediately knows that it shares key $k_{c_i,l}$ with node $x$ upon receiving node $x$'s broadcast message. Node $y$ then simply calculates $k_{c_i,l}$ following Eq. (6.8). If node $y$ also contains key $k_{c_i,l}$, then there is no need for calculation. If there are more than one shared key, the final pairwise key is simply computed as the hash value of the shared keys. The concatenation sequence of the shared keys can be easily enforced to ensure the same output hash value. For example, if $ID_x < ID_y$, then the keys sent by node $x$ becomes the first in the concatenation. In case that two neighbor nodes share no common key directly, we use the same path-key establishment technique as described in [28] to establish a pairwise key between them. Note that in our setting, no shared key is established when two nodes only share one or more key chains, that is, we do not count in the situations that for any two nodes $i$ and $j$, $\mathcal{R}_{i,2} \cap \mathcal{R}_{j,2} = \emptyset$ and $\mathcal{R}_{i,1} \cap \mathcal{R}_{j,2} = \emptyset$ and $\mathcal{R}_{i,2} \cap \mathcal{R}_{j,1} = \emptyset$ and $\mathcal{R}_{i,1} \cap \mathcal{R}_{j,1} \neq \emptyset$. We treat this case the same as that the two nodes do not share any key and use the path-key establishment technique to establish a shared key between them. At this point, each node now shares at least a key with each of its neighbor nodes, respectively. We use the same method as in [17] to generate the link key $k_{link} = hash(k_1|k_2|...|k_i)$ to secure the communication between two sensor nodes, where $i$ ($q \leq i \leq r_0 + r_1$) is the number of keys it actually shares with a particular neighbor node.

## 6.3 Performance Analysis and Simulation

In this section, we evaluate the proposed two schemes in terms of required storage space (i.e., key ring size) at the sensor node, given the required key sharing probability

$p_{required}$. For a WSN of network size $n$ and neighborhood size $n'$, $p_{required}$ can be calculated using Eq. (6.3). Then the key pool size $K$ and key ring size $R$ can be properly chosen according to Eq. (6.5) [28] and Eq. (6.6) [17], respectively. We first develop the equations to calculate the probability that two nodes sharing at least one or $q$ keys for the proposed two schemes. We next compare the performance of the proposed schemes with that of [28] and [17], respectively. From the description of the scheme we know that key ring $\mathcal{R}$ contains two parts: $\mathcal{R}_1$ and $\mathcal{R}_2$ in addition to a public seed. Hence, $R$ is calculated as follows:

$$R = |\mathcal{R}_1| + |\mathcal{R}_2| + 1 = r_0 + r_1 + 1 \tag{6.10}$$

**Connectivity Calculation**:

We consider the probabilities that any two nodes, say $n_i$ and $n_j$, share at least one key (for the basic scheme) and at least $q$ keys (for the $q$-composite scheme).

For any node, say $n_i$, the number of possible key ring assignments can be calculated as follows:

$$(I) = \binom{L}{r_0}\binom{L-r_0}{r_1}\binom{K/L}{1}^{r_1}$$

For the other node, say $n_j$, the number of possible key ring assignments that do not share any key with node $n_i$ can be calculated as follows. Note that the two nodes may share common key chains.

$$(II) = \sum_{s=0}^{r_0}\binom{L-r_0-r_1}{r_0-s}\binom{r_0}{s}\sum_{i=0}^{r_1}\binom{L-2r_0-r_1+s}{r_1-i}$$
$$\binom{r_1}{i}\binom{\frac{K}{L}}{1}^{r_1-i}\binom{\frac{K}{L}-1}{1}^{i}$$

Similarly, the number of possible key ring assignments at the other node $n_j$ that share exactly $x$ ($1 \le x \le r_0 + r_1$) keys with node $n_i$ (excluding key chain to key chain overlapping) can be computed as follows:

$$(III) = \sum_{t=0}^{r_0}\sum_{s=0}^{r_0-t}\binom{L-r_0-r_1}{r_0-s-t}\binom{r_0}{s}\binom{r_1}{t}\sum_{i=0}^{r_0-s}\sum_{j=0}^{min(r_1-t,r_0-s)}$$

$$\binom{L-2r_0-r_1+s+t}{r_1-i}\binom{r_1-t}{j}\binom{r_0-s}{i}$$
$$\sum_{m=0}^{j}\binom{j}{m}\binom{\frac{K}{L}}{1}^{r_1-j}\binom{\frac{K}{L}-1}{1}^{j-m}$$

where $t+i+m=x$ and $t+i+m \leq r_0+r_1-t$.

Therefore, the probability that any two nodes share no key is

$$Pr\{|\mathcal{R}_i \cap \mathcal{R}_j| = 0\} = \frac{(II)}{(I)}$$

and the probability that any two nodes share exactly $x$ keys is

$$Pr\{|\mathcal{R}_i \cap \mathcal{R}_j| = x\} = \frac{(III)}{(I)}$$

.

Hence, for the basic scheme, we have

$$p_{required} = 1 - \frac{(II)}{(I)} \tag{6.11}$$

For the proposed $q$-composite scheme (q=2), we have

$$p_{required} = 1 - Pr\{|\mathcal{R}_i \cap \mathcal{R}_j| = 0\} - Pr\{|\mathcal{R}_i \cap \mathcal{R}_j| = 1\} \tag{6.12}$$
$$= 1 - \frac{(II)}{(I)} - \frac{(III)(x=1)}{(I)} \tag{6.13}$$

**Performance Evaluation**:

In order to thoroughly examine the performance of the proposed two schemes, we vary the values of $r_0$ and $r_1$ under different network size $n$, key pool size $K$, and the number of key chains $L$ to see how the connectivity varies accordingly. The key ring size $R$ is calculated as $r_0 + r_1 + 1$. Also note that in the proposed schemes, the value of $L$ is a function of that of network size $n$. The value of $L$ determines the security strength against node capture as will be discussed in detail in the next section. The network size is first set as $n = 10,000$. The key pool size $K$ is set to 5, 10, and 50 times of the network size. The number of key chains $L$ is set to

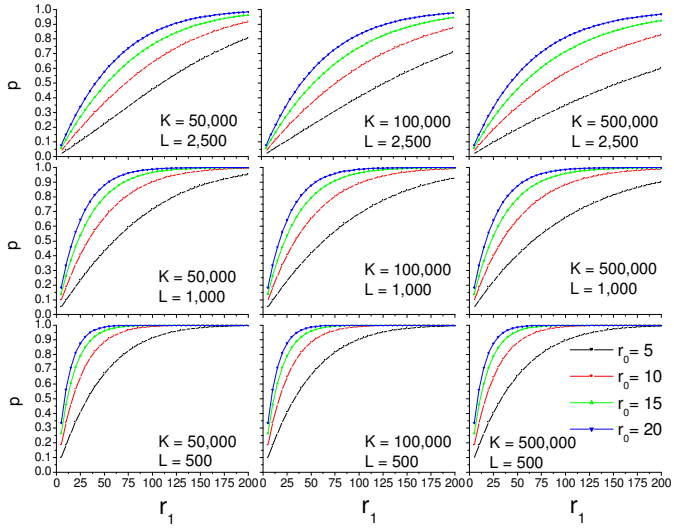


Figure 6.2: The proposed basic scheme: $p$ vs. $r_0$ and $r_1$ under different values of $K$ and $L$, when network size $n$ is $10,000$.

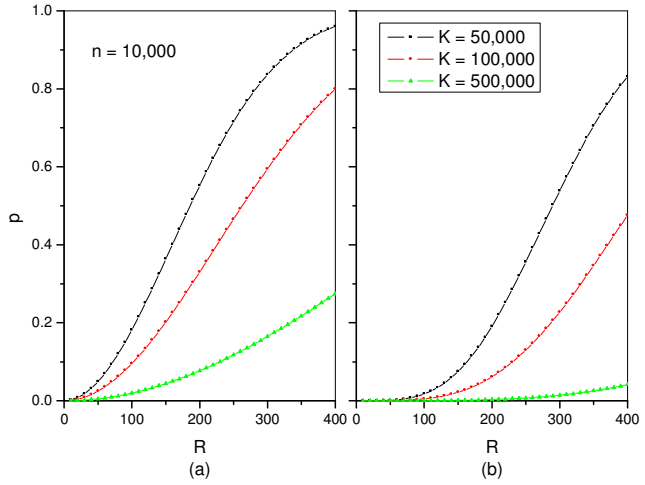


Figure 6.3: (a) Performance of Gligor's Scheme and (b) performance of Chan's Scheme ($q = 2$) when network size $n = 10,000$.

0.05, 0.1 and 0.25 times of the network size. Fig. 6.2 shows the performance of the proposed basic scheme at $n = 10,000$. Fig. 3(a) illustrates the performance of Eschenauer et al.'s scheme at the same network size. The proposed basic scheme offers a great performance improvement as compared to Eschenauer et al.'s scheme. For example, When $n = 10,000$ and $p_{required} = 0.5$, $R$ is required to be around 260 given $K = 100,000$ in [28]; on the other hand, under the same settings $R$ can be as low as 30 in the proposed scheme, although this choice is not good as it has a low security strength against node capture, as we will show in Section 6.4. However, when similar security strength is assumed, the required key ring size in the proposed scheme is around 50% less than that of Eschenauer et al.'s scheme as will be shown in Section 6.4. The evaluation of the proposed $q$-composite scheme is shown in Fig. 4 and as comparison, the performance of Chan et. al.'s $q$-composite scheme under the same settings is illustrated in Fig. 3(b). The performance improvement is again very significant. For instance, when $n = 10,000$ and $p_{required} = 0.5$, $R$ is required to be around 275 ($q = 2$) given $K = 50,000$ in [17]; on the other hand, in the proposed scheme $R$ can be as low as 50 ($q = 2$) in the proposed scheme.

The improvement of the proposed two schemes goes higher as the network size $n$ grows. For example, when $n = 50,000$ and $p_{required} = 0.5$, the proposed basic scheme requires as low as 100 keys with $K = 250,000$ as shown in Fig. 5, while 410 keys are required in Eschenauer et. al.'s scheme for comparable security strength. This fact shows that our scheme is highly scalable to the larger network sizes. At the same time, a requirement of $R = 410$ implies that the scheme is no longer practical under the given network size due to the extremely limited storage space of the sensor nodes.

The above figures (Fig. 2 and Fig. 4) also illustrate how the performance of the proposed two schemes vary under different system settings, i.e., different values of $K$, $L$ and $(r_0, r_1)$ pairs. We find that under a given network size $n$, the performance of the proposed schemes decreases as either $K$ or $L$ increases. From Eq. (6.15) developed

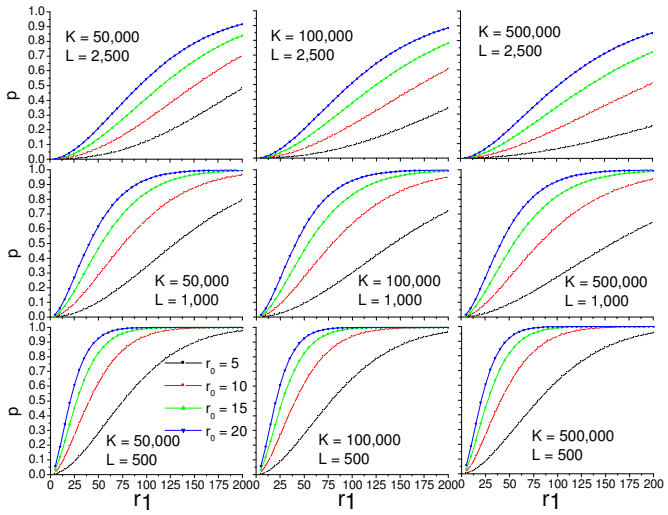


Figure 6.4: The proposed $q$-composite scheme: $p$ vs. $r_0$ and $r_1$ under different values of $K$ and $L$, when network size $n = 10,000$ and $q = 2$.

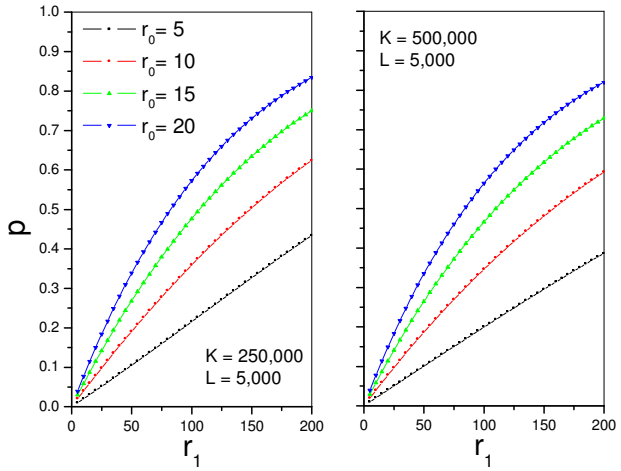


Figure 6.5: The proposed basic scheme: $p$ vs. $r_0$ and $r_1$, when network size $n = 50,000$.

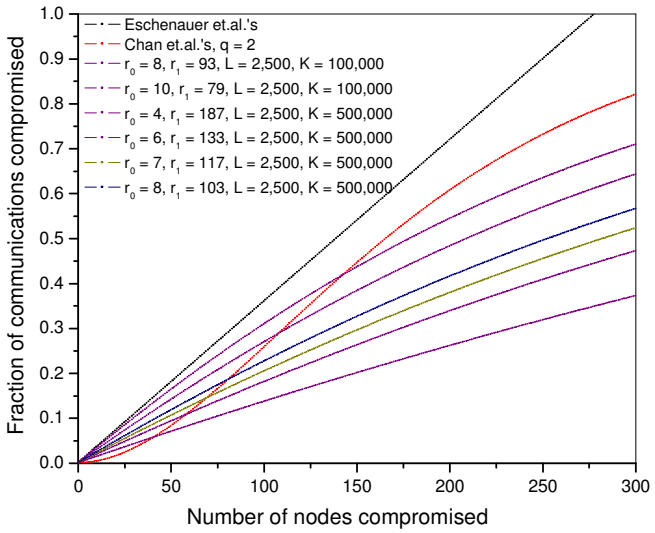


Figure 6.6: Security strength of the proposed basic scheme with $n = 10,000$, $p_{required} = 0.5$ and $R_{max} = 192$

below, we know that the values of $K$ and $L$ also determine how resilient the proposed schemes is against node capture. On one hand, we desire smaller values of $K$ and $L$ to achieve better key sharing probability with $R$ fixed; on the other hand, the proposed schemes present better resilience property against node capture when larger values of $K$ and $L$ are used. Therefore, this can be formulated as a constrained optimization problem:

*Under the given system parameters of networks size n and neighborhood size n', minimize R, where* $R = r_0 + r_1 + 1$ *as defined in Eq. (6.10) and the values of* $(r_0, r_1)$ *are subject to Eq. (6.11) or Eq. (6.13).*

## 6.4 Security Strength Analysis

To study the security strength of the proposed scheme, we first prove that without the knowledge of the corresponding key chain generation key $g_i$, whatever number of keys of a key chain that are compromised will not affect the security of the remaining

keys in that key chain.

*Lemma: For a given key chain $\mathcal{C}_i$ of size $K/L$, the knowledge of any combination of $\frac{K}{L}-1$ keys except for the key in question can not result any advantage on the knowledge of the remaining key without knowing the corresponding key chain generation key $g_i$.*

Proof: In the proposed scheme, a key chain is generated using the keyed hash function following Eq. (6.8). Hence, any key $k_{c_i,l}$ inside a key chain holds the following relationship with other keys of the same key chain:

$$\cdots, k_{c_i,l} = \mathcal{H}(k_{c_i,l-1}, g_i), k_{c_i,l+1} = \mathcal{H}(k_{c_i,l}, g_i), \cdots$$

Therefore, it is computationally infeasible to compute $k_{c_i,l}$ from either $k_{c_i,l+1}$ or $k_{c_i,l-1}$ without the secret key $g_i$ because the keyed hash function is used. On the other hand, it is also computationally infeasible to recover the key chain generation key $g_i$ from any combination of its generated keys because of the same reason. □

Next we study the resilience property of the proposed scheme against node capture by calculating the fraction of links in the network that are compromised due to key revealing resulted from node capture. In the proposed scheme, since each node actually has the knowledge of $\frac{r_0 K}{L} + r_1$ keys, the probability that a given key does not belong to a node is $1 - (\frac{r_0}{L} + \frac{r_1}{K})$. Therefore, if there are $m$ compromised nodes, the probability that a given key is not compromised should be $(1 - (\frac{r_0}{L} + \frac{r_1}{K}))^m$. The expected fraction of total keys compromised is thus $1 - (1 - (\frac{r_0}{L} + \frac{r_1}{K}))^m$. If the communication link between two nodes has its link key $k_{link}$ computed from $s$ ($s \geq q$) shared keys, the probability of that link being compromised is then $(1 - (1 - (\frac{r_0}{L} + \frac{r_1}{K}))^m)^s$ and hence, in the worst case the compromising probability is

$$(1 - (1 - (\frac{r_0}{L} + \frac{r_1}{K}))^m)^q \tag{6.14}$$

Therefore, averagely the compromising probability is

$$\sum_{s=q}^{m} (1 - (1 - (\frac{r_0}{L} + \frac{r_1}{K}))^m)^s \frac{P(|\mathcal{R}_i \cap \mathcal{R}_j| = s)}{\sum_{t=q}^{m} P(|\mathcal{R}_i \cap \mathcal{R}_j| = t)} \tag{6.15}$$

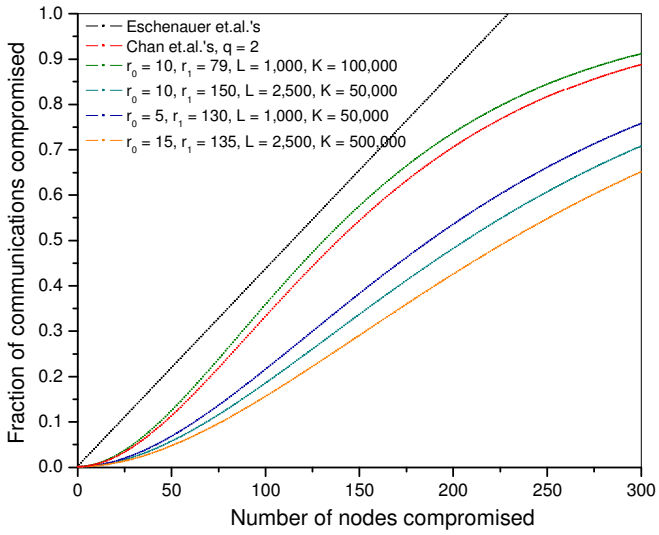


Figure 6.7: Security strength of the proposed $q$-composite scheme with $n = 10,000$, $p_{required} = 0.5$, $q = 2$ and $R_{max} = 161$

Eq. (6.15) also represents the fraction of additional communications that an adversary can compromise based on the key information retrieved from $m$ captured nodes in the worst case. Fig. 6 shows the security strength of the proposed basic scheme, where $n = 10,000$, $p_{required} = 0.5$ and $R_{max} = 192$. Obviously, the proposed scheme offers a much better resilience property while requiring a much smaller key ring size when compared with Eschenauer and Gligor's. Fig. 7 illustrates the security strength of the proposed $q$-composite scheme, where $n = 10,000$, $p_{required} = 0.5$, $q = 2$ and $R_{max} = 161$. Again the proposed $q$-composite scheme offers a much better resilience property as compared to that of Chan et. al.'s. To exactly illustrate how much is the improvements gained by the proposed scheme, we now fix the key ring size $R$ for each scheme and other system settings remain the same. Fig. 8 shows the security strength of the proposed basic scheme, when $n = 10,000$, $p_{required} = 0.5$ and key ring size $R$ is fixed as 90. We can see that when the fraction of the compromised communication has reached to 100% in Eschenauer, the proposed basic scheme only has a value of 38% under the same settings. Fig. 9 shows the significant resilience

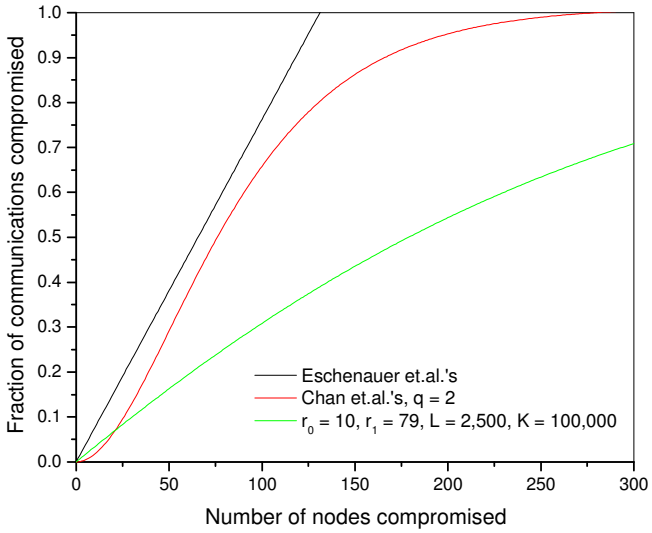


Figure 6.8: Security strength of the proposed basic scheme with $n = 10,000$, $p_{required} = 0.5$ and $R = 90$

improvement of the proposed $q$-composite scheme when $n = 10,000$, $p_{required} = 0.5$, $q = 2$ and key ring size $R$ is fixed as 90. To compromise 10% communications among the remaining network nodes, only 25 compromised nodes are required; however, 50 nodes are required in the proposed scheme. The improvement is around 100%. More importantly, the proposed $q$-composite scheme holds a much better security strength under both small scale attack and large scale attack, which overcomes the shortcomings presented in Chan et. al.'s scheme, that is, achieving better security strength under small scale attack while trading off increased vulnerability in the face of a large scale attack on network nodes. This situation is illustrated in Fig. 10.

## 6.5 Summary

In this chapter, we have proposed a new approach for random key pre-distribution in WSNs. The novelty of this approach is that, instead of requiring the sensor nodes store all the assigned keys, the majority of the keys are represented and stored in

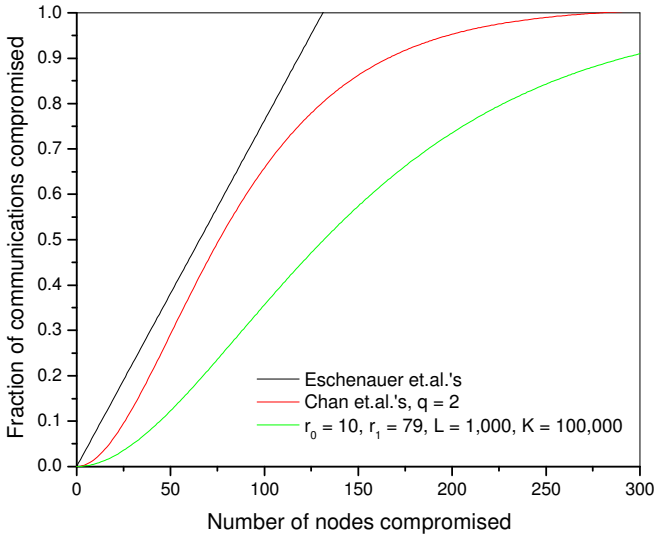


Figure 6.9: Security strength of the proposed $q$-composite scheme with $n = 10,000$, $p_{required} = 0.5$, $q = 2$ and $R = 90$.

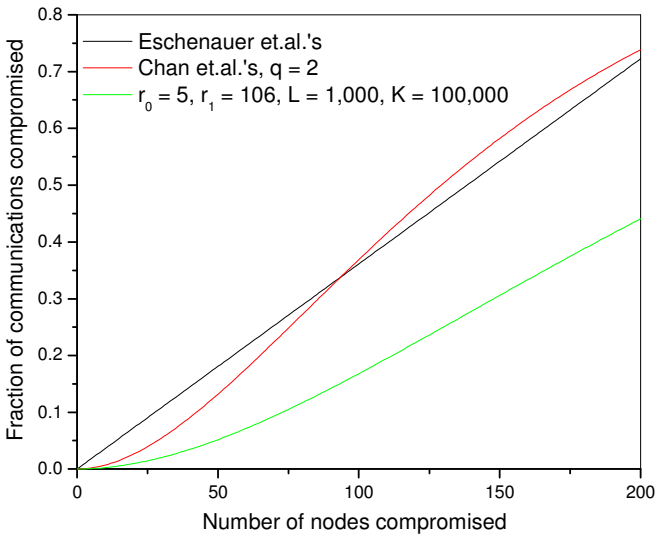


Figure 6.10: Security strength of the proposed $q$-composite scheme with $n = 10,000$, $p_{required} = 0.33$, $q = 2$ and $R = 112$.

terms of key generation key sets with a very small size by carefully designing the key pool, which significantly reduces storage space while holding the same security strength. The proposed scheme is hence, highly scalable to the larger network sizes. The proposed scheme outperforms the previous random key pre-distribution schemes under both small scale and large scale attacks, especially when the network size is large ($\geq 10,000$) as shown by our thorough analysis.

# Chapter 7

# Conclusion

This chapter summarizes the contributions of the research presented in this dissertation and suggests directions for future work.

## 7.1 Contributions

This dissertation studies communication security in WSNs with respect to three important aspects: broadcast/multicast security, data report security, and random key pre-distribution.

- Multicast Security: We identified the problem of multiuser broadcast authentication in WSNs and pointed out a serious security vulnerability inherent to the symmetric-key based $\mu$TESLA-like schemes [73, 54, 57, 58]. We then proposed several PKC-based schemes to address the proposed problem with minimized computational and communication costs. We achieved our goal by integrating several cryptographic building blocks, such as the Bloom filter, the partial message recovery signature scheme, and the Merkle hash tree, in an innovative manner. We also analyzed both the performance and security resilience of the proposed schemes. A quantitative energy consumption analysis was given in de-

tail and demonstrated the effectiveness and efficiency of the proposed schemes. Our research on this topic appears in [85, 82, 92, 86].

To address multicast encryption problem, we first analyzed and classified the multicast group semantics that are inherently demanded by WSNs. We then proposed the GPLD scheme which, to our best knowledge, is the first multicast encryption scheme of its kind that supports various multicast group semantics and is tailored for WSNs. GPLD advances the current state-of-the-art by enabling dynamic changing and simultaneous formation of multiple multicast groups. We developed a novel multicast encryption technique called *global-partition, local-diffusion*. This technique effectively minimizes global (sink-to-sensor) group key distribution and re-keying traffic, while maintaining its support to various multicast group semantics. The efficiency and security properties of GPLD were justified through both analysis and simulations. Our research on this topic appears in [87].

- Data Report Security: To address data report security, we proposed a novel location-aware multi-functional key management framework called LEDS [83]. LEDS efficiently embeds the location (cell) information of each sensor into all types of symmetric secret keys owned by that node, and thus provides end-to-end security guarantee. Each legitimate event report in LEDS is endorsed by multiple sensing nodes and is encrypted with a unique secret key shared between the event sensing nodes and the sink. Furthermore, the authenticity of the corresponding event sensing nodes can be individually verified by the sink. This novel setting successfully eliminates the possibility that the compromise of nodes other than the sensing nodes of an event report may result in security compromise of that event report. LEDS possesses efficient en-route false data filtering capability to deal with the infamous bogus data injection attack, which at the same time significantly reduces energy cost as unnecessary forwarding

is eliminated. LEDS also provides high level assurance on data availability by dealing with both report disruption attack and selective forwarding attack, simultaneously. Our research on this topic appears in [83, 84].

For applications related to large-scale spatial phenomena monitoring, we further studied the problem of securing event boundary detection in WSNs and showed why existing boundary detection schemes fail in adversarial environments [91]. We presented a Secure Event Boundary Detection (SEBD) scheme [91], which is to our best knowledge the first protocol of its kind to secure event boundary detection in WSNs. SEBD withstands many types of attacks. We propose an enhanced statistic model for localized event boundary detection with proactive faulty measurements correction. Our model is more accurate and robust against node compromise and random fault as compared to existing schemes [19, 48, 23]. Moreover, it is nonparametric without relying on any prior knowledge of node compromise and fault probability, which, however, is required by existing schemes to achieve optimal results [19, 23]. We used extensive simulations to evaluate SEBD and show a very good performance and security strength. Our research on this topic appears in [89, 91].

- Random Key Pre-distribution: We proposed a highly efficient random key pre-distribution scheme, which combines the random key pre-distribution technique and the hash chain technique [88]. The novelty of our scheme lies in that, instead of requiring each sensor node to store all the chosen keys, the majority of the keys a node possesses are represented and stored in the form of a small number of key-generation keys by carefully designing the key pool, and therefore, the storage overhead is significantly reduced while the same security strength holds. Compared with the existing schemes, the proposed scheme is more scalable and more secure in the sense that 1) Under the given resilience requirement against node capture, the proposed scheme requires a much smaller key ring size than

the previous schemes; 2) Under the given maximum allowed key ring size, the proposed scheme has a much better resilience property against node capture than the previous schemes. Our research on this topic appears in [88, 90].

## 7.2 Future Direction

In this section, we briefly mention areas of future work based on our thesis. We outline three directions as follows.

- Secure Distributed Data Storage and Retrieval: In this dissertation, we have focused on securing the information in communication; securing information in storage are not adequately addressed. In the context of ubiquitous computing, WSNs are envisioned to provide ubiquitous information sensing, storing, and content delivering services. For the purpose of efficient data management, an increasing number of distributed in-network data storage and retrieval schemes have been proposed recently [101]. This makes the absence of mechanisms for securing stored information becoming a more and more severe issue. Hence, more research efforts should be put on this problem.

- Secure Data Aggregation: In many applications, the raw information sensed by individual sensors should be aggregated for the purpose of reducing the communication cost and energy expenditure in data collection [91, 110, 18]. In this dissertation, we studies a special form of data aggregation, i.e., event boundary detection. However, data aggregation in WSNs can be of different forms as desired by the underlying applications. Each different type of data aggregation may require customized secure aggregation techniques. Moreover, common techniques should also be developed to reduce the complexity of the protocol stack. More research should be done along this direction.

- Privacy-aware Security Services: Current security research in WSNs rarely consider privacy problem. However, data and network communication privacy can be a big concern in many applications [96, 79, 80, 78]. As WSNs are envisioned to become more and more pervasive, privacy-aware security services should be further developed.

# Bibliography


[1] Crossbow technology inc. wireless sensor networks. http://www.xbow.com/.

[2] Intel pxa255 processor electrical, mechanical, and thermal specification. http://www.intel.com/design/pca/applicationsprocessors/manuals/278780.htm.

[3] Texas instruments inc., msp430 family of ultra-lowpower 16-bit risc processors. http://www.ti.com.

[4] Tinyos operation system. http://www.tinyos.net/.

[5] I. F. Akyildiz and I. H. Kasimoglu. Wireless sensor and actor networks: Research challenges. *Ad Hoc Networks Journal (Elsevier)*, 2(4):351–367, Oct. 2004.

[6] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Commun. Mag.*, 40(8):102–116, Aug. 2002.

[7] M. Aydos, T. Yanik, and C. K. Koc. An high-speed ecc-based wireless authentication protocol on an arm microprocessor. In *16th Annual Computer Security Applications Conference*, pages 401–409, New Orleans, Louisiana, 2000.

[8] D. Boneh, H. Shacham, and B. Lynn. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4).

[9] O. Chipara C. Fok C. Lu, G. Xing and S. Bhattacharya. A spatiotemporal query service for mobile users in sensor networks. In *IEEE ICDCS*, Columbus,, 2005.

[10] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *IEEE INFOCOM*, 1999.

[11] S. Capkun and J.P. Hubaux. Secure positioning in wireless networks. *IEEE Journal on Selected Areas in Communications*, Feb. 2006.

[12] D.W. Carman, P.S. Kruus, and B.J. Matt. Constraints and approaches for distributed sensor network security. NAI Labs Technical Report 00-010, Sep. 2000.

[13] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *ACM SIGCOMM Workshop Data Comm. Latin America and the Caribbean*, Costa Rica, Apr. 2001.

[14] A. Chadha, Y. Liu, and S. Das. Group key distribution via local collaboration in wireless sensor networks. In *IEEE SECON*, 2005.

[15] H. Chan and A. Perrig. Security and privacy in sensor networks. *IEEE Computer*, 36(10):103–105, Oct. 2003.

[16] H. Chan and A. Perrig. Pike: Peer intermediaries for key establishment. In *IEEE INFOCOM*, Miami, FL, USA, March 2005.

[17] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security & Privacy*, Oakland, CA, May 2003.

[18] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation for sensor networks. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS 2006)*, October 2006.

[19] K. K. Chintalapudi and R. Govindan. Localized edge detection in sensor fields. *Ad Hoc Networks Journal*, pages 59–70, 2003.

[20] C.-Y. Chong and S. P. Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proc. of IEEE*, Aug 2003.

[21] T. Clouqueur, K.K. Saluja, and P. Ramanathan. Fault tolerance in collaborative sensor networks for target detection. *IEEE Transactions on Computers*, 53(3):320–333, 2004.

[22] J. Deng, R. Han, and S. Mishra. Secure code distribution in dynamically programmable wireless sensor networks. In *Proc. of ACM/IEEE IPSN'06*, 2006.

[23] M. Ding, D. Chen, K. Xing, and X. Cheng. Localized event detection in sensor networks. In *IEEE INFOCOM*, Miami, FL, USA, Mar. 2005.

[24] Y. Dodis and N. Fazio. Public key broadcast encryption for stateless receivers. In *ACM Workshop on Digital Rights Management (DRM)*, 2002.

[25] W. Du, J. Deng, Y. Han, S. Chen, and P.K.Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *IEEE INFOCOM*, HongKong, China, March 2004.

[26] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *ACM CCS*, Washington, DC, Oct. 2003.

[27] W. Du, R. Wang, and P. Ning. An efficient scheme for authenticating public keys in sensor networks. In *6th ACM International Symposium on Mobile Ad*

*Hoc Networking and Computing (MobiHoc)*, pages 58–67, Urbana-Champaign, IL, USA, May 2005.

[28] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. In *ACM CCS*, Washington, DC, Nov. 2002.

[29] D. Estrin, D. Culler, and K. Pister. Connecting the physical world with pervasive networks. *IEEE Percasive Computing*, Jan-Mar 2002.

[30] D. Estrin, A. Sayeed, and M. Srivastava. Wireless sensor networks. In *ACM Mobicom*, 2002. Tutorial.

[31] R. Gennaro and P. Rohatgi. How to sign digital streams. *Information and Computation*, 165(1).

[32] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag, 2004.

[33] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *ASPLOS IX*, 2000.

[34] Y. Hu, A. Perrig, and D. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *IEEE INFOCOM*, San Francisco, CA, April 2003.

[35] D. Huang, M. Mehta, D. Medhi, and L. Harn. Location-aware key management scheme for wireless sensor networks. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04)*, 2004.

[36] J. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proc. of ACM SenSys'04*, Baltimore, 2004.

[37] C. Intanagonwiwat, R. Govindan, D. Estrin, and J. Heidemann. Directed diffusion for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 11(1):2–16, Feb. 2003.

[38] T. Itoh and S. Tsujii. A fast algorithm for computing multiplicative inverse in $gf(2^m)$ using normal bases. *Information and Communication*, 78:171–177, 1988.

[39] A. Perrig J. McCune, E. Shi and M. Reiter. Detection of denial-of-message attacks on sensor network broadcasts. In *IEEE Symposium on Security and Privacy*, 2005.

[40] J. Jung, T. Park, and C. Kim. A forwarding scheme for reliable and energy-efficient data delivery in cluster-based sensor networks. *IEEE Communication Letters*, 9(2):112–114, Feb. 2005.

[41] A. Kansal, D. Potter, and M. Srivastava. Performance aware tasking for environmentally powered sensor networks. In *ACM Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, New York, NY, USA, Jun. 2004.

[42] A. Kansal and M. Srivastava. An environmental energy harvesting framework for sensor networks. In *ACM/IEEE ISLPED*, 2003.

[43] Jens-Peter Kaps. *Cryptography for Ultra-Low Power Devices*. PhD thesis, ECE Department, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, May 2006.

[44] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad Hoc Networks*, 1(2), 2003.

[45] B. Karp and H.T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *ACM MOBICOM*, Boston, Aug. 2000.

[46] H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication. In *Internet RFC 2104*, 1997.

[47] B. Krishnamachari and S. Iyengar. Efficient and fault-tolerant feature extraction in wireless sensor networks. In *IPSN 2003*, volume LNCS 2634, pages 488–501, 2003.

[48] B. Krishnamachari and S. Iyengar. Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Transactions on Computers*, 53(3):241–250, Mar. 2004.

[49] J. Almeida L. Fan, P. Cao and A. Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networks (TON)*, 8(3).

[50] P. E. Lanigan, R. Gandhi, and P. Narasimhan. Secure dissemination of code updates in sensor networks. In *Proc. of ACM SenSys'05*, 2005. poster.

[51] P. E. Lanigan, R. Gandhi, and P. Narasimhan. Sluice: Secure dissemination of code updates in sensor networks. In *The 26th International Conference on Distributed Computing Systems (ICDCS'06)*, 2006.

[52] L. Lazos and R. Poovendran. Energy-aware secure multicast communication in ad-hoc networks using geographic location information. In *IEEE ICASSP*, 2003.

[53] L. Lazos and R. Poovendran. Serloc: Secure range-independent localization for wireless sensor networks. In *ACM WiSe'04*, Philadelphia, PA, Oct. 2004.

[54] D. Liu and P. Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *Proc. NDSS'03*, San Diego, CA, Feb. 2003.

[55] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *ACM CCS*, Washington, DC, Oct. 2003.

[56] D. Liu and P. Ning. Location-based pairwise key establishments for static sensor networks. In *ACM SASN*, Fairfax, VA, Oct. 2003.

[57] D. Liu and P. Ning. Multi-level $\mu$tesla: Broadcast authentication for distributed sensor networks. In *ACM Transactions in Embedded Computing Systems (TECS)*, volume 3 of *4*, 2004.

[58] D. Liu, P. Ning, S. Zhu, and S. Jajodia. Practical broadcast authentication in sensor networks. In *2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2005)*, Jul. 2005.

[59] Donggang Liu. *Security Mechanisms for Wireless Sensor Networks*. PhD thesis, CS Department, North Carolina State University, Raleigh, North Carolina, USA, July 2005.

[60] Konrad Lorincz, David Malan, Thaddeus Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoff Mainland, Steve Moulton, and Matt Welsh. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing*, pages 16–23, Oct-Dec 2004.

[61] S. Lumetta and M. Mitzenmacher. Using the power of two choics to improve bloom filters. preprint.

[62] Samuel Madden, Michael J. Franklin, Joseph Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[63] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM WSNA'02*, Atlanta GA, Sep. 2002.

[64] R. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Research in Security and Privacy*, 1980.

[65] M. Mitzenmacher. Compressed bloom filters. *IEEE/ACM Transactions on Networks (TON)*, 10(5).

[66] D. Naccache and J. Stern. Signing on a postcard. In *Proc. of Financial Cryptography'00*, volume 1962 of *LNCS*, pages 121–135. Springer, 2000.

[67] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO*, 2001.

[68] NIST. Proposed federal information processing standard for digital signature standard (dss). *Federal Register*, 56(169):42980–42982, 1991.

[69] NIST. Digital hash standard. In *Federal information processing standards publication 180-1*, 1995.

[70] NIST. Fips-197: Advanced encryption standard. 2001.

[71] R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. In *IPSN 2003*, volume LNCS 2634, pages 80–95, 2003.

[72] S.-J. Park, R. Vedantham, R. Sivakumar, and I.F.Akyildiz. A scalable approach for reliable downstream data delivery in wireless sensor networks. In *ACM MOBIHOC'04*, Tokyo, Japan, May 2004.

[73] A. Perrig, R. Canetti, J. Tygar, and D. Song. The tesla broadcast authentication protocol. *RSA CryptoBytes*, 2002.

[74] A. Perrig, D. Song, and D. Tygar. Elk: a new protocol for efficient large-group key distribution. In *IEEE S&P*, May 2001.

[75] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, and D. Culler. Spins: Security protocols for sensor networks. *ACM Wireless Networks*, Sep. 2002.

[76] R. Pietro, L. Mancini, Y. Law, S. Etalle, and P. Havinga. Lkhw: A directed diffusion-based secure multicast scheme for wireless sensor networks. In *ICPPW'03*, 2003.

[77] A. Chandrakasan R. Amirtharajah. Self-powered signal processing using vibration-based power generation. *IEEE Journal of Solid-State Circuits*, 33:687–695, 1998.

[78] K. Ren and W. Lou. Privacy enhanced access control in pervasive computing environments. In *2nd International Conference on Broadband Networks (BroadNets 2005)*, Boston, MA, USA, October 2005.

[79] K. Ren and W. Lou. Privacy-enhanced, attack-resilient access control in pervasive computing environments with optional context authentication capability. *ACM Mobile Networks and Applications (MONET) (special issue on Wireless Broadband Access)*, 12:79–92, 2007.

[80] K. Ren, W. Lou, K. Kim, and R. Deng. A novel privacy preserving authentication and access control scheme for pervasive computing environment. *IEEE Transactions on Vehicular Technology (TVT)*, 55(4):1373–1384, July 2006.

[81] K. Ren, W. Lou, and P. Moran. A proactive data security framework for mission-critical wireless sensor networks. In *IEEE military communication conference (MILCOM 2006)*, Washington DC, October 2006.

[82] K. Ren, W. Lou, K. Zeng, and P. Moran. On broadcast authentication in wireless sensor networks. *IEEE Transactions on Wireless Communications (TWC)*, 2007. Accepted.

[83] K. Ren, W. Lou, and Y. Zhang. Leds: Providing location-aware end-to-end data security in wireless sensor networks. In *IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.

[84] K. Ren, W. Lou, and Y. Zhang. Leds: Providing location-aware end-to-end data security in wireless sensor networks. *IEEE Transactions on Mobile Computing (TMC)*, 2007. Under Revision.

[85] K. Ren, W. Lou, and Y. Zhang. Multi-user broadcast authentication in wireless sensor networks. In *Fourth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2007)*, San Diego, CA, USA, June 2007.

[86] K. Ren, W. Lou, and Y. Zhang. Multi-user broadcast authentication in wireless sensor networks. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2007. Under Submission.

[87] K. Ren, W. Lou, B. Zhu, and S. Jajodia. Secure and efficient multicast in wireless sensor networks allowing ad-hoc group formations. 2007. Under Submission.

[88] K. Ren, K. Zeng, and W. Lou. On efficient key pre-distribution in wireless sensor networks. In *IEEE military communication conference (MILCOM 2005)*, Atlantic City, NJ, USA, October 2005.

[89] K. Ren, K. Zeng, and W. Lou. Fault-tolerant event boundary detection in wireless sensor networks. In *IEEE Global Telecommunications Conference (GLOBECOM 2006)*, San Francisco, CA, USA, November-December 2006.

[90] K. Ren, K. Zeng, and W. Lou. A new approach for random key pre-distribution in large-scale wireless sensor networks. *Journal of Wireless Communication and Mobile Computing (WCMC)*, 6(3):307–318, 2006. (Special Issue on Wireless Networks Security).

[91] K. Ren, K. Zeng, and W. Lou. Secure and fault-tolerant event bounary detection in wireless sensor networks. *IEEE Transactions on Wireless Communications (TWC)*, 2007. Accepted.

[92] K. Ren, K. Zeng, W. Lou, and P. Moran. On broadcast authentication in wireless sensor networks. In *International Conference on Wireless Algorithms, Systems, and Applications (WASA 2006)*, China, 2006.

[93] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2).

[94] Arvind Seshadri, Mark Luk, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. Scuba: Secure code update by attestation in sensor networks. In *ACM Workshop on Wireless Security (WiSe 2006)*, September 2006.

[95] A. Shamir. Identity based cryptosystems and signature schemes. In *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.

[96] M. Shao, S. Zhu, W. Zhang, and G. Cao. pdcs: Security and privacy support for data-centric sensor networks. 2007.

[97] E. Shi and A. Perrig. Designing secure sensor networks. *Wireless Communication Magazine*, 11(6), Dec. 2004.

[98] J. Spencer. *The Strange Logic of Random Graphs – Algorithms and Combinatorics 22*. Springer-Verlag, 2000.

[99] IEEE P1363a Standard. Standard specifications for public key cryptography. 2000. http://grouper.ieee.org/groups/1363/index.html.

[100] Ivan Stojmenovic. Geocasting with guaranteed delivery in sensor networks. *IEEE Wireless Comm.*, 11(6):29–37, Dec. 2004.

[101] N. Subramanian, C. Yang, and W. Zhang. Securing distributed data storage and retrieval in sensor networks. 2007.

[102] N. H. Vaidya and M. J. Miller. A mac protocol to reduce sensor network energy consumption using a wakeup radio. *IEEE Transactions on Mobile Computing*, 4(3):228–242, 2005.

[103] S. Čapkun and Jean-Pierre Hubaux. Secure positioning of wireless devices with application to sensor networks. In *IEEE INFOCOM'05*, Miami, FL, Mar. 2005.

[104] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Chang. Energy analysis for public-key cryptography for wireless sensor networks. In *IEEE PerCom'05*, Pisa, Italy, Mar. 2005.

[105] G. Wang, G. Cao, and T. La Porta. Movement-assisted sensor deployment. *IEEE Transactions on Mobile Computing*, 5(6):640–652, June 2006.

[106] C. Wong, M. Gouda, and S. lam. Secure group communication using key graphs. In *ACM SIGCOMM*, 1998.

[107] A. Wood and J. Stankovic. Denial of service in sensor networks. *IEEE Computer*, Oct. 2002.

[108] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *ACM MobiHoc*, 2005.

[109] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh. Toward resilient security in wireless sensor networks. May 2005.

[110] Y. Yang, X. Wang, S. Zhu, and G. Cao. Sdap: A secure hop-by-hop data aggregation protocol for sensor networks. 2006.

[111] F. Ye, H. Luo, S. Lu, and L. Zhang. Stastical en-route filtering of injected false data in sensor networks. In *IEEE INFOCOM'04*, Hong Kong, China, Mar. 2004.

[112] F. Ye, G. Zhong, S. Lu, and L. Zhang. Gradient broadcast: A robust data delivery protocol for large scale sensor networks. *ACM Wireless Networks (WINET)*, 11(2), 2005.

[113] Kaan Yüksel, Jens-Peter Kaps, and Berk Sunar. Universal hash functions for emerging ultra-low-power networks. In *Proceeding of The Communications Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, San Diego, CA, January 2004. Society for Modeling and Simulation International (SCS).

[114] W. Zhang and G. Cao. Group rekeying for filtering false data in sensor networks: A predistribution and local collaboration-based approach. In *IEEE INFOCOM*, 2005.

[115] W. Zhang, H. Song, S. Zhu, and G. Cao. Least privilege and privilege deprivation: Towards tolerating mobile sink compromises in wireless sensor networks. In *ACM MOBIHOC*, USA, May 2005.

[116] Y. Zhang and W. Lee. Intrusion detection in ad-hoc networks. In *ACM Mobicom*, 2000.

[117] Y. Zhang, W. Liu, Y. Fang, and D. Wu. Secure localization and authentication in ultra-wideband sensor networks. *IEEE Journal on Selected Areas in Communications, Special Issue on UWB Wireless Communications - Theory and Applications*, 24(4):829–835, April 2006.

[118] Y. Zhang, W. Liu, W. Lou, and Y. Fang. Location-based security mechanisms in wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 24(2):247–260, February 2006.

[119] L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network Magazine*, 13(6):24–30, November-December 1999.

[120] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *IEEE Symp. Security Privacy*, Oakland, CA, May 2004.

[121] S. Zhu, S. Setia, S. Xu, and S. Jajodia. Gkmpan: An efficient group rekeying scheme for secure multicast in ad-hoc networks. In *ACM Mobiquitous'04*, Boston, MA, USA, Aug. 2004.

[122] S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing pairwise keys for secure communications in ad hoc networks: A probabilistic approach. In *IEEE ICNP'03*, Atlanta, GA, Nov. 2003.

[123] S. Zhu, S. Xu, S. Setia, and S. Jajodia. Lhap: A lightweight hop-by-hop authentication protocol for ad-hoc networks. In *23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03)*, Providence, Rhode Island, USA, May 2003.